TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 06/23 pp2040–2054
DOI: 10.26599/TST.2024.9010227
Volume 31, Number 4, August 2026

Research Article

Digital Twin-Enabled Edge Federated Learning for Data Streams

Zhenzhen Xie, Junjie Pang, Yan Huang, Olusesi Balogun, Dongxiao Yu, and Zhipeng Cai*

Abstract: With the significant advancement in the Internet of Things (IoT), Streaming Federated Learning (SFL) as a novel distributed learning approach can deal with time-varying streaming data among multiple sources. Standard SFL protocol is a collaborative training framework that enables many clients bounded with different online data sources to participate in a continuous training task. However, existing works ignore the cold-start problem and insufficient training data obstacle. Besides, due to the client heterogeneity and forgetting problem, the global model faces performance degradation during the time-series streaming data. In our work, we propose a digital twin-enabled SFL, a novel federated learning system with digital twin support to augment training data on demand. Instead of adopting an asynchronous federated learning protocol or buffer technique to wait for clients to have enough data, Generative adversarial network-based digital twins are introduced to construct a virtual replica for each federated learning client to generate a synthetic dataset based on the real data stream. We conduct the experiments using real-world datasets to evaluate the proposed SFL framework. The results under multiple data stream scenarios and various client behaviors demonstrate that our work outperforms the state-of-the-art baseline.

Key words: Federated Learning (FL); Digital Twins (DTs); streaming data; conditional Generative Adversarial Network (cGAN)

1 Introduction

Due to the significant advancement in the Internet of Things (IoT) over the past decades, there are nearly 14.3 billion active IoT endpoints worldwide, and the connections are expected to continue their steep growth in the foreseeable future. As such, data sources are becoming ubiquitous, e.g., autonomous vehicles, environment monitoring, smart transportation systems, and smart grids. Besides, data are generated from these distributed sources continuously and endlessly. Considering data sources mainly located on mobile and small devices with increasing privacy concerns today[1], a new distributed learning paradigm called Streaming Federated Learning (SFL) is proposed to facilitate privacy-preserving collaborative machine learning of complex models among massive end devices. Compared to traditional distributed learning solutions, SFL highlights the following distinguished features: (1) privacy protection, (2) streaming data, and (3) Multi-access Edge Computing (MEC)-based implementation.

• Zhenzhen Xie and Dongxiao Yu are with School of Computer Science and Technology, Shandong University, Qingdao 266237, China. E-mail: xiezz21@sdu.edu.cn; dxyu@sdu.edu.cn.
• Junjie Pang is with College of Computer Science and Technology, Qingdao University, Qingdao 266070, China. E-mail: pangjj@qdu.edu.cn.
• Yan Huang is with College of Computing and Software Engineering, Kennesaw State University, Atlanta, GA 30060, USA. E-mail: yan.huang@kennesaw.edu.
• Olusesi Balogun and Zhipeng Cai are with Department of Computer Science, Georgia State University, Atlanta, GA 30302, USA. E-mail: obalogun6@student.gsu.edu; zcai@gsu.edu.
* To whom correspondence should be addressed.
Manuscript received: 2024-02-07; revised: 2024-06-08; accepted: 2024-11-11

Many works have implemented SFL by extending existing online learning models into the SFL paradigm[2–5]. These methods aim to deal with the concept drift problems and investigate the device heterogeneity challenges in SFL. Nevertheless, existing SFL studies are still inefficient in response to real-world requirements due to impractical assumptions on the data conditions and resource conditions of Federated Learning (FL) clients. Meanwhile, the volume of FL clients can not always be sufficient for a specific FL request. Thus, the apparent research gap in FL studies lies between real-world conditions and expectations leads to the following problems: First, the uneven computation and communication resources could exacerbate the instability of SFL training, such as Fig. 1a indicates a general scenario (different SFL clients can have various data distribution and data generation rates) that is not considered in existing SFL works. Moreover, considering limited resources and other running applications on the SFL client's end devices, different cache sizes and update rules are adopted and make the SFL client miss or drop some incoming data streams, thus leading to the global model performance degradation or a significantly large time cost to wait for full enough training data.

From our observation, we conclude that resolving the above problems faces the following challenges:

Firstly, data streams from different clients could be highly different in data volume and distribution[6], leading to the difficulty of accumulating enough data samples for specific domain or tasks[7] and thus resulting in performance degradation for the specific FL training task. For example, the sensor of Client 1 exhibits a relatively slow data generation rate and possesses a narrow coverage scope, resulting in its prolonged exposure to merely two categories of data samples (Class 1 and Class 2) for an extended duration. Meanwhile, the sensor of Client 2 has a much higher data generation rate and larger coverage (Class 3–Class 10), so they can collect much more data than Client 1 in the same period. Since the data distribution Client 1 does not overlap with Client 2, even if asynchronous FL protocol is adopted to alleviate the heterogeneity among data sources, the global model can still have limited capability on Class 1 and Class 2 data samples.

Secondly, another restriction of current FL approaches is that all the clients must upload the entire local training model to the FL server for further aggregation. It will lead to privacy leakage because Refs. [8, 9] showed that model sharing increases the gradient leakage risks, and many FL attacks[10] can infer the individual training data effectively.

Finally, applying traditional online learning models in SFL to deal with streaming data causes catastrophic forgetting problems: when new patterns are detected in the latest data stream, the previous patterns will be forgotten/replaced. Thus, the local model can only consider the data stream updates in the most recently received batch, which indirectly leads to the potential occurrence of catastrophic forgetting within the global model. Note that, the inherent data heterogeneity and privacy settings in FL can exacerbate the following problem: the catastrophic forgetting phenomenon could be more frequent since the local data source of FL clients can have arbitrary inputs, resulting in unseen classes or features. Worse, under the existing privacy settings, the FL server has limited capability to prevent catastrophic forgetting by using traditional data-centric

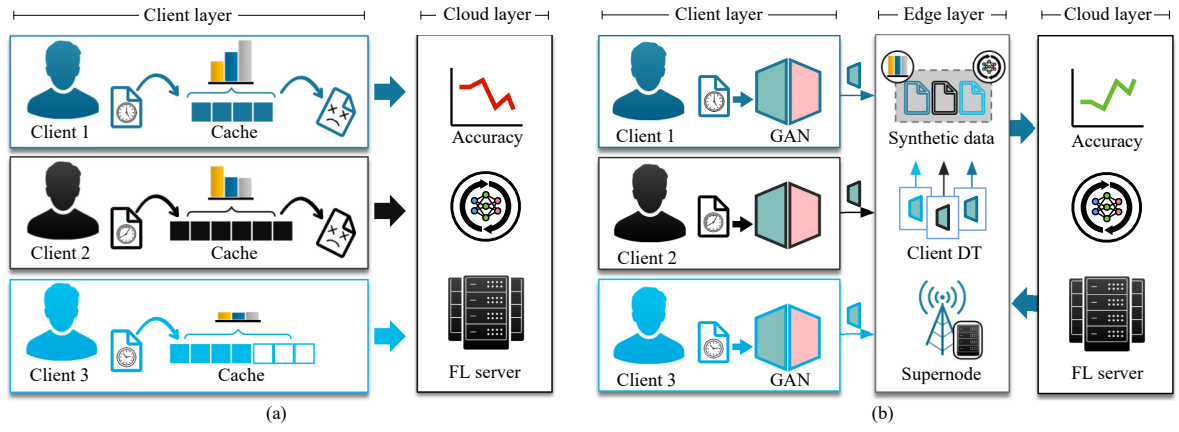


Fig. 1 SFL with clients heterogeneity. (a) Streaming federated learning under client heterogeneity; (b) Digital twin-enhanced SFL with cGAN-based data augmentation. GAN: Generative Adversarial Network.

methods.

In this work, we propose a Digital Twin (DT)-enabled FL, a novel FL system with DT support to resolve all the above challenges. Instead of adopting asynchronous FL protocol or buffer technique to wait for clients to have enough data, we introduce DT to construct a virtual replica for each FL client and use conditional Generative Adversarial Network (cGAN)[11] to generate synthetic data set based on the real data stream as shown in Fig. 1b. We further alleviate the catastrophic forgetting problem by leveraging memory replay on each FL client. Specifically, we allow the FL clients to create a dataset containing the latest data stream and replayed samples from the previous stream, aiming to detect new patterns without overwriting existing learned knowledge.

The contributions are summarized in the following:

● We are the first to integrate DTs into the SFL framework to resolve the client heterogeneity problem from a data perspective. Our work is implemented in the MEC environment, while each client has a client DT running on the edge layer to augment the training data and participate in the SFL training request. Our design resolves both the data imbalance and insufficient data dilemma since the cGAN-based DT can automatically adjust the data distribution without increasing individual SFL clients' training costs.

● Considering the streaming data feature and limited storage in SFL clients can cause catastrophic forgetting, we follow the memory GAN concept to generate historical data samples. When there is a new data stream coming, the SFL client and his replica (client DT) can replay the past instead of only focusing on the recent information.

● To further enhance the privacy protection level, a group of client DTs on the same edge server forms a supernode to participate in an FL training task. Therefore, the SFL clients can hide the raw data and model parameters to achieve a high-level privacy-preserving capability against deep gradient leakage attacks.

● We conduct extensive experiments to validate the efficacy of the proposed framework and analysis conclusions. Experiments on various public datasets show that our proposed framework outperforms state-of-the-art baselines under different client distributions.

The rest of this paper is organized as follows. Section 2 investigates the research related to SFL, DT and GAN. Section 3 presents preliminary concepts and the key observations and challenges of SFL. Section 4 explains the detailed structure and methodology of the proposed framework. Section 5 provides experimental results and analysis. Section 6 concludes our work.

2 Related Work

In this section, we investigate the literature related to SFL, DT, and data augmentation.

SFL is a novel distributed machine learning approach that enables decentralized model training on data streams. Similar to the typical FL settings, different stack-holders (FL clients) collaboratively learn a global model over individual devices under the constraint that the data is updated, stored, and processed locally[12–14]. Existing studies focus on cache management[12], data selection[13], and asynchronous FL[15] to improve SFL training efficiency. Most of the previous studies assume that the SFL paradigm has sufficient participating clients or that SFL clients can always have a consistent pattern of data input, which is difficult to satisfy in the MEC environments due to device heterogeneity. Moreover, in the local training step, the state-of-the-art SFL implementations require clients to maintain communication with the FL server continuously and overlook the catastrophic forgetting difficulties, resulting in gradient leakage risk and global model performance degradation. To the best of our knowledge, we are among the first to provide a novel SFL framework over the MEC network that introduces synthetic data generation methods to resolve the data sparsity and data imbalance challenge. By making full use of local data streams, Our work releases the constraints on SFL clients' participation while having the capability of generating training data on demand.

DT-based FL. DT is a digital replica of a physical entity constructed and updated by continuous interactions with the physical world to ensure synchronization[16–19]. Since the DTs can reflect the running states of a physical system, many recent works[20–23] on FL over MEC networks integrates DT to build a DT network for collecting real-time resource conditions, e.g., bandwidth conditions, available computation resource of end devices and communication load in edge networks. The information about the real-time conditions and behaviors of MEC can be used for resource scheduling and network load balancing, and thus help the FL server to decide on a better client selection strategy or resource optimization

solution. However, in SFL, the application of DTs remains unexplored. We are among the first to investigate constructing client twin on-edge servers in a hierarchical SFL architecture over the MEC network. The integration of DT and SFL can facilitate the enhanced utilization of computational resources available on edge servers, thereby augmenting the training data and subsequently elevating SFL's training efficiency.

Data augmentation with GAN. As computational power increases, deep learning models have made significant advances in deep network architectures and led to an explosion in various kinds of applications, such as healthcare, supply-chain management, and autonomous vehicles. It is common knowledge that, despite the computing resources, data volume and quality are the primary factors to ensure deep learning models can perform well on downstream tasks[24–26]. Rather than accumulating sufficient data for training, GAN-based data augmentation emerged as a promising method to deal with the small data set challenge and reduce over-fitting on deep learning models. Amongst recent studies, DCGANs[27], CycleGANs[28], and Conditional GANs[11] are the most commonly used methods to generate new training data. Moreover, GAN-based data augmentation methods can be used for resolving class imbalance problems. Given the above features, a line of research leverages GAN to generate synthetic data samples to improve the training efficiency of FL[26, 29–31]. However, existing GAN-based FL approaches mainly focus on learning static datasets, and ignore the possibility of using synthetic data samples to accelerate the SFL process.

3 Background and Key Challenge

In this section, we provide the necessary background and summary of important observations and challenges.

3.1 Preliminary

FL for data streams. FL[32] is a collaborative training framework that enables a number of clients bounded with different data sources and end devices to participate in a particular training task. Unlike other distributed learning approaches that need to directly train models on raw data, FL adopts "model-sharing" collaborative training instead of "data sharing", which enhances the privacy protection level and prompts collaborations among different stakeholders[33]. In particular, many participants (or FL clients) can solve the learning task together through a hub-and-spoke topology to generate an optimized global model without exposing private data sources. For a new FL training task, (1) the FL central server trains a global model based on a public dataset for initialization and then distributes this model to the FL clients; (2) after receiving the global model, each FL client can update the global model based on individual data sources to generate local updates; (3) based on a decided synchronization setting, all these local updates are sent to the FL central server for aggregation, and the global model is improved; (4) Steps 2 and 3 are repeated until the global model converges or achieves the expected performance.

Most previous works assume clients have static, high-quality, and sufficient training data to respond to an FL training task. However, in the real world, given that most FL clients rely on end devices with limited capacity but strong sensing capability (various kinds of built-in sensors)[34, 35], we note that the setting and research focus of SFL could be more practical than static FL designs. SFL has a similar training process to traditional FL; however, it assumes every client relies on an online data source and gathers time-varying training data over time.

DT. A DT is a digital representation of a physical asset, initially developed to automatically aggregate, analyze, and visualize complex information through continuous interactions with the physical entity. Given the features of real-time interaction, environment/ process simulation, and future conditions prediction, many FL designs adopt DT to obtain communication efficient FL by using DT to monitor and predict the MEC network status and finally decide on an improved resource scheduling plan. Unlike existing studies that use DT as the replica of the training environment (e.g., MEC network), in our work, we investigate constructing DT models for each FL client to improve communication capability while alleviating the training data shortage challenge without violating privacy protection rules.

GAN. GAN is a deep learning approach for generative modeling composed of a generator network and a discriminator network. The generator network aims to produce a candidate instance to deceive the discriminator. Meanwhile, the discriminator needs to evaluate this candidate instance compared to the real data distribution. The result of training a GAN involves

obtaining equilibrium between the generator and discriminator. Finally, the generator can generate synthetic data, but the discriminator cannot distinguish whether the data is real or generated. In our work, we introduce GAN into FL client DTs to enable the data augmentation function. Based on real streaming data from FL clients, GAN-based FL client DTs offer the capability of augmenting training data on demand, thereby mitigating the waiting time during training and consequently enhancing both training efficiency and model performance. Moreover, the synthetic data generated by GAN can further enhance the privacy protection level. By hiding real raw data and gradients of FL clients, our SFL design can resist privacy attacks such as deep leakage on gradients while maintaining a stable training performance.

3.2 Observation and challenges

Observation 1: dynamic resource conditions and time-varying data stream. First, SFL considers the FL clients have to deal with dynamic datasets that the local model must be trained on time-varying data under dynamic resource conditions. We note that most FLs are in the MEC environments. Specifically, the FL server located on the cloud serves as the primary coordinator, the micro cloudlet or micro datacenter on edge provide computation and communication resource near where the data is generated, and end devices with various kind of sensors generate streaming data to describe how the practical environment changes. This scenario involves carrying out FL protocol in an environment where data is arriving continually with different data generation rates among devices.

Observation 2: obstacle of avoiding catastrophic forgetting. To deal with the data streams, SFL has to consider the limited memory capacities and adopt a buffer/cache to store new data samples. Thus, different from most existing FL frameworks on static datasets without consideration of storage and cache, SFL involves the cache update rules and data sampling strategy to ensure the local updates can always adapt to recent data streams. However, since the training data is gradually accumulated, despite a well-designed cache mechanism, the SFL still needs to reduce the gap between the long-term and short-term data distribution.

Observation 3: Contradiction between insufficient training data and performance enhancement expectation. We note that the fundamental concept of FL resembles the concept of "crowdsourcing", as both rely on high-quality individual contributions to achieve effective collaboration. In other words, the model performance of a specific FL training request is bounded by every FL client's communication power, communication status, and data source[36]. However, there is a clear contradiction between model gains expectation and heterogeneity conditions of FL clients. Worse, in SFL settings, each FL client has a different data arrival rate, and the generated volume of data could be varied. It exacerbates the challenge of insufficient training data in FL, and reliance on existing asynchronous communication methods cannot entirely mitigate this predicament since some stragglers may not have sufficient training data for an extended period of time. Instead of using a straggler dropout strategy, our work intends to investigate an SFL framework that can make full use of every data stream and augment the raw data on demand to maintain the FL training efficiency.

Remarks. Motivated by these observations and analysis of existing studies, our work expects to provide a novel SFL framework that achieves training efficiency by resolving the data sparsity difficulty and data insufficiency dilemma while not increasing individual FL clients' training costs.

4 Framework of DT Enabled FL for Data Streams

This paper proposes a DT-enabled SFL framework aiming at resolving the data sparsity challenge while providing privacy enhancement in the MEC environment. To make full use of streaming data from FL clients, we construct client DT on the edge servers as the replica of individual clients and provide on-demand training data augmentation to respond to FL training requests. Furthermore, considering the streaming feature and catastrophic forgetting obstacle, our work employs experience replay to the local training process. It thus ensures our design fully utilizes limited clients storage resources while avoiding retraining from scratch when the data stream changes. In this section, we first introduce problem and the overview of our proposed SFL framework. Then, we give the details of constructing FL client DTs, the local learning process, and global model aggregation.

4.1 Problem statement

Given an SFL system with m clients as workers and a central server as a coordinator. We assume each Client

k has an online data source that can generate dynamic data streams for local training during the entire training process. At time slot t, Client k received the data set can be denoted as $\mathcal{D}_k^t$. For simplicity, we further assume that the long-term distribution of $\mathcal{D}_k^t$ is ξ. Then, the objective function of SFL can be written as

$$\min_{w} \mathcal{G}(F_1(\mathcal{D}_1; w), F_2(\mathcal{D}_1; w), \ldots, F_m(\mathcal{D}_m; w))$$

where $F_k(\cdot)$ is the local objective function of client k on local data set $\mathcal{D}_k$, $k = 1, 2, \ldots, m$. Note that, $\mathcal{D}_k$ is the set of data $(\mathcal{D}_k^1 \cup \mathcal{D}_k^2 \cup \cdots \mathcal{D}_k^T)$ that Client k can received during the whole training process. $\mathcal{G}(\cdot)$ is the aggregation function, where $\mathcal{G} = \sum_{i=1}^{m} p_i F_i(\mathcal{D}_i; w)$ and the sum of the weights denoted as $\sum_{i=1}^{m} p_i = 1$.

Considering the heterogeneity, uncertainty, and dynamic nature of SFL clients, it is impractical to ensure all the clients can have stable data generation rates without missing any data stream. Limited cache resources and unstable network status can also be another reason that violate the SFL training performance. Worse, the forgetting issue further leads to performance degradation.

Thus, we introduce DT-enabled SFL frameworks in MEC environments and investigate data augmentation techniques to resolve the data sparsity and imbalance challenge. In the next sections, we give the details of our methodology and implementation.

4.2 Overview

Figure 2 illustrates the overall framework of our SFL design. Our proposed framework works in the MEC environment, which includes three layers as follows:

Client layer contains many SFL clients carrying various types of end devices. These mobile devices can collect time-varying data streams based on sensors or other online data resources at different data generation rates. To further utilize the computation capability on the end devices, each of them is provided with a basic sequential learning GAN to further process the dynamic raw data. Generator′ denotes the parameters of the generator updated in the previous time step. It is used to generate synthetic samples and integrate them with the currently arrived data, thereby implementing an experience replay mechanism. Moreover, by incorporating the memory replay technique, we can ensure that the GAN model can be updated by the most recent data streams while preventing forgetting issues.

Edge layer includes many edge servers to run all the FL client twins. Each twin is the replica of the FL

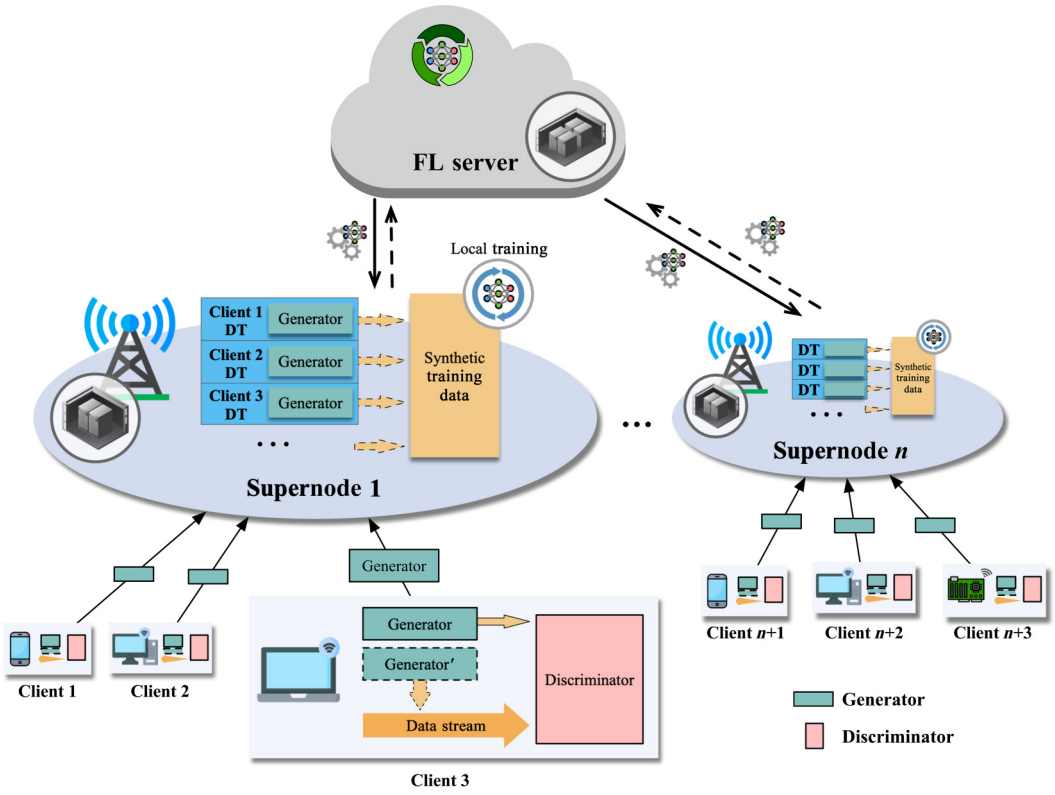


Fig. 2 SFL framework with DT support.

client, and it interacts with the FL clients continually to update itself to maintain synchronization. Moreover, the FL client DT has a replica of the core components of the FL client's GAN-generator. Supported by the computation resource behind edge servers, the client DT can generate synthetic data to augment the real data stream. Furthermore, considering each edge server contains a cluster of FL client DTs, we regard the DT cluster as a supernode. Instead of using FL clients to join the FL training directly, we let the supernodes participate in a specific FL training request to accomplish the local training task with the synthetic data stream. Such a design can further enhance the privacy protection level of individual FL clients since the supernode uses many DT data as the local training data instead of training the FL local model in every single DT. Meanwhile, the edge server can also provide computation resources for running client twins near stream data production.

Cloud layer: The FL server is located in the cloud layer, which is the coordinator throughout the FL training process. Due to the superior computation resources, the computation-intensive tasks in the FL training process, like model aggregation and optimization, can also be offloaded to the remote cloud.

We further illustrate the workflow of our framework: when there is an incoming data stream at FL Client i, the local GAN cGAN_i of Client i will be updated immediately. Then, Client i uploads the latest generator to Client i's DT while keeping the discriminator locally. Once the cGAN_i is changed, the generator of cGAN_i in Client i's DT needs to update itself accordingly to maintain synchronization. Next, the client DTs in the edge server E_i naturally form a supernode N_i to participate in the FL training task; when there is an FL training request, SFL client DT can generate synthetic training data based on the latest streaming data for SFL local training. Finally, each supernode N_i follows the standard FL training process to output an optimized global model G without waiting for accumulating training data.

4.3 GAN-based local pre-training with memory replay

We note that most end devices have limited capacity to store streaming data, and different FL client could have various data generation speed. The streaming feature and unbalanced resource distribution among clients motivate us to investigate a "smart buffer", which can both retain the knowledge within historical streaming data and prioritize the value of newly generated data streams. For implementing such a "smart buffer" while maintaining competitive training efficiency, each FL client is provided with a cGAN to process incoming data streams. For clarity, we term the cGAN-based streaming data learning as local pre-training in order to distinguish it from SFL local training. Figure 3 illustrates the cGAN-based local pre-training while Algorithm 1 describes the detailed procedure. In the pre-training process, we first initialize Generator and Discriminator networks, and define loss functions for the generator, discriminator, and classifier (Lines 1 and 2). Then, we use current samples with replayed sample from previous time step to achieve the training set (Lines 3 and 4). The above step iterates through epochs (Lines 5 and 6). Generate a fake sample using the generator and pass fake sample through discriminator to get output and predicted label (Lines 7 and 8). Compute generator loss and update generator, Pass real batch sample through discriminator to get output and predicted label (Lines 9–11). Then, we compute discriminator loss and classifier loss and update the discriminator (Lines 12–14). Finally, generator parameters are uploaded to the supernode for further processing (Line 17).

To initialize the cGAN training, each client first trains a local cGAN model based on their raw data. Then, the initial GAN model is updated when a new data stream is generated. To ensure that the generated data matches the categories of the raw data on SFL clients, the input conditional information is the class

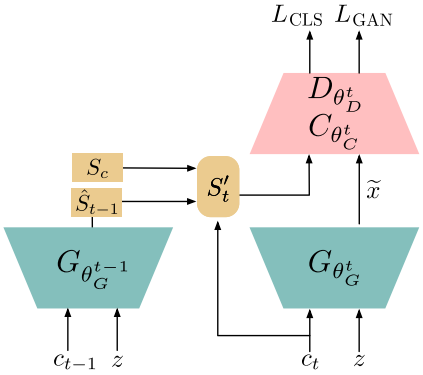


Fig. 3 Local training with memory GAN. $G_{\theta_G^t}$, $D_{\theta_D^t}$, and $C_{\theta_C^t}$ are the generator, discriminator, and classifier networks at timestamp t, respectively. c_t is the class information at timestamp t.

Algorithm 1 cGAN training with experience replay

Initialize generator G and discriminator D networks;
Define generator loss L_G, discriminator loss L_D, and classifier loss L_C;
Generate replayed sample $\hat{S}_{t-1} = G(z, c_{t-1})$;
$S'_t = \text{Concat}(S_c, \hat{S}_{t-1})$;
for epoch in num_epochs **do**
  for batch in S'_t **do**
    fake_sample = $G(z, c)$;
    dis_fake, label_fake = D(fake_sample);
    Compute L_G based on dis_fake;
    Update generator G;
    dis_real, label_real = D(batch);
    Compute L_D based on dis_real, dis_fake;
    Compute L_C based on label_real, label_fake;
    Update discriminator D;
  end for
end for
Upload G parameter to supernode

labels. Then, we use $S = \{S_1, S_2, \ldots, S_C\}$ as the training dataset to generate synthetic data in a certain class, where represents all data samples of Class c.

Each client cGAN consists of three components: generator, discriminator, and classifier. The generator is used to output synthetic data samples according to the conditions: To generate a synthetic data sample $\widetilde{x} = G(z, c; \theta_G)$ by the generator, where θ_G, z, and c represent the parameter of the generator, the input noise, and the class information accordingly. The discriminator D is parametrized by θ_D and aims to classify whether the output of the generator is real or fake, considering the conditional information. The classifier C, which is parametrized by θ_C, is for predicting the class label of an input data sample x by $\widetilde{c} = C(x; \theta_C)$. During the cGAN training, the interactions among the generator and discriminator can be regarded as an adversarial game, and both of the two components can be optimized by alternating the optimization. The optimization problem of the generator is as follows:

$$\min_{\theta_G} \left(L_G^{\text{GAN}}(\theta, S) + \lambda_{\text{CLS}} L_G^{\text{CLS}}(\theta, S) \right);$$

$$L_G^{\text{GAN}}(\theta, S) = E\left[\log\left(1 - D(G(z, c; \theta_G); \theta_D)\right)\right],$$

$$L_G^{\text{CLS}}(\theta, S) = E\left[J\left(C(G(z, c; \theta_G); \theta_C), y_c\right)\right];$$

where $E[\cdot]$ represents the expectation operator $L_G^{\text{GAN}}(\theta, S)$ and $L_G^{\text{CLS}}(\theta, S)$ are the loss function of cGAN and classifier, accordingly, and S is the training dataset. θ represents the collection of network parameters, including θ_G, θ_D, and θ_C. λ_{CLS} is a hyperparameter used to balance the influence of the two losses. z follows a normal distribution and c follows a Gaussian distribution. $J(\cdot)$ is the cross-entropy function, and y_c indicates the one-hot encoding vector of class label c. Similar to the generator, the discriminator and classifier have the following optimization functions:

$$\min_{\theta_D, \theta_C} \left(L_D^{\text{GAN}}(\theta, S) + \lambda_{\text{CLS}} L_D^{\text{CLS}}(\theta, S) \right),$$

$$L_D^{\text{GAN}}(\theta, S) = E_{(x,c)\sim S}\left[\log(1 - D(x; \theta_D))\right] + E\left[\log D(G(z, c; \theta_G); \theta_D)\right].$$

$$L_D^{\text{CLS}}(\theta, S) = E_{(x,c)\sim S}\left[J(C(x; \theta_C), y_c)\right] + E\left[J(C(G(z, c; \theta_G); \theta_C), y_c)\right].$$

where $L_D^{\text{GAN}}(\theta, S)$ and $L_D^{\text{CLS}}(\theta, S)$ are the GAN loss function and classifier loss function of the discriminator, respectively. $E_{(x,c)\sim S}$ denotes the expectation over real data samples (x, c) sampled from the training data set S. θ_D represents the discriminator parameters. θ_G represents the generator parameters.

Considering the streaming data feature, we follow and introduce memory replay in client cGAN to avoid catastrophic forgetting. In each training iteration, we generate an extended dataset $S'_t = S_c \cup \hat{S}_{t-1}$ instead of using s that only contains the latest data streams as the training dataset. $\hat{S}_{t-1}$ is generated by the generator$_{t-1}$, and in $\hat{S}_{t-1}$, each $c\,(c \in \{1, 2, \ldots, t-1\})$ has M historical samples $\hat{S}_{t-1}$ to ensure there is not a data class missed in the extended dataset. t denotes a timestamp. The optimization function of memoryGAN is as follows:

$$\min_{\theta_G^t} \left(L_G^{\text{GAN}}(\theta, S'_t) + \lambda_{\text{CLS}} L_G^{\text{CLS}}(\theta, S'_t) \right),$$

$$\min_{\theta_D^t, \theta_C^t} \left(L_D^{\text{GAN}}(\theta, S'_t) + \lambda_{\text{CLS}} L_D^{\text{CLS}}(\theta, S'_t) \right).$$

When there is a new incoming data stream, we let the cGAN of SFL clients update and upload the latest version to the related client DTs at the nearest edge server. Note that, the data batch size for cGAN training and the SFL client DTs deployment could impact the pre-training efficiency and performance. We leave these problems in our future work.

4.4 SFL training with client DTs

In our SFL framework, the SFL client only needs to

collect time-varying training data, obtain local pre-training with incoming data streams, and update the corresponding DTs continuously. When there is an SFL training request, the client DT would substitute SFL clients in engaging in the training process. When the client DTs receive the request, they can augment the raw training data based on the latest version of generators. To alleviate the imbalance of data volume among different classes of data, we allow the generator to output more synthetic data for such classes. We note that the existing GAN-based data augmentation methods could also generate synthetic data for unseen domains, which may further resolve the data heterogeneity problem in FL. However, such a problem is the common problem for both the conventional FL with static data and SFL, so the detailed methodology is out of the scope of this paper.

This training process is described in Algorithm 2, includes the following steps: In the beginning, the SFL central server trains a global model using a voluntary data source for initialization, then opens the platform for SFL clients to further optimization. When a new SFL client enters the SFL paradigm, the SFL central server allows the new client to register her corresponding DT on the neatest edge server. Next, the SFL central server distributes the global model to all existing client DTs. Each client DT outputs the synthetic training data from DT's generator. When the supernode gathers all DTs' training data, the SFL local training is triggered and the updated local model is sent to the SFL server for aggregation. After that, the SFL central server aggregates all the local updates using a specific aggregation algorithm (like FedAvg) to generate an updated global model and distributes this model to each client DT again. This iteration repeats several times until the global model achieves the expected performance.

Algorithm 2 SFL training with client DTs

```
Initialize global model M;
Define FL parameters;
for iteration in num_iterations do
    Server distributes M parameters to edge server;
    for edgeserver in num_edgeservers do
        Local model M′ = M;
        Generator sample Ŝ from DT;
        Training model M′ with Ŝ;
        Update M′ parameter to server;
    end for
    Aggregate local model updates to the global model M;
end for
```

Finally, the global model can always obtain the local model with the latest data stream without waiting for all the SFL clients to accumulate enough training data. Moreover, our proposed SFL framework can provide enhanced privacy protection since the local updates are from synthetic training data rather than real raw data. Note that, each SFL client can download the global model from the supernode at any time, since different clients may need a global model with different accuracy. For example, an SFL client with an urgent need can directly request the current version of the global model, without waiting for the final one.

5 Experiment

In this section, we evaluate the proposed SFL framework, SFL_DT in the presence of data streams. The experimental results on different datasets demonstrate the superiority of the proposed framework over the baseline methods and explain the necessity of the generative model-based DT implementation. Compared with other FL training protocols, SFL_DT consistently achieves better accuracy against different data stream settings by using DT to augment training data.

5.1 Datasets and setting

To study the performance and robustness of our proposed framework SFL_DT, we consider two data stream scenarios of changes in class distribution and missing data phenomenon. The datasets and detailed settings are introduced as follows.

5.1.1 Datasets

Our experiments are based on the MNIST[※], FMNIST[37], and CIFAR10[†] datasets, which are commonly used datasets for FL training tasks. The details are as follows:

MNIST: It consists of 60 000 training samples and 10 000 testing samples, comprising a total of ten categories. Each sample is a handwritten digit image with 28 × 28 pixels.

FMNIST: The dataset contains images of fashion products in ten categories, including T-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. Each image has 28 × 28 pixels. There are 60 000 training images and 10 000 testing images.

[※]https://www.kaggle.com/datasets/hojjatk/mnist-dataset
[†]https://www.cs.toronto.edu/kriz/cifar.html

CIFAR10: It contains color images in ten categories, including cars, birds, cats, deer, dogs, frogs, horses, ships, trucks, and airplanes. Each image has 32 × 32 pixels. There are 50 000 training images and 10 000 testing images, with 6000 images per category.

5.1.2 Baselines

We choose three FL baselines from the commonly used FL protocols, which include FedAvg[38], FedProx[39], and FedBN[40]. Specifically, FedAvg is the most widely used FL method that aims at saving communication costs. FedProx is designed to better handle the reality of non-IID data distributions in FL with reasonable communication costs by introducing a proximal term in the local objective. FedBN is another modified version of FedAvg by local batch normalization that is tailored for non-IID data distributions. We also consider the ideal setting for SFL, where all the clients have no memory limits so that they can store all the history training data locally, and we denote it as SFL_All To better explain why we introduce memory GAN in our SFL framework, we also consider the standard GAN-based local training as a baseline and denote it as Local sGAN.

5.1.3 Parameter settings

To implement our proposed SFL framework, we use a typical generative model structure (cGAN) consisting of a generator, discriminator, and classifier. The generator's architecture consists of an embedding layer, a fully connected layer, and multiple convolutional layers. Specifically, the generator maps the input into feature representations through the embedding layer, followed by linear and convolutional layers that progressively adjust the feature map dimensions. Upsampling layers are used to gradually increase the image resolution, and the final images are generated through convolutional layers, with a Tanh activation function ensuring the pixel values are in the range of [−1, 1]. The generator takes random noise and the labels of conditions as input and produces 32×32 pixels images. In the MNIST and FMNIST datasets, the feature map dimensions start from $128\times8\times8$, gradually transforming through Conv2d layers to channels. In contrast, for the CIFAR10 dataset, the feature map dimensions start from $512\times4\times4$ and go through more convolutional layers, allowing for more refined feature extraction and processing.

Moreover, we use the baseline ResNet18 as the global model and discriminator. The learning rate for all tasks is initially set to 0.1 and exponentially decays to 0.01. The clients train their local models using the Stochastic Gradient Descent (SGD) optimizer, The batch size is set to 32 to keep the batch size consistent across all experiments.

5.2 Results and analysis

5.2.1 Overall results

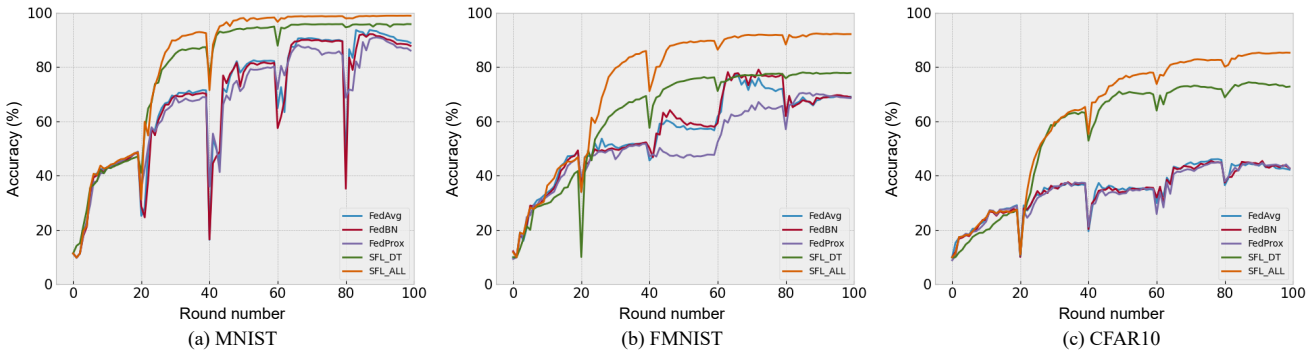
We carry out experiments on all three datasets and compare them with three commonly used FL approaches. We set up 20 clients and the local data used for training by each client is data streams rather than static datasets, where the data class changes every 20 rounds. Table 1 shows the overall experimental results, that is, the average accuracy of each method. Specifically, for the MNIST dataset, the accuracy of the baselines is around 90%, indicating that they can still maintain good performance when dealing with static or relatively simple data, but their performance is limited when facing dynamic changes in the data. Our proposed solution (SFL_DT) achieve an accuracy of 95.67%, significantly higher than the baseline methods. On the FMNIST dataset, the accuracy of the baseline methods remains around 69%, indicating that they have begun to show signs of catastrophic forgetting when facing the streaming data environment. Our algorithm significantly improves the accuracy to 77.76%. On the CIFAR10 dataset, the performance of the baseline methods is particularly limited, with accuracy hovering around 43%, highlighting their shortcomings in complex, dynamic data environments. While our proposed solution (SFL_DT) achieves an accuracy of 73.40%, significantly outperforming the baselines. The above results validate our algorithm's ability to handle dynamic data streams by incorporating generative model base DT in the SFL framework.

Figure 4 shows the accuracy variations during the training process for each method. It can be seen that the three methods, FedAvg, FedBN, and FedProx, exhibit periodic fluctuations in accuracy due to the changes in streaming data classes every 20 rounds. On the MNIST

Table 1 Accuracy comparison of SFL_DT with baselines over the three datasets.

(%)

Method	MNIST	FMNIST	CIFAR10
FedAvg	90.68 ± 1.21	69.10 ± 0.13	43.15 ± 0.70
FedBN	89.37 ± 1.14	69.13 ± 0.40	43.68 ± 0.73
FedProx	88.21 ± 1.52	69.19 ± 0.45	43.85 ± 0.49
SFL_ALL	98.91 ± 0.03	92.18 ± 0.07	85.19 ± 0.13
SFL_DT (ours)	95.67 ± 0.28	77.76 ± 0.11	73.40 ± 0.40



(a) MNIST (b) FMNIST (c) CFAR10

Fig. 4 Accuracy comparison of SFL_DT with baselines over the three datasets.

dataset, the baseline methods show particularly noticeable fluctuations at number of round of 40 and 80, indicating their poor adaptability to sudden changes in data distribution. When the data distribution of the data streams is changed, the models need to retrain to adapt to the new data distribution, resulting in a temporary drop in accuracy. On the FMNIST and CIFAR10 datasets, the fluctuations are more pronounced, with a significant decrease in model accuracy, highlighting the difficulty these methods face when dealing with class changes in more complex datasets.

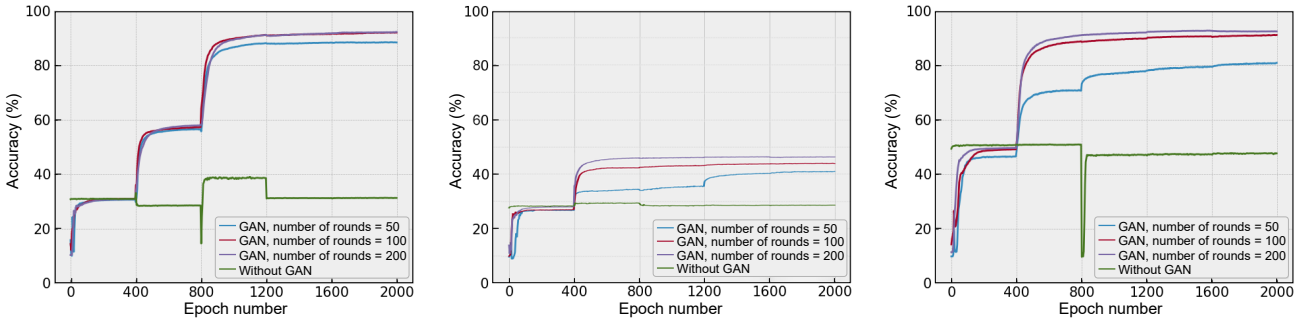
Compared to the baseline methods, our proposed method shows greater robustness on all datasets and higher accuracy, second only to the SFL_ALL method, and even surpassing the performance of SFL_ALL on some datasets. Although our method also exhibits periodic fluctuations across the three datasets, the amplitude of these fluctuations is significantly smaller than that of the baselines. This indicates that our method can quickly adjust and adapt to new data distributions by generating historical training data when dealing with class changes in streaming data, thus avoiding catastrophic forgetting and rapidly recovering to a high level of accuracy after fluctuations occur. Moreover, the experimental results also show that dynamic data streams can cause periodic fluctuations in model accuracy, with traditional baseline methods displaying significant fluctuations and poor adaptability to these changes. In contrast, our proposed SFL_DT method demonstrates higher stability and adaptability, better coping with class changes in streaming data, and quickly recovering accuracy.

5.2.2 Alleviating the effects of missing training data

In addition to unseen class challenges in streaming data, missing data streams are another common phenomenon. This situation is particularly common in IoT environments, where missing data streams phenomenon may result from sudden device disconnections or insufficient storage capacity. Therefore, we design another experiment to analyze whether our method can minimize the performance loss of client models in the presence of data loss. In our experimental scenario, we utilize 20 clients, with streaming data in which the data distribution every 20 rounds. Additionally, each client has a 30% probability of losing the most recent data for the current cycle.

In this experiment, we use DT with three different generative model settings to generate historical training data, where each generative model trained for 50, 100, and 200 rounds. At the same time, clients trained only on the most recent local data streams serving as the comparison baseline.

We select the local training results of three representative clients to analyze the experimental results and depict the results in Fig. 5. It indicates that as the number of training rounds increases, the accuracy of the local models using our method significantly improves, with the method using the generative model trained for the highest number of rounds (200 rounds) performing the best. This suggests that a higher number of training rounds can generate higher-quality synthetic data, thereby significantly enhancing the accuracy of the model even if a missing data phenomenon occurs. We also observe that the performance of standard FL methods without using synthetic data is the poorest, with a relatively smooth yet overall low accuracy curve. Due to the frequent loss of unseen classes, the model struggles to effectively learn and adapt to the new data distribution, leading to a significant drop in performance.



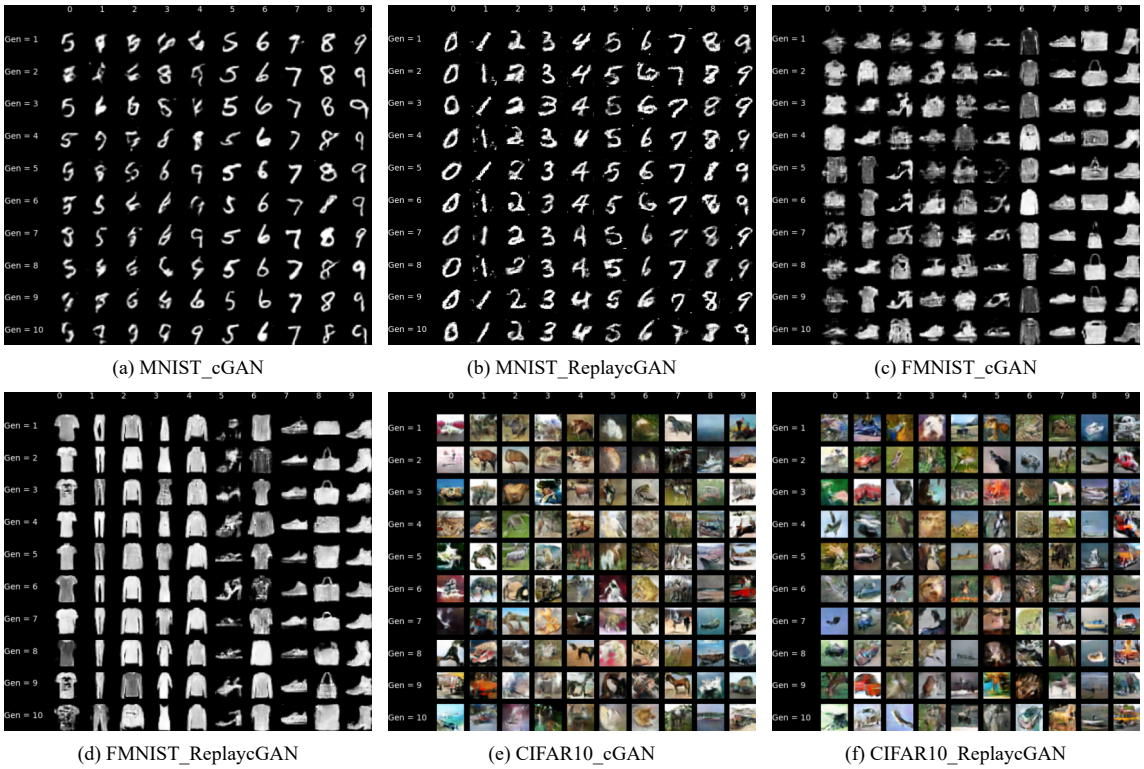
(a) MNIST-Client 1 (b) MNIST-Client 2 (c) MNIST-Client 3

Fig. 5 Accuracy comparison by various FL protocol of each client under missing data conditions.

5.2.3 Analysis of the generative model with experience replay

We analyze the ability of the generative model to prevent catastrophic forgetting under a single client condition and depict the results in Fig. 6. We use a generative model with experience replay on all three datasets (MNIST, F-MNIST, and CIFAR10), and we compare the performance with using a generative model without a replay mechanism. To validate that the replay mechanism can better adapt to the dynamic data distribution, we construct a streaming data scenario where data from two distinct time intervals arrive sequentially. In this scenario, each interval contains data from certain classes. Since the datasets we use contain ten classes in total, we assume that data from the first five classes arrive during the first interval, while data from the remaining five classes arrive during the second interval. In other words, we have two experiences and each experience has five classes. Through this scenario, we observe that the clarity of the generated images during the second interval differs across the three datasets. Methods incorporating experience replay can maintain high quality for both new and seen classes when generating images, while methods without experience replay exhibit significant catastrophic forgetting.

We can observe that the standard generative model could only access the image data received during the



(a) MNIST_cGAN (b) MNIST_ReplaycGAN (c) FMNIST_cGAN

(d) FMNIST_ReplaycGAN (e) CIFAR10_cGAN (f) CIFAR10_ReplaycGAN

Fig. 6 Analysis for the generative model with experience replay.

current interval, leading to catastrophic forgetting, where the generated images more similar to the most recently received categories. In contrast, the generative model with experience replay incorporates pseudo-images from all categories encountered in previous time steps, along with the most recently received images, into the training process. As a result, the generated images can maintain quality comparable to all historical images, rather than generating images that are only similar to the most recently received ones.

5.2.4 Analysis of the impact of supernodes

Because SFL in MEC environment often include various number of edge servers and participating clients, we evaluate several values for number of clients $n = \{8, 16, 32\}$ and number of supernodes $m = \{2, 4, 8, 16\}$. We depict the global model accuracy under different number of client and supernodes in Fig. 7. We observe that the global model's accuracy increases as the total number of clients grows. To further evaluate the impact of supernodes, we compare our approach under the same client number but a different number of supernodes, and we find that when the individual supernode has more clients, the model accuracy increases. The results show that when the number of supernodes is decreased, the SFL tends to be similar to centralized learning, and thus has minimal performance loss.

6 Conclusion

This study set out to construct a novel SFL framework, where each SFL client represents an online data source so that the local training data changes in both data volume and data classes over time. The data streams enable the FL server can always have fresh data and maintain synchronization with the physical conditions. However, this dynamic nature can also exaggerate the heterogeneity and cold-start problem since the SFL clients need more time to accumulate a high-quality and sufficient amount of training data for local training. To deal with the data insufficiency challenge without violating privacy protection rules, our work proposes a novel SFL framework with DT support. In our design, every SFL client has a digital replica client DT that substitutes for themselves in participating in SFL training. Moreover, all the client DTs remain synchronized with the SFL client all the time. By making full use of the edge server's resources, the client DTs are deployed on the edge layer so that the client DTs have sufficient computation resources to automatically use cGAN to augment the training data on demand. Experiments show that our design can achieve competitive model performance, meanwhile, preventing forgetting and achieving privacy enhancement. Future works include extending our DT-enabled SFL framework to resolve the personalized SFL problem.

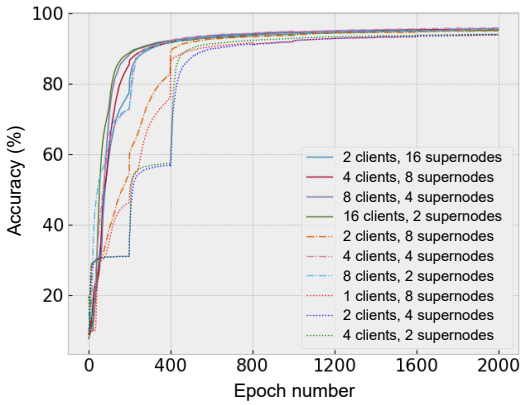


Fig. 7 Accuracy comparison under different number of supernodes.

Acknowledgment

This work was supported in part by the Young Scientists Fund of the Natural Science Foundation of Shandong Province (No. ZR2022QF134).

References

[1] K. Zhang, P. W. Tsai, J. Tian, W. Zhao, X. Cai, L. Gao, and J. Chen, Towards privacy in decentralized IoT: A blockchain-based dual response DP mechanism, *Big Data Mining and Analytics*, vol. 7, no. 3, pp. 699–717, 2024.

[2] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, Asynchronous online federated learning for edge devices with Non-IID data, in *Proc. 2020 IEEE Int. Conf. Big Data* (*Big Data*), Atlanta, GA, USA, 2020, pp. 15–24.

[3] H. Yu, Z. Chen, X. Zhang, X. Chen, F. Zhuang, H. Xiong, and X. Cheng, FedHAR: Semi-supervised online learning for personalized federated human activity recognition, *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3318–3332, 2023.

[4] A. Nandi and F. Xhafa, A federated learning method for real-time emotion state classification from multi-modal streaming, *Methods*, vol. 204, pp. 340–347, 2022.

[5] V. C. Gogineni, S. Werner, F. Gauthier, Y. F. Huang, and A. Kuh, Personalized online federated learning for IoT/CPS: Challenges and future directions, *IEEE Internet Things Mag.*, vol. 5, no. 4, pp. 78–84, 2022.

[6] Z. Cai and T. Shi, Distributed query processing in the edge-assisted IoT data monitoring system, *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12679–12693, 2021.

[7] M. Yan, T. Z. Joey, and W. T. Ivor, Collaborative knowledge infusion for low-resource stance detection, *Big Data Mining and Analytics*, vol. 7, no. 3, pp. 682–698, 2024.

[8] L. Zhu and S. Han, Deep leakage from gradients, in *Federated Learning: Privacy and Incentive*, Q. Yang, L. Fan, and H. Yu, eds. Cham, Germany: Springer, 2020, pp. 17–31.

[9] A. Wainakh, F. Ventola, T. Müßig, J. Keim, C. G. Cordero, E. Zimmer, T. Grube, K. Kersting, and M. Mühlhäuser, User-level label leakage from gradients in federated learning, *Proc. Priv. Enhanc. Technol.*, vol. 2022, no. 2, pp. 227–244, 2022.

[10] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, Evaluating gradient inversion attacks and defenses in federated learning, in *Proc. 35th Int. Conf. Neural Information Processing Systems*, Virtual Event, 2021, p. 553.

[11] M. Mirza and S. Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784, 2014.

[12] H. Wang, J. Bian, and J. Xu, On the local cache update rules in streaming federated learning, *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10808–10816, 2024.

[13] Y. Jin, L. Jiao, Z. Qian, S. Zhang, and S. Lu, Budget-aware online control of edge federated learning on streaming data with stochastic inputs, *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3704–3722, 2021.

[14] Z. Xiong, Z. Cai, D. Takabi, and W. Li, Privacy threat and defense for federated learning with non-i.i.d. data in AIoT, *IEEE Trans. Ind. Inf.*, vol. 18, no. 2, pp. 1310–1321, 2022.

[15] E. Baccarelli, M. Scarpiniti, A. Momenzadeh, and S. S. Ahrabi, AFAFed-asynchronous fair adaptive federated learning for IoT stream applications, *Comput. Commun.*, vol. 195, pp. 376–402, 2022.

[16] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji, Digital twin modeling, *J. Manuf. Syst.*, vol. 64, pp. 372–389, 2022.

[17] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, Characterising the digital twin: A systematic literature review, *CIRP J. Manuf. Sci. Technol.*, vol. 29, pp. 36–52, 2020.

[18] S. Haag and R. Anderl, Digital twin–proof of concept, *Manuf. Lett.*, vol. 15, pp. 64–66, 2018.

[19] C. Wang, Z. Cai, and Y. Li, Sustainable blockchain-based digital twin management architecture for IoT devices, *IEEE Internet Things J.*, vol. 10, no. 8, pp. 6535–6548, 2023.

[20] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, Communication-efficient federated learning for digital twin edge networks in industrial IoT, *IEEE Trans. Ind. Inf.*, vol. 17, no. 8, pp. 5709–5718, 2021.

[21] L. Jiang, H. Zheng, H. Tian, S. Xie, and Y. Zhang, Cooperative federated learning and model update verification in blockchain-empowered digital twin edge networks, *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11154–11167, 2022.

[22] Y. Qu, L. Gao, Y. Xiang, S. Shen, and S. Yu, FedTwin: Blockchain-enabled adaptive asynchronous federated learning for digital twin networks, *IEEE Netw.*, vol. 36, no. 6, pp. 183–190, 2022.

[23] A. M. V. V. Sai, C. Wang, Z. Cai, and Y. Li, Navigating the digital twin network landscape: A survey on architecture, applications, privacy and security, *High-Confid. Comput.*, vol. 4, no. 4, p. 100269, 2024.

[24] L. Taylor and G. Nitschke, Improving deep learning with generic data augmentation, in *Proc. 2018 IEEE Symp. Series on Computational Intelligence* (*SSCI*), Bangalore, India, 2018, pp. 1542–1547.

[25] C. Shorten and T. M. Khoshgoftaar, A survey on image data augmentation for deep learning, *J. Big Data*, vol. 6, no. 1, p. 60, 2019.

[26] Z. Xiong, W. Li, and Z. Cai, Federated generative model on multi-source heterogeneous data in IoT, in *Proc. 37th AAAI Conf. Artificial Intelligence*, Washington, DC, USA, 2023, pp. 10537–10545.

[27] A. Radford, L. Metz, and S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, in *Proc. 4th Int. Conf. Learning Representations*, San Juan, Puerto Rico, 2016.

[28] C. Chu, A. Zhmoginov, and M. Sandler, CycleGAN, a master of steganography, *arXiv preprint* arXiv:1712.02950, 2017.

[29] J. Zhang, C. Chen, B. Li, L. Lyu, S. Wu, S. Ding, C. Shen, and C. Wu, DENSE: Data-free one-shot federated learning, in *Proc. 36th Int. Conf. Neural Information Processing Systems*, New Orleans, LA, USA, 2022, p. 1556.

[30] B. Xin, W. Yang, Y. Geng, S. Chen, S. Wang, and L. Huang, Private FL-GAN: Differential privacy synthetic data generation based on federated learning, in *Proc. 2020 IEEE Int. Conf. Acoustics*, Speech and Signal Processing (ICASSP), Barcelona, Spain, 2020, pp. 2927–2931.

[31] A. Wijesinghe, S. Zhang, S. Qi, and Z. Ding, UFed-GAN: A secure federated learning framework with constrained computation and unlabeled data, *arXiv preprint* arXiv:2308.05870, 2023.

[32] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, Federated learning for internet of things: A comprehensive survey, *IEEE Commun. Surv. Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[33] Z. He, L. Wang, and Z. Cai, Clustered federated learning with adaptive local differential privacy on heterogeneous IoT data, *IEEE Internet Things J.*, vol. 11, no. 1, pp. 137–146, 2024.

[34] Y. Zhan, P. Li, L. Wu, and S. Guo, L4L: Experience-driven computational resource control in federated learning, *IEEE Trans. Comput.*, vol. 71, no. 4, pp. 971–983, 2022.

[35] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, Realizing the heterogeneity: A self-organized federated learning framework for IoT, *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3088–3098, 2021.

[36] M. Yao, S. Qi, Z. Tian, Q. Li, Y. Han, H. Li, and Y. Qi, Quantifying bytes: Understanding practical value of data assets in federated learning, *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 135–147, 2025.

[37] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-MNIST: A novel image dataset for benchmarking machine learning

algorithms, arXiv preprint arXiv:1708.07747, 2017.
[38] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas, Communication-efficient learning of deep networks from decentralized data, in *Proc. 20th Int. Conf. Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.
[39] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, Federated optimization in heterogeneous networks, in *Proc. 3rd Conf. Machine Learning and Systems*, Austin, TX, USA, 2020.
[40] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, FedBN: Federated learning on Non-IID features via local batch normalization, in *Proc. 9th Int. Conf. Learning Representations*, Virtual Event, 2021.

Zhenzhen Xie received the MS and PhD degrees in computer science from Jilin University, China, in 2014 and 2021, respectively. She is currently an associate researcher at School of Computer Science and Technology, Shandong University, China. Her research areas are edge computing, Internet of Things, and federated learning.

Junjie Pang received the MS and PhD degrees from Jilin University, China, in 2013 and 2017, respectively. He is currently an assistant professor at College of Computer Science and Technology, Qingdao University, China. He is the recipient of the Initiative Postdocs Program of Shandong Province, China. His research interests include federated learning, reinforcement learning, and next-generation networking.

Zhipeng Cai received the MS and PhD degrees from University of Alberta, Canada, in 2004 and 2008, respectively, and the BS degree in computer science from Beijing Institute of Technology, China, in 2001. He is currently an assistant professor at Department of Computer Science, Georgia State University, Atlanta, GA, USA. He has published more than 50 journal papers, including more than 20 IEEE/ACM Transactions papers, such as in *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE/ACM Transactions on Networking*, and *IEEE Transactions on Mobile Computing*. He is the recipient of an NSF CAREER Award. He is an editor/guest editor for Algorithmica, Theoretical Computer Science, *Journal of Combinatorial Optimization*, and *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. He is a fellow of the IEEE. His research agenda focuses on networking, privacy, and big data.

Yan Huang received the PhD degree from Georgia State University, Atlanta, GA, USA, in 2019. He is currently an assistant professor, Kennesaw State University, Atlanta, GA, USA. He is broadly interested in privacy and security, with particular emphasis on deep learning aided privacy protection solutions and cybersecurity challenges in the IoT environment. Now, his research agenda focuses on improving the FLs performance and efficiency and alleviating the contradictions between multiple training tasks.

Olusesi Balogun received the BEng degree in computer engineering from Obafemi Awolowo University, Nigeria, in 2014. He is currently pursuing the PhD degree at Department of Computer Science, Georgia State University, Atlanta, GA, USA. His research interests include insider threat detection, moving target defense, access control models, and security and privacy in cyber-physical systems.

Dongxiao Yu received the BS degree from Shandong University, China, in 2006, and the PhD degree in 2014 from The University of Hong Kong, China, in 2014. He became an associate professor at School of Computer Science and Technology, Huazhong University of Science and Technology, China, in 2016. He is currently a professor at School of Computer Science and Technology, Shandong University, China. His research interests include wireless networks, distributed computing, and graph algorithms.