Adaptive Personalized Federated Learning for Non-IID Data with Continual Distribution Shift

Sisi Chen[1], Weijie Liu[1], Xiaoxi Zhang[1], Hong Xu[2], Wanyu Lin[3], Xu Chen[1]
[1]*Sun Yat-sen University,* [2]*The Chinese University of Hong Kong,* [3]*The Hong Kong Polytechnic University*
Email: {chenss8, liuwj55}@mail2.sysu.edu.cn, {zhangxx89, chenxu35}@mail.sysu.edu.cn
hongxu@cuhk.edu.hk, wan-yu.lin@polyu.edu.hk

***Abstract*—Federated Learning (FL) has surged in popularity, allowing machine learning models to be collaboratively trained using decentralized client data, all while upholding privacy and security standards. However, leveraging locally-stored data introduces challenges related to data heterogeneity. While many past studies have addressed this non-IID problem, they often overlook the dynamic nature of each individual client's data or disrupt its continuous shift. In this paper, our emphasis is on the challenges posed by temporal data distribution shift alongside non-IID data across clients, a more prevalent yet complex situation in real-world FL. We propose to analytically capture the evolving nature of each local data distribution, by modeling them as a time-varying composite of multiple latent Gaussian distributions. We then employ the expectation maximization (EM) algorithm to deduce the distribution model parameters based on the prevailing observed training data, ensuring that the learned mixture proportion weights mirror a consistent trajectory. Additionally, by embedding an adaptive data partitioning method into the EM algorithm and using each partition to train a distinct sub-model, we realize an intuitive and novel personalized FL paradigm. This refines the FL training by exploiting the heterogeneity and temporal shifts of clients' datasets. We derive analytical results to guarantee the convergence of our training method. Comprehensive tests across diverse datasets and distribution configurations also underscore our enhanced efficacy compared to several state-of-the-art.**

I. Introduction

Federated learning (FL) is a privacy-preserving distributed machine learning approach that enables multiple clients, such as mobile devices, to train a large model collaboratively without sharing their sensitive training data. In a conventional FL workflow, clients obtain the latest model parameters from a central global server. They then can utilize their local data to refine these models, and the updated local models are sent back periodically to the server for aggregation. Importantly, this local update process typically requires a number of iterations. As delineated in previous works such as [1] and [2], a large number of local steps can amplify the discrepancy between the objectives of individual clients and the overarching global model. This can slow down the overall convergence rate. One primary cause behind this challenge is the inherent heterogeneity in local data distributions across clients, resulting in the non-IID (non-independently and identically distributed) data predicaments. A multitude of research efforts have been directed towards understanding and alleviating the adversities posed by the non-IID data with the FL framework [3], [4].

In real-life scenarios, a significant contributor to the non-IID challenge is temporal data distribution shift, which is unfortunately overlooked in most FL convergence studies. Prolonged, uninterrupted training spanning days or even weeks can witness varying data distributions for a client at different temporal intervals [5], [6]. For instance, in a keyboard prediction training task, an office clerk's keyboard might oscillate between professional topics during the day and personal themes at night. Similarly, wearable devices for FL-based health monitoring could record peak activities in the morning, contrasting with calmer nighttime metrics. Streaming services also might see users flocking to light content on weekdays, while longer films become popular over weekends. Such temporal fluctuations in the training data, evident across myriad domains, potentially complicating the global training convergence.

While some state-of-the-art FL algorithms are designed to mitigate the negative effects of data heterogeneity across clients [7]–[9], they still assume that the distribution of any given client is static. Temporal shift at any given client's dataset is not well addressed. A few recent works started to study the distribution shift problem over time [10], e.g., dividing the time in a day into multiple blocks and training distinct models for individual time blocks [11]–[13]. However, this approach neglects the continuity in practical data shifts, which leads to the degradation of model training performance. To tackle this challenge, our basic idea is to first effectively track potentially continuous transitions in each client's data distribution, and then enable personalized models for different clients so as to adapt to data heterogeneity across clients. In achieving this, we draw inspiration from personalized FL studies [14], [15] where a customized model is trained for each client, and from clustered FL works where clients with similar data distributions are grouped together [1], [16], [17]. Our intuition is that a client's time-varying data distribution can be fit by a mixture of multiple base distributions. But the coefficients of any one client that controls the contribution of each individual base distribution within the mixed distribution (we call them mixture proportions) can continuously fluctuate and present differences from those of other clients. Therefore, we need to learn the data mixture patterns on the fly and adapt the local model training process to clients' diverse data. To the best of our knowledge, this is the first FL model training algorithm to cope with heterogeneous distributions

across clients and continuous temporal data shift within each individual client's dataset simultaneously with supreme experimental results.

We summarize our main **contributions** as follows:

1) *FL with learning time-varying data distributions (Section III)*: We provide a new FL training paradigm to capture changing datasets over training rounds, where we model a local distribution as a Gaussian Mixture Model (GMM) of K independent base Gaussian distributions. We integrate the EM algorithm into FL model training, where the GMM parameters for modeling the data mixture pattern of different clients are learned on the fly along with clients' local FL training processes. We next construct personalized training K sub-models simultaneously using properly selected data from each client's dataset. These sub-models are then integrated together for periodic testing and final inference use. Our method can track the data dynamics and exploit the combined capability of multiple sub-models per client, rather than selecting a single model, e.g., either a daytime or a nighttime model [11], [13], for each client. Therefore, we allow more complex temporal data differences in successive training rounds.

2) *Experimental validation (Section IV)*: We conduct extensive experiments across diverse datasets featuring heterogeneous and time-evolving data partitions for FL clients. The results show that our proposed method improves both training stability and the model accuracy compared with several state-of-the-art benchmarks.

II. Related Work

Non-IID data in FL often leads to a deterioration in accuracy due to model divergence. A line of work has been devoted to alleviating this issue, such as using data-sharing or data augmentation strategies, and incorporating personalization approaches [3], [15], [18]–[22]. While most of these methods can mitigate the issue of degraded performance caused by non-IID data, they often fail to account for the time-varying aspects that are present in many real-world scenarios.

Time-varying data distribution shift in FL escalates the training challenge with non-IID data, which was first identified in [10]. [11] and its variants [12] proposed to segment the time into multiple blocks. While they improved the performance, the practical application remains challenging. First, determining the clients' temporal state is difficult, especially during the transitions between two blocks. Second, the time block partition breaks the continuity of the evolving process. Recently, [13] recognized the gradual distribution shift in clients' population and proposed a temporal prior for the smooth transition, but they assumed a single distribution per client, which goes against reality. In contrast, we allow each client to possess data from multiple different base distributions and the proportion of each base distribution shifts gradually.

Clustered FL: Our algorithm shares similarities with clustered FL, which partitions clients into clusters, targeting individual objectives while sharing information to accelerate training [17], [23]. FedSoft assumes real-world local distributions resemble mixtures of various distributions [16]. Different from the traditional clustered FL approach where cluster center updating and FL training processes uniformly treat the local data of each client, our approach partitions a client's local data to train separate sub-models using respective subsets, to mitigate interference from irrelevant data.

III. Adaptive Personalized FL Training

In this section, we first introduce our FL model training architecture, where we model the time-varying distributions of clients' training datasets. We then describe our proposed FL training algorithm, named **pFedEM**, to manage non-IID and non-stationary data distributions in FL training.

A. Training Paradigm with Mixed Datasets

We consider an FL setting for a set of client $\mathcal{I}$ where the local data distributions $\mathcal{D}_i^r, \forall i \in \mathcal{I}$ are independently and continuously shift over time r. Since the FL training process consists of a number of discrete iterations, without the loss of generality, we use r in the following to index the number of training rounds. The actual local dataset of client i, defined as D_i^r, used for training in round $r \in [1, R]$, is IID sampled from $\mathcal{D}_i^r$ at communication round r. The ultimate goal is to minimize the expected loss over all the clients under their full datasets collected in the training process, say R rounds:

$$\min_{\omega} L(\omega) = \sum_{i \in \mathcal{I}} p_i L_i(\omega), \tag{1}$$

where $L_i(\omega)$ is the expected loss of client i and defined as

$$L_i(\omega) = \frac{1}{|D_i|} \sum_{(x,y) \in D_i} \ell(\omega, (x, y)), \sum_{i \in \mathcal{I}} p_i = 1, \tag{2}$$

where $D_i = \cup_r^R D_i^r$, $p_i = \frac{|D_i|}{\sum_{i=1}^{|\mathcal{I}|} |D_i|}$, and $\ell(\cdot)$ is the loss function for a single data sample (x, y), e.g., cross-entropy or mean square error.

Mixed datasets and multiple models. To characterize the inter-client data heterogeneity and also the intra-client data variation, we model the local data distribution $\mathcal{D}_i^r$ of client i to be a mixture of K independent *base* distributions, with an unknown mixture pattern to be learned, formalized in Section III-B. We then propose to train K distinct models at each client and the central server. Our algorithm illustrated in Section III-B will decide how to partition the clients' data and choose the right data to train each individual model. The k-th training model maintained at the server after R iterations is denoted by ω_k, and each k-th final model at the client is ω_{ik}.

Therefore, the optimization goal of each client i is:

$$\min_{\omega_1, ..., \omega_K} L_i(\omega_1, ..., \omega_K) = \frac{1}{K} \sum_{k=1}^{K} L_{ik}(\omega_k), \tag{3}$$

where we use $h_{\omega_k}(x)$ to represent the k-th model's prediction under data feature x, and we define the average empirical loss at client i under the k-th model to be:

$$L_{ik}(\omega_k) = \frac{1}{|D_{ik}^r|} \sum_{(x,y) \in D_{ik}^r} \ell\left(h_{\omega_k}(x), y\right), \tag{4}$$

where D_{ik}^r is an unknown time-varying subset of D_i^r for training the k-th model at client i and is drawn from the k-th unknown distribution $\mathcal{D}_k$. We then model client i's training dataset at the r-th round as $D_i^r = \bigcup_{k=1}^K D_{ik}^r$. When each round starts, each client $i \in \mathcal{I}$ first downloads these K models and performs a certain number of local update iterations to train each model, using data samples selected from the dataset D_i^r, determined by our algorithm introduced in Section III-B. The server then aggregates K updated models uploaded by each client. For the k-th model and the $(r+1)$-th communication round, the server aggregates the parameters as:

$$\omega_k^{r+1} = \omega_k^r + \sum_{i=1}^{|\mathcal{I}|} p_i(\omega_{ik}^{r+1} - \omega_{ik}^r), \tag{5}$$

where ω_{ik}^{r+1} represents the uploaded parameter from client i for the k-th model to be updated in round $r+1$. The objective function of training the k-th model at the server is:

$$L_k(\omega_k) = \frac{1}{|D_k^r|} \sum_{i \in \mathcal{I}} |D_{ik}^r| L_{ik}(\omega_k), \tag{6}$$

where $|D_k^r| = \sum_{i \in \mathcal{I}} |D_{ik}^r|$, denoting the total size of data sampled from base distribution k at r over all the clients.

B. FL with Time-varying and Non-IID Datasets

We propose to model the data distribution as a linear combination of K independent *base* distributions, each of which is denoted by $\mathcal{D}_k$. For client i, we consider his local data distribution $\mathcal{D}_i^r$ at round r as the linear combination of K independent distributions. Let π_{ik}^r denotes the mixture proportion of $\mathcal{D}_k$ in $\mathcal{D}_i^r$. The data distribution of client i at any time r is:

$$\mathcal{D}_i^r = \sum_{k=1}^K \pi_{ik}^r \mathcal{D}_k, \text{ with: } \sum_{k=1}^K \pi_{ik}^r = 1. \tag{7}$$

Note that for all clients, the k-th data distribution at time r are identical, but both the *base* distributions $\mathcal{D}_k^r$ and mixture proportions π_{ik}^r can change over time. The heterogeneity across clients can also be characterized, since their coefficients $\pi_i^r = \{\pi_{i1}^r, ..., \pi_{iK}^r\}, \forall i \in \mathcal{I}$ vary with each other.

Inspired by the Central Limit Theorem [24], we model each $\mathcal{D}_k$ as a distinct Gaussian distribution, thus $\mathcal{D}_i^r$ can be modelled as the Gaussian Mixture Model (GMM) with K distribution components. We then adopt the expectation–maximization (EM) algorithm [25] to fit the GMM parameters. Each client will first estimate its own GMM parameters based on its local data distribution. Then, in order to enhance the prediction on base distributions by leveraging all the clients' datasets, we conduct GMM parameter aggregation per round at the server. The aggregated parameter π_k will also be used for the final inference (see (15)). This GMM parameter estimation procedure is carried out as the training proceeds on the fly, consisting of the E step, M step, and GMM aggregation phases.

In general, EM is an approximate algorithm to solve the maximum log-likelihood on joint probability of multiple random variables while only some of them are observable. In our scenario, we encounter that each client's data come from a mixture of multiple latent Gaussian distributions, and we need to figure out what each Gaussian distribution looks like and how likely the client's data samples come from each of the distributions to enable data partitioning. Since each client' data D_i^r has a varying data mixture pattern over training round, our EM algorithm for estimating the distribution $\mathcal{D}_k$'s and π_{ik}^r should be carried out round by round.

E Step: We calculate $\gamma_{ik}^r(z)$, the posterior probability that, for each sample $z = f(x), x \in D_i$, the data feature x is generated from the k-th Gaussian distribution, given the observed z. Here, $f(\cdot)$ is the backbone network for feature extraction to extract discriminative features. Estimating $\gamma_{ik}^r(z)$ can save the computation and storage compared to estimating $\gamma_{ik}^r(x)$ for all x, due to a largely reduced number of dimensions of z compared to x.

$$\gamma_{ik}^r(z) = \frac{\pi_{ik}^r \mathcal{N}(z|\mu_{ik}^r, \sigma_{ik}^r)}{\sum_{k=1}^K \pi_{ik}^r \mathcal{N}(z|\mu_{ik}^r, \sigma_{ik}^r)}. \tag{8}$$

M Step: Each client i evaluates the GMM parameters for each data mixture component, including the unknown mean μ_{ik} and variance σ_{ik}^2 of the k-th latent Gaussian distribution, as well as the mixture proportion π_{ik}, in each round r using the posterior probabilities $\gamma_{ik}^r(\cdot)$ calculated in the E Step.

$$\mu_{ik}^{r+1} = \frac{\sum_{n=1}^{|D_i^r|} \gamma_{ik}^r(z_n) z_n}{\sum_{n=1}^{|D_i^r|} \gamma_{ik}^r(z_n)}, \tag{9}$$

$$[\sigma_{ik}^2]^{r+1} = \frac{\sum_{n=1}^{|D_i^r|} \gamma_{ik}^r(z_n)(z_n - \mu_{ik}^{r+1})^2}{\sum_{n=1}^{|D_i^r|} \gamma_{ik}^r(z_n)}, \tag{10}$$

$$\pi_{ik}^{r+1} = \frac{\sum_{n=1}^{|D_i^r|} \gamma_{ik}^r(z_n)}{|D_i^r|}, \tag{11}$$

where z_n is the feature for sample $x_n \in D_i$.

GMM Aggregation: At the aggregation stage, the server aggregates all GMM parameters uploaded from clients for each component separately. For the k-th global component,

$$\mu_k^{r+1} = \sum_{i \in \mathcal{I}} \frac{|D_{ik}^r|}{\sum_{j \in \mathcal{I}} |D_{jk}^r|} \mu_{ik}^{r+1}, \tag{12}$$

$$[\sigma_k^2]^{r+1} = \sum_{i \in \mathcal{I}} \frac{|D_{ik}^r|}{\sum_{j \in \mathcal{I}} |D_{jk}^r|} [\sigma_{ik}^2]^{r+1}, \tag{13}$$

$$\pi_k^{r+1} = \frac{\sum_{i \in \mathcal{I}} |D_{ik}^r|}{\sum_{q=1}^K \sum_{j \in \mathcal{I}} |D_{jq}^r|}. \tag{14}$$

Workflow of pFedEM. At the beginning of a communication round, the server broadcasts all parameters to all clients (line 3 of Algorithm 1). Each client will first update the GMM parameters with an EM step (lines 6–7 of Algorithm 1). The obtained posterior for each training sample will be applied to partition the data set into K clusters (lines 9–11 of Algorithm 1). Next, the client runs multiple local steps for updating K network components, each of which is only updated using the corresponding data cluster (line 12 of Algorithm 1). After

Algorithm 1: Personalized FL Algorithm for Non-IID Data with Continuous Temporal Shifts (pFedEM)

Input: Number of communication rounds R; Number of local iterations J; Number of components for the model K

Output: Network parameters: $\omega^r = (\omega_1^r, ..., \omega_K^r)$; GMM parameters: $\theta_k^r = (\mu_k^r, [\sigma_k^2]^r, \pi_k^r)$

Initialize: Network ω^0, GMM parameters θ_k^0

1 $r \leftarrow 0$;
2 **repeat**
3 Server broadcasts ω^r and $\theta_k^r, \forall k \in [1, K]$;
4 **foreach** *client* $i \in \mathcal{I}(r)$ **do**
5 $\theta_{ik}^{r,0} \leftarrow \theta_k^r, \forall k \in [1, K]$;
6 Compute posterior γ_{ik}^r for all local samples with Eq. (8);
7 Update GMM parameters $\mu_{ik}^{r+1}, [\sigma_{ik}^2]^{r+1}, \pi_{ik}^{r+1}$ using Eq. (9)–(11);
8 $D_{ik} \leftarrow \emptyset$, $\forall k \in [1, K]$;
9 **foreach** $x_n \in D_i$ **do**
10 $c_n \leftarrow \arg\max_k \gamma_{ik}^r$;
11 $D_{ic_n} \leftarrow D_{ic_n} \cup \{x_n\}$;
12 Run J local updates for each K network models by optimizing Eq. (4) and get $\omega_{ik}^{r+1}, \forall k \in [1, K]$;
13 Upload all ω^{r+1} and θ_k^{r+1} to the server;
14 Aggregate models of each component with Eq. (5);
15 Aggregate GMM parameters with Eq.(12)–(14);
16 **until** $r < R$;

that, all clients upload new parameters to the server (line 13 of Algorithm 1) who aggregates all network and GMM parameters (lines 14–15 of Algorithm 1). At the inference stage, a prediction at test time is obtained by model interpolation,

$$h_\omega(x) = \sum_{k=1}^{K} \pi_k h_{\omega_k}(x). \tag{15}$$

It is worth noting that this is different from the training stage where each sample is only applied for one model's training.

IV. Experiments and Results

In this section, we conduct extensive experiments on four benchmark datasets with two data distribution shift modes.

A. Experimental Setup

1) Dataset and models: We use two pairs of datasets: EMNIST (Digits and Letters) and CIFAR (CIFAR-10 and CIFAR-100). Each pair includes two datasets to generate a continuous distribution shift by changing their ratios. We train the MoblieNet-v2 [26] for CIFAR and a two-layer convolutional neural network [27] for EMNIST.

2) Distribution shift setting: We design two data distribution shift modes: linear shift and cosine shift, spanning $R = 200$ communication rounds across multiple cycles. In a linear shift cycle with CIFAR dataset, the portion of CIFAR-10 increases linearly from 0 to 1, and then decreases to 0. In contrast, the portion of CIFAR-100 decreases linearly from 1 to 0 and rises back to 1. Similarly, for a single cosine shift cycle, the portion of different datasets also increases and decreases but follows a cosine pattern.

3) Baselines: To verify the effectiveness of our method, we compare it with four pioneering baselines:

- FedAvg [28]: The vanilla FL method which trains one global model for all clients.
- FedEM [22]: A federated multi-task learning approach that characterizes each local distribution as a mixture of multiple unknown underlying distributions.
- FedGMM [15]: A personalized FL method using GMM to fit the input data distribution across different clients with assumptions similar to those in FedEM [22].
- CFL [23]: A clustered FL approach that groups clients into clusters to alleviate the distribution divergence issue among different clients' local data.

We apply the same backbone model and hyperparameters among baselines and our method for fair comparisons.

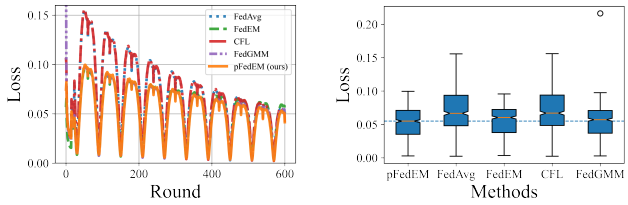
4) Hyperparameters: For all experiments, we conduct three sets of training experiments for $|\mathcal{I}| = 80$ clients, where each training round consists of $J = 20$ local iterations: one with 200 rounds of training, including a single cycle period, and another with 200 rounds of training, including three cycle periods. Additionally, we conduct an experiment with 600 rounds, including ten cycle periods. We set the number of base distributions $K = 2$ to represent the daytime and nighttime respectively. We set the learning rate 0.01 for EMNIST, 0.03 for CIFAR, and the batch size 128 unless otherwise specified.

B. Experimental results and interpretation

We first evaluate **pFedEM** on EMNIST to demonstrate its superiority in terms of training loss, test accuracy, and adaptability in a larger K scenario. We then further evaluate on CIFAR to demonstrate the superior performance of **pFedEM** with complex datasets and extra shifting cycles. Additionally, we demonstrate the stability of **pFedEM** and showcase its ability to capture the data distribution shifts during the training.

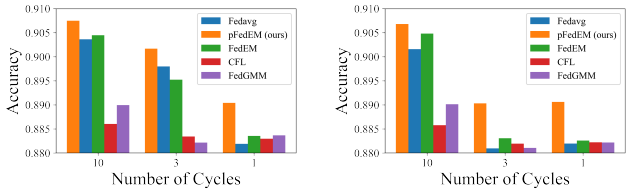
1) Performance on EMNIST Dataset: Fig. 1 compares the **Training Loss** on 10-cycle cosine shift setting, showing the aggregated loss per round and the spread of loss values respectively. Although **pFedEM** has similar loss values with FedEM and FedGMM, the training loss of **pFedEM** drops faster than that of FedEM and FedGMM, achieving the minimum in the last cycle. Additionally, **pFedEM** has narrower spreads, and the lowest mean and median loss values, indicating its faster convergence speed and closer approach to the global optimum.

Test Accuracy. We evaluate the final trained model on EMNIST consisting of 50% digits and 50% letters. As shown in Fig. 2, **pFedEM** achieved the highest test accuracy over all baselines in both shift settings. FedAvg and FedEM exhibited



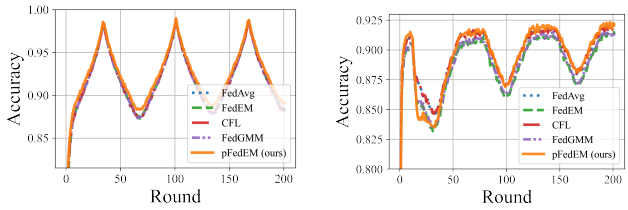
(a) Training Loss each round (b) Statistics of Loss Valuss

Fig. 1. Train Loss with Cosine 10 Cycles on EMNIST Dataset.



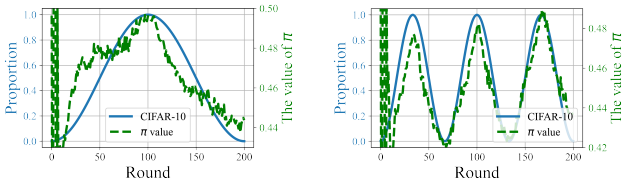
(a) Linear shift (b) Cosine shift

Fig. 2. Final Test Accuracy with EMNIST using 50% Digits and 50% Letters



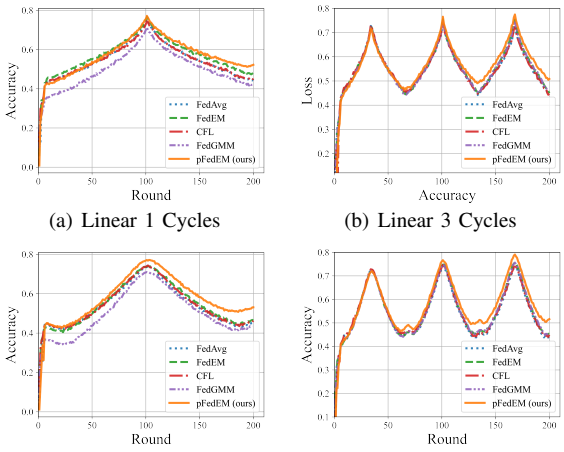
(a) EMNIST ($K = 2$) (b) EMNIST, FMNIST($K = 3$)

Fig. 3. Test Accuracy on EMNIST and FMNIST with 3-cycle Linear Shift



(a) 1-Cycle (b) 3-Cycle

Fig. 4. π value on CIFAR dataset (Cosine shift)



(a) Linear 1 Cycles (b) Linear 3 Cycles

(c) Cosine 1 Cycles (d) Cosine 3 Cycles

Fig. 5. Test Accuracy on CIFAR Dataset

similar performance, while CFL consistently performed the worst. These results also highlight the limitations of traditional FL methods in handling data distribution shift issues.

Extending to $K = 3$. The paper focuses on $K = 2$, aligning with a straightforward intuition: the training data exhibits two distinct time-varying distributions corresponding to daytime and nighttime. However, our algorithm can adapt to more complex situations by setting K to larger values. To demonstrate this, we extened K to 3 by adding the Fashion-MNIST (FMNIST) dataset. Fig. 3 compares the per-round test accuracy for $K = 2$ and 3 with the 3-cycle linear shift. Remarkably, **pFedEM** consistently performs the best, showing its superiority and adaptability across different distribution patterns. Future research can further investigate the challenges and implications of different data distribution shift patterns, which are beyond the scope of this paper.

2) Performance on CIFAR Dataset: **Test Accuracy.** To further evaluate **pFedEM** under complex datasets, we conduct experiments on the CIFAR, with fine-grained examination of experimental results in Fig. 5, where the data distributions on test and training sets remain identical throughout the training process. The test accuracy of all algorithms fluctuates during the whole training process due to the distinct learning complexities of two datasets. In general, **pFedEM** consistently improves model performance: The peak model accuracy in each cycle surpasses that of the preceding cycle, and the minimum values also exhibit the same upward trend. In contrast, other baselines do not sustain such an upward trend. As shown in Fig. 5(a) and Fig. 5(c), in the scenario with a slower distribution shift (1 cycle in 200 rounds), **pFedEM** can consistently achieve the highest test accuracy throughout the training process, particularly in the second half of the cycle. As the distribution shift accelerates in the setting shown in Fig. 5(b) and Fig. 5(d) (3 cycles in 200 rounds), all methods, except **pFedEM**, fail to continue improving the model performance after the first cycle. We mention that such a phenomenon is more pronounced under cosine mode, which is closer to the real distribution shift process as the distribution shift is always nonlinear in reality.

The ability to learn the distribution shift patterns. To further understand **pFedEM**, Fig. 4 present the variation of π in GMM, which represents the learned ratio of the first dataset, during training on CIFAR. **pFedEM** can successfully capture the distribution shift in different cycle settings, verifying its ability to automatically learn shift patterns.

We further extended the training process to 600 rounds spanning 10 cycles. As shown in Fig. 6, the superiority of **pFedEM** has become more pronounced with the inclusion of additional cycles and rounds. These results show that **pFedEM** can mitigate the adverse impacts of continuous data distribution shifts, especially in a prolonged training process.

3) Stability: We compute the standard deviations of the per-round test accuracy under the two distribution shift modes for each algorithm in Table I, where the minimum values are underlined. Compared with all baselines, **pFedEM** attains the minimum standard deviations under all settings, demonstrating

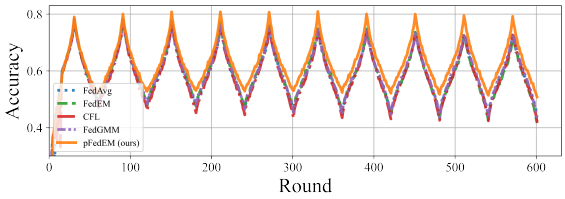


Fig. 6. Test accuracy with linear distribution shift for 10 cycles on CIFAR.

the superiority of our **pFedEM** in terms of training

TABLE I
STANDARD DEVIATION OF TEST ACCURACY IN 3 CYCLES.

Dataset	Shift	FedEM	FedAvg	FedGMM	CFL	pFedEM
EMNIST	**Linear**	0.0316	0.0318	0.0315	0.0317	0.0296
	Cosine	0.0395	0.0399	0.0393	0.0397	0.0371
CIFAR	**Linear**	0.0803	0.0772	0.0793	0.0773	0.0764
	Cosine	0.0990	0.0976	0.0978	0.0969	0.0938

V. CONCLUSION

In this paper, we propose **pFedEM**, a new FL training algorithm to alleviate the data distribution shift problem over time, by modeling each local distribution as a GMM. Incorporating the EM algorithm into the FL setting, **pFedEM** can capture the continuous distribution shift process during training to train distinct models with partitioned training datasets. Extensive experiments demonstrate its superior performance.

ACKNOWLEDGMENT

This work was supported by NSFC grant (No. 62102460), Guangzhou Science and Technology Plan Project (No. 202201011392), Guangdong Basic and Applied Basic Research Foundation (No. 2023A1515012982), and Young Outstanding Award under the Zhujiang Talent Plan of Guangdong Province. This work was supported in part by Guangdong Basic and Applied Basic Research Foundation (No. 2023B1515120058); Guangzhou Basic and Applied Basic Research Program (No. 2024A04J6367). Xiaoxi Zhang is the corresponding author.

REFERENCES

[1] M. Duan, D. Liu, X. Ji, R. Liu, L. Liang, X. Chen, and Y. Tan, "Fedgroup: Efficient federated learning via decomposed similarity-based clustering," in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE, 2021, pp. 228–237.

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

[3] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[4] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.

[5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[6] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

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

[8] 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.

[9] K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, J. Rush, and S. Prakash, "Federated reconstruction: Partially local federated learning," *Advances in Neural Information Processing Systems*, 2021.

[10] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[11] H. Eichner, T. Koren, B. McMahan, N. Srebro, and K. Talwar, "Semi-cyclic stochastic gradient descent," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1764–1773.

[12] Y. Ding, C. Niu, Y. Yan, Z. Zheng, F. Wu, G. Chen, S. Tang, and R. Jia, "Distributed optimization over block-cyclic data," *arXiv preprint arXiv:2002.07454*, 2020.

[13] C. Zhu, Z. Xu, M. Chen, J. Konečnỳ, A. Hard, and T. Goldstein, "Diurnal or nocturnal? federated learning of multi-branch networks from periodically shifting distributions," in *International Conference on Learning Representations*, 2021.

[14] O. Marfoq, G. Neglia, R. Vidal, and L. Kameni, "Personalized federated learning through local memorization," in *International Conference on Machine Learning*. PMLR, 2022, pp. 15 070–15 092.

[15] Y. Wu, S. Zhang, W. Yu, Y. Liu, Q. Gu, D. Zhou, H. Chen, and W. Cheng, "Personalized federated learning under mixture of distributions," *arXiv preprint arXiv:2305.01068*, 2023.

[16] Y. Ruan and C. Joe-Wong, "Fedsoft: Soft clustered federated learning with proximal local updating," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 8124–8131.

[17] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 586–19 597, 2020.

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

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

[20] M. Duan, D. Liu, X. Chen, Y. Tan, J. Ren, L. Qiao, and L. Liang, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *International conference on computer design*. IEEE, 2019, pp. 246–254.

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

[22] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, "Federated multi-task learning under a mixture of distributions," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 434–15 447, 2021.

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

[24] H. Fischer, *A history of the central limit theorem: from classical to modern probability theory*. Springer, 01 2011, vol. 4.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977. [Online]. Available: http://www.jstor.org/stable/2984875

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[27] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

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