

Data Heterogeneity-Robust Federated Learning via Group Client Selection in Industrial IoT

Zonghang Li, Yihong He, Hongfang Yu, *Member IEEE*, Jiawen Kang, Xiaoping Li, Zenglin Xu, *Senior Member IEEE*, Dusit Niyato, *Fellow IEEE*

Abstract—Nowadays, the industrial Internet of Things (IIoT) has played an integral role in Industry 4.0 and produced massive amounts of data for industrial intelligence. These data locate on decentralized devices in modern factories. To protect the confidentiality of industrial data, federated learning (FL) was introduced to collaboratively train shared machine learning models. However, the local data collected by different devices skew in class distribution and degrade industrial FL performance. This challenge has been widely studied at the mobile edge, but they ignored the rapidly changing streaming data and clustering nature of factory devices, and more seriously, they may threaten data security. In this paper, we propose FEDGS, which is a hierarchical cloud-edge-end FL framework for 5G empowered industries, to improve industrial FL performance on non-i.i.d. data. Taking advantage of naturally clustered factory devices, FEDGS uses a gradient-based binary permutation algorithm (GBP-CS) to select a subset of devices within each factory and build homogeneous super nodes participating in FL training. Then, we propose a compound-step synchronization protocol to coordinate the training process within and among these super nodes, which shows great robustness against data heterogeneity. The proposed methods are time-efficient and can adapt to dynamic environments, without exposing confidential industrial data in risky manipulation. We prove that FEDGS has better convergence performance than FedAvg and give a relaxed condition under which FEDGS is more communication-efficient. Extensive experiments show that FEDGS improves accuracy by 3.5% and reduces training rounds by 59% on average, confirming its superior effectiveness and efficiency on non-i.i.d. data.

Index Terms—AI, Federated Learning, Industrial IoT, Data Heterogeneity, Client Selection, Cluster Learning.

I. INTRODUCTION

In recent years, the Internet of Things (IoT) has played an increasingly integral role in the industrial community. Taking logistics sorting and automatic object identification as examples. Optical character recognition (OCR) cameras on logistics pipelines detect and read characters on packing boxes in order to sort them [1]. At the same time, the

surrounding surveillance cameras are constantly monitoring, automatically identifying objects through optical recognition of the characters on their badges, and confirming whether the machines, robots, vehicles, and workers in the factory are legal entrants [2]. These optical sensors collect a huge amount of industrial data. In order to tap the value of these data, advanced data mining technologies are needed, especially machine learning (ML). However, gathering the industrial big data to the cloud leads to unbearable transmission overhead, and also violates data privacy regulations. Taking the idea of task offloading, federated learning (FL) [3] sinks model training from the cloud to the edge. OCR cameras use local optical data to train local OCR models, then upload their local model updates to the cloud to update the global model. The global model is then synchronized to OCR cameras. These steps are repeated until the global model converges. In this way, FL preserves data confidentiality because the raw data does not leave the devices.

The combination of industrial IoT (IIoT) and FL opens a door for smart industry [4]–[6]. FL provides powerful privacy-preserving tools for mining decentralized industrial data, and IIoT technologies such as smart sensors and mobile robots provide rich resources (e.g., data, computation) for FL. Despite these benefits, compared with OCR in natural scenes, FL in industries requires higher accuracy to ensure the reliability of industrial operations. However, sensors' local data distributions can be highly heterogeneous due to differences in times, locations, functions, and so on. Taking the logistics industry of cross-border e-commerce as an example, the Singapore warehouse transports more packing boxes to Singapore than other countries, so the optical characters in the word “Singapore” appear more times than other characters. For this reason, the number of optical images of each character captured by different OCR cameras (in different warehouses) is skewed and inconsistent. Other examples are device failure detection [7] and object detection [8], which also prove the existence of data heterogeneity in real-world IIoT. These skewed distributed data are called non-independent and identically distributed (non-i.i.d.) and can lead to FL performance degradation [9], which becomes more challenging when local data of sensors are constantly changing.

The non-i.i.d. data challenge has inspired the research field of heterogeneous FL, especially the field of mobile edge computing (MEC) [5], [10]–[19], which currently remains open. These kinds of literature have achieved great success in the context of MEC, but the following characteristics of IIoT make them still limited:

¹This work was supported in part by National Key Research and Development Program of China (2019YFB1802800), PCL Future Greater-Bay Area Network Facilities for Large-Scale Experiments and Applications (LZC0019).

²Zonghang Li, Yihong He and Hongfang Yu are with School of Information and Communication Engineering, University of Electronic Science and Technology of China. (Emails: lizhu@uestc, heyh.uestc, yuhf@networklab@gmail.com). Xiaoping Li is with School of Mathematical Science, University of Electronic Science and Technology of China. (Email: lixiaoping.math@uestc.edu.cn). Zenglin Xu is with School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), China. (Email: xuzenglin@hit.edu.cn). Dusit Niyato and Jiawen Kang is with School of Computer Science and Engineering, Nanyang Technological University, Singapore. (Email: dniyato, kavinkang@ntu.edu.sg). Corresponding author: Hongfang Yu.

³This work was done in part at Nanyang Technological University (NTU).

- 1) **Higher requirements for data security.** Industries (e.g., manufacturing, logistics, and transportation) often face even more serious data threats due to owning a vast amount of valuable information, so they possess the most urgent and critical requirement to increase security to protect data. Therefore, any form of disclosure [10]–[13] or tampering [14]–[16] of the confidential raw data is not allowed.
- 2) **Rapidly changing streaming data on data-intensive sensors.** Data-intensive IIoT sensors such as OCR cameras require high sampling rates to capture the real-time phenomenon information and produce large amounts of data. In order to save storage space, these data will overwrite the old data that has been processed, forming a data stream similar to a first-in-first-out (FIFO) data queue. In such a dynamic environment, static approaches no longer work for IIoT, for example, [17], and the K-Center clustering algorithm in [18].
- 3) **Natural geographical clustering property.** In modern industrial parks, IIoT devices in each factory are geographically adjacent, which makes them naturally clustered into groups and interconnected by highly reliable networks, for example, through regional 5G base stations (see Fig. 1). However, this valuable property is often ignored, and the rich communication resources at the edge are not fully utilized [12], [19], [20], which limits the improvement of industrial FL.

The above characteristics distinguish “FL in IIoT” from “FL in non-IIoT” (e.g., FL in Edge). Few literature has been proposed to tackle the non-i.i.d. data challenge of FL in IIoT, such as approaches based on centroid distance weighted averaging [7], reinforcement learning [21], and kmeans-based cohorts [22]. However, none of them take into account the changing local data distribution or the natural geographic clustering property of devices in IIoT. More importantly, they do not address the fundamental problem causing FL model performance to degrade, namely the divergence in class distributions [9].

To address the root cause of non-i.i.d. data, this paper aims to propose an effective approach to minimize the divergence in class distributions among heterogeneous devices. Taking advantage of the natural property of geographical clustering, we can select a subset of devices in each factory to construct “FL super nodes” with consistent class distributions. These super nodes can be treated as homogeneous clients participating in FL training, without exposing the confidential industrial FL process in risky data manipulation. However, designing such an approach is not trivial. Firstly, selecting a subset of devices in each group to minimize the class distribution divergence among groups is a 0-1 integer programming problem with vector weight constraints, which is proved to be NP-complete. More challenging, this procedure needs to be invoked frequently to adapt to rapidly changing local data and the mobility of mobile IIoT devices (e.g., robots, drones), which places high demands on execution latency. Secondly, even if class distributions among FL super nodes are forced to be homogeneous, devices’ local data in each FL super node

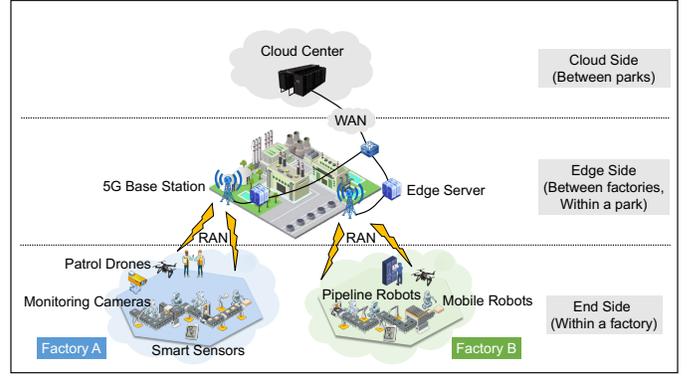


Fig. 1: A network architecture example in the modern industrial park. End devices within each factory submit locally trained ML models to nearby 5G edge base stations. Edge servers synchronize these models and upload the synchronized models to the cloud server for global synchronization. The cloud center can be located on the cloud to synchronize ML models from multiple industrial parks, or it can be a micro data center located in an industrial park to synchronize ML models of edge servers in this park.

can still be skewed. If not handled properly, these challenges will still degrade FL model performance.

To minimize the data heterogeneity among groups and realize efficient client selection, this paper proposes a novel gradient-based binary permutation optimizer GBP-CS to solve the above NP-complete client selection problem. GBP-CS runs a constraint-preserving gradient descent optimization procedure directly in the 0-1 integer space, and can build homogeneous FL super nodes in a very short time. Then, we propose Federated Group Synchronization (FEDGS), which is a hierarchical cloud-edge-end FL framework for 5G empowered modern industries, to improve industrial FL performance on non-i.i.d. data. FEDGS uses a compound-step synchronization protocol to train ML models, which can suppress data heterogeneity within and among FL super nodes. More specifically, FEDGS uses a single-step synchronization protocol (e.g., SSGD [23]) within super nodes because of its robustness against data heterogeneity, and a multi-step synchronization protocol (e.g., FedAvg [24]) among homogeneous super nodes to reduce communication overhead. Theoretical analysis shows that FEDGS has both the convergence upper bound and optimality gap better than FedAvg in the presence of non-i.i.d. data, and can be more time-efficient under a relaxed condition. Finally, we evaluate FEDGS on the most widely adopted non-i.i.d. benchmark dataset FEMNIST [25], and compare it with 10 advanced approaches, including FedAvg, FedMMD [26], FedFusion [27], FedProx [28], IDA [29], CGAU [30], FedAvgM [31], and FedAdagrad, FedAdam, FedYogi from [32]. The main contributions of this paper are summarized as follows.

- We propose a hierarchical cloud-edge-end FL framework FEDGS for 5G empowered modern industries, which uses a novel compound-step synchronization protocol to coordinate the training process within and among groups. The new protocol is robust against data heterogeneity and can effectively improve industrial FL performance.
- We propose a novel GBP-CS algorithm to select a subset

of devices from each group to build homogeneous FL super nodes, which can find a desirable selection strategy in a very short time. GBP-CS is a general optimizer for constrained 0-1 integer programming problems and can be used for other practical cases such as game matching.

- We analyze the convergence rate and optimality gap of FEDGS and give a relaxed condition under which FEDGS is more time-efficient than FedAvg. Theoretical results show that FEDGS not only converges closer to the optimal, but also faster.
- Extensive experiments compared to 10 advanced approaches show that FEDGS improves FL accuracy by 3.5% and reduces training rounds by 59% on average. The results highlight the superior effectiveness and efficiency of FEDGS on non-i.i.d. data.

II. RELATED WORK

In this section, we categorize related works into four types according to the techniques they use.

Data Sharing and Augmentation. This type of approach aims to minimize the class distribution divergence among devices by sharing or augmenting FL clients’ local datasets. For the sharing-based approaches, Zhao *et al.* propose to distribute a small portion of globally shared data (e.g., open available data) to clients’ devices [9]. Yao *et al.* collect metadata shared by voluntary clients to perform controllable meta updating [10]. Yoshida *et al.* reward FL clients for contributing local datasets and propose a hybrid learning mechanism wherein the server updates the model using the shared data and clients’ local models [11]. These approaches achieve a considerable improvement in FL accuracy, but they are suspected of leaking private data due to the need to share clients’ local datasets. Besides, open-available datasets do not always exist, especially in fields where data is highly confidential.

For the augmentation-based approaches, Duan *et al.* observe that the imbalance among different classes can also degrade FL accuracy [14]. Hence, they augment classes with fewer samples by simply random offset, rotation, cropping, and scaling. Jeong *et al.* [33] propose to generate new samples using a globally trained conditional generative adversarial network (CGAN) to build unskewed local datasets. Similarly, Wang *et al.* generate synthetic data in the minority class based on linear interpolation to re-balance local datasets on edge devices [15]. These approaches avoid the leakage of FL clients’ private data. However, they still bring credibility crises. Speculative clients can use the synthetic data generated out of thin air to participate in FL training while hiding their original data. Also, they can pretend that the synthetic data is a large volume of high-quality data for more rewards. Therefore, these operations (i.e., data sharing, data augmentation) are high risk and should be prohibited.

Hyperparameter Tuning. Hyperparameters play an important role in FL training convergence. Some works have been explored in hyperparameter tunings, such as tuning the number of local iterations and the learning rate. Wang *et al.* point out that the optimal performance can be achieved when the number of local iterations equals one [34]. However, the

constrained resources (e.g., bandwidth, time, power) prevent us from doing this. In practice, large local iterations are more commonly used. For example, Yu *et al.* carefully set the number of local iterations and obtain a considerable convergence rate [35]. In addition, Li *et al.* point out that decaying the learning rate is necessary for FL convergence with large local iterations [36]. For a strongly convex and smooth objective function, FedAvg can converge to the optimal after applying learning rate decay, with a convergence rate of $\mathcal{O}(1/\mathcal{T})$, where \mathcal{T} is the total number of local updates on a single device. These works give rigorous proofs for convergence analysis, which guides follow-up optimization on FL. However, carefully tuning these hyperparameters (e.g., number of local iterations, learning rate, and decay rate) requires multiple attempts and incurs high time costs.

Client-Side Adaption. This type of approach emphasizes that FL clients should adaptively retain global knowledge while improving local knowledge. Some examples are given to integrate them. Yao *et al.* point out that the global model contains more global knowledge and should be kept as a reference, rather than simply thrown away. Based on this idea, they adopt a two-stream model to transfer the global knowledge to the local model [26]. By minimizing the maximum mean discrepancy (MMD) loss, the two-stream model can extract more generalized features and learn better local representations. Then, in [27], they use the 1×1 convolution, vector weighted average, and scalar weighted average operators to fuse the global and local features. Li *et al.* point out that too many local updates will cause the FL training to diverge, especially under the non-i.i.d. data setting [28]. Hence, they add a proximal penalty term to local objective loss functions to constrain the local model to be closer to the global model and avoid excessive divergence. Rieger *et al.* point out that clients express representations in different patterns and their shared knowledge may be obfuscated after synchronization [30]. Hence, they adopt conditional gated activation units to enable clients to condition their units. In this way, clients can identify whether the global feature is expressed and how to modulate the global pattern. These approaches impose more storage footprint and computation on resource-constrained client-side devices, requiring higher resource allocation and also higher energy consumption.

Server-Side Adaption. This type of approach explores how local models can be adaptively aggregated and how the global model can be adaptively optimized on the server-side. Yeganeh *et al.* aggregate clients’ local models according to their weights, by capitalizing an adaptive weighting approach based on the inverse distance between the local model parameters and the averaged model parameters [29]. By using this approach, out-of-distribution models will be weighed down and the global model can have a lower variance. The authors also explored the combination with other metrics, such as the training accuracy and the data size. However, these variants did not perform well in our experiments, probably because some honest but “out-of-distribution” devices were over-suppressed. On the other hand, inspired by the ability of momentum accumulation to dampen oscillations [37], Hsu *et al.* adopt the momentum optimizer on the server-side and observe a

TABLE I: Summary of Main Symbols.

Symbol	Explanation
\mathcal{L}	Loss function (e.g., cross-entropy).
R	Maximum training rounds.
T	Number of iterations in each round.
K, K^m	Number of devices (in factory m).
L	Number of devices to be selected per factory.
L_{rnd}	Number of devices to be randomly pre-sampled per factory.
L_{sel}	Number of devices to be selected by GBP-CS per factory.
M	Number of factories (also the number of groups).
F	Number of classification classes.
η	Local learning rate.
ω_t	Parameters of the global ML model at t -th iteration.
ω_t^m	Parameters of the ML model on BS m at t -th iteration.
$\omega_t^{m,k}$	Parameters of the local ML model on device k in factory m at t -th iteration.
$\mathcal{D}^{m,k}$	Local dataset of device k in factory m .
$\mathcal{D}_t^{m,k}$	A mini-batch data of device k in factory m at t -th iteration.
$N^{m,k}$	Size of local dataset of device k in factory m .
$n, n^{m,k}$	Batch data size (of device k in factory m).
n^m	Total size of data batches in factory m .
$\mathbf{a}_t^{m,k}$	Data size vector of F label classes of mini-batch data $\mathcal{D}_t^{m,k}$.
\mathbf{A}_t^m	Data size matrix of $\mathbf{a}_t^{m,k}$ of $K^m - L_{\text{rnd}}$ devices.
\mathbf{b}_t^m	Total data size vector of the pre-sampled L_{rnd} devices.
\mathcal{C}^m	Set of all devices in factory m .
\mathcal{C}_t^m	Set of L selected devices in factory m at t -th iteration.
$\mathcal{P}^{m,k}$	Local data distribution of device k in factory m .
$\mathcal{P}_t^{m,k}$	Local data distribution of mini-batch data $\mathcal{D}_t^{m,k}$.
\mathcal{P}_t^m	Mean data distribution of $\mathcal{P}_t^{m,k}$ over selected devices \mathcal{C}_t^m .
$\mathcal{P}^{\text{real}}$	Real-world global data distribution.

significant improvement in FL accuracy [31]. Then, Reddi *et al.* introduce three advanced adaptive optimizers (i.e., Adagrad [38], Adam [39], and Yogi [40]) to update the server-side global model [32]. These adaptive federated optimizers enable the use of adaptive learning rates for different gradients and achieve great success, but unfortunately, they also require careful tuning of initial learning rates, and we observed drastic accuracy oscillation in the experiments.

III. SYSTEM MODEL

IoT devices in modern industrial parks can be divided into two types: Fixed devices (e.g., monitoring cameras, temperature and humidity sensors) and mobile devices (e.g., patrol drones and logistics robots). In the industrial park, due to the advantages of improved performance, reduced communication cost, and decentralized scalability, FL plays an important role in many industrial applications. For example, an anomaly detection application based on on-device federated monitors could be applied for IIoT scenarios, where sensing and monitoring devices may locate in harsh environments with high voltage and high radiation, and they may move around the factory, making them impractical to access wired networks [41]. 5G mobile networks have been considered to be enhanced to support key performance features of industrial applications such as high throughput, low latency, and high scalability [42]. These features enable industrial FL applications to transmit model data of a large number of IIoT devices at a high cycle frequency, with a high data rate and low latency. Therefore, we consider a hierarchical cloud-edge-end network architecture empowered by 5G cellular wireless networks for training industrial FL applications, as shown in Fig. 1.

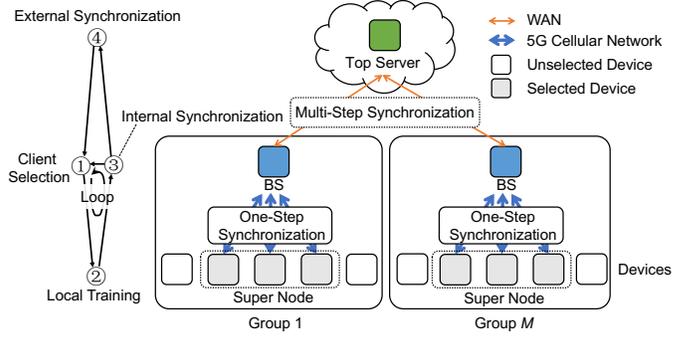


Fig. 2: Overview of FEDGS framework and workflow. ① Each BS selects L devices to form a super node. ② Each selected device trains its local model for one mini-batch SGD step. ③ Each BS synchronizes local models in its group. ④ Top server synchronizes models of BSs. ①②③ loop T times every ④.

In this case, a modern industrial park has M factories, each factory m has K^m smart devices. We consider the devices in the same factory as a group. These devices are equipped with embedded computing chips to perform lightweight processing, for example, training ML models on local data streams. The devices connect to the nearby base station (BS) (also identified as m) through 5G cellular wireless networks. Then, BSs communicate FL model data with the cloud center through the Internet. Specifically, the device k of factory m collects streaming sensory data in real-time and changes the local dataset $\mathcal{D}^{m,k}$, whose class distribution is defined as $\mathcal{P}^{m,k}$. The class distributions of different devices can be highly heterogeneous due to diverse local usage patterns. Thus, we have $\mathcal{P}^{m_1,k_1} \neq \mathcal{P}^{m_2,k_2} \neq \mathcal{P}^{\text{real}} (\forall m_1 \neq m_2, k_1 \neq k_2)$, where $\mathcal{P}^{\text{real}}$ is the real-world global class distribution. The goal of industrial FL is to find the optimal model parameters ω^* that can minimize the global loss function,

$$\omega^* \triangleq \arg \min_{\omega} \sum_{m=1}^M \sum_{k=1}^{K^m} \mathcal{L}(\omega, \mathcal{D}^{m,k}). \quad (1)$$

The traditional workflow is briefly described below. In each round, a small subset of devices is randomly selected to participate in FL training. These devices utilize local datasets for several epochs to train their local ML models and upload these local models to connected BSs. BSs aggregate these local models and upload the aggregated models to the top server in the cloud. The top server globally aggregates BSs' models, updates the global ML model, and synchronizes the updated model back to all BSs and end devices. These steps are repeated until the global model parameters ω^* converge. However, this approach causes performance degradation to industrial FL due to data heterogeneity among devices, and it ignores the streaming nature of industrial data. Hence, in Section IV, a data heterogeneity-robust federated group synchronization approach is presented to address this issue.

We summarize the main symbols used in this paper in Table I. The framework and workflow of the proposed FEDGS are illustrated in Fig. 2.

IV. FEDGS: FRAMEWORK AND WORKFLOW

The core idea is to strategically select a small subset of devices in each group to form FL super nodes with homogeneous data distributions. Then, these super nodes can be regarded as homogeneous clients to participate in FL. To resolve the heterogeneity in local datasets of devices inside each super node, the one-step synchronization protocol (e.g., SSGD) can be useful because it was proved to be equivalent to centralized SGD, which gives it robustness against data heterogeneity. Meanwhile, the multi-step synchronization protocol (e.g., FedAvg) can be used to keep FEDGS communication efficient, since the class distributions of super nodes are aligned. In this way, the problem of data heterogeneity is decomposed from the entire population of devices to a small number of devices in multiple groups, making it efficiently and effectively solved by the compound-step synchronization protocol. In this way, the performance degradation of industrial FL is addressed.

The detailed design is given in Alg. 1. In the initialization stage, the top server first initializes the global ML model parameters ω_0 and synchronizes $\omega_0^m \leftarrow \omega_0$ to BSs. Then, it collects local class distributions $\mathcal{P}^{m,k} (\forall m, k)$ from all devices to estimate the real-world global class distribution $\mathcal{P}^{\text{real}}$,

$$\mathcal{P}^{\text{real}} = \text{norm} \left(\sum_{m \in M} \sum_{k \in K^m} N^{m,k} \mathcal{P}^{m,k} \right), \quad (2)$$

where $N^{m,k}$ is the local data size of device k in group m , and $\text{norm}(\cdot)$ is a probability normalization function.

Client Selection. In each iteration t , each BS m selects L devices from its group \mathcal{C}^m to obtain a homogeneous super node \mathcal{C}_t^m via the Select-Clients-Via-GBP-CS interface. The detailed algorithm GBP-CS is presented in Section V.

Local Training. In each group m , each selected device k fetches a mini-batch of data $\mathcal{D}_t^{m,k}$ from local dataset $\mathcal{D}^{m,k}$ with batch size $n^{m,k}$. These mini-batch streaming data are one-shot and will not be used again. Then, the device k downloads the model $\omega_{t-1}^{m,k} \leftarrow \omega_{t-1}^m$ from the connected BS m and trains $\omega_t^{m,k}$ for one mini-batch gradient descent step with learning rate η ,

$$\omega_t^{m,k} \leftarrow \omega_{t-1}^{m,k} - \frac{\eta}{n^{m,k}} \nabla_{\omega} \mathcal{L} \left(\omega_{t-1}^{m,k}, \mathcal{D}_t^{m,k} \right). \quad (3)$$

Internal Synchronization. The locally trained model $\omega_t^{m,k}$ will be uploaded to BS m for internal synchronization,

$$\omega_t^m \leftarrow \sum_{k \in \mathcal{C}_t^m} \frac{n^{m,k}}{n^m} \omega_t^{m,k}, \quad (4)$$

where $n^m = \sum_{k \in \mathcal{C}_t^m} n^{m,k}$ is the total data size of all used mini-batches in \mathcal{C}_t^m . Then, the BS m updates its model with ω_t^m and synchronizes ω_t^m in its group.

We call the above client selection, local training, and internal synchronization as a *one-step synchronization iteration* because the local update on each device is only performed once before each synchronization. The one-step synchronization will loop T iterations before each round of external synchronization.

Algorithm 1 Federated-Group-Synchronization (Main)

Input: Number of iterations in each round T ; Maximum training rounds R ; Number of groups M ; Number of selected devices per group L .

Output: Well-trained global FL model ω_{TR} .

- 1: Initialize ω_0 and $\omega_0^m \leftarrow \omega_0$ and estimate $\mathcal{P}^{\text{real}}$ by Eq. (2);
 - 2: **for** each internal synchronization $t = 1, \dots, TR$ **do**
 - 3: **for** each BS m in $1, \dots, M$ in parallel **do**
 - 4: **Client Selection:** Select L devices from group \mathcal{C}^m to form a homogeneous super node \mathcal{C}_t^m : $\mathcal{C}_t^m \leftarrow \text{Select-Clients-Via-GBP-CS}(L, \mathcal{C}^m, \mathcal{P}^{\text{real}})$;
 - 5: **for** each device k in \mathcal{C}_t^m in parallel **do**
 - 6: **Local Training:** Fetch a mini-batch of data $\mathcal{D}_t^{m,k}$ and update the local model $\omega_t^{m,k}$ by Eq. (3);
 - 7: **end for**
 - 8: **Internal Synchronization:** BS m aggregates local models $\{\omega_t^{m,k} | \forall k \in \mathcal{C}_t^m\}$ by Eq. (4) and update ω_t^m ;
 - 9: **if** $t \% T == 0$ **then**
 - 10: **External Synchronization:** The top server globally aggregates $\{\omega_t^m | \forall m\}$ by Eq. (5), and synchronizes the updated global model ω_t to BSs: $\omega_t^m \leftarrow \omega_t$;
 - 11: **end if**
 - 12: **end for**
 - 13: **end for**
 - 14: **return** ω_{TR} ;
-

External Synchronization. For every time that the one-step synchronization is performed T iterations, BSs can upload their model ω_t^m to the top server for global aggregation,

$$\omega_t \leftarrow \frac{1}{M} \sum_{m \in M} \omega_t^m. \quad (5)$$

The globally aggregated model ω_t will be used to update the global model on the top server and synchronized to BSs.

Since the external synchronization is performed every T one-step synchronization iterations, we call it a *multi-step synchronization round*. The above steps will be repeated R rounds (i.e., TR iterations) to obtain the converged model parameters $\omega^* \leftarrow \omega_{TR}$.

The above workflow can be seen as an equivalent version of FedAvg, which performs local updates on FL super nodes for T iterations with larger batch sizes, but homogeneous local datasets among super nodes. By capitalizing on an effective client selection strategy to make these super nodes homogeneous, the FL training process can be robust against data heterogeneity, and FL model performance can be improved. In the following section, we give our solution GBP-CS.

V. CLIENT SELECTION VIA GBP-CS

In this section, we formulate the client selection problem as a 0-1 integer programming problem with vector weight constraints, and present our novel Gradient-based Binary Permutation approach, namely GBP-CS, to solve this problem in an acceptable short time, with a desirable solution.

A. Problem Modeling

Given a factory (i.e., group) m with K^m industrial devices \mathcal{C}^m ($|\mathcal{C}^m| = K^m$). The next batch data of device k follows the distribution $\mathcal{P}_t^{m,k}$ ($\neq \mathcal{P}^{\text{real}}$) and the data size vector of F label classes is $\mathbf{a}_t^{m,k} = n^{m,k} \mathcal{P}_t^{m,k} \in \mathbb{Z}^{F \times 1}$. Our goal is to select L devices from group m at iteration t to form a FL super node \mathcal{C}_t^m whose overall class distribution \mathcal{P}_t^m satisfies,

$$\min_{\mathcal{P}_t^m \triangleq \mathbb{E}[\{\mathcal{P}_t^{m,k} | \forall k \in \mathcal{C}_t^m\}]} \|\mathcal{P}_t^m - \mathcal{P}^{\text{real}}\|_{L_2}. \quad (6)$$

Note that if $\mathcal{P}_t^{m,k}$ is fixed, Eq. (6) will always find a fixed device set \mathcal{C}_t^m and other devices $\mathcal{C}^m \setminus \mathcal{C}_t^m$ will have no chance to be selected. To keep the randomness of client selection so that each device has the same probability to be selected, we use a trick to randomly pre-sample L_{rnd} devices before strategically selecting the remaining $L_{\text{sel}} = L - L_{\text{rnd}}$ devices. Formally speaking, in group m , we first sample L_{rnd} devices at random to obtain $\mathcal{C}_{\text{rnd}}^m$, whose next data batches have a total data size of vector $\mathbf{b}_t^m \in \mathbb{Z}^{F \times 1}$. Then, L_{sel} devices are further selected from the remaining $K^m - L_{\text{rnd}}$ devices $\mathcal{C}^m \setminus \mathcal{C}_{\text{rnd}}^m$, whose data size matrix is $\mathbf{A}_t^m = [\mathbf{a}_t^{m,1}, \mathbf{a}_t^{m,2}, \dots, \mathbf{a}_t^{m,K^m - L_{\text{rnd}}}] \in \mathbb{Z}^{F \times (K^m - L_{\text{rnd}})}$, with the goal to minimize Eq. (6).

We use the following mathematical model to describe the above problem. Let $\mathbf{e}_{K^m - L_{\text{rnd}}}^T \in 1^{1 \times (K^m - L_{\text{rnd}})}$ and $\mathbf{e}_F^T \in 1^{1 \times F}$, the objective is to find a solution $\mathbf{x}_t^m \in \mathbb{Z}^{(K^m - L_{\text{rnd}}) \times 1}$, where $\mathbf{x}_t^m(i) \in \{0, 1\}$ and $\mathbf{e}_{K^m - L_{\text{rnd}}}^T \cdot \mathbf{x}_t^m = L_{\text{sel}}$, that

$$\min_{\mathbf{x}_t^m} \left\| \frac{\mathbf{A}_t^m \mathbf{x}_t^m + \mathbf{b}_t^m}{\mathbf{e}_F^T (\mathbf{A}_t^m \mathbf{x}_t^m + \mathbf{b}_t^m)} - \mathcal{P}^{\text{real}} \right\|_{L_2}, \quad (7)$$

$$\text{s.t. } \mathbf{x}_t^m(i) \in \{0, 1\}, \quad (8)$$

$$\mathbf{e}_{K^m - L_{\text{rnd}}}^T \cdot \mathbf{x}_t^m = L_{\text{sel}}. \quad (9)$$

Let batch sizes of all data batches be the same $n = n^{m,k}$ ($\forall m, k$), then we have $\mathbf{e}_F^T (\mathbf{A}_t^m \mathbf{x}_t^m + \mathbf{b}_t^m) = nL$ and a simplified model,

$$\min_{\mathbf{x}_t^m} \|\mathbf{A}_t^m \mathbf{x}_t^m - \mathbf{y}_t^m\|_{L_2}, \quad (10)$$

$$\text{s.t. } \mathbf{y}_t^m = nL \mathcal{P}^{\text{real}} - \mathbf{b}_t^m, \quad (11)$$

$$\mathbf{x}_t^m(i) \in \{0, 1\}, \quad (12)$$

$$\mathbf{e}_{K^m - L_{\text{rnd}}}^T \cdot \mathbf{x}_t^m = L_{\text{sel}}. \quad (13)$$

Note that in the mathematical model above, we considered data size and data quality in the objective (i.e., minimizing the distribution divergence) for client selection, but assumed that IIoT devices have similar hardware capabilities. However, if system heterogeneity should be considered, ESync [43] is compatible and can be useful. Then, we give a simple proof to show that the above problem is NP-complete.

Lemma 1 (Problem A): Given an integer matrix \mathbf{A} and an integer vector \mathbf{y} , the goal is to find whether there is a 0-1 vector \mathbf{x} such that $\mathbf{A}\mathbf{x} = \mathbf{y}$. This 0-1 integer programming problem is NP-complete [44].

Proposition 1 (Problem B): Let Lemma 1 hold and constrain the number of 1 in \mathbf{x} to be L_{sel} (Eq. (13)). This variant problem is at least NP-complete.

Proof 1: To solve problem A, we can solve problem B for K times with $L_{\text{sel}} = 1, 2, \dots, K$, where K is the size of

Algorithm 2 Select-Clients-Via-GBP-CS

Input: Number of selected devices per group L ; Device set \mathcal{C}^m of group m ; Global class distribution $\mathcal{P}^{\text{real}}$.

Output: L selected devices \mathcal{C}_t^m .

- 1: Pre-sample L_{rnd} devices $\mathcal{C}_{\text{rnd}}^m$ at random, and construct \mathbf{b}_t^m from $\mathcal{C}_{\text{rnd}}^m$ and \mathbf{A}_t^m from $\mathcal{C}^m \setminus \mathcal{C}_{\text{rnd}}^m$;
- 2: Initialize $s \leftarrow 1$, \mathbf{y}_t^m by Eq. (11), and \mathbf{x}_1 by Eq. (14);
- 3: Calculate the distance $d_s = \|\mathbf{A}_t^m \mathbf{x}_s - \mathbf{y}_t^m\|_{L_2}$;
- 4: **repeat**
- 5: Calculate the gradient $\mathbf{g}_s = \nabla_{\mathbf{x}} d_s$;
- 6: Select an index pair $(i_{0 \rightarrow 1}, i_{1 \rightarrow 0})$ by Eq. (15)-(16);
- 7: Make a copy $\mathbf{x}_{s+1} \leftarrow \mathbf{x}_s$ and permute $\mathbf{x}_{s+1}(i_{0 \rightarrow 1})$ and $\mathbf{x}_{s+1}(i_{1 \rightarrow 0})$ by Eq. (17);
- 8: Update the distance $d_{s+1} = \|\mathbf{A}_t^m \mathbf{x}_{s+1} - \mathbf{y}_t^m\|_{L_2}$;
- 9: $s \leftarrow s + 1$;
- 10: **until** $d_s > d_{s-1}$;
- 11: Construct the set of L selected devices $\mathcal{C}_t^m = \mathcal{C}_{\text{rnd}}^m \cup \mathcal{C}_{\text{sel}}^m$, where $\mathcal{C}_{\text{sel}}^m$ is defined as Eq. (18);
- 12: **return** \mathcal{C}_t^m ;

vector \mathbf{x} . This input transformation has a linear complexity $\mathcal{O}(K)$. Problem B outputs YES if Eq. (10) could reach 0, otherwise, it outputs NO. This output transformation has a constant complexity $\mathcal{O}(1)$. Hence, problem A can reduce to problem B in a polynomial complexity, which makes problem B also NP-complete.

We can see from Proposition 1 that it is almost impossible to find the optimal solution in a polynomial complexity. To make FEDGS time efficient, a sub-optimal but fast solution is preferred, as described in the following subsection.

B. Gradient-based Binary Permutation Client Selection

To make the NP-complete client selection problem solvable, in this paper, we propose a novel gradient-based approximate approach, namely GBP-CS. The core idea is to permute (0,1) pairs of binary selection variables in \mathbf{x} with the steepest opposite gradients. In other words, the variable $\mathbf{x}(i)$ with the selection value of 0 and the smallest gradient will be permuted with the variable $\mathbf{x}(j)$ with the selection value of 1 and the largest gradient. In this way, the number of variables whose selection value equals one (i.e., the vector weight constraint) is maintained, and constraints (12) and (13) can be satisfied. This binary permutation operation will be performed iteratively to minimize the objective Eq. (10).

The pseudo code of GBP-CS is given in Alg. 2. Given the data size matrix \mathbf{A}_t^m and $\mathbf{y}_t^m = nL \mathcal{P}^{\text{real}} - \mathbf{b}_t^m$, GBP-CS first initializes the solution variable \mathbf{x}_1 as follows,

$$\mathbf{x}_1 \triangleq \left\{ \mathcal{T}_{L_{\text{sel}}}(\tilde{\mathbf{x}}_1) | \tilde{\mathbf{x}}_1 = (\mathbf{A}_t^m)^{-1} \mathbf{y}_t^m \right\}, \quad (14)$$

where $(\mathbf{A}_t^m)^{-1} \mathbf{y}_t^m$ is the Moore-Penrose Inverse solution, $\mathcal{T}_{L_{\text{sel}}}(\tilde{\mathbf{x}}_1)$ means to set the largest L_{sel} values of $\tilde{\mathbf{x}}_1$ to 1, and the others to 0. Then, GBP-CS calculates the objective distance $d_s = \|\mathbf{A}_t^m \mathbf{x}_s - \mathbf{y}_t^m\|_{L_2}$ and the gradient $\mathbf{g}_s = \nabla_{\mathbf{x}} d_s$.

The gradient $\mathbf{g}_s(i)$ indicates the opposite direction in which $\mathbf{x}_s(i)$ should be updated. The greater the absolute value of

$\mathbf{g}_s(i)$, the smaller the d_s can be obtained by updating $\mathbf{x}_s(i)$, so $\mathbf{g}_s(i)$ appears as a key gradient [45]. Based on this idea, GBP-CS selects a pair of selection variables ($\mathbf{x}_s(i_{0 \rightarrow 1}), \mathbf{x}_s(i_{1 \rightarrow 0})$) with opposite key gradients for permutation. More specifically, $i_{0 \rightarrow 1}$ is the identity of the device with the selection value of 0 and the smallest gradient,

$$i_{0 \rightarrow 1} \triangleq \arg \min_i \{ \mathbf{g}_s(i) | \mathbf{x}_s(i) = 0, \forall i \in [1, K^m - L_{\text{rnd}}] \}. \quad (15)$$

Similarly, $i_{1 \rightarrow 0}$ is the identity of the device with the selection value of 1 and the largest gradient,

$$i_{1 \rightarrow 0} \triangleq \arg \max_i \{ \mathbf{g}_s(i) | \mathbf{x}_s(i) = 1, \forall i \in [1, K^m - L_{\text{rnd}}] \}. \quad (16)$$

Then, GBP-CS permutes the values of $\mathbf{x}_s(i_{0 \rightarrow 1})$ and $\mathbf{x}_s(i_{1 \rightarrow 0})$ to obtain a new solution \mathbf{x}_{s+1} ,

$$\mathbf{x}_{s+1}(i_{0 \rightarrow 1}) = 1, \quad \mathbf{x}_{s+1}(i_{1 \rightarrow 0}) = 0. \quad (17)$$

Eqs. (15)-(17) will be repeated until the objective distance d_s no longer decreases. Finally, we can construct

$$\mathcal{C}_{\text{sel}}^m \triangleq \{ \text{device } i | \mathbf{x}^*(i) = 1, \mathbf{x}^* = \mathbf{x}_{s-1}, \forall i \in [1, K^m - L_{\text{rnd}}] \}, \quad (18)$$

and obtain the set of L selected devices $\mathcal{C}_t^m = \mathcal{C}_{\text{rnd}}^m \cup \mathcal{C}_{\text{sel}}^m$.

GBP-CS has a complexity of $\mathcal{O}(F^3 + \alpha F^2 + \alpha^2 F \tau + L)$, where $\alpha = K^m - L_{\text{rnd}}$ and τ is the number of GBP-CS iterations. In our experiment, GBP-CS can obtain a very desirable solution close to the optimal and has a high execution efficiency comparable to the random sampling approach.

VI. PERFORMANCE ANALYSIS

In this section, we analyze the optimality gap and convergence rate of FEDGS, and qualitatively compare them with those of FedAvg in the presence of non-i.i.d. data. Then, we give the condition under which FEDGS is time efficient.

A. Convergence Analysis

Assumption 1: The local function \mathcal{L} are μ -strongly convex, β -smooth, and ρ -Lipschitz.

Proposition 2: For the gradient \mathbf{g}_t^m on FL node m in the federated setting and the gradient \mathbf{g}_t^c in the centralized setting, we have an upper bound δ^m of the gradient divergence $\|(\mathcal{P}_t^m)^T \cdot \mathbf{g}_t^m - (\mathcal{P}^{\text{real}})^T \cdot \mathbf{g}_t^c\|$ proportional to the distribution divergence $\|\mathcal{P}_t^m - \mathcal{P}^{\text{real}}\|$.

Proof 2:

$$\begin{aligned} & \left\| (\mathcal{P}_t^m)^T \cdot \mathbf{g}_t^m - (\mathcal{P}^{\text{real}})^T \cdot \mathbf{g}_t^c \right\| \\ &= \left\| \begin{aligned} & (\mathcal{P}_t^m)^T \cdot \mathbf{g}_t^m - (\mathcal{P}_t^m)^T \cdot \mathbf{g}_t^c \\ & + (\mathcal{P}_t^m)^T \cdot \mathbf{g}_t^c - (\mathcal{P}^{\text{real}})^T \cdot \mathbf{g}_t^c \end{aligned} \right\| \\ &\leq \left\| (\mathcal{P}_t^m)^T \cdot (\mathbf{g}_t^m - \mathbf{g}_t^c) \right\| + \left\| (\mathcal{P}_t^m - \mathcal{P}^{\text{real}})^T \cdot \mathbf{g}_t^c \right\| = \delta^m. \end{aligned}$$

It is easy to know that δ^m captures the impact of divergence in class distributions $\|\mathcal{P}_t^m - \mathcal{P}^{\text{real}}\|$. Generally speaking, the smaller the distribution divergence, the smaller the upper bound of the gradient divergence δ^m .

Proposition 3: Let $\delta \triangleq \mathbb{E}[\delta^m]$, the convergence upper bound of FEDGS is $\mathcal{O}\left(\frac{1}{R(T - \delta h(T))}\right)$, and its optimality gap is bounded by $\mathcal{O}\left(\frac{1}{TR} + \delta h(T) + o\left(\sqrt{\frac{\delta h(T)}{T}}\right)\right)$.

Proof 3: As mentioned above, the convergence performance of FEDGS is theoretically equivalent to that of FedAvg, in which M FL super nodes run mini-batch SGD with batch size nL for T local iterations in each round. Then, the convergence upper bound of FEDGS after TR iterations can be inferred from Lemma 2 in [34],

$$\mathcal{L}(\omega_{TR}) - \mathcal{L}(\omega^*) \leq \frac{1}{TR \left(\eta \varphi - \frac{\rho \delta h(T)}{T \varepsilon^2} \right)},$$

where $h(T) \triangleq \frac{1}{\beta}((\eta\beta + 1)^T - 1) - \eta T$. When $\eta \leq \frac{1}{\beta}$, the optimality gap $G = \mathcal{L}(\omega^{(f)}) - \mathcal{L}(\omega^*)$ is bounded by

$$\begin{aligned} G &\leq \frac{1}{2\eta\varphi TR} + \rho\delta h(T) + \sqrt{\frac{1}{4\eta^2\varphi^2 T^2 R^2} + \frac{\rho\delta h(T)}{\eta\varphi T}} \\ &\leq \frac{1}{\eta\varphi TR} + \rho\delta h(T) + \sqrt{\frac{\rho\delta h(T)}{\eta\varphi T}}. \end{aligned}$$

Since GBP-CS forces FL super nodes to have aligned class distributions, FEDGS has an upper bound of the gradient divergence smaller than FedAvg $\delta_{\text{FEDGS}} < \delta_{\text{FedAvg}}$. Therefore, it is easy to infer from Proposition 3 that, FEDGS has both the convergence upper bound and the optimality gap smaller than those of FedAvg, thus it can improve the FL convergence speed and accuracy performance.

B. Time-Efficiency Condition

We analyze the time cost of FEDGS in each round (T one-step synchronization iterations) and that of FedAvg in each round (T local iterations), and give the condition under which FEDGS can achieve higher time efficiency than FedAvg.

Time Cost of FEDGS. The time cost of FEDGS T_{FEDGS} is determined by communication, computation and client selection. The communication delays are brought by internal synchronizations $T_{\text{comm}}^{\text{int}}$ and external synchronizations $T_{\text{comm}}^{\text{ext}}$. For the internal synchronization, the delay of uploading local models of size S from L devices to their BS is $\frac{SL}{B_{\text{up}}^{\text{int}} \log_2(1 + \gamma_{\text{BS}})}$, and the delay of synchronizing the model of size S from the BS to L devices is $\frac{SL}{B_{\text{down}}^{\text{int}} \log_2(1 + \gamma_{\text{device}})}$, where $B_{\text{up}}^{\text{int}}$ and $B_{\text{down}}^{\text{int}}$ are the uplink and downlink bandwidths between devices and BS, γ_{BS} and γ_{device} are the received signal-to-noise ratio (SNR) of BS and devices. For the external synchronization, the delay of uploading models of size S from M BSs to the top server is $\frac{SM}{B_{\text{up}}^{\text{ext}} \log_2(1 + \gamma_{\text{top}})}$, and the delay of synchronizing the global model of size S from the top server to M BSs is $\frac{SM}{B_{\text{down}}^{\text{ext}} \log_2(1 + \gamma_{\text{BS}})}$, where $B_{\text{up}}^{\text{ext}}$ and $B_{\text{down}}^{\text{ext}}$ are the uplink and downlink bandwidths between BSs and the top server, γ_{top} is the SNR of the top server. Hence, we have the communication

time cost for each internal synchronization and each external synchronization as follows,

$$T_{\text{comm}}^{\text{ext}} = \frac{SM}{B_{\text{up}}^{\text{ext}} \log_2(1 + \gamma_{\text{top}})} + \frac{SM}{B_{\text{down}}^{\text{ext}} \log_2(1 + \gamma_{\text{BS}})}, \quad (19)$$

$$T_{\text{comm}}^{\text{int}} = \frac{SL}{B_{\text{up}}^{\text{int}} \log_2(1 + \gamma_{\text{BS}})} + \frac{SL}{B_{\text{down}}^{\text{int}} \log_2(1 + \gamma_{\text{device}})}. \quad (20)$$

Let the delay of each local update be T_{comp} and the delay of client selection procedure be T_{select} . Each round the internal synchronization is performed T times, the total delay is

$$T_{\text{FEDGS}} = T_{\text{comm}}^{\text{ext}} + T \cdot (T_{\text{select}} + T_{\text{comm}}^{\text{int}} + T_{\text{comp}}). \quad (21)$$

Time Cost of FedAvg. The time cost of FedAvg T_{FedAvg} is mainly determined by communication and computation because the client selection procedure is simple random sampling so that the selection delay is negligible. The communication delays come from the uplink and downlink model transmission between the top server and devices. The delay of uploading local models of size S from ML devices to the top server is $\frac{SML}{B_{\text{up}}^{\text{ext}} \log_2(1 + \gamma_{\text{top}})}$, and the delay of synchronizing the global model of size S from the top server to ML devices is $\frac{SML}{B_{\text{down}}^{\text{ext}} \log_2(1 + \gamma_{\text{device}})}$. Hence, we have the communication time cost for each round of synchronization as follows,

$$\tilde{T}_{\text{comm}}^{\text{ext}} = \frac{SML}{B_{\text{up}}^{\text{ext}} \log_2(1 + \gamma_{\text{top}})} + \frac{SML}{B_{\text{down}}^{\text{ext}} \log_2(1 + \gamma_{\text{device}})}. \quad (22)$$

Then, the total time cost when FedAvg performs T local updates and one round of synchronization is

$$T_{\text{FedAvg}} = \tilde{T}_{\text{comm}}^{\text{ext}} + T \cdot T_{\text{comp}}. \quad (23)$$

In order to simplify the analysis result, we make the following assumptions.

Assumption 2: (a) The uplink and downlink bandwidths are equal: $B_{\text{up}}^{\text{ext}} = B_{\text{down}}^{\text{ext}} = B^{\text{ext}}$, $B_{\text{up}}^{\text{int}} = B_{\text{down}}^{\text{int}} = B^{\text{int}}$; (b) The SNRs of the top server, BSs, and devices are equal: $\gamma_{\text{top}} = \gamma_{\text{BS}} = \gamma_{\text{device}} = \gamma$.

Then, we can give the following condition for hyperparameter setting, under which FEDGS can achieve higher time efficiency than FedAvg.

Proposition 4: Let Assumption 2 hold and $\beta = \log_2(1 + \gamma)$, the time cost per T iterations in each round satisfies $T_{\text{FEDGS}} < T_{\text{FedAvg}}$ if $\frac{TL}{M(L-1)} < \frac{B^{\text{int}}}{B^{\text{ext}}}$, where

$$T_{\text{FEDGS}} = \frac{2SM}{\beta B^{\text{ext}}} + T \left(T_{\text{select}} + \frac{2SL}{\beta B^{\text{int}}} + T_{\text{comp}} \right), \quad (24)$$

$$T_{\text{FedAvg}} = \frac{2SML}{\beta B^{\text{ext}}} + TT_{\text{comp}}. \quad (25)$$

Proof 4: Eq. (24) can be obtained by combining Eqs. (19)-(21), and Eq. (25) can be obtained by combining Eqs. (22)-(23). Let $T_{\text{FEDGS}} - T_{\text{FedAvg}} < 0$, we have

$$\begin{aligned} & \frac{2SM(1-L)}{\beta B^{\text{ext}}} + \frac{2TSL}{\beta B^{\text{int}}} + TT_{\text{select}} < 0 \\ \Rightarrow & \frac{B^{\text{ext}}}{B^{\text{int}}} SL + T_{\text{select}} \cdot \frac{\beta B^{\text{ext}}}{2} < \frac{SM(L-1)}{T}. \end{aligned}$$

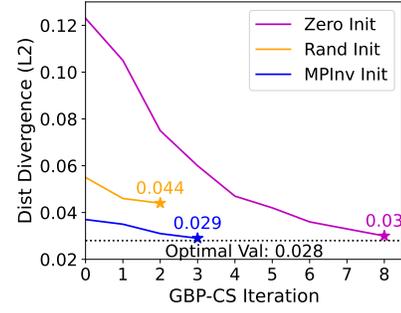


Fig. 3: Distribution divergence optimization curves of GBP-CS with different initializers.

In our experiment, GBP-CS is quite fast, whose time cost (15 milliseconds) is negligible compared to other delays. Therefore, we assume $T_{\text{select}} \approx 0$ to simplify the result and obtain

$$\frac{B^{\text{ext}}}{B^{\text{int}}} SL < \frac{SM(L-1)}{T} \Rightarrow \frac{TL}{M(L-1)} < \frac{B^{\text{int}}}{B^{\text{ext}}}.$$

In modern industrial applications, 5G enables indoor industrial use cases that were impossible before, supported by high data rate, ultra-low delay, and extreme density of wireless communications [46]. In reality, the data rate of 5G edge is about 10-100 times of that in WAN, that is, $B^{\text{int}}/B^{\text{ext}} \in [10, 100]$. Therefore, we can easily set T, M, L to satisfy the condition in Proposition 4, so that the time efficiency of FEDGS can be guaranteed.

VII. EXPERIMENTAL EVALUATION

A. Experiment Setup

Environment and Hyperparameter Setup. In the experiment, we consider an IIoT application where OCR technology is used in the identification of packing boxes, machines, robots, vehicles, and workers, through recognizing the optical characters on their badges. To this end, we aim to train a high-accuracy OCR model in the federated setting, where sensors' local character images are confidential and skewed in the class distribution. The real-world FEMNIST [25] dataset is chosen to train our federated OCR model, as it is built by partitioning 805,263 optical digit and character images into 3,550 devices, following non-i.i.d.-like class distributions and uneven data sizes. Our experiment platform contains $K = 350$ OCR cameras and $M = 10$ factories, each factory m has $K^m = 35$ OCR cameras (hereinafter referred to as devices). In each iteration, $L = 10$ devices are selected from each factory to participate in the federated OCR training. A four-layer convolutional neural network [Conv2D(32), MaxPool, Conv2D(64), MaxPool, Dense(2048), Dense(62)] is used as the training model because it is lightweight and suitable for resource-constrained industrial devices. Unless otherwise specified, we use the standard mini-batch SGD to train local ML models, with the learning rate $\eta = 0.01$, the batch size $n = 32$, the number of iterations per round $T = 50$ and the maximum number of rounds $R = 500$.

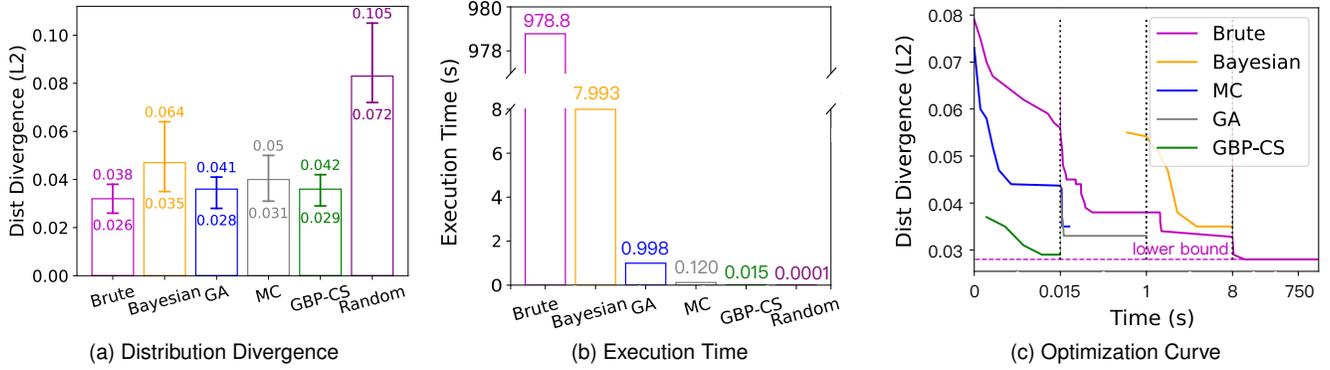


Fig. 4: Comparison of (a) distribution divergence, (b) execution time, and (c) optimization curve among different samplers.

GBP-CS Initialization. The choice of the initial point \mathbf{x}_1 in GBP-CS is critical to the quality of the solution, because a bad initial point may cause GBP-CS to fall into a local minimum. In the experiment, $L_{\text{rnd}} = 2$ devices are pre-sampled at random, and the other $L_{\text{sel}} = L - L_{\text{rnd}} = 8$ devices are selected using GBP-CS, with the following three initialization methods.

- 1) *Random Initialization.* Set L_{sel} values in \mathbf{x}_1 to 1 at random and leave other values at 0.
- 2) *Zero Initialization.* All values in \mathbf{x}_1 are first initialized to 0. Then, a warm-up step is performed to meet the vector weight constraint Eq. (13), in which one value $\mathbf{x}_1(i)$ with the smallest gradient is set to 1 iteratively until the number of value 1 in \mathbf{x}_1 reaches L_{sel} . The warm-up step requires additional L_{sel} iterations.
- 3) *Moore-Penrose Inverse Initialization (MPIInv).* MPIInv is first used to solve the least square solution $\tilde{\mathbf{x}}_1 = \mathbf{A}^{-1}\mathbf{y}$ of the unconstrained objective function $\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{y}\|_{L_2}$. Then, L_{sel} elements with the largest values in $\tilde{\mathbf{x}}_1$ are set to 1 and others are left at 0 to obtain the initial point \mathbf{x}_1 .

Comparison Algorithms. To highlight the efficiency and effectiveness of the proposed GBP-CS, we consider the following five benchmark client selection methods for comparison.

- 1) *Random Sampler (Random):* From each group, L_{sel} devices are uniformly and randomly sampled.
- 2) *Monte Carlo Sampler (MC):* Repeat the random sampler 1000 times and the solution minimizes Eq. (10) is used.
- 3) *Brute Sampler (Brute):* Brutely search for the optimal L_{sel} devices by traversing all feasible solutions to meet Eqs. (10)-(13).
- 4) *Bayesian Sampler (Bayesian):* Search for a sub-optimal L_{sel} devices using Bayesian optimization [47] to meet Eqs. (10)-(13). By default, we set the number of initial points to 5 and exploration iterations to 25.
- 5) *Genetic Sampler (GA):* Search for a sub-optimal L_{sel} devices using genetic algorithm [48] to meet Eqs. (10)-(13), in which the constrained 0-1 vector solutions are regarded as genes and suffer from selection, crossover, mutation and elimination. By default, we set the population size to 100, the mutation probability to 0.001, and

the number of generations to 100.

Except for the baseline FedAvg [24], other nine advanced approaches are also experimentally compared with FEDGS in the presence of non-i.i.d. data. They are FedMMD [26], FedFusion [27], FedProx [28], IDA [29], CGAU [30], FedAvgM [31], and FedAdagrad, FedAdam, FedYogi from [32].

Implementation. We implement FEDGS on a standard FL simulator Leaf-MX¹ (an MXNET [49] implementation of LEAF [25]). The code implementation is open-available on Github: <https://github.com/Lizonghang/fedgs>.

B. Results and Discussion

Comparison of initialization methods in GBP-CS. The optimization curves of the class distribution divergence of Zero, Random, and MPIInv initializers are shown in Fig. 3. Both Zero and MPIInv initializers successfully find high-quality solutions (0.029 and 0.030, respectively) close to the optimal of the brute force search (0.028). Instead, the Random initializer falls into a poor local optimal (0.044). Furthermore, the MPIInv initializer is much faster because it does not require an additional warm-up procedure like the Zero initializer. Therefore, GBP-CS is default initialized with MPIInv initializer.

Comparison among GBP-CS and other samplers. Since the procedure of GBP-CS client selection is performed every iteration, both the quality and time cost of the solution are critical to FEDGS performance.

We first compare the distribution divergence (defined as Eq. (6)) among GBP-CS and other five benchmark samplers. Generally speaking, the smaller the gap between the class distribution \mathcal{P}_t^m of the group m and the global distribution $\mathcal{P}^{\text{real}}$, the smaller the distribution divergence and the better the sampler. The distribution divergence of $M = 10$ factories is shown in Fig. 4a. As expected, the most commonly used random sampler in FedAvg leads to a high divergence in class distribution (0.072 ~ 0.105) and causes non-i.i.d. data among groups, while the brute force sampler can always minimize the divergence (0.026 ~ 0.038). The random sampler and the brute force sampler give the upper and lower bounds of the distribution divergence, and solutions of other samplers should

¹Leaf-MX: <https://github.com/Lizonghang/leaf-mx>

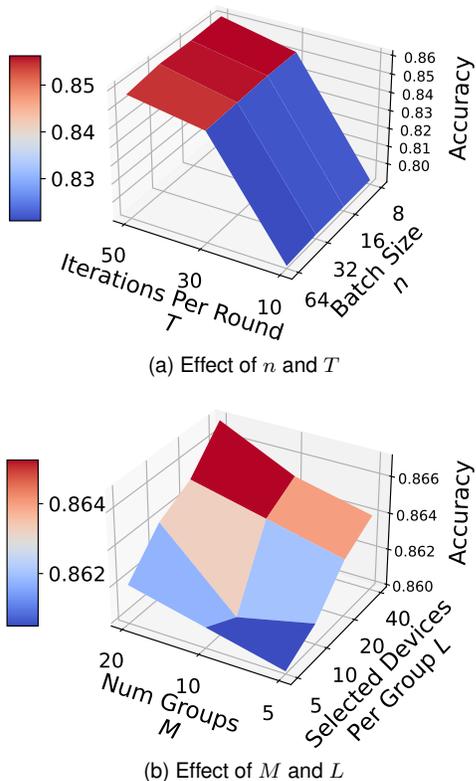


Fig. 5: Accuracy surface of FEDGS over different (a) batch size n and iterations per round T ; (b) number of groups M and number of selected devices per group L .

locate in this interval, among which GA and GBP-CS samplers perform best (0.028 \sim 0.041 and 0.029 \sim 0.042, respectively).

In terms of execution time, FEDGS prefers samplers with a very short execution time, because high-frequency client selection may introduce a non-negligible latency to FEDGS and significantly slow down FL training. Fig. 4b compares the execution time of the above samplers. Let us focus on the brute force, GA, and GBP-CS samplers, because their solutions are of the best quality. The brute force sampler requires 979 seconds to find the optimal solution, whose latency is too long to be acceptable. Therefore, FEDGS prefers a sub-optimal solution in an acceptable short time. GA and GBP-CS samplers seem to be good choices, and the proposed GBP-CS sampler is 66 \times faster than the GA sampler, with a negligible 15 milliseconds and a loss of distribution divergence by only 0.001.

To highlight GBP-CS more intuitively, we draw the optimization curve of distribution divergence over execution time in Fig. 4c. The results show that the proposed GBP-CS sampler converges to a high-quality solution 0.029 closest to the optimal 0.028 in the shortest time, demonstrating the superior effectiveness and efficiency of GBP-CS.

Effects of hyperparameters in FedGS. Hyperparameters may have great effects on FEDGS. To explore these effects, we perform a grid search on experimental hyperparameters, including the batch size n , the number of iterations per round T , the number of devices selected per group L , as well as the environmental hyperparameter, the number of groups M . Fig.

TABLE II: Test accuracy, test loss and convergence speed of FEDGS vs ten federated approaches.

	Test Accuracy	Test Loss	Rounds To 82%
FedAvg (Baseline)	82.1%	0.587	478
FedProx	82.0%	0.586	497
IDA	81.0%	0.628	\times
IDA+INTRAC	81.0%	0.618	\times
IDA+FedAvg	80.5%	0.687	\times
CGAU	83.3%	0.509	202
FedMMD	83.0%	0.564	378
FedFusion+Conv	81.7%	0.624	\times
FedFusion+Multi	82.0%	0.591	486
FedFusion+Single	80.7%	0.627	\times
FedAvgM	84.4%	0.820	68
FedAdagrad	83.8%	0.583	264
FedAdam	85.0%	0.662	71
FedYogi	84.6%	0.590	76
FEDGS	86.0%	0.435	147

5a visualizes the test accuracy over different n and T settings, where n is chosen from $\{8, 16, 32, 64\}$ and T is chosen from $\{10, 30, 50\}$. The results show that a moderately large T can improve the accuracy of FEDGS, while the batch size n has little effect. Fig. 5b visualizes the test accuracy over different M and L settings, where M is chosen from $\{5, 10, 20\}$ and L is chosen from $\{5, 10, 20, 40\}$. Without loss of generality, both more groups and more selected devices can bring gains in FL model accuracy because more devices' data is included. In this paper, $n = 32$, $T = 50$ and $L = 10$ are used by default to meet the condition in Proposition 4. Please note that $M = 10$ is determined by the real-world environment instead of an adjustable hyperparameter.

Comparison among FEDGS and other federated approaches. We take ten advanced federated approaches for comparison to show the state-of-the-art performance of the proposed FEDGS in the presence of non-i.i.d. data. The test accuracy, test loss, and training rounds required to reach the accuracy of 82% are listed in Table II, and detailed training curves are given in Fig. 6. Unless otherwise specified, all the comparison approaches use the local epoch $e = 5$ by default. In the following, we will compare these approaches and analyze their results, respectively.

FEDGS vs FedProx. FedProx adds a proximal term to local loss functions to penalize divergent local models. We tune the penalty constant $\mu = \{0.05, 0.1, 0.5, 1.0\}$ to find the best result in Figs. 6a and 6d. However, FedProx performs poorly in our case, with the accuracy of 82.0%, not even exceeding the baseline accuracy of 82.1% of FedAvg. The reason may be that the proximal penalty term will slow convergence by forcing local models closer to the starting point [28]. Instead, the proposed FEDGS improves the baseline accuracy by 3.9% and achieves the accuracy of 86.0%.

FEDGS vs IDA. IDA weighs model parameters of devices based on their inverse distance to the averaged model parameter during aggregation. We combine IDA with inverse training accuracy coefficients (IDA+INTRAC) and normalized data size coefficients (IDA+FedAvg) as suggested by the authors.

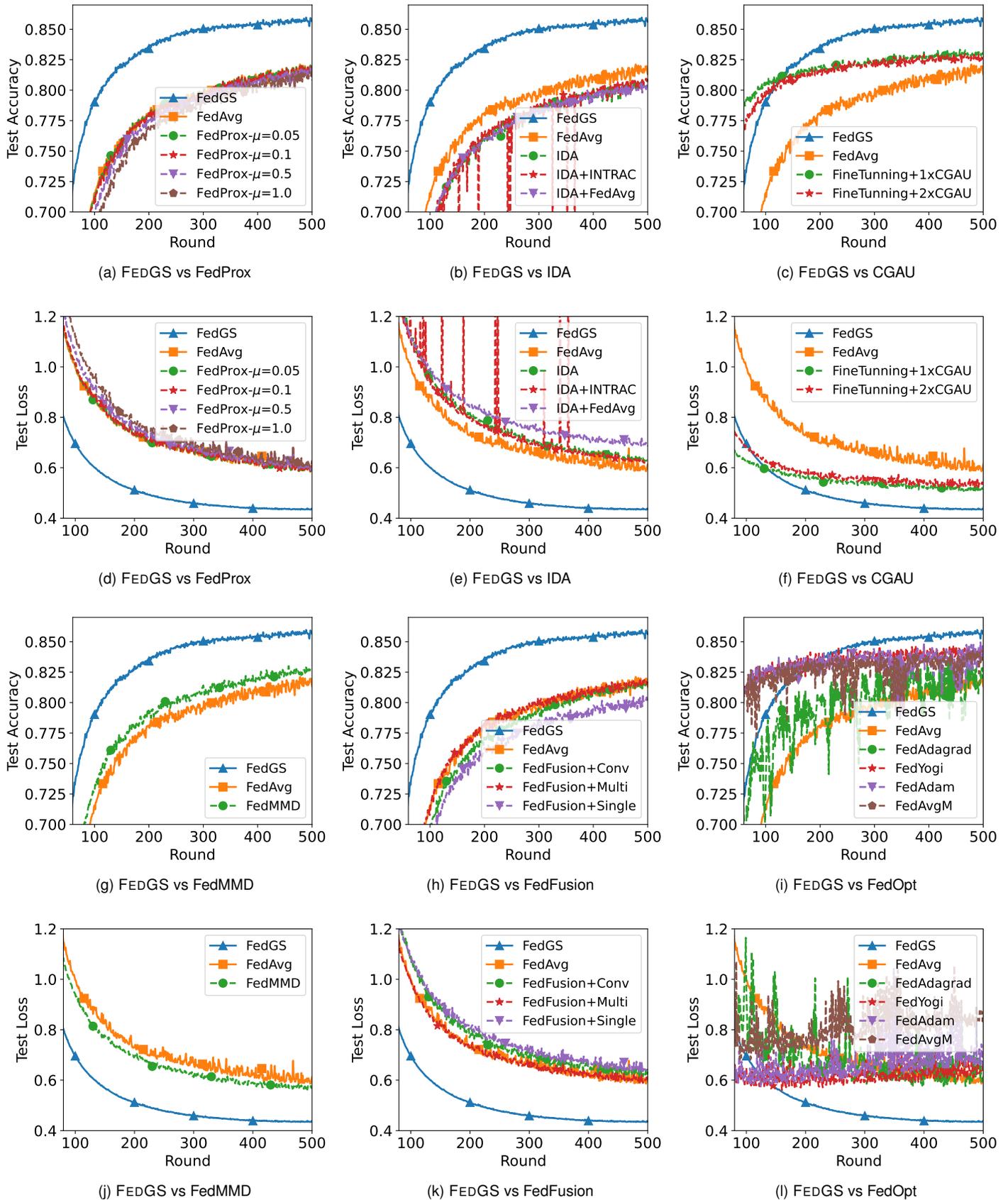


Fig. 6: Comparison of FEDGS and FedAvg, FedProx, IDA, CGAU, FedMMD, FedFusion, FedAvgM, FedAdagrad, FedAdam, FedYogi.

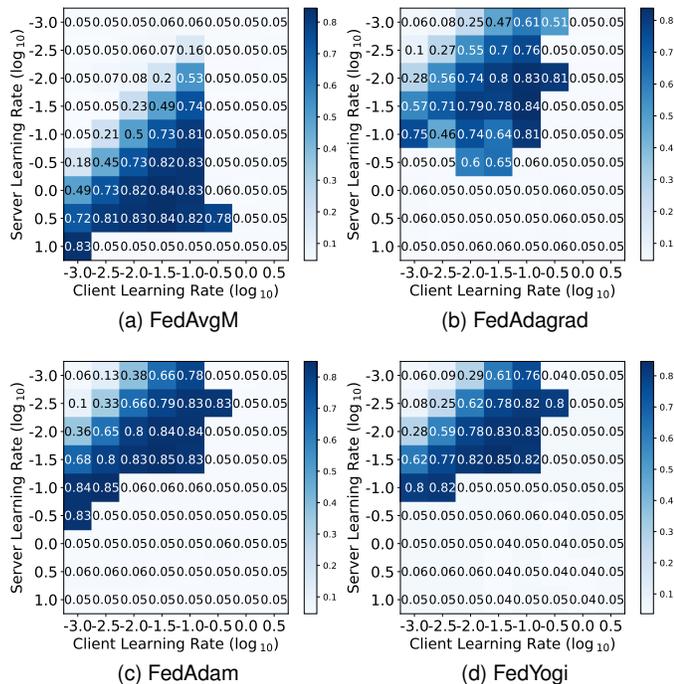


Fig. 7: Accuracy Heatmap of (a) FedAvgM, (b) FedAdagrad, (c) FedAdam and (d) FedYogi.

However, Figs. 6b and 6e shows that IDA-series approaches suffer an accuracy degradation (80.5% \sim 81.0%). That is because devices with large parameter deviations are over-suppressed, causing the global model to lose data knowledge on these devices. Besides, IDA should cache model parameters uploaded by all devices until the average model parameter and inverse distance coefficients are calculated, which takes up huge memory space on the server.

FEDGS vs CGAU. CGAU uses gated activation units on top of a pre-trained model to enable client-specific expression of heterogeneous data. We train a 1-layer and a 2-layer CGAU classifier with 256 units, respectively (namely FineTuning+1 \times CGAU and FineTuning+2 \times CGAU). The dropout layer is not used as the authors did because we observed a 3.3% drop in accuracy after using them. Figs. 6c and 6f show that FineTuning+1 \times CGAU achieves a higher accuracy of 83.3%, which improves the baseline accuracy by 1.2% and benefits from the fast convergence speed of the pre-trained model. Despite these gains, the proposed FEDGS can still achieve 2.7% higher accuracy, lower test loss, and faster convergence.

FEDGS vs FedMMD. FedMMD uses transfer learning [50] to better merge the knowledge of the global model into the local model. As suggested by the authors, we use the MMD distance and the penalty coefficient $\gamma = 0.1$. As shown in Figs. 6g and 6j, FedMMD improves the baseline accuracy by 0.9% and achieves the accuracy of 83.0%, but the proposed FEDGS further improves that by another 3%.

FEDGS vs FedFusion. FedFusion fuses the global and local features using operators such as 1 \times 1 convolution (FedFusion+Conv), vector weighted average (FedFusion+Multi) and

scalar weighted average (FedFusion+Single). However, the results in Figs. 6h and 6k show that FedFusion+Multi and FedFusion+Conv only achieve the accuracy similar to the baseline (82.0% and 81.7%, respectively), and FedFusion+Single even decreases the accuracy by 1.4%. Instead, the proposed FEDGS is obviously better and faster.

FEDGS vs FedOpt. FedOpt is a general paradigm for a series of adaptive federated optimizers, which dynamically adjusts learning rates of all gradients to accelerate convergence, including FedAvgM, FedAdagrad, FedAdam, and FedYogi. Preliminary experiments show that the convergence performance of these approaches has indeed significantly improved, but they are particularly sensitive to initial learning rates. As the authors did, we search for the best setting of the client-side and server-side initial learning rates in Fig. 7 and give the best results in Figs. 6i and 6l. Other hyperparameters follow the authors' setting, for example, $\beta = 0.9$ for FedAvgM, $\beta_1 = \beta_2 = 0$ for FedAdagrad, $\beta_1 = 0.9, \beta_2 = 0.99$ for FedAdam and FedYogi, and $\tau = 0.001$. The results show that these approaches can improve the baseline accuracy by 1.7% \sim 2.9% with fast convergence speed, especially FedAdam. However, the accuracy of the proposed FEDGS is still 1% higher.

To sum up, FedAvgM, FedAdagrad, FedAdam, and FedYogi are generally better than other comparison approaches (some of which cannot even reach the accuracy of 82%, marked with “ \times ” in Table II). Instead, FEDGS achieves the state-of-the-art accuracy of 86.0%, which is 3.9% higher than the baseline and 3.5% higher than the averaged accuracy. In addition, FEDGS can also reach the accuracy of 82% in only 147 rounds, which is 3.3 \times faster than FedAvg and reduces training rounds by 59% on average. These comprehensive experiments prove the effectiveness and efficiency of FEDGS.

VIII. CONCLUSION

FL in IIoT is emerging as a field of great value with increasing interest from both academia and industry, however, it still faces the challenge of non-i.i.d. data, which currently remains open. In this paper, we propose FEDGS, which is a hierarchical cloud-edge-end FL framework for 5G empowered modern industries. To minimize the divergence in data distributions among factories, we propose a constrained gradient-based optimizer, namely GBP-CS, to select a subset of devices in each factory to construct homogeneous FL super nodes. GBP-CS can find a desirable selection strategy in a very short time, and can also be used for other practical cases such as game matching. Then, to eliminate the impact of residual non-i.i.d. data within the super nodes, we use a compound-step synchronization protocol to coordinate the training process. This protocol uses the data heterogeneity-insensitive one-step synchronization protocol within the super nodes to suppress the negative impact of data heterogeneity, then uses the multi-step synchronization protocol among the super nodes to reduce communication frequency. The proposed approach takes into account the natural geographical clustering property of factory devices and can adapt to rapidly changing streaming data at runtime, without exposing confidential data in high-risk

data manipulation. Theoretical analysis shows that FEDGS has both the convergence rate and optimality gap better than the benchmark FedAvg, and can be more time-efficient under a relaxed hyperparameter condition. Extensive experiments compared to ten advanced approaches demonstrate the state-of-the-art performance of FEDGS on non-i.i.d. data.

REFERENCES

- [1] Chen, Fei, Bo Li, Rong Dong, *et al.*: “High-performance OCR on packing boxes in industry based on deep learning.” In: *Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, pp. 1018-1030. Springer, Cham, 2018.
- [2] Liukkonen, Mika, and Tsung-Nan Tsai: “Toward decentralized intelligence in manufacturing: Recent trends in automatic identification of things.” *International Journal of Advanced Manufacturing Technology (IJAMT)* 87, no. 9, pp. 2509-2531, 2016.
- [3] Kairouz, Peter, H. Brendan McMahan, Brendan Avent, *et al.*: “Advances and open problems in federated learning.” *Foundations and Trends in Machine Learning* 14, no. 1-2, pp. 1-210, 2021.
- [4] Hiessl, Thomas, Daniel Schall, Jana Kemnitz, *et al.*: “Industrial federated learning: Requirements and system design.” In: *International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, pp. 42-53, Springer, Cham, 2020.
- [5] Lim, Wei Yang Bryan, Nguyen Cong Luong, Dinh Thai Hoang, *et al.*: “Federated learning in mobile edge networks: A comprehensive survey.” *IEEE Communications Surveys & Tutorials (COMST)* 22, no. 3, pp. 2031-2063, 2020.
- [6] Pham, Quoc-Viet, Kapal Dev, Praveen Kumar Reddy Maddikunta, *et al.*: “Fusion of federated learning and industrial internet of things: A survey.” *arXiv preprint arXiv:2101.00798*, 2021.
- [7] Zhang, Weishan, Qinghua Lu, Qiuyu Yu, *et al.*: “Blockchain-based federated learning for device failure detection in industrial IoT.” *IEEE Internet of Things Journal (IOTJ)* 8, no. 7, pp. 5926-5937, 2020.
- [8] Luo, Jiahuan, Xueyang Wu, Yun Luo, *et al.*: “Real-world image datasets for federated learning.” *arXiv preprint arXiv:1910.11089*, 2019.
- [9] Zhao, Yue, Meng Li, Liangzhen Lai, *et al.*: “Federated learning with non-iid data.” *arXiv preprint arXiv:1806.00582*, 2018.
- [10] Yao, Xin, Tianchi Huang, Rui-Xiao Zhang, *et al.*: “Federated learning with unbiased gradient aggregation and controllable meta updating.” In: *Workshop on Federated Learning for Data Privacy and Confidentiality (FL-NeurIPS 2019, in conjunction with NeurIPS 2019)*, 2019.
- [11] Yoshida, Naoya, Takayuki Nishio, Masahiro Morikura, *et al.*: “Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data.” In: *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1-7, IEEE, 2020.
- [12] Zhang, Wenyu, Xiumin Wang, Pan Zhou, *et al.*: “Client selection for federated learning with non-iid data in mobile edge computing.” *IEEE Access* 9, pp. 24462-24474, 2021.
- [13] Zhao, Zhongyuan, Chenyuan Feng, Wei Hong, *et al.*: “Federated learning with non-iid data in wireless networks.” *IEEE Transactions on Wireless Communications (TWC)*, 2021.
- [14] Duan, Moming, Duo Liu, Xianzhang Chen, *et al.*: “Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications.” In: *IEEE 37th International Conference on Computer Design (ICCD)*, pp. 246-254, 2019.
- [15] Wang, Han, Luis Muñoz-González, David Eklund, *et al.*: “Non-iid data re-balancing at IoT edge with peer-to-peer federated learning for anomaly detection.” In: *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, pp. 153-163, 2021.
- [16] Wen, Hui, Yue Wu, Chenming Yang, *et al.*: “A unified federated learning framework for wireless communications: Towards privacy, efficiency, and security.” In: *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops*, pp. 653-658, IEEE, 2020.
- [17] Sattler, Felix, Klaus-Robert Müller, and Wojciech Samek: “Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints.” *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2020.
- [18] Wang, Hao, Zakhary Kaplan, Di Niu, *et al.*: “Optimizing federated learning on non-iid data with reinforcement learning.” In: *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1698-1707, IEEE, 2020.
- [19] Zeng, Shenglai, Zonghang Li, Hongfang Yu, *et al.*: “Heterogeneous federated learning via grouped sequential-to-parallel training.” In: *27th International Conference on Database Systems for Advanced Applications (DASFAA)*, 2022.
- [20] Nishio, Takayuki, and Ryo Yonetani: “Client selection for federated learning with heterogeneous resources in mobile edge.” In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1-7, IEEE, 2019.
- [21] Pang, Junjie, Yan Huang, Zhenzhen Xie, *et al.*: “Realizing the heterogeneity: A self-organized federated learning framework for IoT.” *IEEE Internet of Things Journal (IOTJ)* 8, no. 5, pp. 3088-3098, 2020.
- [22] Hiessl, Thomas: “Cohort-based federated learning services for industrial collaboration on the edge.” TechRxiv, Preprint, 2021.
- [23] Zinkevich, Martin, Markus Weimer, Alexander J. Smola, *et al.*: “Parallelized stochastic gradient descent.” In: *24th Conference on Neural Information Processing Systems (NeurIPS)* 4, no. 1, p. 4, 2010.
- [24] McMahan, Brendan, Eider Moore, Daniel Ramage, *et al.*: “Communication-efficient learning of deep networks from decentralized data.” In: *20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273-1282, 2017.
- [25] Caldas, Sebastian, Sai Meher Karthik Duddu, Peter Wu, *et al.*: “Leaf: A benchmark for federated settings.” In: *33rd Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2019.
- [26] Yao, Xin, Chaofeng Huang, and Lifeng Sun: “Two-stream federated learning: Reduce the communication costs.” In: *2018 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1-4, 2018.
- [27] Yao, Xin, Tianchi Huang, Chenglei Wu, *et al.*: “Towards faster and better federated learning: A feature fusion approach.” In: *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 175-179, 2019.
- [28] Li, Tian, Anit Kumar Sahu, Manzil Zaheer, *et al.*: “Federated optimization in heterogeneous networks.” In: *Conference on Machine Learning and Systems (MLSys)*, 2018.
- [29] Yeganeh, Yousef, Azade Farshad, Nassir Navab, *et al.*: “Inverse distance aggregation for federated learning with non-iid data.” In: *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, pp. 150-159, Springer, Cham, 2020.
- [30] Rieger, Laura, Rasmus M. Th Høegh, and Lars K. Hansen: “Client adaptation improves federated learning with simulated non-iid clients.” In: *International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML*, 2020.
- [31] Hsu, Tzu-Ming Harry, Hang Qi, and Matthew Brown: “Measuring the effects of non-identical data distribution for federated visual classification.” In: *International Workshop on Federated Learning for Data Privacy and Confidentiality in Conjunction with NeurIPS*, 2019.
- [32] Reddi, Sashank, Zachary Charles, Manzil Zaheer, *et al.*: “Adaptive federated optimization.” In: *International Conference on Learning Representations (ICLR)*, 2021.
- [33] Jeong, Eunjeong, Seungeun Oh, Hyesung Kim, *et al.*: “Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data.” In: *Workshop on Machine Learning on the Phone and other Consumer Devices*, Montréal, Canada, 2018.
- [34] Wang, Shiqiang, Tiffany Tuor, Theodoros Salonidis, *et al.*: “Adaptive federated learning in resource constrained edge computing systems.” *IEEE Journal on Selected Areas in Communications (JSAC)* 37, no. 6, pp. 1205-1221, 2019.
- [35] Yu, Hao, Sen Yang, and Shenghuo Zhu: “Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning.” In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* 33, no. 1, pp. 5693-5700, 2019.
- [36] Li, Xiang, Kaixuan Huang, Wenhao Yang, *et al.*: “On the convergence of fedavg on non-iid data.” In: *International Conference on Learning Representations (ICLR)*, 2020.
- [37] Nesterov, Yu: “Gradient methods for minimizing composite functions.” *Mathematical Programming* 140, no. 1, pp. 125-161, 2013.
- [38] Ward, Rachel, Xiaoxia Wu, and Leon Bottou: “AdaGrad stepizes: Sharp convergence over nonconvex landscapes.” In: *International Conference on Machine Learning (ICML)*, pp. 6677-6686, 2019.
- [39] Kingma, Diederik P., and Jimmy Ba: “Adam: A method for stochastic optimization.” In: *International Conference on Learning Representations (ICLR)*, 2015.
- [40] Zaheer, Manzil, Sashank Reddi, Devendra Sachan, *et al.*: “Adaptive methods for nonconvex optimization.” In: *32nd Conference on Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2018.
- [41] Liu, Yi, Sahil Garg, Jiangtian Nie, *et al.*: “Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach.” *IEEE Internet of Things Journal (IOTJ)*, 8, no. 8, pp. 6348-6358, 2020.

- [42] Akpakwu, Godfrey Anuga, Bruno J. Silva, Gerhard P. Hancke, *et al.*: "A survey on 5G networks for the internet of things: Communication technologies and challenges." *IEEE Access* 6, pp. 3619-3647, 2017.
- [43] Li, Zonghang, Huaman Zhou, Tianyao Zhou, *et al.*: "ESync: Accelerating intra-domain federated learning in heterogeneous data centers." *IEEE Transactions on Services Computing* (TSC), (2020).
- [44] Rice, Bart: "The 0-1 integer programming problem in a finite ring with identity." *Computers and Mathematics with Applications* 7, no. 6, pp. 497-502, 1981.
- [45] Zhou, Huaman, Zonghang Li, Qingqing Cai, *et al.*: "DGT: A contribution-aware differential gradient transmission mechanism for distributed machine learning." *Future Generation Computer Systems* (FGCS) 121, pp. 35-47, 2021.
- [46] Varga, Pal, Jozsef Peto, Attila Franko, *et al.*: "5G support for industrial IoT applications: Challenges, solutions, and research gaps." *Sensors* 20, no. 3, p. 828, 2020.
- [47] Nogueira, Fernando, *et al.*: "Bayesian optimization: Open source constrained global optimization tool for python." [Online]: <https://github.com/fmfn/BayesianOptimization>, 2014.
- [48] Whitley, Darrell: "A genetic algorithm tutorial." *Statistics and Computing* 4, no. 2, pp. 65-85, 1994.
- [49] Chen, Tianqi, Mu Li, Yutian Li, *et al.*: "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems." In: *30th Conference on Neural Information Processing Systems* (NeurIPS), 2016.
- [50] Pan, Sinno Jialin, and Qiang Yang: "A survey on transfer learning." *IEEE Transactions on Knowledge and Data Engineering* (TKDE) 22, no. 10, pp. 1345-1359, 2009.