FedDCS: Federated Learning Framework based on Dynamic Client Selection

Shutong Zou*, Mingjun Xiao*✉,Yin Xu*, Baoyi An*, Jun Zheng*
*Scoool of Computer Science and Technology / Suzhou Institute for Advanced Study
University of Science and Technology of China Hefei, China
✉Correspondence to: xiaomj@ustc.edu.cn

***Abstract*—Federated Learning, through which a server can coordinate a crowd of clients to accomplish a machine learning task, has been recognized as a promising paradigm for privacy preserving decentralized learning in recent years. Most federated learning researches are on the basis of IID data, and more and more researches focus on the Non-IID data problem. However, they have not considered the process of data acquisition. In fact, in real applications, the training data are typically collected by clients in real scenarios. In this paper, we propose a Federated Learning framework based on Dynamic Client Selection, called FedDCS, to deal with the real scene Non-IID data machine learning problem in federated learning. The objective of FedDCS is to utilize a parameter estimation algorithm to select the optimal clients to join the collaboration and finally acquire a better global machine learning model. Moreover, we introduce Intel SGX TEE to keep client data from privacy leakage. We theoretically prove the convergence of the estimation algorithm. Our extensive experiments on several real-world data sets demonstrate the superior performance of FedDCS.**

***Index Terms*—federated learning, Gaussian Mixture Model, machine learning.**

I. Introduction

Federated Learning (FL) is a newly-emerging distributed machine learning technique which allows multiple decentralized edge devices to train a global model without exchanging data [1]. Compared to the traditional machine learning methods, FL expands the training machine learning model from on a single data set to on multiple data sets distributed among multi-parties. Since FL can achieve a higher accuracy, and meanwhile ensure that the data privacy will not be revealed to others, it attracts much research attention.

A typical FL framework consists of a server node and a set of clients. Each client will train a machine learning local model based on its local data set. Then, the server aggerates the different local models from the clients to acquire a global model which is capable to deal with general data and has better performance on them. Since each client possesses a local data set with diverse distribution, it will obtain an independent local mode, so that the performance of global machine learning model may be significantly affected by the heterogeneous local models.

Traditional FL methods mainly focus on the model training on Independently and Identically Distributed (IID) data sets. For example, Mcmahan *et al.* firstly proposes the concept of Federated Learning and designs a framework FedAvg [1], which can achieve a high accuracy for jointly training a global model on IID data sets. However, in real applications, most of the data sets are Non-IID. This is because data sets are generally held by different clients, and each individual data set has different size, different classes and different distributions. Actually, the convergence speed and global machine learning accuracy of traditional FL frameworks including FedAvg may drastically drop when training models on Non-IID data sets.

In recent years, some FL methods have been proposed to deal with the Non-IID problem. Mcmahan *et al.* propose a FL algorithm, called FedSGD [1]. By changing the number of batch size and local iterations, it performs much better than FedAvg when training models on Non-IID data sets [2]. Another algorithm HybridFL is proposed to improve performance on classification tasks, and it conducts a cooperative mechanism to mitigate the performance degradation on Non-IID data sets. Nevertheless, those researches have not considered the process of data collection, which makes the learning effect unsatisfactory.

In this paper, we focus on how to design a framework that can adapt to the real-scene-based Non-IID data. We consider such a scenario where some clients locate in several separated places, and conduct the data collection tasks for FL. The data set each client collected is Non-IID and the collection task is a long term process. However, the existing FL frameworks have not taken the data collection into consideration, which results in a long convergence time and unsatisfactory training results.

Exmaple : Figure 1 shows the situation of clients collecting data from real world. Assuming client c_1 , c_2 and c_3 located in different places P_1, P_2, and P_3, and the collecting task is not done at one time. We assume in different locations P and different time periods T, the data distribution might change. Based on above assumptions, the data D_1 collected by c_1 has different Gaussian distribution from D_2 and D_3. Moreover, the subset of D_1 which gathered in T_1 is different from T_2.

Based on the aforementioned insights, we tackle the challenging real-scenarios-based federated learning problem by selecting clients to participate in model training through parameter prediction. We make several technical contributions:

1) We propose a novel Federated Learning framework based on Dynamic Client Selection method, namely FedDCS, whose central idea is the parameter estimation based dynamic selection mechanism. FedDCS allows the clients to join the training freely, and we utilize the parameter estimation algorithm to select participants dynamically.

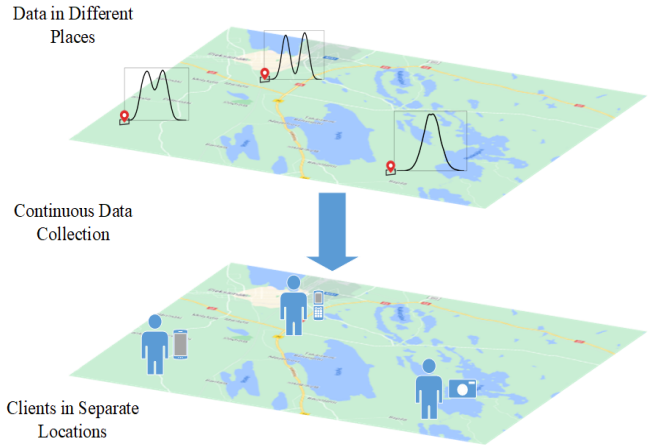


Fig. 1. The clients collect data from real world

To the best of our knowledge, this is the first work that combines parameters estimation and dynamic selection mechanism to solve real-scene data-based Non-IID problem in FL.

2) We introduce Gaussian Mixture Model (GMM) to indicate the distribution of real-world Non-IID data sets, and design an estimation algorithm to predict the parameters of GMM holds by clients. GMM can better clarify the continuously collected data. Moreover, we introduce Intel SGX TEE to ensure data privacy and the authenticity of uploading data.
3) We prove that the convergence of estimation algorithm, in limited rounds, this process will converge to a local maximum.
4) We conduct extensive experiments on FedDCS: we test the convergence speed of the estimation algorithm and time loss of utilizing Intel SGX to execute estimation algorithm in different buffer sizes. The comparison experiment between FedAvg, FedSGD, and FedDCS demonstrates the superior performance of our framework.

The remainder of the paper is organized as follows. In Sec. 2, we review the related works. In Sec. 3, we introduce the preliminaries and definitions. Modeling and the problem formulation will be introduced in Sec. 4. We elaborate the FedDCS in Sec. 5. In Sec. 6, we conduct the convergence analysis of estimation algorithm. The simulations and evaluations are presented in Sec. 7. We make the conclusion in Sec. 8.

II. Related Work

Since FL is proposed in 2016 by google, centralized FL frameworks have been gradually improved. Federated learning framework FATE [3] and TensorFlowFederated [4] are the representatives. Leaf Project [5] has provided multiple training environments which can simulate different Non-IID and IID scenarios. According to the frameworks, a central server is responsible for managing clients to accomplish machine learning tasks and coordinate the whole steps of algorithm. Each client keeps data in local storage so that they can preserve the privacy. In centralized frameworks, all the selected nodes are supposed to send updates to central server, leading to communication congestion, and the limited bandwidth also slows the global model convergence.

To avoid the communication bottleneck, some researchers have proposed new frameworks. Pokhrel *et al.* [6] and Li *et al.* [7] have proposed a blockchain-based decentralized FL framework. Roy *et al.* [8] proposed a peer-to-peer decentralized FL algorithm. Jeong *et al.* [9], Itahara *et al.* [10] and Li *et al.* [11] have proposed knowledge distillation methods to minimize the communication overhead. In a decentralized FL framework, nodes are supposed to coordinate with each other to obtain the global model, and the model updates will be exchanged only between nodes. Without communicating with a server, clients can perform the whole process themselves. Although decentralized frameworks ease the communication pressure to a certain extent, privacy leakage, untrusted nodes and Non-IID data sets can drastically influence training performance.

Non-IID data has been a hot issue since FL was proposed. Non-IID data can significantly influence the training results [12], much work has been done. FedProx [13] works well on Non-IID, it dynamically adjusts the local training iterations of each round, aiming to reduce the communication pressure and maximize the use of local computing resources. Zhao *et al.* [14] and Lu *et al.* [15] have proposed global sharing methods to provide a supplement for missing data, which aim to achieve faster convergence and higher global accuracy. Similarly, Federated Transfer Learning [16] allows knowledge to be shared without compromising user privacy. Hsieh *et al.* [17] have presented SkewScout, which can control the communication frequency according to the data distribution offset. FedAMP [18] is a framework that can satisfy individual model requirements. Differing from the conventional FL framework, FedAMP can customize self-adaptive modes, requiring clients to join in machine learning. SCAFFOLD [19] is a new algorithm specialized for Non-IID data training, and it aims to improve training performance in client-shift circumstances.

Besides using share data set to fulfill the shifting clients, there are researches focus on client selection. Nishio *et al.* [20] focus on selecting clients with limited computational resources. Zhang *et al.* [21] propose a dynamic fusion-based FL algorithm. Tuor *et al.* [22] design a selection mechanism aiming to choose the relevant clients.

Unlike the existing work, our study explores selection mechanisms that do not require a local training model. Our method is particularly effective when it meets data sets collected from real scenarios, typically following Gaussian distribution.

III. Preliminaries and Definitions

A. Federated Learning Framework

Centralized Federated Learning Frameworks have been proposed to achieve a satisfying situation by both user privacy security and machine learning efficiency. It consists of a server node and a series of clients. Federated Learning requires a central server to coordinate clients training a global model.

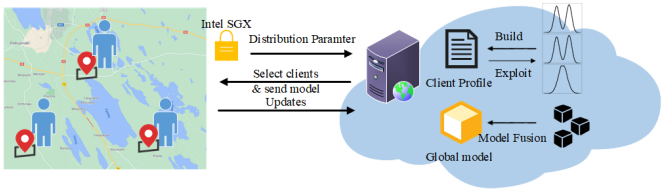


Fig. 2. The workflow of FedDCS

In general, the server is responsible for dispensing the initial global model and aggerating updates from clients. While the server does not accumulate data from clients, the clients store data locally to preserve privacy.

We denote machine learning model as W, to update W, clients perform SGD or other gradient descent algorithm, this process can be depicted by

$$W_j^{l+1} \leftarrow W_j^l - \nabla F(W_j, D_j) \tag{1}$$

where j represents client j, e denotes the e-th iteration of local training and $F(W, D)$ means the SGD methods trains model with local data set D.

After each client finished local training, server executes global aggregation task. This process can be performed for serval epochs. Let t denote global epochs, the fusion of global model can be calculated by

$$W^{t+1} \leftarrow \sum_{j \in M} W_j^t \tag{2}$$

B. Intel SGX Trusted Execution Environment

Intel Software Guard Extensions [23] is a set of new instructions on recent model of intel CPUs, which can confer reliable hardware-level security protection. Intel SGX can protect preselected codes and data fragments, developers can put data that need to be guarded into enclave. Enclave is a data storage zone that is isolated from software running outside of enclave. With the isolation protection, codes in enclave can keep confidentiality and integrity while the malicious software runs on the same system. Enclave memory is handled in plaintext only inside the CPU and is encrypted by the processor whenever it leaves so that even the OS can't access the data.

Intel SGX allows remote access to verify the software running in the enclave, and can communicate with it securely. And the software running in enclave can generate a report at times, which consists of supplementary data. The report is digitally signed with a hardware-level protected key. Remote computers have access to verify the report in order to prove that the measured software is running in Intel SGX enclave.

We assume the clients have access to Intel SGX Trusted Execution Environment, so that the preset programs can be executed in enclave without tampering. Moreover, the central server can verify the reports generated by software remotely, and the clients can also keep data locally, which satisfies both sides.

C. Gaussian Mixture Model

A mixture model is a probabilistic model for representing the presence of subpopulations in the overall population, it contains a series of complex data which belong to multiple regions with high probability mass. Data in mixture model are generated by randomly choosing mixture components and sampling from the distribution which associated with this component. Under this definition, we can do exact inference in the model. Gaussian distribution is well known in statistics, it follows the probability function

$$p(x|\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x-u)^2}{2\sigma^2}) \tag{3}$$

where x is univariate, and u represents mean value or expectation of x, and σ is standard deviation. According to the previous definitions, we introduce Gaussian Mixture Model. Assuming a set of observation data $x_1, x_2, \ldots, x_n$ which is sampled from K single gauss models, and α_k is the probability that observation data belongs to the K-th gauss model. $\phi(x|\theta_k)$ is the probability of K-th gauss model. With parameters above, GMM follows probability density function:

$$p(x|\theta) = \sum_{k=1}^{K} \alpha_k \phi(x|\theta_k) \tag{4}$$

for each single gauss model, θ represents the combination of parameters u, σ and α.

IV. Modelling and Problem Formulation

Our objective is to achieve better performance on real-world- based Non-identically distributed (Non-IID) data sets in FL and preserve client privacy in the same time.

Consider a real scenario where a series of clients $C = \{c_1, c_2, \ldots, c_M\}$ are collecting data in different places $P = \{P_1, P_2, \ldots P_M\}$ and different time periods $T = \{T_1, T_2, \ldots T_K\}$. After the collection task is finished, each client c_j holds the data set D_j. We assume in time period T_k the subset of data set D_j collected by client c_j follows Gaussian distribution p_k, that is to say, each data set D_j is a combination of data from $T_1, T_2 \ldots, T_K$, which follows $p_1, p_2, \ldots, P_K$ respectively. Obviously, those data sets are Non-IID, in detailed depiction, they have a prior probability shifting problem [24], which can result in sharply decrease of global model performance. Here we utilize GMM to describe the complex distribution, according to equation (4), the data set D_j follows probability:

$$p(x|\theta) = \sum_{k=1}^{K} \alpha_k \cdot p_k \tag{5}$$

where α_k represents the ratio of k-th distribution to the total observed data x.

After finishing the data collecting, clients intend to join the collaborating machine learning task. As shown in figure 2, the server is supposed to take over the entire process including selecting clients, dispensing global models, and fuse global model. This procedure will iterate several times till the final

TABLE I
DESCRIPTION OF MAJOR NOTATIONS

Variable	Description
D, D_j	the data set, data set own by client c_j
C, c_j	the clients set, the j-th client.
M, m	the number of total clients, number of selected clients
n, N	the number of observation data samples, the number of total observation data
k, K	the k-th Gaussian distribution, the number of all Gaussian distributions.
α_k	the ratio of k-th distribution to total observed data.
θ_k	the combination of parameters u_k, σ_k, α_k.
$\phi(x\|\theta)$	the probability function of gauss model.
x, x_i, X	the observation data samples,the i-th data sample of x, the observation data.
Y	the hidden variable.
Z	the complete data, which consists of X and Y.
$Q(\theta)$	the estimation-step function of estimation algorithm.
$L(\theta\|x)$	the likelihood function of GMM.
t	the t-th round of global epoch.
l	the l-th iteration of local training
W, W_j	the machine learning model, the machine learning model own by client c_j
δ, δ_s	the difference between current model and standard model, the threshold of δ.
p, p_k	the probability function, the probability function of k-th component in mixture model.

convergence. Each client c_j stores a data set D_j locally and the server can't obtain any information about the collected data. Under this circumstance, the server cannot screen out the clients who hold the low-quality data sets, and thus the accuracy of global model drops drastically.

We dedicate efforts to improving the machine learning accuracy and archiving faster convergence under the above condition.

V. FEDDCS FRAMEWORK

In this section, we propose a Federated Learning Framework FedDCS, which consists of Estimation Algorithm, Intel SGX Trusted Execution Environment, Dynamic Selection Mechanism and Model Fusion Mechanism.

In order to obtain the data distribution of all clients which can enable the server to evaluate the value of clients, FedDCS utilize Estimation algorithm to acquire the GMM parameters from the Intel SGX Enclave, which can preserve data privacy. Based on those augments, Dynamic Selection mechanism can screen out the valuable data sets, which can facilitate the performance of machine learning model. The details of above techniques will be elaborated in following sections.

A. Estimation Algorithm

GMM is a generation model, it assumes that data are generated from multiple Gaussian distributions. In this paper we offer each distribution a weight α_k, and when data generate, GMM randomly select a distribution according to the α_k, then, GMM creates data copies which follow chosen distributions.

According to the previous definition, server can't acquire the GMM parameters directly, therefore, we utilize the estimation algorithm to predict those arguments. To achieve the prediction task, we have to seek the solution for equation (5). We firstly introduce likelihood function, assuming we have samples $X = \{x_1, x_2, \ldots, x_n\}$ from the same distribution and they are independent identically distributed, following probability p. So we get joint probability density function $L(\theta|x) = p(x|\theta) = \prod_{i=1}^{n} p(x_i|\theta)$ which is regarded as the likelihood function of θ.

Since the value of function L can change with θ, we utilize Maximum Likelihood Estimation $\theta^* = argmaxL(\theta|X)$ to figure out the optimal value of θ to maximize function L. To make the equation computable, we change it to $\ln(L(\theta|x)) = \sum_{i=1}^{n} \ln p(x_i|\theta)$ and to seek the max value of L, we take the derivative of log-likelihood function $\frac{\partial}{\partial\theta} \ln L(\theta \mid X) = 0$.

According to the previous equations, the log likelihood function of GMM can be written as:

$$\ln(L(\theta|X)) = \ln \prod_{i=1}^{N} p(x_i|\theta) = \sum_{i=1}^{N} \ln[\sum_{k=1}^{K} \alpha_k p_k(x_i|\theta_k)] \quad (6)$$

Obviously, it's quite hard seeking the max value by directly deriving the log likelihood function, but if we know which distribution the observed data x_i stems from, the equation (6) can be solved. We introduce a hidden variable $Y = y_1, y_2, \ldots, y_N$, and $y_i \in 1, 2, \ldots, K$, where $y_i = k$ indicates thta the sample x_i originates from distribution k. $\ln(L)$ can be written as

$$\ln(L(\theta|x, y)) = \ln \prod_{i=1}^{N} p(x_i, y_i|\theta) = \sum_{i=1}^{N} \ln(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})) \quad (7)$$

We denote $Z = (X, Y)$ to represent the complete data, X is the observation data and Y is the hidden variable. $p = (y|x, \theta)$ is the density of hidden variable given the observed data X.

$$p(y|x, \theta) = \frac{p(z|\theta)}{p(x|\theta)} = \frac{p(x, y|\theta)}{p(x|\theta)} \quad (8)$$

We use iterative method to find the max value of $L(\theta|X)$ and denote θ^e as e-th round iteration optimal parameter of function as $L(\theta|X)$. We define $Q(\theta|\theta^e)$ as the expectation of loglikelihood function under the condition of observation data $X = x_1, x_2, \ldots, x_n$ as follows

$$\begin{aligned} Q(\theta|\theta^{(e)}) &= E[\log L(\theta|Z)|x, \theta^{(e)}] \\ &= \int [\log p(z|\theta)] p(y|x, \theta^{(e)}\, dy \end{aligned} \quad (9)$$

While the observation data X is given, we are going to solve the equation about Y. We seek the condition expectations on Y and obtain an equation only related to θ.

The estimation algorithm begins with $\theta^{(0)}$, and executes the iterations of Estimation step and Maximization step. estimation step aims to calculate the conditional expectation of $Q(\theta|\theta^{(e)})$ and Maximization step aims to maximum $Q(\theta|\theta^{(e)})$. Assume we have gotten $\theta^{(e-1)}$ in $e-1$-th iteration, in GMM we get

$$\theta^{(e-1)} = (\alpha_1^{(e-1)}, \ldots, \alpha_K^{(e-1)}, \theta_1^{(e-1)}, \ldots, \theta_K^{(e-1)})$$

According to equation (9), the estimation step is:

$$\begin{aligned} Q(\theta|\theta^{(e-1)}) &= E[\ln(L(\theta|X,Y))] \\ &= \sum_y \ln(L(\theta|X,y))p(y|X,\theta^{(e-1)}) \\ &= \sum_y \sum_{i=1}^{N} \ln(\alpha_{y_i} p_{y_i},(x_i|\theta_{y_i}))p(y|x_i,\theta^{(e-1)}) \end{aligned} \tag{10}$$

Since we already know that $p(y_i = k|x_i, \theta^{(e-1)})$ represent the probability of sample x_i originating from k-th distribution, the equation (10) can be written as:

$$\begin{aligned} Q(\theta|\theta^{(e-1)}) &= \sum_{k=1}^{K}\sum_{i=1}^{N} \ln(\alpha_k p_k(x_i|\theta_k))p(k|x_i,\theta^{e-1}) \\ &= \sum_{k=1}^{K}\sum_{i=1}^{N} \ln(\alpha_k)p(k|x_i,\theta^{(e-1)}) \\ &+ \sum_{k=1}^{K}\sum_{i=1}^{N} \ln(p_k(x_i|\theta_k))p(k|x_i,\theta^{(e-1)}) \end{aligned} \tag{11}$$

Based on Bayes Theorem, $p(k|x_i, \theta^{(e-1)})$ can be written as

$$p(k|x_i,\theta^{(e-1)}) = \frac{p_k(x_i|\theta_k^{e-1})}{\sum_{k'=1}^{K} p_k^{'}(x_i|\theta_{k'}^{(e-1)})} \tag{12}$$

and $p(k, x_i|\theta_k^{(e-1)})$ can be calculated by equation (8).

Maximization step utilizes derivation to seek the optimum parameters,

$$\begin{aligned} &\frac{\partial Q\left(\theta \mid \theta^{(e-1)}\right)}{\partial \mu_k} = \frac{\partial}{\partial \mu_k}\left[\sum_{i=1}^{N} \ln\left(p_k\left(x_i \mid \theta_k\right)\right) p\left(k \mid x_i, \theta^{(e-1)}\right)\right] \\ &= \frac{\partial}{\partial \mu_k}\left[\sum_{i=1}^{N}\left(\ln\left(\frac{1}{\sqrt{2\pi\sigma_k^2}}\right) - \frac{(x_i-\mu_k)^2}{2\sigma_k^2}\right) p\left(k \mid x_i, \theta^{(e-1)}\right)\right] \\ &= \sum_{i=1}^{N}\left[\frac{1}{\sigma_k^2}\left(x_i-\mu_k\right) p\left(k \mid x_i, \theta^{(e-1)}\right)\right] \\ &= \frac{1}{\sigma_k^2}\sum_{i=1}^{N}\left(x_i p\left(k \mid x_i, \theta^{(e-1)}\right)\right) - \frac{\mu_k}{\sigma_k^2}\sum_{i=1}^{N} p\left(k \mid x_i, \theta^{(e-1)}\right) = 0 \end{aligned} \tag{13}$$

from equation (13), we get u_k, and similar, we derive $Q(\theta|\theta^{e-1})$ with respect to σ_k^2. We get u_k and σ_k^2 as follows

$$\mu_k = \frac{\sum_{i=1}^{N}\left(x_i p\left(k \mid x_i, \theta^{(e-1)}\right)\right)}{\sum_{i=1}^{N}\left(p\left(k \mid x_i, \theta^{(e-1)}\right)\right)} \tag{14}$$

$$\sigma_k^2 = \frac{\sum_{i=1}^{N}\left(\left(x_i-\mu_k\right)^2 p\left(k \mid x_i, \theta^{(e-1)}\right)\right)}{\sum_{i=1}^{N}\left(p\left(k \mid x_i, \theta^{(e-1)}\right)\right)} \tag{15}$$

Note that α_k can't be calculated by using derivation directly, we introduce Lagrange multiplier $\sum_{k=1}^{K}\alpha_k - 1$, and we get equation as follows

$$\begin{aligned} &\frac{\partial}{\partial \alpha_k}\left[Q\left(\theta \mid \theta^{(e-1)}\right) + \lambda\left(\sum_{k=1}^{K}\alpha_k - 1\right)\right] \\ =&\frac{\partial}{\partial \alpha_k}\left[\sum_{k=1}^{K}\sum_{i=1}^{N}\ln\left(\alpha_k\right) p\left(k \mid x_i, \theta^{(e-1)}\right) + \lambda\left(\sum_{k=1}^{K}\alpha_k - 1\right)\right] \\ =&\frac{1}{\alpha_k}\sum_{i=1}^{N} p\left(k \mid x_i, \theta^{(e-1)}\right) + \lambda = 0 \end{aligned} \tag{16}$$

finally we get:

$$\alpha_k = \frac{\sum_{i=1}^{N}\left(p\left(k \mid x_i, \theta^{(e-1)}\right)\right)}{N} \tag{17}$$

B. Dynamic Selection Mechanism

We obtain parameters from estimation algorithm in previous section, with those parameters, a dynamic selection mechanism will be designed.

Firstly, we take effort dedicated to fitting parameters u, σ and α to a more general one δ, and a preset value δ_s, since u and σ stems from GMM, this mechanism will rebuild a GMM on the basis of those data, and calculate a difference of current distribution and standard distribution. In detailed depiction, this mechanism will compute the difference of the occurring frequency between label i and standard ones.

Argument δ_s set a threshold for selecting clients, *i.e.*, we only choose nodes whose $\delta < \delta_s$, and selected clients will join the training, then perform FedAvg and gather all local models to get a global model. The clients can execute the estimation algorithm once they have finished collection tasks, and the server can centralized processes the received parameters. The argument δ_s can be changed in a predetermined range so that the server can set the severity if the selected requirements. The Algorithm 1 gives an explicit description of the workflow.

As server can't acquire samples from the nodes, the rebuilding of GMM depends entirely on the parameters calculated in previous section. We set random number $n < N$ as the sample space, where N represents the total data size, and n must be greater than 500, or the rebuilt model would be quite different from the original one.

C. Model Fusion

In each epoch, the server node aggerates information and performs selection mechanism, then, those nodes will execute local training and run FedAvg algorithm to gather models and fuse into a global one. We make a detailed description of this process in Algorithm 2.

Note that in order to match the dynamic selection mechanism, once the client is chosen, the central server will not stop the current machine learning task, but send the last updates to the selected clients in the next epoch. Moreover, the Server only dispense initial parameters to the selected node, And also, only the selected nodes execute training works and upload models, which saves the bandwidth, increases spectrum efficiency and decreases communication pressure. In addition, the selection mechanism doesn't require any local computation so that the unselected nodes can save computing resources.

Algorithm 1: Dynamic Selection Mechanism

Input: M clients, each hold a set of private data. One Server
Output: m selected clients
Data: inital model W_g and preset threshold δ_s
for *client* c_j *in* M *Clients* **do**
  run estimation algorithm in SGX Enclave;
  upload parameter θ_j to Server;
  wait for response;
end
while *server get each* θ_j **do**
  rebuild GMM G with θ_j;
  calculate δ_j from G_j ($G_j \to \delta_j$);
  if $\delta_j < \delta_s$ **then**
    send initial model M_g to client j;
  else
    send rejection message to client j;
  end
end

Algorithm 2: Model Fusion

Input: M clients, each hold a set of private data D_j. One Server hold a global model.
Output: a trained global model W_g
Initialization;
From Algorithm 1, get selected clients C_m;
Server send initial model W_g to C_m;
for *each round* $t = 1, 2, \dots$ **do**
  for *each client* $c_j \in C_m$ **do**
    $W_j^{k+1} \leftarrow W_j^k - \nabla F(W, D_j)$
  end
  $W_g^t = \sum_{j=1}^{m} \frac{1}{m} W_j$
end

VI. CONVERGENCE ANALYSIS OF ESTIMATION ALGORITHM

In this section, we analyze the convergence of estimation algorithm.

Theorem 1. *Suppose that* $P = \{p_1, p_2, \dots, p_n\}$ *and* $Q = \{q_1, q_2, \dots, q_n\}$ *are two probability distributions. The following inequality holds:*

$$-\sum_{i=1}^{n} p_i \log p_i \le -\sum_{i=1}^{n} p_i \log q_i \tag{18}$$

the inequality changes to equality under the condition that if and only if $p_i = q_i$

Proof.

$$\begin{aligned} 0 &\ge \sum_{i=1}^{n} p_i \log q_i - \sum_{i=1}^{n} p_i \log p_i \\ &= \sum_{i=1}^{n} p_i \log(q_i/p + i) \end{aligned} \tag{19}$$

according to $\ln(x) \le x - 1$, the equal sign holds if and only if $x = 1$, the equation changes to:

$$\begin{aligned} &\sum_{i=1}^{n} p_i \log(q_i/p + i) \ge \sum_{i=1}^{n} p_i \log(q_i/p_i - 1) \\ &= \sum_{i=1}^{n} (q_i - p_i) = \sum_{i=1}^{n} q_i - \sum_{i=1}^{n} p_i = 0 \Longleftrightarrow q_i = p_i \end{aligned} \tag{20}$$

□

Theorem 2. $f(x), g(x)$ *is the given probability distribution that has the same support.*

$$H(f \mid g) = \int \log\left(\frac{g(x)}{f(x)}\right) g(x) dx$$

and we get:

$$H(f|g) \le 0, " = " \Longleftrightarrow f(x) = g(x)$$

Proof.

$$\begin{aligned} H(f|g) &= \int \log(\frac{g(x)}{f(x)}) g(x)\, dx \\ &\ge \int (\frac{f(x)}{f(x)} - 1) g(x)\, dx \\ &= -\int f(x)\, dx + \int g(x)\, dx = 0 \end{aligned} \tag{21}$$

□

Theorem 3. *Assume we get the estimated parameters* $\{\theta^e\}$*, and each maximization step increases the value of loglikelihood function* $L(\theta|x)$

$$\log(p(x|\theta^{(e+1)})) \ge \log(p(x|\theta^{(e)})) \tag{22}$$

Proof. According to equation (8) and equation (9), we get:

$$\log(p(z|\theta)) + c = \log(p(x|\theta)) + \log(p(y|x,\theta)) + c$$

$$Q(\theta|\theta^{(e)}) = \int_Y [\log(p(z|\theta))]\, p(y|x,\theta^{(e)}) dy$$

so we can get the following equations:

$$\begin{aligned} &Q(\theta|\theta_e) = \int_Y \Big(\log(p(x|\theta^{(e)})) + \log(p(y|x,\theta))\Big) \log(p(y|x,\theta^{(e)})) dy \\ &= \log(p(x|\theta^{(e)})) + \int [\log(p(y|x,\theta))] \log(p(y|x,\theta^{(e)})) dy \\ &\log(p(x|\theta^{(e+1)})) - \log(p(x|\theta^{(e)})) = Q(\theta|\theta^{(e+1)}) - Q(\theta|\theta^{(e)}) \\ &+ \int \log\left(\frac{\log(p(y|x,\theta^{(e)}))}{\log(p(y|x,\theta^{(e+1)}))}\right) \log(p(y|x,\theta^{(e)})) dy \end{aligned} \tag{23}$$

according to equation (22) and (23), we know that $\Delta Q(\theta^{(e+1)}|\theta^{(e)}, X) \ge 0$. On the basis of Theorem2, we get :

$$\int \log\left(\frac{\log(p(y|x,\theta^{(e)}))}{\log(p(y|x,\theta^{(e+1)}))}\right) \log(p(y|x,\theta^{(e)})) dy \ge 0 \tag{24}$$

□

From Theorem3 we know the EM algorithm will finally converges to a global or local maximum, the parameter sequence $\Theta = \left\{\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(e)}\right\}$ will eventually converges to the maximum likelihood estimation.

TABLE II
TIME LOSS OF SGX IN DIFFERENT BUFFER SIZE

Buffer Size	20kb	10mb	20mb
CPU Cycles (StartUp)	1.43×10^3	1.7×10^8	6.9×10^8
CPU Cycles (ShutDown)	3×10^2	1.5×10^7	2.1×10^7
Buffer Size	40mb	160mb	
CPU Cycles (StartUp)	1.3×10^9	5.25×10^9	
CPU Cycles (ShutDown)	4.13×10^7	1.4×10^8	

TABLE III
ESTIMATION RESULTS

Epoch	Mu	Sigma	Alpha
Standard	(2.1,1.63,1.74)	(0.001,0.062,0.086)	(0.2,0.57,0.23)
1	(1.51, 1.77, 1.80)	(0.987 1.407 1.11)	(0.01,0.6,0.39)
5	(1.86, 1.78, 1.78)	(0.201, 0.177, 0.178)	(0.0,0.54,0.45)
10	(1.91, 1.78, 1.78)	(0.196,0.177, 0.179)	(0.0,0.54,.045)
100	(2.10, 1.67, 1.7)	(0.001,0.083, 0.085)	(0.2,0.49,0.30)
400	(2.10, 1.65, 1.77)	(0.001,0.072, 0.075)	(0.2, 0.45, 0.34)

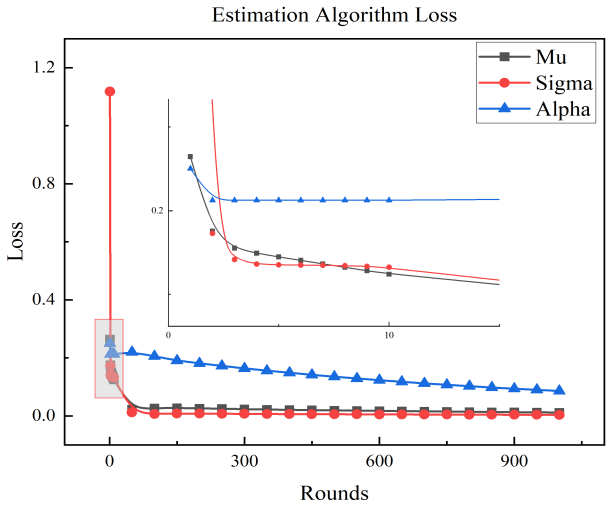


Fig. 3. Estimation Algorithm Loss

VII. PERFORMANCE EVALUATION

In this section, we evaluate the performance of FedDCS and compare it with the popular Federated Learning algorithm, FedAvg and FedSGD. To make our experiment more comprehensive, we utilize two kind of CNN networks and three kinds of data sets to test the performance of FL framework.

A. Environment Setting

1) Data Setting: We utilize CIFAR-10 and MNIST data sets to verify the performance of FedDCS. CIFAR-10 data set is a collection of images that are commonly used to train machine learning algorithm, which contains 60,000 32*32 images in 10 different classes, and the images are 3-channel RGB color ones. Similarly, MNIST is a large database of handwritten digits 0 to 9 and it contains 60,000 training images and 10,000 testing images. Those images are 28*28 one channel (black and white). To simulate the real scene, we create 10 clients in this experiment, each holds a set of images. While those data are Non-IID, they follow GMM. For each kind of data set, we randomly choose a series of normal distribution parameters and assign them to clients. We assume each client owns a GMM data set which consists of three Gaussian distributions. The unstableness catastrophically destroys the performance of FedAvg and also significantly influences local model training.

Let us take CIFAR-10 for example to introduce how we apply the Non-IID data setting. We firstly random generate a series of normal distribution parameters $\theta_i = \{u, \sigma, \alpha\}$ and sample from CIFAR-10 with θ. The sample categories are serialized as the one-hot form, number 0-9 corresponds to the category of CIFAR-10. Then we construct a GMM based on the three selected Gaussian distributions. For convenience of computation, each distribution holds the same size of image sets, in different data sets, the set size can vary from 600 to 6000.

2) Model Setting: AlexNet [25] and LeNet-5 [26] are the two kinds of improved convolutional neural networks (CNN). LeNet-5 is firstly proposed to solve the handwriting recognition problem, it is composed of 7 layers, including 3 convolutional layers, 3 pooling layers, and 2 fully connected layers. Compared to LeNet-5, AlexNet enriches the CNN layers and adds LRN. With Relu function, the AlexNet can converge faster and cost less than LeNet-5.

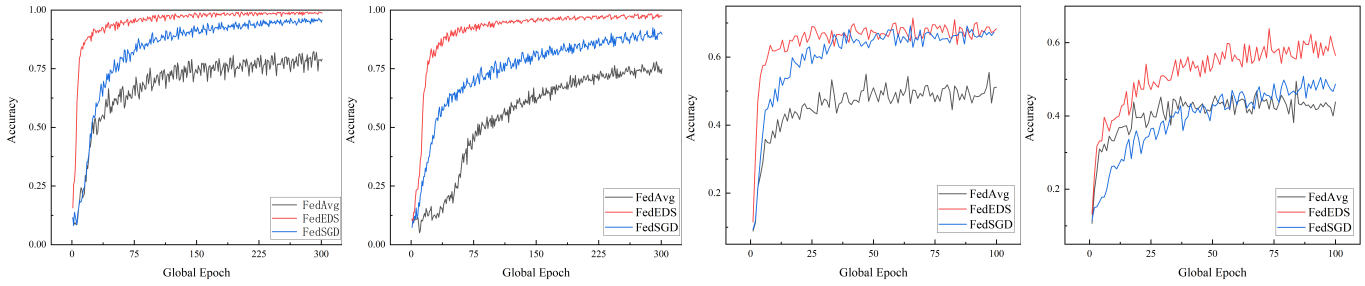
3) Comparsion Setting: We assume there exists at least one client which holds a lower quality, it typically can't achieve the preset threshold in the selection mechanism. Under this condition, we compare the performance of FedDCS, FedAvg, and FedSGD on the Non-IID data sets in different δ_s. In addition, we measured the time consumption of running algorithm programs in Intel SGX Enclave, which aims to prove the loss is acceptable.

4) Environment Setting: We utilize TensorFlow Federated (TTF) framework to simulate the server and client model. Each client has similar level hardware: CPU-Intel Core 8700K, RAM-64GB, GPU-1080Ti (SLI, CUDA10.0), OS-Ubuntu20.04. TensorFlow-1.14.

Each client is allocated with an Intel CPU, which enables them to run the preset EM algorithm program in Enclave in parallel. Clients and server are deployed separately, they communicate through HTTP protocol. And the clients are distributed deployed, each client can only sense its own data. In this paper we assume the total number of clients is 10, and we design different GMM on each client to stimulate the real-world distribution. Moreover, we set the local training iteration for different CNN and different data set. For CIFAR-10 with AlexNet and LeNet-5, we set the local iteration $l = 4$ and $l = 6$ respectively. And for MNIST with AlexNet and LeNet, we set $l = 2$ and $l = 4$.

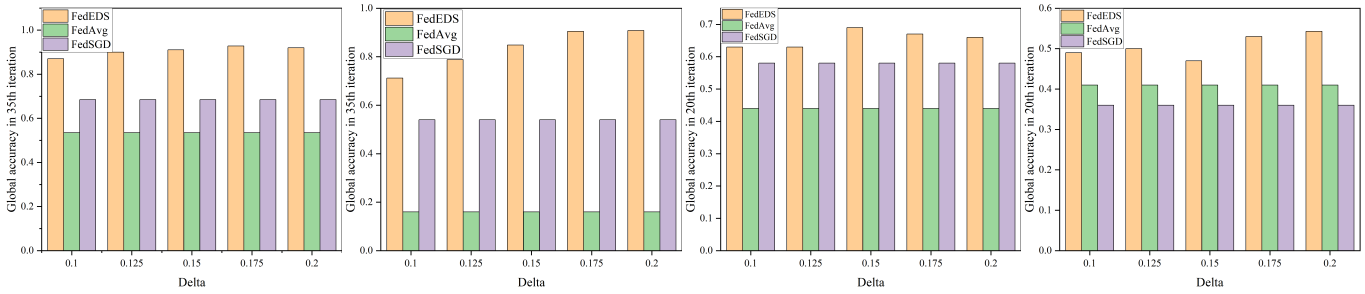
B. Results on Practical Non-IID data Setting

1) Estimation Results: In this section, we bring out an experiment which aims to estimate parameters of a 3 branch GMM. We assume a 3-branch GMM follows probability



(a) MNINST with AlexNet (b) MNINST with LeNet-5 (c) CIFAR-10 with AlexNet (d) CIFAR-10 with LeNet-5

Fig. 4. Global Accuracy of FedAvg, FedSGD and FedEDS



(a) MNINST with AlexNet (b) MNINST with LeNet-5 (c) CIFAR-10 with AlexNet (d) CIFAR-10 with LeNet-5

Fig. 5. Global Accuracy of FedAvg, FedSGD and FedEDS in Different δ_s

function as follows

$$p(x|\alpha,u,\sigma) = 0.572N(x|1.74,0.0865) + 0.227N(x|1.63,0.062) + 0.2N(x|2.1,0.001)$$

We firstly generate a series of samples which follow the above function and denote it as the observation data x. Then we execute the estimation algorithm for serval rounds. After the iterative calculation, we compare the results with the actual parameters. For a more intuitive view of the comparison results, we use $\frac{(\sum_k^K u_k - K u_r)}{K u_r}, k = \{1, 2, \ldots, K\}$ to calculate the average error ratio of the parameters and show the variation of the errors in each round.

Obviously, through the algorithm, we get anideal result. Table 2 shows the estimation values in 400 rounds and figure 3 shows the difference between standard value and estimation value in 1000 rounds.

The estimation algorithm runs on each independent device, while the apacity of calculation can be scarce, the process reaches convergence in limited rounds and the convergence speed is also acceptable.

2) Time Loss: Since the estimation part is running in Intel SGX Enclave, we test the time consumption of executing algorithm under different data scales. We disabled automatic CPU frequency scaling, Turbo Boost, and hyper-threading to avoid inconsistent performance behaviour. We test the startup and shutdown overhead of Enclave in different buffer size, aiming to figure out the time loss of running program in SGX. We set the buffer size as 20kb, 10Mb, 40Mb and 160Mb, and record the consumed CPU cycles, table 2 shows the experiment result.

While the upper limit of data size in each client is 600 (10% of 6000), which is far less than the 20kb, therefore, we can see that the time loss of Intel SGX is completely acceptable.

3) Training Results: We first test the global accuracy on 10 different clients, apparently, as shown in figure 4, FedDCS has higher global accuracy and faster convergence compared to FedAvg and FedSGD on both CIFAR-10 and MNIST datasets. Take CIFAR-10 with AlexNet for example, while the global accuracy of final convergence between FedSGD and FedDCS is very close, the FedDCS achieve a faster convergence. Then we test the accuracy of FedDCS in different δ_s settings, and compare it with FedAvg and FedSGD in the same rounds. Figure 5 shows the results, our method has better performance and the optimum threshold is 0.175.

VIII. CONCLUSION

In this paper, we tackle the challenging problem of real-scene-based Non-IID data FL and develop FedDCS that introduces a parameter-estimation-based dynamic selection mechanism to significantly facilitate the collaboration effectiveness between clients without infringing their privacy. We analyze the distribution of real-world data and utilize GMM to build the Non-IID data distribution model, which builds the foundation of the estimation algorithm. Moreover, we introduce Intel SGX to keep data from privacy leakage risk and conduct extensive experiments to prove the time loss is acceptable.

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 *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[2] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[4] M. Abadi, A. Agarwal, and P. Barham, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[5] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," 2019.

[6] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4734–4746, 2020.

[7] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, 2020.

[8] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[9] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S. Kim, "Federated distillation and augmentation under non-iid private data," *NIPS Wksp. MLPCD*, 2018.

[10] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data," *arXiv preprint arXiv:2008.06180*, 2020.

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

[12] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.

[13] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020.

[14] 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.

[15] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[16] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, 2020.

[17] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4387–4398.

[18] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-iid data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

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

[20] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–7.

[21] W. Zhang, T. Zhou, Q. Lu, X. Wang, C. Zhu, H. Sun, Z. Wang, S. K. Lo, and F.-Y. Wang, "Dynamic fusion-based federated learning for covid-19 detection," *IEEE Internet of Things Journal*, 2021.

[22] T. Tuor, S. Wang, B. J. Ko, C. Liu, and K. K. Leung, "Data selection for federated learning with relevant and irrelevant data at clients," *arXiv preprint arXiv:2001.08300*, 2020.

[23] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[24] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

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