General Federated Class-Incremental Learning With Lightweight Generative Replay

Yuanlu Chen, Alysa Ziying Tan, *Member, IEEE*, Siwei Feng, Han Yu, *Senior Member, IEEE*, Tao Deng, Libang Zhao, and Feng Wu

***Abstract*—Federated class-incremental learning (FCIL) aims to allow federated learning (FL) systems to consistently learn new tasks with classes that change dynamically, without forgetting knowledge from previous classes. In FCIL scenarios, both heterogeneity in both label and data distribution across clients and catastrophic forgetting caused by continual emergence of new classes can significantly affect the performance of a FL system. Existing FCIL methods assume only changes in class distribution over time for each single client while ignoring class-specific domain distribution. Furthermore, these methods often rely on storing old class exemplars to mitigate catastrophic forgetting, potentially raising privacy concerns and computational burdens. In this article, we propose a FCIL framework called generative federated class-incremental learning (GenFCIL) that effectively addresses the aforementioned challenges. First, we introduce a lightweight generator that promotes knowledge sharing among clients and preserves the accumulated knowledge from all clients. By collecting classes and their associated data from each client, the generator effectively tackles data heterogeneity, facilitating information transfer across clients, and mitigating catastrophic forgetting in a replay-free manner. Importantly, the lightweight nature of the generator ensures that it does not impose excessive memory and computation requirements. Second, to tackle challenges from shifts in both class distribution and class-specific domain distribution in general FCIL scenarios, which may exacerbate catastrophic forgetting, we incorporate and update multiple logit scores from clients focusing on their old and new overlapping classes to incorporate more intraclass information. Experimental results show that GenFCIL effectively alleviates the impact of catastrophic forgetting and heterogeneity.**

***Index Terms*—Catastrophic forgetting, class-specific domain distribution, data heterogeneity, federated class-incremental learning, replay free.**

Manuscript received 10 May 2024; revised 15 June 2024; accepted 22 July 2024. Date of publication 29 July 2024; date of current version 9 October 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62106167; in part by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme under Grant AISG2-RP-2020-019; in part by the Agency for Science, Technology and Research under its RIE2025 Industry Alignment Fund—Industry Collaboration Projects (IAF-ICP) under Award I2301E0026; in part by the Alibaba Group; and in part by NTU Singapore. *(Corresponding author: Siwei Feng.)*

Yuanlu Chen, Siwei Feng, Tao Deng, Libang Zhao, and Feng Wu are with the School of Computer Science and Technology, Soochow University, Suzhou 215000, China (e-mail: 20225227102@stu.suda.edu.cn; swfeng@suda.edu.cn; dengtao@suda.edu.cn; 20225227093@stu.suda.edu.cn; 20224227076@stu.suda.edu.cn).

Alysa Ziying Tan and Han Yu are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: s190109@ntu.edu.sg; han.yu@ntu.edu.sg).

Digital Object Identifier 10.1109/JIOT.2024.3434600

I. Introduction

IN TRADITIONAL centralized machine learning, data is typically collected and sent to a central server for training, which poses privacy risks as data needs to be transmitted from local devices to the server. To address this concern, the paradigm of federated learning (FL) [1] was introduced. FL has gained significant attention in both academia and industry due to its ability to enable collaborative learning of a global model while ensuring privacy protection [2]. However, existing models are limited to static scenarios where the global model learns a single task and local client data remains fixed. In practical settings, there is a need to continuously learn new tasks with dynamically changing classes [3]. This misalignment between realistic demands and existing algorithms has fueled interest in class-incremental learning (CIL). Nevertheless, existing CIL studies typically focus on single agent. In order to combine FL and CIL, federated class-incremental learning (FCIL) is proposed [4].

FCIL aims to design machine learning algorithms to effectively train a global model to learn new classes consecutively. In federated class-incremental learning (FCIL) scenarios, the heterogeneity of label distribution (i.e., disparate label spaces and class distributions) [5], along with varying data distributions across clients [6], [7], can hinder model convergence, introduce bias, and degrade the generalization performance of the global model. As a result, the learned global model experiences significant performance degradation over time. In addition, each client can only access its current data when training local model because of privacy constraints. This may result in the forgetting of knowledge from old classes, known as catastrophic forgetting. Additionally, within each client, there can be overlaps in classes across multiple tasks, and data pertaining to the same classes but obtained from different sessions may be sampled from similar or distinct distributions. These potential shifts in both class distribution and class-specific domain distribution across tasks can exacerbate the problem of catastrophic forgetting. These challenges, as shown in Fig. 1, hinder the development of a reliable global model capable of continual learning.

Existing studies of FCIL [4], [8], [9] mainly rely on storing part of the original data or data generated by generators for the training of subsequent tasks to alleviate negative influences from data heterogeneity and catastrophic forgetting. The proposed approaches usually need a lot of extra memory and more computing resources to store or generate old class

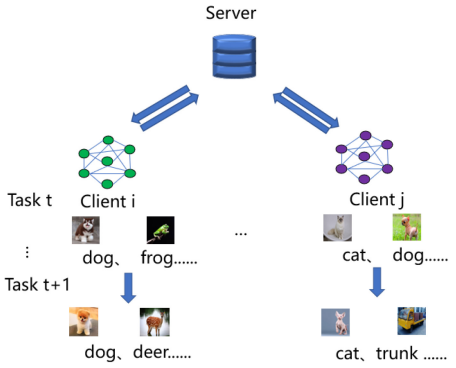

Fig. 1. We investigate the challenges of FCIL in the context of randomly emerging classes, where classes may appear unpredictably. This dynamic setting introduces challenges, such as data heterogeneity, catastrophic forgetting, and shifts in both class distribution (i.e., label sets differing across tasks) and class-specific domain distribution (e.g., the "cat" class may appear with ragdolls in one task and sphynxs in another).

data. However, these methods ignore the possible overlapping of classes between old and new tasks. This situation, which is more responding to the real application scenario, will cause the serious shifts in both class distribution and class-specific domain distribution of FCIL. Moreover, it will further lead to more serious heterogeneity and seriously reduce the effect of the integrated model.

In this article, we propose a FCIL framework called generative FCIL (GenFCIL) to solve the aforementioned challenges. First, we propose the integration of a lightweight generator to assist clients in achieving effective local models in FCIL scenarios. Specifically, during each local task, clients share the set of classes they encounter with the generator. This enables the generator to accumulate comprehensive knowledge about the classes encountered by all clients thus far. Using this accumulated knowledge, the generator generates feature representations for these classes, which are then utilized for subsequent training. These feature representations incorporate information from previous classes and serve as an effective solution to mitigate catastrophic forgetting, eliminating the need to store and preserve old class samples. Importantly, these generated feature representations closely resemble the distribution of real client data, which further reduces the impact of data heterogeneity and enhances the generalization capabilities of the global model. The lightweight design of the generator not only decreases the storage and computational requirements within the FCIL system, but also enhances overall system efficiency. This feature enables the system to operate more efficiently and conserves resources, providing crucial support for stable operation and optimization. Second, it is common to encounter overlapping classes across tasks for individual clients. This means that data belonging to the same class may be sampled from different distributions. For instance, the "dog" class may appear with chihuahuas in one task and huskies in another. Failure to consider such intraclass diversity can result in suboptimal performance of local models, which subsequently affects the overall performance of the global model. To address this challenge, we leverage the representations and classifiers obtained from all clients. The server calculates logits for each client, establishing connections through knowledge transfer, and then fuses their logit scores for the overlapping classes using a knowledge pooler [10]. When a new task is introduced, the server updates the logit scores in a similar manner for each client, with a focus on their old and new overlapping classes. These update scores are then transmitted to assist in training their local models. By fusing the logit scores, our approach effectively handles shifts in both class distribution and class-specific domain distribution while maintaining a balanced representation. This process enables the FCIL system to adapt and learn from the dynamically changing class distributions across tasks, which ultimately enhances the performance of the global model.

The main contributions of this article can be summarized as follows.

1) We introduce GenFCIL, a FCIL framework that leverages a lightweight generator to effectively tackle the challenges of data heterogeneity and catastrophic forgetting in the FCIL scenario without the need for storing old class data.
2) We propose a novel approach that utilizes multiple logit scores to mitigate the impact of shifts in both class distribution and class-specific domain distribution caused by overlapping classes within each client, thereby improving the performance of the proposed framework in complex scenarios.
3) Extensive experiments on four benchmarks demonstrate that GenFCIL consistently outperforms the best baseline in terms of test accuracy.

The remainder of this article is organized as follows. In Section II, we provide an overview of related work. As for Section III, we describe the definition of FCIL and the underlying background of the proposed method. In Section IV, we provide a detailed description of GenFCIL. Section V introduces the data sets, experimental settings, and other relevant details used in our experiments. Finally, we summarize the main contributions and conclude this article in Section VI.

II. Related Work

FCIL aims at designing FL systems with the capability of class-incremental learning. In this section, we provide a detailed overview of methods on FL [11], [12], [13], [14], [15], [16], [17], CIL, and FCIL, as well as the applications of generators in FL.

A. Federated Learning

FL [1], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30] is a decentralized machine learning approach that enables training models without the need for raw data to leave the clients' devices. In a FL system, a central server coordinates multiple clients, including mobile devices and similar entities. These clients retain their respective data locally and conduct model training without sharing the raw data. The central server plays a crucial role in aggregating model updates from the clients, facilitating global model improvement while safeguarding data privacy. The pioneering work of FL [1] introduces a weight-based mechanism for

aggregating local models, enabling collaborative learning of a global model. Subsequent studies attempt to address data heterogeneity in FL. For example, Li et al. [18] incorporated a proximal term for constrained updates during local model training, Li et al. [20] leveraged model-level contrastive learning to correct local training, Dinh et al. [21] employed decoupled model optimization to tackle data heterogeneity, Zhu et al. [22] proposed a generator-based data-free knowledge distillation scheme for heterogeneous FL, Li et al. [31] used local batch normalization to alleviate the feature shift before averaging models, Long et al. [32] clustered clients using their models' parameters, FedRS [33] introduce the concept of "Restricted Softmax" to constrain the adjustment of weights corresponding to absent classes during the local process, FedLC [34] incorporates a finely tuned calibrated cross-entropy loss during local updates by introducing a pairwise label margin, Luo et al. [25] corrected the update directions on both the server-side and worker-side. In recent years, You et al. [29] optimized the interaction between clients and server to facilitate cost-effective model updates and enhance model aggregation, which elegantly addresses the challenges of asynchronous FL for autonomous and intelligent things, demonstrating significant improvements in model accuracy, communication efficiency, and overall performance. Yang et al. [35] introduced an innovative approach to address data heterogeneity in FL, enhancing model performance while preserving privacy through the strategic sharing of partial features. Ye et al. [36] presented an innovative approach to address data heterogeneity in FL by introducing an aggregation mechanism that considers both data set size and the discrepancy between local and global category distributions, demonstrating improved performance and theoretical efficiency. To tackle dynamic FL scenarios where data can accumulate and change gradually, You et al. [30] offered a robust stage-based and layerwise approach that significantly enhances the intelligence and adaptability of Internet of things systems while addressing data protection concerns. However, despite these advancements, existing FL methods still lack an effective mechanism for continuous learning of new classes and mitigating catastrophic forgetting.

B. Class-Incremental Learning

To mitigate catastrophic forgetting, various approaches have been proposed in CIL [37]. These approaches can be broadly categorized into three categories: 1) replay; 2) regularization; and 3) parameter isolation. Replay methods, such as [38] and [39], store exemplars of old classes and utilize them in new tasks, but they require extra memory and processing power [40]. Regularization techniques, like [41] and [42], constrain parameter updates and encourage consistent outputs on previous tasks, without the need for additional resources. Parameter isolation methods, such as [43] and [44], safeguard important parameters from being overwritten during training new tasks. These approaches effectively tackle catastrophic forgetting by preserving performance on prior tasks while updating weights for new tasks. In recent years, there has been a growing interest among researchers in solving the intraclass stability-plasticity dilemma in CIL scenarios. For example, Xie et al. [45] presented a pioneering approach that significantly advances the field of continual learning by introducing a domain-aware learning framework to adeptly handle the challenges of class and domain distribution shifts in real-world applications. However, this approach necessitates the storage of previous data. Wu et al. [10] introduced a novel two-stage training scheme that significantly advances the state-of-the-art in CIL by effectively leveraging strong pretrained models and a proposed score fusion network for robust performance across various challenging scenarios. However, existing CIL methods only focus on single-client scenarios.

C. Federated Class-Incremental Learning

The goal of FCIL is to perform continual learning of new classes in a distributed and privacy-preserving manner using the principles of FL. To address the challenges in FCIL, a proxy server is proposed in [4], while [8] and [9] introduce generative data replay in the FCIL setting. For subsequent studies, Zhang et al. [46] designed a novel method called TARGET, which utilizes the previously trained global model to transfer knowledge from old tasks to the current task at the model level. Additionally, Shenaj et al. [47] proposed a method that tackles an asynchronous setting using prototype-based learning called FedSpace. These methods effectively address catastrophic forgetting and facilitate the acquisition of new knowledge in the FL system. However, most of these methods either store exemplars from old classes or synthesize raw input data to mitigate catastrophic forgetting, which require additional memory space and computing power [4], [8], [46]. In addition, storing or generating raw input to the local models may increase the risk of privacy leakage. Furthermore, practical applications of FL often involve overlapping classes across tasks for each client, which can lead to shifts in both class distribution and class-specific domain distribution among samples belonging to the same classes but from different tasks, thereby potentially degrading the performance of the global model. In this work, we focus on leveraging a generator to address the issues of catastrophic forgetting, data heterogeneity, and shifts in both class distribution and class-specific domain distribution caused by overlapping classes in FCIL without the need for storing previous data.

D. Generator Applications in FL

Generators are versatile tools in computer science and artificial intelligence. In machine learning, they create new data with desired attributes based on input data. This includes generating synthetic images, audio, or text. Notable generator models include generative adversarial networks [48] and variational autoencoders [49]. Traditional applications of generators in FL commonly involve the use of a discriminator and an extractor [8], [50], [51]. The discriminator is a neural network aiming at distinguishing between real and generated samples to provide feedback to the generator to enhance the realism of its outputs. On the other hand, the extractor, also known as an encoder or feature extractor, is responsible for extracting meaningful representations or features from the

input data. Together, these components generate synthetic data and assist in training the model. Nonetheless, conventional generator implementations in FL often demand substantial computational resources and may introduce privacy risks associated with data generation and handling. In order to deal with heterogeneity and catastrophic forgetting without extra memory we try to find a tool to learn the data characteristics of old tasks and apply it to the training of new tasks. Inspired by [22], we propose to use a lightweight generator to mitigate these challenges by reducing overhead, eliminating the need for additional memory, and achieving improved results.

III. Notations and Preliminaries

In this section, we introduce the core concepts and fundamental theories of FCIL, as well as the relevant theoretical background of generator used in GenFCIL.

A. Federated Class-Incremental Learning

FCIL aims to develop a FL algorithm that enables a global model to continuously learn new classes while retaining knowledge of old classes. In this article, we define the input space as $\mathcal{X} \in \mathbb{R}^p$, the representation space as $\mathcal{Z} \in \mathbb{R}^d$ (where $d < p$), and the label space as $\mathcal{Y} \in \mathbb{R}$. In addition, we denote a task as $\mathcal{T}$. Each model, parameterized by $\theta = [\theta^f; \theta^p]$, consists of two components: 1) a representation extractor parameterized by $\theta^f : \mathcal{X} \to \mathcal{Z}$ and 2) a predictor parameterized by $\theta^p : \mathcal{Z} \to \hat{\mathcal{Y}}$, $\hat{\mathcal{Y}}$ means output space.

An FCIL framework usually consists of a central server $\mathcal{S}$ and K clients $\{\mathcal{C}_k\}_{k=1}^K$. Each client $\mathcal{C}_k$ learns sequentially from a series of M_k tasks $\mathcal{T}_k = \{\mathcal{T}_k^t\}_{t=1}^{M_k}$ locally based on its private data $\{\mathcal{D}_k^t\}_{t=1}^{M_k}$ in a class-incremental manner. For the rth global round ($r = 1, 2, \ldots, R$), a portion of the local clients are randomly selected for global model training. Specifically, each $\mathcal{D}_k^t$ contains a set of classes $\mathcal{Y}_k^t$, which is a subset of all possible classes $\mathcal{Y}$. Note that $\mathcal{D}_k^t$ is no longer accessible after task t for each client $\mathcal{C}_k$. Most existing studies on FCIL [4], [8] primarily assume that the label sets for each client's tasks are completely disjoint (i.e., $\mathcal{Y}_k^i \cap \mathcal{Y}_k^j = \emptyset$). However, in practical situations, it is frequently observed that classes overlap across tasks, and training samples with the same label can exhibit variations. To simulate real-world scenarios where edge devices continuously receive new data, we do not impose any constraints on the classes. In this scenario, an extreme case of complete overlap may occur, when $\mathcal{Y}_k^i = \mathcal{Y}_k^j$. These overlapping classes between old and new classes may result in shifts in both class distribution and class-specific domain distribution.

B. Theoretical Background of Generator

A generator is a framework used to estimate generative models through an adversarial process [48]. The goal of the generator is to learn the distribution p_g over data x. To achieve this, researchers define a prior distribution $p_z(z)$ on noise variables, which serves as the input for the generative model G. The purpose of G is to extract meaningful representations from the input data. Additionally, a discriminative model D is employed. D is trained to distinguish between real samples and those generated by G, providing feedback to the generator. The output of $D(x)$ represents the probability that x came from the data rather than from the generator's distribution p_g. In order to maximize the probability of correctly labeling both real and generated samples, the generator G and discriminator D are trained together. The training process aims to optimize the value function $V(G, D)$, where $\min_G \max_D V(G, D) = \mathbb{E}_{z\sim p_{\text{data}}(x)}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$, p_{data} means distribution of data. The generator adjusts its parameters according to this value function to generate more realistic samples. Through this mechanism, the generator can gradually improve the quality of its generated samples. While applying the generator to assist clients in FL, Qi et al. [8] used the generator parameters collected from the selected clients to generate balanced synthetic data. And the final global model is consolidated by the loss of synthetic data and global generator. In this way, each client has its own complex generator, which leads to more computing resources.

IV. Proposed Method

This section first provides an overview of the GenFCIL framework in Section IV-A. The solution to address catastrophic forgetting and data heterogeneity in FCIL by leveraging a generator is discussed in detail in Section IV-B. Our strategy for effectively handling overlapping classes is presented in Section IV-C. And we analyze the advantages, shortcomings and security of GenFCIL in Section IV-D.

A. GenFCIL Framework

Traditional generators typically require large volumes of raw data to ensure the reliability of the generated data. This poses a challenge in FCIL scenarios, as it necessitates either clients uploading their data to a central server housing a global generator or each client maintaining a local generator. The former practice raises privacy concerns, while the latter practice is impractical because clients in an FCIL system often consist of edge devices with constrained computational and storage resources, and their local data might be insufficient for training a dependable generator. Hence, we employ a lightweight generator placed on the server, shifting the training process to the server and avoiding high-computational loads on clients.

Specifically, for each client in the GenFCIL framework, the network is composed of a feature extractor and a fully connected layer for classification. To address the issue of high-data dimensionality and potential calculation overload, we propose a strategy where the features obtained after the feature extraction layers, specifically the outputs before entering the last fully connected layer, are transmitted to the server. It primarily consists of multiple fully connected layers that handle the extracted features. When a new task emerges, clients learn the generated features from the trained generator without additional memory. These extracted features include not only local features but also features from other clients, which significantly enriches the data involved in local model training. As for server, during the aggregation phase, the

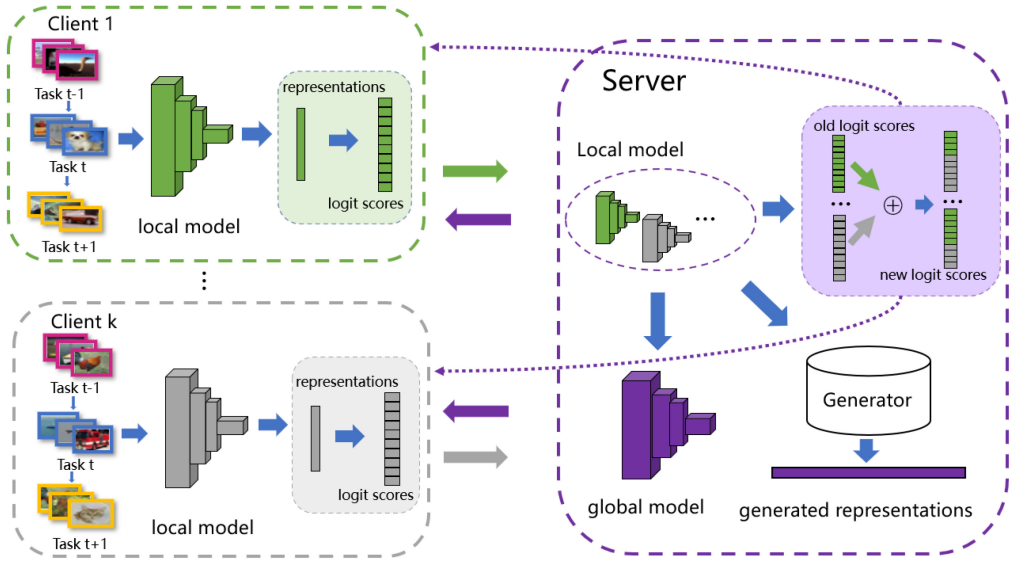

Fig. 2. Overview of our proposed model: Initially, selected clients train their local models and share their trained models and representations with the server. Next, the server uses the collected data and classifiers to train a generator. The generated features are then sent back to the clients for further learning. At the same time, the server computes logit scores, applies processing techniques for overlapping classes, and distributes these scores to all clients alongside the aggregated global model. This iterative process facilitates collaboration and knowledge exchange between clients and the server, effectively addressing the challenges posed by overlapping classes in FCIL.

server computes logit scores based on the features collected from the clients. These logit scores are then returned to each client for further processing. Finally, by employing the federated averaging method, we aggregate all local models by all selected clients to yield the ultimate global model. This comprehensive model is then deployed for evaluating performance on the test data set. An overview of the complete training pipeline is depicted in Fig. 2.

B. Data Generation

Due to the inability of clients to access data from previous tasks, their local models can only be updated with new task data. Unfortunately, this often leads to a significant decline in performance on previous tasks for both local and global models, a phenomenon known as catastrophic forgetting. Additionally, when the data distribution varies across clients, a model trained on one client may struggle to generalize effectively to another client with a different data distribution. Consequently, when the global model is updated using data from a specific client, it may inadvertently discard previously acquired knowledge that is irrelevant to the new client's data distribution. This, in turn, exacerbates the issue of catastrophic forgetting on other clients. To mitigate the adverse effects of catastrophic forgetting without the need to store raw data from old classes, one potential solution is to synthesize data with distributions similar to those encountered previously. To achieve this, we propose to generate data corresponding to the old classes on the server side based on the prototypes of classes that have appeared so far. Subsequently, these synthesized data are sent to clients to combine with real data for local model updates. When receiving new data, clients actively share the corresponding class sets with the generator. The generator thus maintains a comprehensive record of all received classes. When a client finishes local training, it can randomly choose classes from the record. For the selected labels y^t_{kold} from the tth task of client k, the generator produces corresponding representations $\tilde{z}^t_{\text{kold}}$ in response to the clients. This enables clients to acquire old knowledge without the need for additional memory for storing old data.

On the server side, our focus is on extracting information about the overall underlying data distribution, which is not readily observed using conventional FL approaches. To achieve this, a straightforward approach is to learn a conditional distribution $Q_X : \mathcal{Y} \to \mathcal{X}$ to capture and characterize this information

$$Q_X = \underset{G:\mathcal{Y}\to\mathcal{X}}{\operatorname{argmax}} E_{x\sim p(x|y)}\left[\sum_{k=1}^{K} p\left(y|x;\theta_k^t\right)\right] \tag{1}$$

where $p(y|x;\theta_k^t)$ represents the data distribution of the generator for task t and θ_k^t is the local model from client k. However, directly optimizing (1) over the high-dimensional input space $\mathcal{X}$ incurs a heavy computational burden [22]. Therefore, it is important to transform the data by reducing its dimensionality to a more compact form than the raw data space, thereby alleviating the computational burden and protecting client's privacy. A more advanced method involves learning another conditional distribution $Q_Z : \mathcal{Y} \to \mathcal{Z}$ over a more compact dimension

$$Q_Z = \underset{G:\mathcal{Y}\to\mathcal{Z}}{\operatorname{argmax}} E_{z\sim p(z|y)}\left[\sum_{k=1}^{K} p\left(y|z;\theta_k^{p,t}\right)\right] \tag{2}$$

where $\theta_k^{p,t}$ represents the classifier parameter of client k at task t. To achieve this, we propose a generator $G_\omega(\tilde{z}|y)$ that focuses solely on learning the conditional distribution Q_Z rather than Q_X. In contrast to traditional generators that include convolutional layers, our proposed generator is

composed solely of linear layers. Feature integration layers parameterized by ω generate corresponding representations $\tilde{z}$ based on class label y. Through this process, not only does the data dimensionality decrease, but the parameter count of the generator is also reduced, effectively alleviating the computational burden. This reduction in computational burden further enhances communication efficiency within the FCIL framework. Therefore, our ultimate objective is to extract information by training the generator $G_\omega(\tilde{z}|y)$. To achieve this, we optimize the following objective function:

$$\min_{\omega} L_{G_\omega} = E_{z\sim G_\omega(\tilde{z}|y)}\left[\frac{1}{K}\sum_{k=1}^{K} l\Big(h\Big(G\big(y_k^t;\omega\big);\theta_k^{p,t}\Big);y_k^t\Big)\right] \quad (3)$$

where $l(\cdot)$ is cross-entropy loss function, $G(y_k^t;\omega)$ is the generated representations, $h(\cdot)$ is a mapping from $\tilde{\mathcal{Z}}$ to $\hat{\mathcal{Y}}$, and $\tilde{\mathcal{Z}}$ means generated representation space.

To enhance the diversity of knowledge acquired by the clients from the generator, we introduce a random noise vector $\varepsilon \sim \mathcal{N}(0, I)$ into ω, denoted as $G_\varepsilon(\tilde{z}|y)$, as inspired by [22].

As for the client side, we leverage both generated representations and new incoming data to train the local model for the tth task. The loss function for client training with the generator is formulated as

$$L_{\text{Gen}} = \frac{1}{b}\sum_{i=1}^{b} D_{CE}\big(\hat{y}_i, y_i\big) + \alpha \cdot l\Big(h\Big(G_\varepsilon\big(y_k^t;\omega\big);\theta_k^{p,t}\Big);y_k^t\Big) \quad (4)$$

where b is the batch size, $D_{CE}(\cdot,\cdot)$ denotes the binary cross-entropy loss, and α is a parameter for balancing the two terms.

C. Logit Score Computation

Overlapping classes with different distributions at different tasks may lead to shifts in both class distribution and class-specific domain distribution [10]. In FCIL, clients with limited data may encounter the same classes with varying distributions across tasks, which can further exacerbate catastrophic forgetting and data heterogeneity. To address this issue, we propose new logit scores inspired by [10] to tackle the challenges posed by overlapping classes. Considering the top layers of neural networks are more task-specific, we propose that the final classification layer is more vulnerable to shifts in label distribution. Therefore, we have conducted an in-depth study of the classifier layer and identified several issues with the standard Softmax approach, particularly related to missing classes. Additionally, as new data emerges, we aim for the network to retain knowledge from old data while learning new information. Thus, rather than adjusting the entire network structure, we opted for fine-tuning using logit scores. During the local round, clients may receive data from all classes at any time, and the data may come from all classes. Initially, clients train their models locally to achieve high-performance classifiers. Subsequently, they share both the representations $\tilde{z}_k^t$ and the parameters of their local models θ_k^t with the server. In our experiments, we observed that enabling knowledge transfer between clients yields better results compared to individual clients. Building upon this observation, the server leverages the information from other clients to acquire additional knowledge and generate the original logit scores s_k^t. This is achieved by connecting one client's representations to other classifiers and incorporating the resulting logits into the final output. Considering the potential imbalance in data availability among clients, we apply an averaging technique to ensure fairness in the knowledge transfer process

$$s_k^t(z) = \frac{1}{K-1}\sum_{i=1,i\neq k}^{K} h\Big(z_i^t;\theta_k^{p,t}\Big) + h\Big(z_i^t;\theta_i^{p,t}\Big). \quad (5)$$

We form s_k^t by concatenating the logit scores $s_{(k,0)}^t \oplus s_{(k,1)}^t \oplus \cdots \oplus s_{(k,n)}^t$, where $s_{(k,n)}^t$ represents the logit score corresponding to the nth overlapping class. In the presence of overlapping classes, we apply a knowledge pooler, selecting the maximum vector from the overlapping classes to derive the logit Δs_k^t from s_k^t

$$\Delta s_k^t = \Big(\max\Big(s_{(k,1)}^t\Big) \oplus \cdots \oplus \max\Big(s_{(k,n)}^t\Big)\Big)\Big). \quad (6)$$

In essence, when dealing with classes shared among multiple clients, we select the client with the highest probability distribution among those who own these classes. For the remaining classes, we retain the logit scores of the clients. Upon the introduction of new classes, we combine the logit scores from the current Δs_k^t and previous rounds Δs_k^{t-1} to handle overlapping classes [10] and determine the logits

$$\Delta s_k^t = \sigma\Big(\big(\max\big(s_{(k,1)}^t, s_{(k,1)}^{t-1}\big) \oplus \cdots \oplus \max\big(s_{(k,n)}^t, s_{(k,n)}^{t-1}\big)\big)\Big/T\Big) \quad (7)$$

where σ denotes the Softmax activation function and T represents the temperature used for smoothing. Subsequently, the server transmits Δs_k^t to the clients for subsequent training. Finally, we incorporate the computed Δs_k^t into the local training process through the following equation:

$$L_O = l\Big(\Big(\Delta s_k^t \cdot \theta_k^{p,t}; x_k^t\Big); y_k^t\Big). \quad (8)$$

We describe the whole training process in Algorithm 1. The overall loss function in the training process for each client is

$$\min L_g(x,\theta,G) = L_{Gen} + \beta \cdot L_O. \quad (9)$$

Here, β is parameter for balancing the two terms.

D. Analysis and Discussion

1) *Advantages:* GenFCIL utilizes a lightweight generator to assist in the training of the model. In comparison to traditional generators [8], [50], GenFCIL requires fewer parameters due to the simpler structure of its lightweight generator. Unlike the approach of having one generator per client [8], GenFCIL only necessitates a single generator that is placed on the server. This allows GenFCIL to alleviate catastrophic forgetting without requiring additional storage space and enables it to leverage knowledge from other clients. Moreover, GenFCIL tackles the challenge of shifts in both class distribution and class-specific domain distribution, which is a scenario more closely aligned with real-world applications.

Algorithm 1 GenFCIL Framework

Require: Local parameters θ, k^{th} private data $\{x_k, y_k\}$, a tasks $\mathcal{T} = \{\mathcal{T}^t\}_{t=1}^{M_k}$, generator parameters G_ε, the global model $\overline{\theta}$, clients' label set A, batch size b.
for $t = 1$ **to** M_k **do**
  Server randomly selects clients $\{\mathcal{C}_k\}_{k=1}^{K}$ to participate training.
  Server trains generator parameters G_ε by $G_\varepsilon = \frac{1}{|K|}\sum_{k=1}^{K} l\big(G_\varepsilon\big(y_k^t;\omega\big);y_k^t\big)$ using Eq. (3).
  for $k = 1, 2, \ldots, K$ in parallel **do**
    Client $\mathcal{C}_k$ trains locally to obtain $\hat{y}_k^t$, transmits local labels to generator and samples from label set A.
    $L_{Gen} \longleftarrow \frac{1}{b}\sum_{i=1}^{b} D_{CE}(\hat{y}_k^t, y_k^t) + \alpha \cdot l\Big(h\Big(G_\varepsilon\big(y_k^t;\omega\big);\theta_k^{p,t}\Big);y_k^t\Big)$ by Eq. (4).
    Client sends local parameters θ_k^t and representations to the server.
    Server calculates Δs_k^t using Eq. (7) and sends it to the client $\{\mathcal{C}_k\}_{k=1}^{K}$.
  end for
  for $k = 1, 2, \ldots, K$ in parallel **do**
    $L_g \longleftarrow L_{Gen} + \beta L_O$ by Eq. (9).
  end for
  Server updates global parameters with $\overline{\theta} = \frac{1}{K}\sum_{k=1}^{K}\theta_k^t$ and transmits $\overline{\theta}$ to clients.
end for

2) *Limitations:* It is important to note that the lightweight generator employed in GenFCIL may struggle to accurately capture complex data characteristics and generate effective data in cases where the data set is intricate. Therefore, GenFCIL is well-suited for applications involving moderate data complexity and resource-constrained environments.
3) *Security Analysis:* In the training stage, traditional generators with complex structures usually require real data from clients, which causes privacy leakage. To address this, our generator uses fully connected layers and only needs processed data representations. These representations contain less privacy-sensitive information due to their lower dimensionality compared to high-dimensional raw data. Nevertheless, uploading them directly to the server still poses the risk of privacy leakage. To mitigate this, we introduce random Gaussian noise into the representations by following [52]. Although this approach inevitably impacts model performance, our experiments demonstrate that it still significantly outperforms other methods. Additionally, the framework ensures that data from previous tasks is not accessible when a new task arrives, further protecting client privacy.

V. Experiments

In this section, we present an evaluation of our proposed GenFCIL framework on four benchmark data sets: 1) CIFAR-100; 2) CIFAR-10; 3) MNIST; and 4) SVHN. We begin by introducing the data set and then we provide the experiment details. Next, we compare GenFCIL with state-of-the-art approaches on these benchmarks. Finally, we conduct comprehensive experiments to validate the effectiveness of individual components in our design and compare the performance of other factors in GenFCIL.

A. Data Set

CIFAR-100: CIFAR-100 [53] data set is a widely used benchmark in the field of computer vision. It comprises a total of 60 000 RGB images, each measuring 32×32 pixels, These images cover a diverse range of objects and scenes, categorized into 100 distinct classes

CIFAR-10: CIFAR-10 [53] is a commonly used image classification data set containing 60 000 color images sized 32×32 pixels, distributed across 10 classes, with an additional 10 000 images reserved for testing purposes. This data set is typically employed to train and evaluate the performance of image classification algorithms.

MNIST: MNIST [54] is a data set primarily used for digit image classification tasks, encompassing 10 different classes. It consists of a large number of handwritten digit images, each of which is a 28×28 pixel grayscale image. MNIST consists of a training set comprising 60 000 instances and a separate test set comprising 10 000 instances.

SVHN: SVHN [55] data set, which features house numbers captured from Google Street View, includes approximately 73 257 color training images and 26 032 color testing images. The data set exhibits nonuniform image quality and potential issues. All images in the SVHN data set are 32×32 pixels in resolution.

Unlike the traditional class-incremental learning setup, our approach does not strictly adhere to disjoint classes across tasks. Instead, we adopt a more practical scenario where each client randomly selects a subset of classes for each task, allowing for overlapping classes in different tasks. In the case of a 5-task scenario for CIFAR-10, SVHN, MNIST, each class is represented by 300 samples. In the more challenging setting, we challenge GenFCIL further by reducing the number of samples per class to 50 and increasing the total number of tasks to 10.

B. Implementation Details

In our experiments, we utilize ResNet18 [56] as the backbone architecture for all methods. At the beginning of the training process, we set the number of local clients to be 4. For each task, every client randomly selects 40% of the classes from its assigned task's label space for MNIST, CIFAR-10, and SVHN, and 20% for CIFAR-100. Each client then undergoes 20 rounds of training with a dynamic learning rate. And the generator trains itself for 20 rounds using the received data and classifiers at the end of each task. In each round, we set the parameter α to 0.3 for MNIST, SVHN, and CIFAR-10, and 0.5 for CIFAR-100 in the generator component and β to 0.1 in the overall loss function which is used to balance the logical fraction loss in the total loss. During testing, we evaluate the model after each session using the entire test set and report the highest accuracy achieved in each task as the

TABLE I
RESULTS ON MNIST, SVHN, CIFAR-10, AND CIFAR-100. WE REPORT THE ACCURACY OF THE GLOBAL MODEL WHEN ALL CLIENTS COMPLETE THEIR TRAINING ON ALL TASKS, REFERRED TO AS THE GLOBAL ACCURACY

Method	MNIST(sample = 300)	CIFAR-10(sample = 300)	CIFAR100(sample = 50)	SVHN(sample = 300)
Fedavg + lfl	87.27 ± 0.48	61.69 ± 1.6	24.36 ± 0.23	83.03 ± 0.16
Fedavg + lwf	89.37 ± 1.34	57.08 ± 2.16	23.3 ± 0.31	81.18 ± 0.78
Fedprox + lfl	72.88 ± 2.22	30.56 ± 1.45	4.65 ± 0.35	33.25 ± 1.27
Fedprox + lwf	85.31 ± 1.91	40.44 ± 1.02	9.68 ± 0.48	68.06 ± 0.71
Fedprox + mas	82.99 ± 2.4	46.92 ± 0.61	10.78 ± 0.23	71.37 ± 0.16
Fedprox + ewc	85.02 ± 0.59	45.78 ± 0.34	10.69 ± 0.05	71.6 ± 1
Fedavg + ewc	88.64 ± 0.55	61.64 ± 0.26	24.77 ± 0.16	82.02 ± 0.95
Fedavg + mas	89.07 ± 0.72	63.11 ± 0.56	24.66 ± 0.3	82.4 ± 0.25
GLFC	60.09 ± 0.69	60.08 ± 0.69	25.78 ± 0.38	59.40 ± 0.81
FedFed	87.96 ± 0.38	64.16 ± 0.74	22.22 ± 0.23	81.91 ± 0.34
FedDisco	76.35 ± 0.89	58.17 ± 1.92	21.86 ± 2.95	76.96 ± 0.23
GenFCIL	**95.52 ± 1**	**69.8 ± 0.34**	**32.6 ± 0.39**	**83.74 ± 1.27**

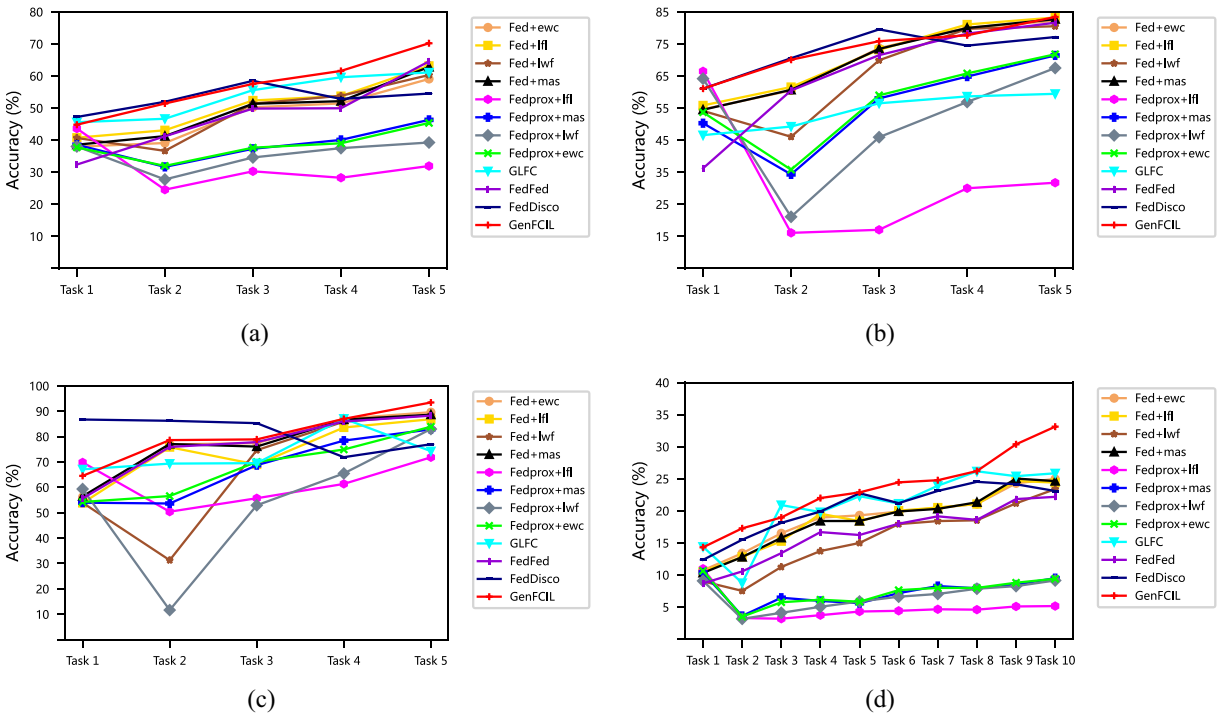

Fig. 3. Qualitative analysis of different incremental tasks on CIFAR-10, SVHN, MNIST, and CIFAR-100.

final result. Our reported results are averaged over 5 runs. Although our experimental setup may appear to reduce the difficulty of the traditional FCIL setting, it more realistically simulates the challenges faced by edge devices with limited data in real-world scenarios. Therefore, our research holds significance in addressing these practical concerns.

C. Performance Evaluation

We combined several classical continual learning methods, namely, EWC [57], LWF [41], LFL [42], and MAS [58], with two classical FL methods: 1) FedAvg [1] and 2) FedProx [18], and used these combined methods, FedFed [35] has some connection with our method, FedDisco [36] which the research problem is more similar to ours and has achieved obvious results and GLFC [4] to compare with GenFCIL. We did not compare our method with the other three FCIL-relevant works mentioned in Section II-C [8], [9], [46], [47]. In [47], the focus is primarily on addressing asynchronous issues in FCIL scenarios. Since this does not align with the specific problems addressed in this article, it is not directly relevant to our research. Approaches proposed in [8], [9], and [46] employ complex generators to synthesize raw input data. While these generators may produce higher quality images, they also require increased computational resources and storage space. This difference in resource requirements could introduce unfairness in the comparison between these methods and ours. In addition, generating raw data may increase the risk of privacy leakage. Considering these factors, we decided not to include these three methods in our comparison. The results are depicted in Table I, which reveal that GenFCIL outperforms the federated continual methods in a FCIL scenario. Furthermore, we visualize the training results for all tasks using both the comparison methods and GenFCIL in Fig. 3(e). The results indicate that the performance of

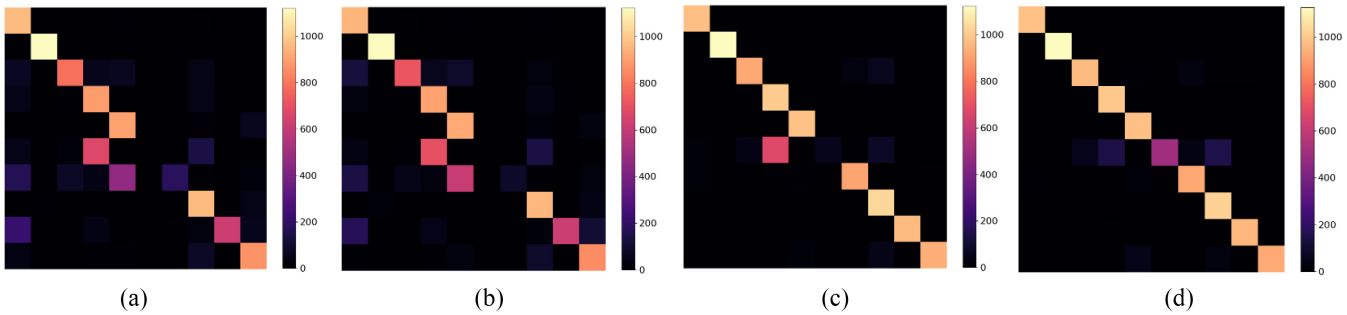

Fig. 4. Confusion matrices for the global model on the server in Fed+lfl, Fed+ewc, Fed+mas, and GenFCIL on MNIST.

some comparison methods experience a rapid drop when the second task is introduced, particularly noticeable in SVHN. Concretely, with FedProx+lfl, FedProx+lwf, FedProx+ewc, FedProx+mas, and Fed+lwf. In contrast, GenFCIL exhibits a steady upward trend, demonstrating its superior stability compared to the other methods. In addition, FedDisco showed excellent performance at first on MNIST, but with the emergence of new tasks, the performance gradually declined, which was not as good as our method. It is worth noting that for all methods, the accuracy is on the rise after the arrival of new tasks, which is different from the continuous decline of accuracy rate in traditional continual learning. This is because we have not restricted the classes between the old and new tasks, and clients may encounter classes with new data that they have learned before during the training process, including old classes and new data, helps to improve the accuracy of the final global model. Our method appears to have less advantage in terms of variance. This is partly due to the simple architecture of the lightweight generator, which limits its learning capacity. Additionally, throughout the training process, the server-side generator learns representations with random Gaussian noise added to protect user privacy. This noise can lead to performance degradation, as it negatively impacts model performance [35] and affects the stability of the generated synthetic data's quality. For the client, the synthetic representation obtained from the generator during each training process has a certain randomness, which further impacts the stability of the final aggregation model to some extent. Consequently, this can lead to an unstable training process. Despite these challenges, our method remains significantly superior to the comparative methods in terms of overall effectiveness.

In Fig. 4(e), we provide the confusion matrices for the global model on the server, comparing the results of some of the comparison methods with GenFCIL on the MNIST data set. We can clearly observe that the three methods of Fed+mas, Fed+lfl and Fed+ewc all perform poorly in the classification of the sixth and seventh pictures, which are significantly lower than GenFCIL. The displayed confusion matrices validate the significantly superior classification performance achieved by GenFCIL compared to other methods and emphasize its extraordinary ability in accurately identifying patterns and categories in data sets.

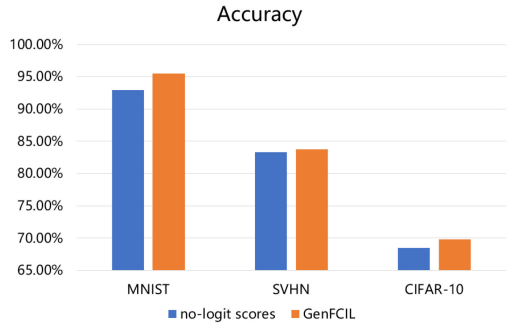

Fig. 5. Effects of logit scores on MNIST, SVHN, and CIFAR-10.

TABLE II
EFFECTS OF THE GENERATOR IN GENFCIL ON MNIST, SVHN, CIFAR-10, AND CIFAR-100

	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
Accuracy(%) MNIST	92.47 ± 1.35	95.52 ± 1	95.32 ± 0.62
Accuracy(%) SVHN	83.4 ± 0.22	83.74 ± 1.27	83.57 ± 0.4
Accuracy(%) CIFAR-10	69.26 ± 0.55	69.8 ± 0.34	69.6 ± 0.49
Accuracy(%) CIFAR-100	27.22 ± 0.32	29.1 ± 0.56	32.6 ± 0.39

D. Ablation Analysis

Effect of Logit Scores: We conduct a comprehensive evaluation of the impact of logit scores in GenFCIL through meticulous ablation experiments, as elucidated in Fig. 5. Taking into account the relatively smaller class distributions within each task of the CIFAR-100 data set in comparison to other data sets, our investigation extends to alternative data sets, including MNIST, SVHN, and CIFAR-10. The outcomes revealed a discernible trend whereby the overall performance of GenFCIL surpassed that of the generator when operating in isolation.

E. Sensitivity Analysis

Effect of Generator: Different values of the hyperparameter α in (4) have a significant impact on the generator's performance, as shown in Table II. An α value of 0.3 yields notably superior results for MNIST, SVHN and CIFAR-10. Beyond this value, the generator's effectiveness plateaus. However, for CIFAR-100, an α value of 0.5 performs the best.

TABLE III
EFFECTS OF GENERATOR SAMPLING SIZE ON MNIST

Effect of Generator sampling size on MNIST	Accuracy(%)
$B_G = 8$	94.1
$B_G = 16$	94.73
$B_G = 32$	93.45
$B_G = 64$	95.52
$B_G = 128$	95.82

TABLE IV
EFFECTS OF GENERATOR NETWORK STRUCTURE ON MNIST

Effect of Generator Network Structure. on MNIST	Accuracy(%)
$d_h = 32$	93.31
$d_h = 64$	94.56
$d_h = 128$	95.26
$d_h = 256$	95.52
$d_h = 512$	93.57

Notably, varying α leads to significant accuracy improvements for MNIST and CIFAR-100, but less so for CIFAR-10 and SVHN. This is due to differences in data set characteristics, where CIFAR-100 faces challenges of data inadequacy and numerous classes and the content of MNIST data set is relatively simple, and it is not difficult for generator to learn. In these cases, the generator's data expansion mechanism assumes a pivotal role in bolstering the performance of the local models, enabling them to learn and generalize patterns more effectively. The generator's contribution in expanding the data not only improves the robustness and accuracy of the local models but also facilitates better decision-making and classification outcomes in complex scenarios. Conversely, CIFAR-10 and SVHN benefit from their larger data sets and simpler nature. Overall, the generator proves valuable in scenarios with limited data and complex tasks.

Effects of the Generator'S Network Architecture and Sampling Size: We investigate the impact of both network structure and sampling size on the generator. In Table III, the results demonstrate an overall performance improvement with an increase in the number of client samples. Furthermore, Table IV delves into exploring various hidden layer configurations (d_h) of the generator while maintaining fixed output layer dimensions (i.e., the generated representation space dimension) and input noise ε dimensions. The outcomes reveal that the performance of models with different hidden layers is somewhat unstable. In a certain range, the more layers of the model structure, the stronger the learning ability of the model, the better it can learn data and generate better synthetic data. However, when the complexity of the data set is not high, the learning ability of the model is too strong, which may lead to over-fitting, thus adversely affecting the training effect. And we observe that generator network structure $d_h = 256$ exhibits superior performance compared to others.

Effect of Training Epochs: In our study, we explored how the number of training epochs influences performance. Various comparison methods were trained for 20, 30, and 40 epochs, and their results were compared in Fig. 6. From the picture, we can observe that the effects of Fed+ewc and Fed+mas are increasing with the increasing number of epoch, but they are

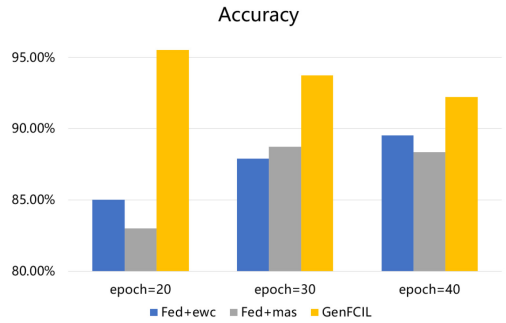

Fig. 6. Effects of the training epochs among Fed+ewc, Fed+mas, and GenFCIL on MNIST.

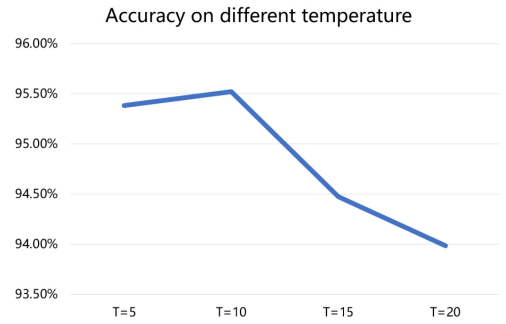

Fig. 7. Effects of the temperature in logit scores on MNIST.

obviously lower than the effect of GenFCIL. As for GenFCIL, different from the other two methods, its effect decreases with the increase of training rounds. The findings revealed that GenFCIL outperformed the other methods when trained for 20 epochs, demonstrating its efficiency and effectiveness. This suggests that GenFCIL's innovative parameter sharing strategy has the potential to reduce communication overhead by delivering superior results in fewer training iterations.

Effect of Temperature in Logit Scores: We conducted a comprehensive analysis to investigate the influence of various temperatures on the logit scores using the MNIST data set. The results in Fig. 7 demonstrate that the model's performance experiences enhancement with increasing temperature values below 10. However, once the temperature exceeds the threshold of 10, we observed a decline in the model's effectiveness as the temperature continues to increase. These findings lead us to identify a temperature of $T = 10$ as the optimal choice for our model, striking a balance between performance enhancement and avoiding diminishing returns.

VI. CONCLUSION

We propose GenFCIL, a general federated class-incremental learning (FCIL) framework designed to overcome the challenges of catastrophic forgetting and facilitate knowledge transfer among clients, all without the need for external data or memory. In addition, we leverage logit scores to address shifts in both class distribution and class-specific domain distribution caused by imbalanced class and data during training. Through extensive experimentation on CIFAR-10, CIFAR-100, MNIST, and SVHN data sets, we demonstrate the remarkable effectiveness of GenFCIL in tackling the complexities of this

emerging scenario. Ablation experiments further validate the contributions and efficacy of each proposed module within our framework. GenFCIL's key strengths lie in its lightweight generator, efficient resource utilization, knowledge sharing capability, and the ability to handle shifts in both class distribution and class-specific domain distribution. It is worth mentioning that this research introduces a novel direction in the field of FCIL by being the first study to consider overlapping classes across tasks. The findings of this study may open up new possibilities and avenues for exploration in the field.

However, it is important to note that the lightweight generator used in GenFCIL may face challenges when accurately capturing intricate data characteristics and generating effective data. Consequently, GenFCIL is better suited for applications involving moderate data complexity and resource limitations. To address this limitation, we aim to enhance GenFCIL's capability to handle intricate data in future work. It would also be interesting to explore the integration of asynchronous collaborative training strategies into the GenFCIL framework to handle the heterogeneity among participating clients.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[2] W. Guo, Y. Zhang, X. Cai, L. Meng, J. Yang, and X. Yuan, "LD-MAN: Layout-driven multimodal attention network for online news sentiment recognition," *IEEE Trans. Multimedia*, vol. 23, pp. 1785–1798, Jun. 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9121752/citations#citationsformoredetails.

[3] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 1234–1241.

[4] J. Dong et al., "Federated class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10164–10173.

[5] T. Sheng et al., "Modeling global distribution for federated learning with label distribution skew," *Pattern Recognit.*, vol. 143, Nov. 2023, Art. no. 109724.

[6] A. Ahmad, W. Luo, and A. Robles-Kelly, "Robust federated learning under statistical heterogeneity via hessian spectral decomposition," *Pattern Recognit.*, vol. 141, Sep. 2023, Art. no. 109635.

[7] Z. Liu, F. Wu, Y. Wang, M. Yang, and X. Pan, "FedCL: Federated contrastive learning for multi-center medical image classification," *Pattern Recognit.*, vol. 143, Nov. 2023, Art. no. 109739.

[8] D. Qi, H. Zhao, and S. Li, "Better generative replay for continual federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2023, pp. 1–17.

[9] S. Babakniya, Z. Fabian, C. He, M. Soltanolkotabi, and S. Avestimehr, "A data-free approach to mitigate catastrophic forgetting in federated class incremental learning for vision tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 66408–66425, 2023.

[10] T.-Y. Wu et al., "Class-incremental learning with strong pre-trained models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9601–9610.

[11] Z. Qi, L. Meng, Z. Chen, H. Hu, H. Lin, and X. Meng, "Cross-silo prototypical calibration for federated learning with non-IID data," in *Proc. ACM Int. Conf. Multimedia*, 2023, pp. 3099–3107.

[12] A. Wuerkaixi et al., "Accurate forgetting for heterogeneous federated continual learning," in *Proc. Int. Conf. Learn. Represent.*, 2023, pp. 1–19.

[13] X. Fang and M. Ye, "Robust federated learning with noisy and heterogeneous clients," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10072–10081.

[14] Z. Wang et al., "FLGo: A fully customizable federated learning platform," 2023, *arXiv:2306.12079*.

[15] F. Zhang et al., "Exploiting class activation value for partial-label learning," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–17.

[16] Y. Mao et al., "Communication-efficient federated learning with adaptive quantization," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–26, 2022.

[17] L. You et al., "FMGCN: Federated meta learning-augmented graph convolutional network for EV charging demand forecasting," *IEEE Internet Things J.*, vol. 11, no. 14, pp. 24452–24466, Jul. 2024.

[18] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, 2020.

[19] L. Lyu et al., "Privacy and robustness in federated learning: Attacks and defenses," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 8726–8746, Jul. 2024.

[20] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 10713–10722.

[21] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with Moreau envelopes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 21394–21405.

[22] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12878–12889.

[23] Q. Huang, J. Zhang, Z. Zeng, D. He, X. Ye, and Y. Chen, "PPDF-FedTMI: A federated learning-based transport mode inference model with privacy-preserving data fusion," *Simulat. Model. Pract. Theory*, vol. 129, Dec. 2023, Art. no. 102845.

[24] Y. Tan, G. Long, J. Ma, L. Liu, T. Zhou, and J. Jiang, "Federated learning from pre-trained models: A contrastive learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 19332–19344.

[25] K. Luo, X. Li, Y. Lan, and M. Gao, "GradMA: A gradient-memory-based accelerated federated learning with alleviated catastrophic forgetting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 3708–3717.

[26] Y. Liu, T. Lin, and X. Ye, "Federated recommender systems based on deep learning: The experimental comparisons of deep learning algorithms and federated learning aggregation strategies," *Expert Syst. Appl.*, vol. 239, Apr. 2024, Art. no. 122440.

[27] M. Ilić, M. Ivanović, V. Kurbalija, and A. Valachis, "Towards optimal learning: Investigating the impact of different model updating strategies in federated learning," *Expert Syst. Appl.*, vol. 249, Sep. 2024, Art. no. 123553.

[28] G. Rjoub, O. A. Wahab, J. Bentahar, and A. Bataineh, "Trust-driven reinforcement selection strategy for federated learning on IoT devices," *Computing*, vol. 106, no. 4, pp. 1273–1295, 2024.

[29] L. You, S. Liu, B. Zuo, C. Yuen, D. Niyato, and H. V. Poor, "Federated and asynchronized learning for autonomous and intelligent things," *IEEE Netw.*, vol. 38, no. 2, pp. 286–293, Mar. 2024.

[30] L. You, Z. Guo, B. Zuo, Y. Chang, and C. Yuen, "SLMFed: A stage-based and layer-wise mechanism for incremental federated learning to assist dynamic and ubiquitous IoT," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 16364–16381, May 2024.

[31] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-IID features via local batch normalization," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–27.

[32] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning: Clients clustering for better personalization," *World Wide Web*, vol. 26, no. 1, pp. 481–500, 2023.

[33] X.-C. Li and D.-C. Zhan, "FedRS: Federated learning with restricted Softmax for label distribution non-iid data," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Min.*, 2021, pp. 995–1005.

[34] J. Zhang et al., "Federated learning with label distribution skew via logits calibration," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 26311–26329.

[35] Z. Yang et al., "FedFed: Feature distillation against data heterogeneity in federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 1–32.

[36] R. Ye, M. Xu, J. Wang, C. Xu, S. Chen, and Y. Wang, "FedDisco: Federated learning with discrepancy-aware collaboration," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 39879–39902.

[37] Z. Qin et al., "Rethinking few-shot class-incremental learning: A lazy learning baseline," *Expert Syst. Appl.*, vol. 250, Sep. 2024, Art. no. 123848.

[38] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2001–2010.

[39] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, pp. 3302–3309, 2018.

[40] W. Sun, Q. Li, J. Zhang, D. Wang, W. Wang, and Y.-A. Geng, "Exemplar-free class incremental learning via discriminative and comparable parallel one-class classifiers," *Pattern Recognit.*, vol. 140, Aug. 2023, Art. no. 109561.
[41] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
[42] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–10.
[43] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7765–7773.
[44] Z. Fu, Z. Wang, X. Xu, D. Li, and H. Yang, "Knowledge aggregation networks for class incremental learning," *Pattern Recognit.*, vol. 137, May 2023, Art. no. 109310.
[45] J. Xie, S. Yan, and X. He, "General incremental learning with domain-aware categorical representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 14351–14360.
[46] J. Zhang, C. Chen, W. Zhuang, and L. Lyu, "TARGET: Federated class-continual learning via exemplar-free distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 4782–4793.
[47] D. Shenaj, M. Toldo, A. Rigon, and P. Zanuttigh, "Asynchronous federated continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2023, pp. 5055–5063.
[48] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.
[49] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.
[50] Y. Wu et al., "FedCG: Leverage conditional GAN for protecting privacy and maintaining competitive performance in federated learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 2334–2340.
[51] S. Augenstein et al., "Generative models for effective ML on private, decentralized datasets," 2019, *arXiv:1911.06679*.
[52] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.
[53] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Rep. TR-2009, 2009.
[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[55] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 1–9.
[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[57] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017.
[58] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 139–154.

Yuanlu Chen receives the B.S. degree in mathematics and applied mathematics from Chengdu University of Technology, Chengdu, China, in 2022. She is currently pursuing the M.S. degree in computer engineering with the School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu, China.

Her research interests include machine learning and computer vision.

Alysa Ziying Tan (Member, IEEE) received the master's degree in intelligent systems from National University of Singapore, Singapore, in 2020, and was awarded the Inaugural IMDA Singapore Digital Postgraduate Scholarship. She is currently pursuing the Ph.D. degree with Alibaba-NTU Joint Research Institute, Nanyang Technological University, Singapore.

Prior to her graduate studies, she gained valuable experience as a Data Scientist, where she led numerous large-scale deep learning and optimization projects across diverse domains, including manufacturing, finance, insurance, and supply chain management. Her research interests include federated learning, continual learning, and computer vision.

Siwei Feng received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA, USA, in 2019.

He is currently an Assistant Professor with the School of Computer Science and Technology, Soochow University, Suzhou, China. He has published over 20 research papers in leading international journals and conferences. His research interests include machine learning and signal processing.

Dr. Feng is a member of CCF.

Han Yu (Senior Member, IEEE) received the Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore, in 2014.

He is a Nanyang Assistant Professor with the School of Computer Science and Engineering, NTU. He held the Prestigious Lee Kuan Yew Postdoctoral Fellowship from 2015 to 2018. He has published over 250 research papers and book chapters in leading international conferences and journals. He is a co-author of the book *Federated Learning*—the first monograph on the topic of federated learning. His research focuses on federated learning and algorithmic fairness.

Dr. Yu's research works have won multiple awards from conferences and journals. He is a Distinguished Member of CCF, and a Senior Member of AAAI.

Tao Deng received the Ph.D. degree from Southwest Jiaotong University, Chengdu, China, in 2019.

He is currently a Lecturer with the School of Computer Science and Technology, Soochow University, Suzhou, China. From July 2019 to June 2022, he was a Research Engineer with Huawei Technologies Company Ltd., Shanghai, China. From September 2016 to January 2018, he was a visiting Ph.D. student with the Department of Information Technology, Uppsala University, Uppsala, Sweden. His research interests include mobile edge computing, wireless caching, and large-scale optimization for wireless networks.

Libang Zhao receives the B.S. degree in computer science and technology from Southwest University of Science and Technology, Mianyang, China, in 2022. He is currently pursuing the M.S. degree in artificial intelligence with the School of Computer Science and Technology, Soochow University, Suzhou, China.

His research interests include machine learning and computer vision.

Feng Wu received the B.S. degree in vehicle engineering from Anhui Agricultural University, Hefei, China, in 2022. He is currently pursuing the M.S. degree in computer engineering with the School of Computer Science and Technology, Soochow University, Suzhou, China.

His research interests include machine learning and computer vision.