Federated Multi-Phase Curriculum Learning to Synchronously Correlate User Heterogeneity

Mingjie Wang ©, Jianxiong Guo ©, Member, IEEE, and Weijia Jia ©, Fellow, IEEE

Abstract—Federated learning (FL) is a decentralized learning method used to train machine learning algorithms. In FL, a global model iteratively collects the parameters of local models without accessing their local data. However, a significant challenge in FL is handling the heterogeneity of local data distribution, which often results in a drifted global model that is difficult to converge. To address this issue, current methods employ different strategies, such as knowledge distillation, weighted model aggregation, and multitask learning. These approaches are referred to as asynchronous FL, as they align user models either locally or posthoc, where model drift has already occurred or has been underestimated. In this article, we propose an active and synchronous correlation approach to address the challenge of user heterogeneity in FL. Specifically, our approach aims to approximate FL as standard deep learning by actively and synchronously scheduling user learning pace in each round with a dynamic multiphase curriculum. A global curriculum is formed by an autoregressive autoencoder that integrates all user curricula on the server. This global curriculum is then divided into multiple phases and broadcast to users to measure and align the domain-agnostic learning pace. Empirical studies demonstrate that our approach outperforms existing asynchronous approaches in terms of generalization performance, even in the presence of severe user heterogeneity.

Impact Statement—We propose an active and synchronous correlation approach to address the challenge of user heterogeneity in FL. The proposed approach aims to approximate FL as standard deep learning by actively and synchronously scheduling user learning pace in each round with a dynamic multiphase curriculum. It shows that the current approaches for handling user heterogeneity in FL referred to as asynchronous FL, often result in a drifted global model that is difficult to converge. In contrast, the proposed approach aligns user models synchronously and actively by scheduling their learning pace, resulting in improved generalization performance and avoiding model drift, even in the presence of severe user heterogeneity. We employ a curriculum learning method based on loss function to integrate the learning state of users on the server, which is then divided into multiple phases and broadcast to users to measure and align the domain-agnostic learning pace. Empirical studies demonstrate that their approach outperforms existing asynchronous approaches in terms of generalization performance. Overall, the proposed active and synchronous correlation approach for handling user heterogeneity in FL offers a promising solution to the current challenges faced by FL, enabling improved convergence and generalization performance.

Index Terms—Curriculum learning (CL), federated learning (FL), heterogeneity data, synchronization.

I. INTRODUCTION

FEDERATED learning (FL) enables the training of a robust deep learning model using large-scale data distributed across multiple decentralized sources without directly accessing the data. The standard FL model, represented by FedAvg [1], updates the global model by iteratively averaging the parameters of local models, thus avoiding direct contact with local user data. This privacy-preserving scheme has made FL a popular paradigm for facilitating real-world applications. To date, FL has been applied to various application areas, such as machinery fault diagnosis [2], the internet of vehicles [3], and healthcare systems [4].

However, when applied to real-world scenarios, the standard FL model faces practical challenges arising from data heterogeneity. Local data in multiple sources or clients often have independent data sampling spaces and unique local data distributions, known as nonindependent and identically distributed (Non-IID) data. This inconsistency between local objective functions and global optimization directions frequently leads to problems. Studies in [5], [6], and [7] have demonstrated that applying the standard FL model to Non-IID data, such as with FedAvg, results in drifting local models and catastrophic forgetting of global information. Consequently, this approach leads to suboptimal convergence speed and model performance. Thus, addressing Non-IID local datasets has become a crucial topic in FL.

From the above analysis, it is clear that the standard FL model with elementwise averaging operations, such as FedAvg, is not ideal for real-world applications. This has attracted numerous researchers to focus on addressing heterogeneity in FL. Current methods can be broadly categorized into four groups as follows.
1) **Alignment of global and local objectives** [8], [9]: This approach aims to mitigate model drift by modifying the local target to align with a desired global target, thus harmonizing global and local objectives. However, this approach’s limitation is that it can only be applied to clients due to inherent structural differences in their partial local data.

2) **Weighted aggregation** [10], [11], [12], [13]: This method achieves aggregation by assigning weights to local models, optimizing fusion performance for the global model at the server. Nevertheless, the drawback of this approach is its reliance on proxy data, and it occurs after model drift has already taken place.

3) **Knowledge distillation (KD)** [14], [15], [16]: This technique combines real data with synthesized data, aiming for a more balanced data distribution at the client or server level to align multiclient domain knowledge. However, this approach’s shortcoming is that the generator model, which produces synthetic data, is also trained on Non-IID data. Consequently, the generator model is affected by the inherent data heterogeneity and model drift.

4) **Multitask learning** [17], [18], [19], [20]: This method enables federated personalization by learning a generic and personalized FL at clients. However, the disadvantage of this approach is that multitask learning and personalized FL inherently contradict each other.

We observe that these approaches either underestimate heterogeneity based on inconsistent principles (client drift mitigation and federated personalization) or address model drift in a posthoc manner (KD and weighted model aggregation). We refer to them as asynchronous FL, considering the fitting degree of the model simultaneously. However, existing posthoc calibration methods produce overly confident predictions under domain shift, and the prerequisite of proxy data can leave such an approach susceptible to overfitting the model.

Given the challenges of user heterogeneity and the limitations of previous asynchronous FL approaches, we adopt a synchronous FL model by approximating the learning speed of the local model to that of the centralized deep learning model, leveraging consistency and online heterogeneity alignment.

Specifically, model aggregation under IID data consistently and efficiently converges to a relative global optimum, as shown in Fig. 1 (asynchronous IID) [21]. In contrast, Fig. 1 (asynchronous Non-IID) depicts the global model $\omega^t$ obtained from the model aggregation at the server, trained by Non-IID data in clients. This process disregards local knowledge incompatibility, leading to knowledge forgetting and performance degradation in the global model [6], [22].

Curriculum learning (CL) is a technique that aims to enhance the generalization ability of models by progressively increasing the difficulty level of the training data. This approach has been demonstrated to improve model performance across a variety of tasks. However, in distributed learning scenarios, such as FL, additional challenges need to be addressed, including ensuring consistency and reliability of local training and global aggregation. In this article, we propose a method called federal curriculum learning (FedCL), which is based on synchronous learning and aims to overcome these challenges. FedCL comprises three components: local CL, global curriculum ensemble, and multiphase correlation. First, each client in the FL system learns a local curriculum to evaluate model-aware difficulty based on confidence. This enables clients to adjust their learning strategies according to individual progress. Second, the global server summarizes the clients’ progress to measure performance change based on global learning progress, which is then broadcast to all users. This ensures that all users benefit from globally consistent learning progress that reflects online synchronization across all users. Finally, based on this global learning progress, we further align the learning progress from the global model to users’ local networks incrementally by incorporating a self-paced learning methodology. Our experiments demonstrate that FedCL outperforms existing FL methods on several benchmark datasets, including CIFAR-10, CIFAR-100, and MNIST. The results indicate that FedCL exhibits better generalization ability and can effectively mitigate the negative impact of heterogeneous data distribution in FL systems. In addition, we analyze the influence of different hyperparameters on FedCL’s performance and provide guidelines for selecting appropriate hyperparameters. In summary, our main contributions can be summarized as follows.

1) In contrast to previous asynchronous learning approaches, we propose a synchronous FL method that aggregates local models before they drift, effectively avoiding model
drift and enabling clients to achieve competitive global model performance.

2) Our method directly synchronizes each local model, arranging a hierarchical curriculum for personalized FL based on efficient progress synchronization and online heterogeneity alignment. Experiments demonstrate that this curriculum reduces the risk of inductive bias and overfitting of local models, leading to better generalization on Non-IID data.

3) Comprehensive empirical studies, supported by theoretical explanations, show that FedCL achieves superior generalization performance using fewer communication rounds. Furthermore, we conduct thorough ablation studies, which corroborate the advantages of FedCL.

4) We perform extensive experiments to validate the effectiveness of our approach in various environments, demonstrating that FedCL exhibits better generalization using fewer communication rounds.

**Organization**: Section II summarizes the works related to this paper. Section III establishes the basic settings of our FL algorithm and introduces that other important approaches. Section IV expands our model algorithm in detail which establishes a multiphase CL approach to deal with user heterogeneity. Besides, Section V sets up the experiments, which compare with other baselines on different datasets and parameters, and then discusses the results. Finally, Section VI summarizes our research and future works.

## II. RELATED WORK

In this article, we discuss FL, a distributed machine learning framework initially proposed by Google. FL enables multiple clients to collaborate in training a model without disclosing their data. FedAvg [1] is a representative FL method, where each client trains the model locally and uploads its parameters to the server. The server then aggregates the local model parameters and returns it to all clients for the next training round. However, FedAvg’s performance significantly declines in the case of Non-IID data, where data imbalance and differing local client distributions lead to global model divergence. This divergence limits the model’s relevance and accuracy, resulting in increased communication rounds required for convergence.

1) **FL With Heterogeneous Data**: Several studies aim to improve FedAvg on heterogeneous data. These studies can be divided into four categories: alignment of global and local objectives, weighted aggregation, KD, and multitask learning. We will introduce them individually here.

   In terms of research on improving the consistency of global and local objectives, FedProx [12] enhances the local training model by incorporating an approximation term into the target during the local training process. It can train and predict various natural distributions with good performance on different natural distribution offsets owned by various clients. FedDC [8] employs learned auxiliary local drift variables to bridge the gap between local and global model parameters, achieving parameter-level consistency. The global-local forgetting compensation [23] model mitigates catastrophic forgetting in FL using class-aware gradient compensation, relation distillation loss, and a privacy-preserving communication mechanism, thus boosting average accuracy. However, differences in local data information structure limit each user’s effectiveness in learning the global goal and the aggregation model’s consistency. Regarding studies on improving weighted aggregation, FedDissolve [15] extracts anonymous data from user model logit-vectors and shares this metadata with users for KD. FedGen [16] proposes a generator to produce data to balance the distribution of corresponding features under the given prediction, reducing the influence of data heterogeneity. The FBL [24] model is designed to mitigate the problem of forgetting old classes in federated semantic segmentation by employing adaptive class-balanced labeling and incorporating intraclient and interclient forgetting-balanced mechanisms. The effectiveness of knowledge extraction is limited by local model and generator performance, and there remains the risk of local model drift. For multitask learning research, CFMTL [17] uses a clustering method to measure local data similarity among groups, allowing clients with similar data distribution to be grouped, transitioning from single tasks to group tasks. Regrettably, multitask learning constrains the upper limit of personalization and global optimization of local models. Moreover, under extreme Non-IID data, multitask learning will degenerate into the primitive FedAvg.

   The existing methods primarily employ asynchronous FL, which modifies partial or posterior user models or specific aggregated drift models, leading to overfitted or underfitted models. Consequently, we propose a synchronous FL method based on CL, aligning the global model with local models in terms of model fitting. CL can assess each local model’s learning situation and the global model’s learning progress, ensuring personalization and consistency of FL under Non-IID data.

2) **Curriculum Learning**: CL is an approach that assigns different weights to samples, guiding the model learning process to follow a sequence from easy to difficult. CL can accelerate the training of machine learning models, reduce the number of iterations, and achieve a better local optimal state [25]. Most CL designs are currently based on the framework of difficulty measurement and training scheduler, with automatic CL offering more flexibility since it can automatically participate and consider its feedback during the process. Automatic CL improvements can be divided into three categories: optimizing the selection of learning samples, addressing dimension and edge problems through self-supervised learning, and optimizing local personalization.

   In research on optimizing the selection of learning samples, the training sample of CL transitioned from relying on human
experts to avoid local minima [26] to automatically selecting strategies using various standard samples during training [27]. Hacohen and Weinshall [28] showed that, under mild conditions, it does not change the corresponding global minimum of the optimization function. The selection criteria of these methods were developed for various learning settings and hypothesis class assumptions, revealing that many CL methods prefer easy-to-learn samples [29], [30].

Regarding optimizing dimension and edge problems, researchers have considered self-supervised learning as a means to address these issues. Specifically, Kumar et al. [30] proposed a method that considers examples with a loss, while Katharopoulos et al. [31] correlated the importance of a sample with its norm of loss gradient of network parameters. In the context of noisy examples, Jiang et al. [32] examined the robustness of CL and proposed curriculum accelerated self-supervised learning, which identifies memorable examples that can be safely deleted without affecting generalization and demonstrates that noisy examples rank higher in terms of the number of forgotten events. In addition, to address the edge problem of CL optimization, a covering model has been proposed that achieves better generalization and improved convergence rates [26].

In the realm of optimizing local personalization, to address the problem of heterogeneous data in virtual reality, many studies have been devoted to training the best personalized models for each customer [33]. A general curriculum learning strategy was developed for the optimization of training dynamics [34]. Yang et al. [35] proposed a model-agnostic hybrid CL strategy.

In our work, CL is utilized to alleviate the heterogeneity of local data and unify client learning progress. Due to the heterogeneity of data, the difficulty lies in helping the model approximate the difficulty distribution of generated data to that of global data. Therefore, we have added some concepts to guide the optimization process through CL, so as to better accomplish the final task in the heterogeneous environment.

III. NOTATIONS AND PRELIMINARIES

This section presents the mathematical formulation and objective function in FL. After introducing the concept of CL, we redefine the FL objective function used in this article. Furthermore, to ensure data privacy during client-to-server transmission, the Gaussian mixture model (GMM) is introduced. The frequently used notations are summarized in Table I.

A. Problem Notations

FL is a machine learning paradigm that aims to protect user privacy by allowing participants to collaborate on modeling without sharing data. Consider a typical FL setting for supervised learning of a multiclass classification problem, we define the input feature space as $\mathcal{X} \subset \mathbb{R}^d$, the latent feature space as $\mathcal{Z} \subset \mathbb{R}^p$ with $d < p$, and the output space as $\mathcal{Y} \subset \mathbb{R}$. $\mathcal{T}$ is a domain (or called task), denoted by $\mathcal{T} := \langle \mathcal{D}, \omega^k \rangle$, which has a data distribution $\mathcal{D}$ over $\mathcal{X}$ and a ground-truth labeling function $c^* : \mathcal{X} \to \mathcal{Y}$. A model can be parameterized by $\omega := \langle \omega^f; \omega^h \rangle$, which consists of two components: a feature extractor $f : \mathcal{X} \to \mathcal{Z}$ parameterized by $\omega^f$ and a predictor $h : \mathcal{Z} \to \Delta^\mathcal{Y}$ parameterized by $\omega^h$, where $\Delta^\mathcal{Y}$ is the simplex over $\mathcal{Y}$. By giving a nonnegative and convex loss function $l : \Delta^\mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, the expected risk of this model parameterized by $\omega$ on domain $\mathcal{T}$ can be defined as follows, that is:

$$L_T(\omega) := \mathbb{E}_{x \sim \mathcal{D}} \left[l \left(h \left(f \left(x; \omega^f \right); \omega^h \right), c^*(x)\right)\right]. \quad (1)$$

B. Federated Learning

In FL, the optimization objective is represented by $\mathcal{L}$ for the global task $\mathcal{T}$ and by $\mathcal{L}_k$ for the local task $\mathcal{T}_k$. The aim of FL is to learn a global model, parameterized by $\omega$, that minimizes the risk on each client task. The global objective function is expressed as follows [1]:

$$\min_{\omega} \mathbb{E}_{\mathcal{T}_k \sim \mathcal{T}} \left[\mathcal{L}_k(\omega)\right] \quad (2)$$

where $\mathcal{T}_k = \langle \mathcal{D}_k, c^* \rangle$. Assuming there are total $K$ clients, denoted by $[K]$. In detail, for each client $k \in [K]$, its local task is $\mathcal{T}_k = \langle \mathcal{D}_k, c^* \rangle$, and we have

$$\mathcal{L}_k(\omega) := \mathbb{E}_{x \sim \mathcal{D}_k} \left[l \left(h \left(f \left(x; \omega^f \right); \omega^h \right), c^*(x)\right)\right] \quad (3)$$

where $\mathcal{T} = \{\mathcal{T}_k\}_{k=1}^K$ is the collection of client tasks $\mathcal{T}_k = \langle \mathcal{D}_k, c^* \rangle$ for each client $k \in [K]$ share the same mapping rules $c^*$.

| Symbol | Definition |
|-------|------------|
| $\beta$ | Global distribution that fits the difficulty scores of all clients |
| $n_k$ | Number of difficulty samples collected for each client $k$ |
| $N$ | Total number of difficulty samples from all clients |
| $N$ | Sorted list of difficulty samples |
| $T_k$ | Global threshold |
| $y(\cdot)$ | Approximate prior distribution |
| $Q^*$ | Conditional distribution with the ground-truth data and current difficulty |
| $y(\cdot)$ | Ground-truth prior |
| $y(\cdot, \beta)$ | Posterior distribution of target labels given a difficulty distribution $\beta$ |
| $\xi$ | Noise vector |
| $\lambda(\cdot)$ | Loss function for classification |
| $h(\cdot)$ | Predictor function |
| $f(\cdot)$ | Feature extraction function |
| $\mathcal{E}_k(\omega^k)$ | Empirical risk of the local model $k$ |
| $\delta_k$ | Local dataset of client $k$ |
| $\mathcal{T}_k$ | Global threshold |
| $T$ | Local maximum steps |
| $B$ | Batch size |
| $a, \gamma$ | Learning rate |
| $c_k$ | Label counter |
| $M_f$ | Set of selected clients |
| $M_h$ | Set of remaining clients |

Table I: Notation Table
and loss function \( l \). In practice, (2) can be empirically optimized by

\[
\min_{\omega} \frac{1}{K} \sum_{k=1}^{K} \hat{L}_k(\omega)
\]

(4)

where we have \( \hat{L}_k(\omega) \), which is the empirical risk over an observable dataset \( D_k \)

\[
\hat{L}_k(\omega) := \frac{1}{|D_k|} \sum_{x_i \in D_k} \left[ l \left( h \left( f \left( x_i; \omega_l^f \right); \omega_h^k \right), c^*(x_i) \right) \right].
\]

(5)

Thus, in this FL model, the global dataset \( \hat{D} \) is distributed to each of the local domains, thus we have \( \hat{D} = \bigcup \{ \hat{D}_k \}_{k=1}^{K} \).

C. Curriculum Learning

CL is a method that trains a model by assigning different weights to samples based on the difficulty of learning each sample. We introduce a CL loss function inspired by confidence-aware [36], which is simply integrated with the original task loss during training to monitor each sample’s loss and dynamically determine sample contributions by applying the core principle of CL. The confidence-aware loss function \( \ell^{cl}(\cdot) \) consists of a difficulty score loss function \( \ell^{cl}(\cdot) \) and a regularization term. The difficulty score follows the task core of CL and mutiply the confidence parameter in a form of the loss function to achieve the weight distribution from easy to difficult. Let \( x_i \) be the \( i \)th sample of dataset \( D_k \) in the \( k \)th client. The difficulty score is calculated as follows:

\[
\bar{l}^{cl} \left( h \left( f \left( x_i; \omega_l^f \right); \omega_h^k \right), c^*(x_i) \right) = \left[ l \left( h \left( f \left( x_i; \omega_l^f \right); \omega_h^k \right), c^*(x_i) \right) - \tau \right] \cdot \sigma_i
\]

(6)

where \( \sigma_i \) is a confidence parameter and \( \tau \) represents a loss-based threshold distinguishing easy and hard samples, we aim to avoid overfitting issues resulting from high model complexity. We add a regularization term \( \lambda (\log \sigma_i)^2 \) to (6) to minimize structural risk, where \( \lambda \) is the hyperparameter. Consequently, we can formulate the confidence-aware loss function as follows:

\[
\ell^{cl} \left( h \left( f \left( x_i; \omega_l^f \right); \omega_h^k \right), c^*(x_i) \right) = \bar{l}^{cl} \left( h \left( f \left( x_i; \omega_l^f \right); \omega_h^k \right), c^*(x_i) \right) + \lambda (\log \sigma_i)^2.
\]

(7)

By using \( \ell^{cl} \) in (7) to redefine local object \( L_k \) in (3) as

\[
\hat{L}_k^{cl}(\omega) := \mathbb{E}_{x \sim D_k} \left[ \ell^{cl} \left( h \left( f \left( x; \omega_l^f \right); \omega_h^k \right), c^*(x) \right) \right].
\]

(8)

A change of local goals also needs to redefine global goals, the redefined global object function in (2) as

\[
\min_{\omega} \mathbb{E}_{\hat{D} \in T} \left[ \hat{L}_k^{cl}(\omega) \right].
\]

(9)

Similarly, we use the confidence-aware loss \( \ell^{cl} \) instead of \( l \) in the empirical risk \( \hat{L}_k \) shown as (5). Then, the \( \hat{L}_k \) can be redefined by \( \hat{L}_k^{cl} \) as

\[
\hat{L}_k^{cl}(\omega) := \frac{1}{|D_k|} \sum_{x_i \in D_k} \ell^{cl} \left( h \left( f \left( x_i; \omega_l^f \right); \omega_h^k \right), c^*(x_i) \right).
\]

(10)

Back to the task in Section III-B, FL finally expresses a standard way of defining a global loss function in the context of optimization. The goal is to find the value of \( \omega \) that minimizes the average loss across all data points as

\[
\min_{\omega} \frac{1}{K} \sum_{k=1}^{K} \hat{L}_k^{cl}(\omega)
\]

(11)

where it redefines the global loss function in (4) by the sum over \( k \) in the expression.

D. Gaussian Mixture Model

The GMM is a probability model positing that all data points are generated by a finite number of unknown parameters, forming a mixture of Gaussian distributions. Theoretically, any continuous distribution can be approximated using a finite number of Gaussian distribution mixes. Each client \( k \) has \( |D_k| \) samples, and for each observation sample \( x_i \in D_k \), we can obtain its difficulty score \( \bar{l}^{cl}(h(f(x_i; \omega_l^f); \omega_h^k), c^*(x_i)) \), as shown in (6).

The distribution of difficulty scores over all samples in client \( k \) can be denoted by \( D_k^{cl} \). We assume that the local difficulty score can be expressed as the superposition of a series of Gaussian distributions.

For each client \( k \), we can estimate the distribution \( D_k^{cl} \) by using the difficulty score \( \bar{l}^{cl}(x_i) \) for each sample \( x_i \in D_k \) where we denote by \( \bar{l}^{cl}(x_i) = \bar{l}^{cl}(h(f(x_i; \omega_l^f); \omega_h^k), c^*(x_i)) \) for convenience. Given a fixed positive integer \( L \), the distribution of \( D_k^{cl} \) can be formulated as

\[
D_k^{cl}(x; \mu_k, \sigma_k^2) = \sum_{l=1}^{L} \alpha_{k,l} \cdot \phi \left( x; \mu_{k,l}, \sigma_{k,l}^2 \right)
\]

(12)

where \( \phi \left( x; \mu_{k,l}, \sigma_{k,l}^2 \right) \) is a Gaussian distribution with the mean \( \mu_{k,l} \) and variance \( \sigma_{k,l}^2 \), \( \alpha_{k,l} \) is the weight of the \( l \)th Gaussian distribution with \( \sum_{l=1}^{L} \alpha_{k,l} = 1 \), and we have \( \mu_k = \{ \mu_{k,1}, \ldots, \mu_{k,L} \} \) and \( \sigma_k^2 = \{ \sigma_{k,1}^2, \ldots, \sigma_{k,L}^2 \} \). Here, we denote by \( \alpha = \{ \alpha_{k,1}, \ldots, \alpha_{k,L} \} \), and then, the parameters of the distribution \( D_k^{cl} \) can be defined as \( \{ \mu_k, \sigma_k^2, \alpha_k \} \). In order to ensure the privacy of clients while enabling the central server to obtain the global difficulty score, the parameters \( \{ \mu_k, \sigma_k^2, \alpha_k \} \) should be transmitted to the server instead of the observable dataset \( \hat{D}_k \) from the client \( k \).

IV. METHODOLOGY

To address the issue of model drift in asynchronous FL approaches, we propose a multiphase CL approach called FedCL that deals with user heterogeneity synchronously. FedCL comprises two components: clients and the server.

For each client, the training process consists of four parts. First, each client employs the CL method to train its model and collect the local difficulty score, which determines the progress of local model convergence. Second, to ensure privacy protection, the local difficulty score is encapsulated using GMM. Third, to facilitate global data distribution synchronization and reduce model drift during local training, we introduce a generator model based on global data and difficulty distribution. The
generator model is trained only on the server and does not impose additional training costs on clients. Finally, the client uploads its label counter, task model parameters, and GMM parameters to the server.

For the server, it utilizes the approximate local difficulty score distribution of GMMs uploaded by clients to obtain the global difficulty score distribution, which updates the global information synchronization strategy. Then, each client’s local information synchronization strategy is updated based on the global information synchronization strategy, determining if further training is needed for each client. Next, the server performs standard model aggregation and trains the generator model. Finally, the aggregated models and local information synchronization strategy are distributed to the selected clients requiring further training.

The overview of the training procedure is shown in Fig. 2. We will explain our approach in detail from three perspectives: global difficulty distribution, generator learning, and synchronization strategy.

A. Global Difficulty Distribution

In order to protect the privacy of local difficulty scores during the uploading process from clients to the server, each client will upload its GMM parameters \((\mu_k, \sigma_k^2, \alpha_k)\) to the server. For the server, after receiving the GMM parameters from all clients, it wants to generate a global distribution \(\beta\) that can fit the difficulty scores of all clients. However, this distribution \(\beta\) is hard to obtain. To solve it, the server collects a certain number of samples \(N_k\) for each client \(k\) according to the GMM determined by its uploaded parameters \((\mu_k, \sigma_k^2, \alpha_k)\). Thus, we can get a collection of difficulty samples generated from GMMs of all clients. Its total number of \(N = \sum_{k=1}^{N_k} N_k\). Then, we sort these \(N\) difficulty samples from low to high to form a sorted list \(\mathcal{N}\). When we want to generate a difficulty sample according to this global distribution \(\beta\), we first uniformly generate an integer number \(r\) from 1 to \(N\), then return the \(r\)th element in the sorted list \(\mathcal{N}\). This process can be used to approximately generate difficulty samples from \(\beta\).

B. Generator Learning

The process of KD involves a student model learning from one or more teacher models. By leveraging KD, the student model can be compressed while simultaneously inheriting knowledge that has been distilled from the teacher(s). One popular approach to data-free KD is through the implementation of generative adversarial networks (GANs). Specifically, a generator is trained to produce imitated training data, which then enhances the original training dataset used during the KD process. The KD has been used in FL to address user heterogeneity [12], [37] by training each local model \(k\), parameterized by \(\omega_k = (\omega_k^f, \omega_k^h)\) as the teacher, then it is combined into the global (student) model, parameterized by \(\omega\). To improve the generalization performance, it leverages a proxy dataset \(\mathcal{D}_p\) to minimize the discrepancy, measured by using Kullback–Leibler (KL) divergence shown as follows:

\[
\min_{\omega} \mathbb{E}_{x \sim \mathcal{D}_p} \text{KL} \left[ \sigma \left( \frac{1}{K} \sum_{k=1}^{K} g(f(x; \omega_k^f); \omega_k^h) \right) || \sigma(g(f(x; \omega^f); \omega^h)) \right]
\]

(13)

where \(\sigma(\cdot)\) is a nonlinear function and \(g(\cdot)\) is the logits output of an predictor \(h\), namely, \(h(z; \omega^h) = \sigma(g(z; \omega^h))\). However, this requires a proxy dataset \(\mathcal{D}_p\), whose quality plays an important role in the final performance. Subsequently, we will demonstrate how KD can be made feasible for Synchronous FL in a data-free manner.

The first part of generator learning is to extract knowledge. To generate samples that align with the overall data distribution and satisfy difficulty distribution requirements, it is necessary to extract knowledge about both the global data distribution \(\mathcal{D}\) and the difficulty distribution \(\beta\). We propose learning the features of the conditional distribution \(Q^\star: \mathcal{Y} \to \mathcal{X}\), which is consistent with the ground-truth data distribution and current difficulty distribution

\[
Q^\star = \arg \max_{Q: \mathcal{Y} \to \mathcal{X}} \mathbb{E}_{y \sim p(y)} \mathbb{E}_{x \sim Q(x|y, \beta)} [\log p(y|x)]
\]

(14)

where \(p(y)\) is the ground-truth prior and \(p(y|x, \beta)\) is the posterior distribution of the target labels given a difficulty distribution \(\beta\). To make (14) solvable, we replace \(p(y)\) and \(p(y|x)\) with their empirical approximations. To approximate \(p(y)\), we have

\[
\hat{p}(y) \propto \sum_k \mathbb{E}_{x \sim \mathcal{D}_k} \left[ \mathbb{I}(c^\star(x) = y) \right]
\]

(15)

where \(\mathbb{I}(\cdot)\) is an indicator function and \(\hat{p}(y)\) can be obtained by requiring the training label counts from clients during the model uploading phase. Meanwhile, to model \(p(y|x)\), we can use the ensemble wisdom from user models as follows:

\[
\log \hat{p}(y \mid x) \propto \frac{1}{K} \sum_{k=1}^{K} \log p(y|x; \omega_k).
\]

(16)

Directly optimizing (14) by using the above approximations, it is prone to overload when \(\mathcal{X}\) is high dimensional and may also reveal information about the user’s data profile. A feasible method is to exploit an induced distribution \(G^\star: \mathcal{Y} \to \mathcal{Z}\) over the latent space, which is more compact than the original space and provides privacy protection. That is

\[
G^\star = \arg \max_{G: \mathcal{Y} \to \mathcal{Z}} \mathbb{E}_{y \sim \hat{p}(y)} \mathbb{E}_{z \sim G(z|y, \beta)} \sum_{k=1}^{K} \log p(y|z; \omega_k^h).
\]

(17)
Following the above explanation, we can learn a conditional generator \( G \) parameterized by \( \theta \) to optimize \( J(\theta) \), where we have \( \min_{\theta} J(\theta) = \)

\[
\mathbb{E}_{y \sim \hat{p}(y)} \mathbb{E}_{z \sim G_{\theta}(z|y, \beta)} \left[ r_{cl} \left( \sigma \left( \frac{1}{K} \sum_{k=1}^{K} g \left( z; \omega_k^h \right) \right), y \right) \right].
\]

(18)

The conditional sample \( z \) is based on the condition of \( y \) and \( \beta \), and optimizing (18) only requires access to the predictor modules \( \omega_k^h \) of user models. Specifically, to enable diversified outputs from \( G(\cdot|y, \beta) \), we introduce a noise vector \( \epsilon \sim \mathcal{N}(0, I) \) to the generator, which is resemblance to the reparameterization technique proposed by prior art [38], so that \( z \sim G_{\theta}(\cdot|y, \beta) \equiv G_{\theta}(y, \beta, \epsilon | \sim \mathcal{N}(0, I)) \).

Given arbitrary target labels \( y \), the proposed generator can yield feature representations \( z \sim G_{\theta}(\cdot|y, \beta) \) that induce ideal predictions from the ensemble of user models. In other words, the generator for global data and difficulty distribution can be seen as an induced image with a consensual distribution that is consistent with the user data and difficulty distribution from a global perspective. It generates samples that reflect both the global data distribution and the difficulty level of each user’s data, thereby achieving the goal of personalized learning in FL.

The second part of generator learning is to distill knowledge. The learned generator \( G_{\theta} \) is then sent to local clients so that each local model can sample from \( G_{\theta} \) to obtain an enhanced representation \( z \sim G_{\theta}(\cdot|y, \beta) \) over the feature space. Thus, the objective of the local model parameterized by \( \omega_k \) is changed to maximize the probability that it will produce an ideal prediction for the enhanced samples. The objective of each local client \( k \) can be defined as \( \min_{\omega_k} J(\omega_k) = \)

\[
\mathcal{L}_k^1(\omega_k) + \frac{1}{|D_k|} \sum_{x_i \in D_k} r_{cl} \left( h \left( f \left( x_i; \omega_k^f \right); \omega_k^h \right), y_i \right).
\]

(19)

where we have

\[
\mathcal{L}_k^1(\omega_k) = \frac{1}{|D_k|} \sum_{x_i \in D_k} r_{cl} \left( h \left( f \left( x_i; \omega_k^f \right); \omega_k^h \right), y_i \right).
\]

(20)

It is the empirical risk given a local dataset \( D_k \). To this end, we propose an approach to realize knowledge transfer without data by interactively learning a generator that mainly relies on global data and difficulty distribution. The generator can then be used to transmit global knowledge to local users.

In summary, the generator for global data and difficulty distribution is a useful approach for inducing a consensual distribution that aligns with the global data distribution and the difficulty level of each user’s data. It essentially generates synthetic samples that reflect both of these factors, thus enabling personalized learning in the context of FL. This is achieved by leveraging the difficulty distribution to control the sampling process and ensure that the generated samples are representative of each user’s data distribution. As a result, the generated samples can be used to augment the training data of individual users, thereby improving the accuracy and generalization of the federated model. In addition, the generator can be trained using a variety of techniques, including GANs, variational autoencoders (VAEs), or other generative models, depending on the specific requirements of the application. Overall, the Generator for global data and difficulty distribution is a powerful tool for addressing the challenges of FL and advancing the state-of-the-art in this area.

C. Synchronization Strategy

We propose a synchronization strategy that capitalizes on the global difficulty score to transform asynchronous FL into synchronous FL. The global information synchronization strategy seeks to harmonize the global learning state, ensuring that all clients learn from a congruous global perspective and update their local models in a coordinated manner, ultimately augmenting the overall performance of the FL algorithm.

Global information synchronization encompasses acquiring the global difficulty score and distributing local information synchronization to each user. For the global difficulty score, the server samples a number of difficulty samples based on the GMMs returned by clients to obtain an approximation of the global difficulty score distribution \( \beta \), as delineated in Section IV-A. For local information synchronization, we define a temperature \( T \in (0, 1) \) as the threshold, signifying the state of global synchronization. We then define a decreasing sorted list \( S = T_1, T_2, \ldots, T_Z \) as all training states, comprising a sequence of thresholds such that \( T_1 \geq T_2 \geq \cdots \geq T_Z \). As expounded in Section IV-A, the server can acquire a sorted list \( \mathcal{N} \) by sampling according to all local GMMs, mirroring the global difficulty distribution \( \beta \). Given a threshold \( T_z \in S \), the server returns the \( T_z \cdot N \)th element (difficulty sample) in \( \mathcal{N} \), denoted as \( \beta_{T_z} \) for convenience. Predicated on the \( (\mu_k, \sigma_k^2, \alpha_k) \) returned by client \( k \), the server reassembles a set of difficulty score samples \( \mathcal{N}_k \), with \( |\mathcal{N}_k| = N_k \). We then compare the samples in \( \mathcal{N}_k \) with the global \( \beta T_z \). If the number of samples in \( x \in \mathcal{N}_k \) is greater than \( v \cdot N_k \), where \( v \) is a temperature coefficient, it suggests that the majority of samples in client \( k \) satisfy the current global state prerequisites. The server then issues a freeze instruction to the corresponding client, delineating a set \( M_f \) consisting of all frozen clients and a set \( M_i \) encompassing all unfrozen clients. Clients in \( M_f \) necessitate further training. When all local models are frozen, it denotes the completion of learning under the current threshold \( T_z \), progressing to the next threshold \( T_z + 1 \), and unfreezing all clients at that point.

Therefore, the server aggregates an approximate unbiased model, as shown in Fig. 2. For each interaction between the server and clients, the server can determine the current frozen set \( M_f \) and unfrozen set \( M_i \). Then, the global parameters should be aggregated as

\[
\omega = \frac{1}{K} \left[ \sum_{k \in M_f} \omega_k + \sum_{k \in M_i} \omega_k \right]
\]

(21)

where the server applies the parameter averaging technique to update global parameters by averaging the parameters of both \( M_f \) and \( M_i \). This strategy aims to solve the problem of model drift caused by overfitting personalized user features. Besides, neglecting the aggregation of frozen models can lead to insufficient representation of general features, resulting in
Algorithm 1: FedCL.

Require: Tasks \( \{T_k\}_{k=1}^K \) and Difficulty score \( D^{cl} \);
Global difficulty distribution \( D^{cl} \) and local difficulty distribution \( D^{cl}_k \sim (\mu_k, \sigma^2_k, \alpha_k) \);
Global parameter \( \omega \) and Local parameters \( \{\omega_k\}_{k=1}^K \);
Generator parameter \( \theta \);
\( \hat{p}(y) \) and global difficulty score distribution \( \beta \) are uniformly initialized;
Global threshold \( \mathcal{S} = \{T_z\}_{z=1}^Z \);
Local maximum steps \( T \) and batch size \( B \);
Learning rate \( \gamma \) and \( \alpha \);

1: for each training state \( T_z \in \mathcal{S} \) do
2: \( M_I \leftarrow 0; \ M_I \leftarrow [K] \); 3: while \( M_I \neq \emptyset \) do
4: Send global \( \omega, \theta, \hat{p}(y), \beta \) to clients in \( M_I \); 5: for each client \( k \in M_I \) in parallel do
6: \( \omega_k \leftarrow \omega \); 7: for \( t = 1 \) to \( T \) do
8: \( \{x, y\}_{i=1}^B \sim T_k \); 9: \( \{z_i \sim G_\theta (\hat{y}_i, \beta), \hat{y}_i \sim \hat{p}(y)\}_{i=1}^B \); 10: Update label counter \( c_k \); 11: \( \omega_k \leftarrow \omega_k - \gamma \nabla \omega_k J(\omega_k) \) in Eqn. (19); 12: end for
13: \( D^{cl}_k \leftarrow (\mu_k, \sigma^2_k, \alpha_k) \); 14: Sends \( \omega_k, D^{cl}_k \), and \( c_k \) to the server; 15: end for
16: Server updates \( \omega \leftarrow \frac{1}{M_I} \sum_{k \in [K]} w_k \), and \( \hat{p}(y) \) based on \( \{c_k\}_{k \in [K]} \); 17: \( \theta \leftarrow \theta - \alpha \nabla \theta J(\theta) \) as in Eqn. (18); 18: \( \beta \leftarrow \{D^{cl}_k\}_{k=1}^K \); 19: Generate a difficulty sample set \( N \) based on \( \beta \) and determine \( \beta_{T_z} \); 20: for each client \( k \in M_I \) do
21: Collect a set of difficulty samples \( N_k \) on \( D^{cl}_k \); 22: if \( \{x \in N_k | x \leq \beta_{T_z} \} \geq N_k \) then
23: \( M_I \leftarrow M_I \cup \{k\} \); 24: end if
25: end for
26: \( M_I \leftarrow [K] \setminus M_I \); 27: end while
28: end for

The aggregated model \( \omega \), parameters of the generator \( \theta \), and \( \hat{p}(y) \), and global difficulty score distribution \( \beta \) to the selected client in \( M_I \). Each selected client \( k \in M_I \) trains its local model based on its original data, using the global learning state \( \beta \) and generated data \( G_\theta (\hat{y}_i, \beta) \) for global information synchronization strategy. The difficulty level for each training state \( \beta \) is determined by the main server. Then, it sends its updated weight \( \omega_k \), parameters of difficulty score distribution \( D^{cl}_k \), and local label counter \( c_k \) to the server. This process is executed on the client side, which is shown in lines 5–15 of Algorithm 1. After the server receives the feedback from clients, it averages the parameters of all local models to update the global model \( \omega \). The parameters of the generator \( \theta \) are also updated based on the updated global model. Finally, the server estimates the global difficulty score distribution \( \beta \) and determines \( \beta_{T_z} \). Then, it updates \( M_I \) and \( M_I \) based on \( \beta_{T_z} \). This process is executed on the server side, which is shown in lines 16–26 of Algorithm 1. The process continues until all clients have been frozen, which implies that all local models are under the same learning state, at which point we proceed to the next training state.

V. EXPERIMENTS

In this section, we empirically verify the effectiveness of FedCL. All experiments are conducted on a machine with an Intel Xeon CPU (4 GHz, 96 cores), 376 GB of DDR4 RAM, Nvidia Tesla V100-SXM2 with 32 GB HBM2 memory, and CUDA 11.6, running on Ubuntu 20.04.3. We summarize the setup of our experiment in Section V-A. Initially, FedCL is compared with six state-of-the-art approaches, including FedAvg [1], FedEnsemble [13], FedGen [16], FedProx [12], FedDistill [15], and FedDistill+[16]. Moreover, all experiments are conducted on three datasets, including MNIST [39], CELEBA [40], and EMNIST [41], with different setting parameters. In Section V-B, we present the visualization of measurement in the training process, the experiment results, and their analysis. At the end of this section, ablation studies are conducted to verify the robustness of our algorithm. The source code can be found at https://github.com/MingjieWang0606/FedCL_Public/.

A. Experimental Setup

1) Datasets: We use real-world datasets to comprehensively evaluate the performance of the proposed model. The datasets include MNIST [39], consisting of a training set of 60000 examples and a test set of 10000 examples; CELEBA [40], comprising 202599 face images with 40 attribute annotations; and EMNIST [41], containing a training set of 697932 examples and a test set of 116323 examples. Following the suggestion of Zhu et al. [16], MNIST and EMNIST are both used for character and digit image classifications. On the other hand, CELEBA consists of celebrity faces, which is suitable for binary-classification tasks, such as predicting whether the celebrities in photos are smiling.

For heterogeneity exploration, we utilize the Dirichlet distribution \( \text{Dir}(\alpha) \) to model Non-IID data distributions, where smaller \( \alpha \) indicates higher data heterogeneity, making the distribution of \( p_k(y) \) more biased for a user \( k \) of MNIST and EMNIST.
Fig. 3. Performance curves on MNIST dataset where a smaller $\alpha$ means larger data heterogeneity. (a) $\alpha = 0.05$. (b) $\alpha = 0.1$. (c) $\alpha = 1$. (d) $\alpha = 10$.

Fig. 4. Performance Curves on CELEBA dataset. (a) $r = 5/10$. (b) $r = 5/25$. (c) $r = 10/25$.

Based on prior works [42]. In other words, a larger $\alpha$ results in greater differences in the distribution of labels for each client. Fig. 3 shows the effect of adopting different $\alpha$ and different training baselines on the statistical heterogeneity for the MNIST dataset. For CELEBA, whose data are originally Non-IID, $r$ denotes the ratio of active users in total users, and $T$ is the local training step.

2) Baselines: We demonstrate the efficiency and effectiveness of FedCL by comparing it with other baselines. First, FedAvg [1] is an aggregated model that iteratively averages the parameters of distributed local user models. Specifically, FedAvg randomly selects some users for sampling, replacing the nonsampled users with the current global model by averaging the gradient updates of these users to form a global update. FedEnsemble [13] is an extension of FedAvg that ensembles all user models’ prediction output by using random permutations to update a group of models and then obtaining predictions through model ensembling by averaging. Moreover, FedGen [16] addresses heterogeneity in FL with KD, and the distilled knowledge is directly adjusted to users’ learning through distribution matching in the potential space. FedProx [12] uses a proximal term in the objective function for stabilization, which improves robustness by regularizing local model training. Finally, FedDistill [15] is a data-free KD approach that shares the average of logit-vectors of labels among users and benefits from the robustness of the Bayesian model ensemble in aggregating users’ predictions. However, the model parameters and its performance drop compared with other baselines. For comparison, FedDistill$^+$ [16] is used to share both network parameters and the logit-vectors of labels among users on average.

In our experiments, we evaluate the performance of the proposed FedCL and the aforementioned baselines using real-world datasets, including MNIST [39], CELEBA [40], and EMNIST [41]. MNIST and EMNIST are used for character and digit image classifications, while CELEBA consists of celebrity faces and is used for a binary-classification task of predicting whether celebrities in photos are smiling. Besides, to ensure the validity and robustness of our experimental results, we employed ten distinct random seeds in our study. We report the mean outcome of these ten trials, along with their associated variance, as our final result.

3) Training Settings: We train and test all datasets using the approaches with the same setting parameters. Hyperparameter configurations of the experiment are shown in Table II. For shared parameters of all models, 200 global communication rounds are performed except for CELEBA, and 20 user models are assigned, with the active-user ratio $r$ being 50%.

| Approach | Hyperparameter | Value |
|----------|----------------|-------|
| Shared Parameters | Learning Rate | 0.01 |
| | Optimizer | SGD |
| | Local Update Steps | 20 |
| | Batch Size | 32 |
| | CELEBA Communication rounds | 100 |
| | Other’s Communication rounds | 200 |
| | # of total users | 20 |
| | # of active users | 10 |
| FEDGEN & FEDCL | Generator Optimizer | adam |
| Shared Parameters | Generator learning rate | $10^{-4}$ |
| | Generator inference size | 128 |
| | User distillation batch size | 32 |
| FEDCL. CL Settings | Curriculum Completed Threshold | 80% |
| | Learning levels of CL | [0.3, 0.6, 0.9] |
| FEDDSTILL & FEDDSTILL$^+$ | Distillation Coefficient | 0.1 |
| FEDPROX | Proximal Coefficient | 0.1 |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
batch size of 32 and a local updating step $T$ of 20, we adopt the learning rate $r = 0.01$ and use stochastic gradient descent (SGD) as the optimizer. For FedGen and FedCL, the optimizer for their generator is the Adam approach, and the learning rate is $10^{-4}$. Since FedDistill and FedDistill$^+$ both share the data of the logit-vector outputs of the user models for KD, the distillation coefficient is 0.1. FedProx has a proximal term with a coefficient of 0.1 in the loss function. All of the above are implemented using PyTorch [43].

**B. Experimental Results**

1) **Accuracy and Analysis:** The data generator is a multilayer perceptron that takes a noise vector $\epsilon$ and a one-hot vector $y$ as inputs and generates a feature representation as output. The deep learning model is used as a classifier, with the last layer acting as the predictor [1]. During training, half of the training dataset is distributed to users, while all testing datasets are reserved for performance evaluation. We also consider diversity loss to increase the diversity of the generator output. The learning curves for training on the MNIST, CELEBA, and EMNIST datasets are shown in Figs. 3–5.

Table III lists the top-1 test accuracy of all approaches with the default setting on MNIST, EMNIST, and CELEBA datasets. In these datasets, FedCL is the best performer and at least 0.2% above the other baselines. The accuracy of FedGen is lower than that of FedCL, but it still yields better results than FedAvg.

2) **Data Heterogeneity:** FedCL and FedGen are two approaches that demonstrate robustness in the face of different levels of user heterogeneity while consistently performing well, as seen in Fig. 6. Specifically, Fig. 3 shows that as the data distribution becomes more heterogeneous, the gain of FedCL is more notable with a smaller $\alpha$. FedCL performs the best because it takes into account the model-aware difficulty of the data, which helps to aggregate the models before model drifts occur. A key strength of both FedCL and FedGen is their ability to induce knowledge distilled to current users, reducing the difference in latent distributions across users. In contrast, FedAvg and FedProx cannot obtain shared information, and the curve of FedDistill drops dramatically due to KD. These results highlight the importance of parameter sharing, which is vulnerable to data heterogeneities. Furthermore, the generator of FedCL is superior to that of FedGen, which may explain why FedEnsemble, which only ensembles predictions from user models, performs slightly lower than FedGen and FedCL. Overall, these findings demonstrate the effectiveness and efficiency of FedCL in addressing the challenges of user heterogeneity in FL.

3) **Learning Efficiency:** Regarding learning efficiency, FedCL has the fastest learning speed to reach optimized performance and outperforms other baselines. FedGen performs lower than FedCL, as each local user directly benefits from the learned knowledge. As given in Table IV, compared with FedGen, FedCL reduces 4.3 rounds on CELEBA and 2.5 rounds on MNIST on average. The average reduction is 11.5 rounds on EMNIST with $T = 20$ and 28.25 rounds on EMNIST with $T = 40$. Our model demonstrates excellent performance in the vast majority of cases and is robust against different levels of communication delays. However, the accuracy advantage of FedCL over FedGen varies among datasets, depending on the dataset distribution. In the case of
TABLE III
PERFORMANCE OVERVIEW UNDER DIFFERENT DATA HETEROGENEITY SETTINGs

| Dataset | Setting | PEDAVG | PEDPROX | PEDHOMOBBLE | PEDDISTILL | PEDGEN | PRDCL |
|---------|---------|--------|---------|-------------|------------|--------|-------|
| CELEBA  | r = 5/10| 86.52±0.43 | 87.34±0.43 | 87.94±0.51 | 78.73±1.21 | 89.45±0.58 | 90.65±0.42 |
|         | r = 5/25| 89.12±0.24 | 88.69±0.22 | 90.13±0.28 | 75.11±1.62 | 89.26±0.42 | 90.83±0.61 |
|         | r = 10/25| 89.07±0.18 | 88.98±0.37 | 90.04±0.21 | 75.91±1.19 | 90.27±0.44 | 91.58±0.64 |
| MNIST   | α = 0.05| 85.62±1.56 | 86.73±1.97 | 89.05±1.04 | 71.34±1.46 | 90.04±0.68 | 91.47±0.81 |
|         | α = 0.1 | 91.13±0.84 | 90.69±0.99 | 91.60±0.56 | 58.03±0.78 | 95.65±0.05 | 96.03±0.12 |
|         | α = 1   | 94.73±0.49 | 94.42±0.37 | 94.77±0.49 | 86.52±1.56 | 97.36±0.14 | 97.56±0.02 |
|         | α = 10  | 95.03±0.51 | 94.81±0.50 | 94.96±0.52 | 86.73±1.97 | 97.52±0.04 | 97.62±0.03 |
| EMNIST  | T = 20  | α = 0.05 | 63.00±1.06 | 62.75±0.44 | 65.52±0.87 | 47.13±0.14 | 71.98±0.10 | 73.65±1.23 |
|         |         | α = 0.1 | 73.65±1.23 | 68.55±0.87 | 69.72±0.72 | 45.84±0.23 | 76.05±0.22 | 78.06±1.25 |
|         |         | α = 1   | 78.08±0.41 | 77.28±0.54 | 78.34±0.41 | 56.12±0.36 | 82.81±0.05 | 84.21±0.15 |
|         |         | α = 10  | 77.81±0.11 | 77.03±0.04 | 77.83±0.03 | 46.34±0.11 | 82.71±0.09 | 83.88±0.29 |
| EMNIST  | T = 40  | α = 0.05| 68.72±0.60 | 66.85±0.57 | 69.57±0.63 | 49.32±0.13 | 72.29±1.27 | 74.91±2.55 |
|         |         | α = 0.1| 75.18±0.54 | 74.05±0.82 | 75.92±0.67 | 48.23±0.42 | 77.43±0.54 | 79.25±1.53 |
|         |         | α = 1   | 83.79±0.30 | 83.22±0.22 | 82.91±0.15 | 56.83±0.53 | 84.01±0.14 | 84.77±0.18 |
|         |         | α = 10  | 83.47±0.20 | 82.91±0.15 | 83.62±0.15 | 48.67±0.53 | 84.24±0.27 | 84.77±0.18 |

The bold sections represent FedCL results.

The results are presented in Table VII. These experiments were carried out on the EMNIST dataset with $T = 20$ and $\alpha = 0.1$. In the first set of hyperparameter experiments, we varied $\lambda$, the hyperparameter of the regularization term $\lambda (\log \sigma_i)^2$ in $f^{cl}$, while keeping $\tau = 10$ fixed. Our results indicated that the best performance was achieved with $\lambda = 0.5$. In the second set of hyperparameter experiments, we varied $\tau$, the loss-based threshold that distinguishes easy and hard samples, while keeping $\lambda = 0.5$ fixed. Our results revealed that the best performance was achieved with $\tau = 10$.

In the third and fourth experiments, we varied the array of training states $S$ and the temperature coefficient $v$ of $N_k$ to represent the number of samples. These experiments were conducted with the best $\tau$ and $\lambda$ values obtained from the first two groups of experiments. Our results demonstrated that the best performance was achieved when $S = 0.3, 0.5, 0.7$ and $v = 0.6$. Overall, our hyperparameter experiments showed that the proposed method is insensitive to the choice of hyperparameters.

4) Hyperparameter Experiments: We conducted hyperparameter experiments to investigate the impact of different hyperparameters on the performance of our proposed method. The results presented in Table VII. These experiments were carried out on the EMNIST dataset with $T = 20$ and $\alpha = 0.1$. In the first set of hyperparameter experiments, we varied $\lambda$, the hyperparameter of the regularization term $\lambda (\log \sigma_i)^2$ in $f^{cl}$, while keeping $\tau = 10$ fixed. Our results indicated that the best performance was achieved with $\lambda = 0.5$. In the second set of hyperparameter experiments, we varied $\tau$, the loss-based threshold that distinguishes easy and hard samples, while keeping $\lambda = 0.5$ fixed. Our results revealed that the best performance was achieved with $\tau = 10$.

In the third and fourth experiments, we varied the array of training states $S$ and the temperature coefficient $v$ of $N_k$ to represent the number of samples. These experiments were conducted with the best $\tau$ and $\lambda$ values obtained from the first two groups of experiments. Our results demonstrated that the best performance was achieved when $S = 0.3, 0.5, 0.7$ and $v = 0.6$. Overall, our hyperparameter experiments showed that the proposed method is insensitive to the choice of hyperparameters.

5) Ablation Experiments: Extended analysis indicated that FedCL is robust across different thresholds for curriculum satisfaction. As given in Table VI, our default threshold is $80\%$. The CELEBA dataset experiment with $r = 5/25$, we observed a decreased performance of FedCL compared with FedGen. We believe this could be due to a number of reasons. CELEBA is a particularly complex and high-dimensional dataset. When the ratio $r$ is set to $5/25$, it indicates a relatively higher rate of clients dropping out. FedCL, with its focus on CL, might face difficulties in efficiently learning from this relatively small and possibly unrepresentative subset of clients. On the other hand, FedGen, which is more robust to client dropout, could still maintain a relatively higher efficiency due to its generalized FL approach that does not rely on the CL strategy. For other approaches, their trends consistently underperform FedCL by a significant margin, which becomes more pronounced with high data heterogeneity. Therefore, FedCL not only achieves fast convergence but also has less communication workload when given a wise parameter sharing strategy. Our approach can directly benefit every local user with learned knowledge, being more explicit and consistent.
TABLE IV
NUMBER OF ROUNDS OF DIFFERENT APPROACHES TO ACHIEVE THE SAME ACCURACY AS RUNNING FedAvg FOR 200 ROUNDS (MNIST AND EMNIST) OR 100 ROUNDS (CELEBA)

| Dataset | Approach | FedAvg | FedProx | FedEnsemble | FedDistill | FedGen | FedCL |
|---------|----------|--------|---------|-------------|------------|--------|-------|
| CELEBA  | $\tau = 5/10$ | 100    | 23      | 17          | \          | 20     | 18    |
|         | $\tau = 5/25$ | 100    | 58      | 16          | \          | 15     | 17    |
|         | $\tau = 10/25$ | 100    | 37      | 22          | \          | 19     | 16    |
| MNIST   | $\alpha = 0.05$ | 200    | 58      | 37          | \          | 46     | 41    |
|         | $\alpha = 0.1$  | 200    | 73      | 66          | \          | 24     | 22    |
|         | $\alpha = 1$    | 200    | \       | 194         | \          | 18     | 16    |
|         | $\alpha = 10$   | 200    | \       | \           | \          | 9      | 8     |
| EMNIST  | $T = 20$ | \   | \       | \           | \          | \      | \     |
|         | $\alpha = 0.05$ | 200    | 173     | 143         | \          | 58     | 43    |
|         | $\alpha = 0.1$  | 200    | \       | 162         | \          | 64     | 50    |
|         | $\alpha = 1$    | 200    | \       | 187         | \          | 58     | 46    |
|         | $\alpha = 10$   | 200    | \       | \           | \          | 48     | 43    |
| EMNIST  | $T = 40$ | \   | \       | \           | \          | \      | \     |
|         | $\alpha = 0.05$ | 200    | \       | 162         | \          | 79     | 65    |
|         | $\alpha = 0.1$  | 200    | 197     | 126         | \          | 76     | 63    |
|         | $\alpha = 1$    | 200    | \       | 177         | \          | 154    | 82    |
|         | $\alpha = 10$   | 200    | \       | 179         | \          | 81     | 67    |

The bold sections represent FedCL results.

TABLE V
PERFORMANCE OVERVIEW UNDER DIFFERENT THRESHOLD FOR MOVING TO NEXT DIFFICULTY LEVEL

| Performance w.r.t. different P-Threshold on EMNIST $T = 20$ $\alpha = 0.1$ |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| P-Value         | 60              | 70              | 80              | 90              |
| Accuracy        | FedCL=78.06±1.25|
| PREDCL          | 78.02±0.75      | 77.68±0.83      | 78.06±1.25      | 77.35±0.61      |

TABLE VI
PERFORMANCE OVERVIEW UNDER DIFFERENT DIFFICULTY LEVEL DIVISION

| Performance w.r.t. different Stage-num on EMNIST $T = 20$ $\alpha = 0.1$ |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| Stage-num       | 2               | 3               | 4               | 5               |
| Accuracy        | FedCL=78.06±1.25|
| PREDCL          | 78.05±1.22      | 78.01±0.73      | 78.06±1.25      | 77.66±0.93      |

TABLE VII
PERFORMANCE OVERVIEW UNDER DIFFERENT HYPERPARAMETERS

| Performance Overview under different Hyperparameters on EMNIST $T = 20$ $\alpha = 0.1$ |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| $\lambda$       | 0.2             | 0.5             | 0.8             | 1.0             |
| $\tau = 10$     | 77.29±0.43      | 78.06±1.25      | 77.33±0.23      | 78.01±0.81      |
| $\tau = 0.5$    | 77.56±1.02      | 78.06±1.25      | 78.02±0.52      | 78.84±0.64      |
| $\tau = 0.5$, $\tau = 10$ | 77.56±0.15   | 77.86±0.23      | 78.06±1.12      | 77.31±0.54      |
| $\nu$           | 0.5             | 0.6             | 0.7             | 0.8             |
| $\lambda = 0.5$, $\tau = 10$ | 76.01±0.52   | 77.31±0.71      | 78.06±1.25      | 77.51±0.82      |

which means when over 80% of clients have finished over 80% of the corresponding curriculum data, the training process will move to the next difficulty level. In Table V, the performance of FedCL was measured under different thresholds of 60, 70, 80, and 90, and it outperformed FedGen for all four situations by at least 1.5% accuracy.

Furthermore, we explored different numbers of difficulty level settings in Table VI. In Table V, there are three steps set as default. For example, the difficulty level will move from 0.3 to 0.6 and 0.9. For other numbers of steps, it can be deduced by analogy. For step numbers shown in Table V, FedCL outperformed FedGen overall by at least 1.5% accuracy. Therefore, compared with FedGen, FedCL has a stronger generalization ability.

To further investigate the role of CL loss within the synchronous framework, we experimented with utilizing CL loss exclusively for synchronization without employing it for the model of user updates, namely, FedCL (w/o CL). The results are presented in Table VIII. We observed that CL loss does indeed have a tangible impact on the model’s performance. However, it also substantiates that the primary performance improvement in FedCL is derived from our synchronization framework.
VI. CONCLUSION

In this article, we introduce FedCL, a practical and effective approach for enhancing the performance of federated deep learning models on Non-IID datasets. The proposed approach employs synchronous FL, a new learning concept that leverages the learning progress of the model to address heterogeneity issues and mitigate model drift. Through extensive experiments on four benchmarks, we demonstrate that FedCL outperforms state-of-the-art methods, achieving better generalization performance with fewer communication rounds. Besides, in the real-world setting, leveraging its synchronous attributes, FedCL harbors the capacity to revolutionize sectors, such as healthcare, IoT, and finance, by facilitating the extraction of robust, privacy-preserving insights from heterogeneous, distributed data points. We propose that future research could focus on the tradeoff analysis between privacy and utility based on synchronous FL and the integration of the learning progress of the model for guaranteed data privacy. We conclude that FedCL is a promising FL method that can effectively tackle challenges in distributed learning scenarios and improve the performance of FL systems in real-world applications.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proc. Artif. Intell. Statist., 2017, pp. 1273–1282.
[2] W. Zhang, X. Li, H. Ma, Z. Luo, and X. Li, “Federated learning for machinery fault diagnosis with dynamic validation and self-supervision,” Knowl. Based Syst., vol. 213, 2021, Art. no. 106679.
[3] W. Y. B. Lim et al., “Towards federated learning in 5G-enabled internet of vehicles: A multi-dimensional contract-matching approach,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 8, pp. 5140–5154, Aug. 2021.
[4] J. Li et al., “A federated learning based privacy-preserving smart healthcare system,” IEEE Trans. Ind. Inform., vol. 18, no. 3, pp. 2021–2031, Mar. 2022.
[5] A. Khaled, K. Mishchenko, and P. Richtarik, “Tighter theory for local SGD on identical and heterogeneous data,” in Proc. Int. Conf. Artif. Intell. Statist., 2020, pp. 4519–4529.
[6] G. Lee, Y. Shin, M. Jeong, and S.-Y. Yun, “Preservation of the global knowledge by not true soft knowledge distillation in federated learning,” 2021, arXiv:2106.03997.
[7] T.-M. H. Hsu, H. Qi, and M. Brown, “Measuring the effects of non-identical data distribution for federated visual classification,” 2019, arXiv:1909.06335.
[8] L. Gao et al., “Feddc: Federated learning with non-IID data via local drift decoupling and correction,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn., 2022, pp. 10112–10121.
[9] L. Yang, J. Huang, W. Lin, and J. Cao, “Personalized federated learning on non-IID data via group-based meta-learning,” ACM Trans. Knowl. Discov. Data, vol. 17, no. 4, 2022, Art. no. 49.
[10] Y. Deng et al., “Improving federated learning with quality-aware user incentive and auto-weighted model aggregation,” IEEE Trans. Parallel Distrib. Syst., vol. 33, no. 12, pp. 4515–4529, Dec. 2022.
[11] H. Wu and P. Wang, “Fast-convergent federated learning with adaptive weighting,” IEEE Trans. Cogn. Commun. Netw., vol. 7, no. 4, pp. 1078–1088, Dec. 2021.
[12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” IEEE signal Process. Mag., vol. 37, no. 3, pp. 50–60, May 2020.
[13] N. Shi, F. Lai, R. A. Kontar, and M. Chowdhury, “Fed-ensemble: Improving generalization through model ensembling in federated learning,” 2021, arXiv:2107.10663.
[14] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, “Fine-tuning global model via data-free knowledge distillation for non-IID federated learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn., 2022, pp. 10164–10173.
[41] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, “EMNIST: Extending MNIST to handwritten letters,” in Proc. IEEE Int. Joint. Conf. Neural Netw., 2017, pp. 2921–2926.

[42] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, “Ensemble distillation for robust model fusion in federated learning,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 2351–2363.

[43] A. Paszke et al., “Pytorch: An imperative style, high-performance deep learning library,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 8026–8037.

Mingjie Wang received the B.S. degree from Department of Computer Science and Technology, Longdong University, China, in 2021. He is currently working toward the M.Phil. degree with the Department of Data Science and Technology, BNU-HKBU United International College (UIC), Zhuhai, China. His current research interests include time-series, NLP, machine learning, and deep learning.

Jianxiong Guo (Member, IEEE) received the B.E. degree from the School of Chemistry and Chemical Engineering, South China University of Technology, Guangzhou, China, in 2015, and the Ph.D. degree from the Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA, in 2021. He is currently an Assistant Professor with the Advanced Institute of Natural Sciences, Beijing Normal University, Beijing, China, and also with the Guangdong Key Lab of AI and Multi-Modal Data Processing, BNU-HKBU United International College, Zhuhai, China. He has authored or coauthored more than 40 peer-reviewed papers. His research interests include social networks, wireless sensor networks, combinatorial optimization, and machine learning.

Dr. Guo is a Member of ACM/CCF. He has been a Reviewer for many famous international journals/conferences.

Weijia Jia (Fellow, IEEE) received the B.Sc. and M.Sc. degrees from Center South University, Changsha, China, in 1982 and 1984, respectively, and the Master of Applied Science and Ph.D. degrees from Polytechnic Faculty of Mons, Mons, Belgium, in 1992 and 1993, respectively, all in computer science. He is currently a Chair Professor and the Director of BNU-UIC Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai, China, and the VP for Research of BNU-HKBU United International College (UIC) and has been the Zhiyuan Chair Professor of Shanghai Jiao Tong University, Shanghai, China. He was the Chair Professor and the Deputy Director of State Key Laboratory of Internet of Things for Smart City, University of Macau, Macau, China. From 1993 to 1995, he was with German National Research Center for Information Science (GMD), Bonn (St. Augustine), Germany, as a Research Fellow. From 1995 to 2013, he was with the City University of Hong Kong, Hong Kong, as a Professor. His contributions have been recognized as optimal network routing and deployment, anycast and QoS routing, sensors networking, AI (knowledge relation extractions, NLP, etc.), and edge computing. He has authored or coauthored more than 600 publications in the prestige international journals and conferences and research books and book chapters.

Dr. Jia was the recipient of the best product awards from the International Science and Tech. Expo (Shenzhen) in 2011–2012 and the First Prize of Scientific Research Awards from the Ministry of Education of China in 2017 (list 2). He was the Area Editor for various prestige international journals, Chair and PC Members Keynote Speaker for many top international conferences. He is a Distinguished Member of CCF.