OVERCOMING CATASTROPHIC FORGETTING IN FEDERATED CLASS-INCREMENTAL LEARNING VIA FEDERATED GLOBAL TWIN GENERATOR

Anonymous authors
Paper under double-blind review

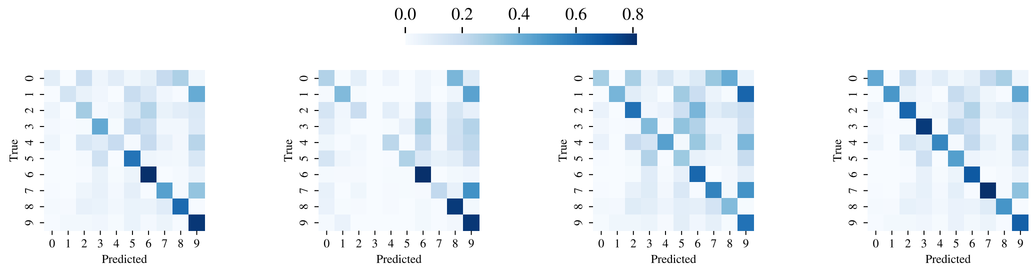
ABSTRACT

Federated Class-Incremental Learning (FCIL) increasingly becomes essential in the decentralized setting, where it enables multiple participants to collaboratively train a global model to perform well on a sequence of tasks without sharing their private data. In FCIL, conventional Federated Learning algorithms such as FedAvg often suffer from catastrophic forgetting, resulting in significant performance declines on earlier tasks. Recent works based on generative models produce synthetic images to help mitigate this issue across all classes. However, these approaches' testing accuracy in previous classes is still much lower than recent classes, i.e., having better plasticity than stability. To overcome these issues, this paper presents Federated Global Twin Generator (FedGTG), an FCIL framework that exploits generative-model training on the global side without accessing client data. Specifically, the server trains a data generator and a feature generator to create two types of information from all seen classes. Then, it sends the synthetic data to the client. The clients then use feature-direction-controlling losses to make the local models retain knowledge and learn new tasks well. We extensively analyze the robustness of FedGTG on natural images and its ability to converge to flat local minima and achieve better predicting confidence (calibration). Experimental results on CIFAR-10, CIFAR-100, and tiny-ImageNet demonstrate the improvements in accuracy and forgetting measures of FedGTG as well as the robustness of domain shifts compared to previous frameworks.

1 INTRODUCTION

Federated Learning (FL) (McMahan et al., 2016; Bonawitz et al., 2019) is a Machine Learning setting that facilitates collaborative learning while maintaining privacy. Despite its significant achievements on various domains (Doshi & Yilmaz, 2022; Lin et al., 2021; Liu & Yang, 2021; Nguyen et al., 2019), FL observes several critical challenges, including resource limitation and data heterogeneity. Moreover, the client's local data distribution is assumed to remain unchanged, but the real-world scenarios (Aljundi, 2019) can be different, where the client's task, data, and domain can be changed. To overcome such challenges, Federated Class-Incremental Learning (FCIL) (Dong et al., 2022; 2023) is an innovative approach that combines the principles of FL and Class-Incremental Learning (CIL) (Rebuffi et al., 2017) to enable models to learn continuously from decentralized data sources while adapting to new information over time without forgetting previous knowledge (French, 1999). This approach addresses data privacy challenges and ensures the model can evolve as new data types or tasks emerge without accessing historical data. In CIL, exemplar-based approaches (Rebuffi et al., 2017; Chaudhry et al., 2019; Buzzega et al., 2020) preserve a limited number of samples from previous tasks to prevent forgetting, whereas exemplar-free approaches (He et al., 2018; Liu et al., 2020; Magistri et al., 2024) do not retain any samples from prior tasks.

In the FL setting, where privacy issues pose significant challenges, the exemplar-free category is particularly interesting since users cannot store historical data. Recent works in this field, such as TARGET (Zhang et al., 2023), FedCIL (Qi et al., 2023) and MFCL (Babakniya et al., 2024) tend to generate synthetic data and combine with distilling regularizers (Hinton et al., 2015; Liu et al., 2020) to balance the trade-off between retaining knowledge and learning new tasks. However, experimental results have shown that these works still witness catastrophic forgetting, i.e., bias



(a) TARGET (b) MFCL (c) $G_D + G_F$ (d) FedGTG (Our)

Figure 1: Confusion matrix among FCIL algorithms: **(a)** TARGET, **(b)** MFCL, **(c)** only the application of two generators to FL, and **(d)** FedGTG, testing on CIFAR-10 after training is completed. While TARGET and MFCL have bad predicting performance on initial classes and two generators struggle to learn new tasks, FedGTG achieves a better stability-plasticity trade-off.

towards recent classes, as shown Figure 1a and 1b. This is because the model trained by MFCL and TARGET is closely linked to its data generator. When the model begins to lose previous knowledge, it impacts the data generator, leading to the production of poor-quality synthetic images of earlier classes. Consequently, the model's performance on these classes in later tasks will significantly drop (Babakniya et al., 2024).

To address this problem, we propose **Federated Global Twin Generator** (FedGTG), an FCIL framework that does not store client data. Specifically, after completing one task, the server trains two generative models and shares them with clients on subsequent tasks to create synthetic examples and features of previous classes. On the client side, we propose a synthetic logit distillation using generated features for retaining knowledge and a fine-tuning loss using both real and generated data to be able to predict all classes. However, using only these two objectives still hinders the model's ability to obtain new knowledge, as shown in Figure 1c. We argue that this issue happened as the feature directions were not constrained. Therefore, we add a feature-direction-controlling loss, which helps the model have more plasticity in learning new tasks. As a result, FedGTG outperforms previous methos in accuracy and forgetting measures as shown in Figure 1d and Section 4.2.

We summarize our contributions below:

- We propose an FCIL framework that trains a data generator and a feature generator on the server side. These generators are distributed to the clients to mitigate forgetting.
- To help the model have a better stability-plasticity trade-off, we propose direction-controlling objectives on the client side.
- We conducted extensive experiments to demonstrate the effectiveness of our method in popular benchmarks and handling domain shifts.
- Moreover, we analyze the robustness of FedGTG compared with recent FCIL algorithms on natural images, its abilities to converge to flat minima, achieve better predicting confidence, and maintaining the effectiveness across different client sizes.

2 Related work

2.1 Continual Learning

Catastrophic forgetting (French, 1999) is a significant issue in machine learning where training a model on new data makes it forget previously learned knowledge. This issue is central to the field of CL, whose primary objective is to build models to acquire new knowledge while retaining information from older tasks. Numerous strategies have been explored to mitigate this problem, including the implementation of regularization terms (Li & Hoiem, 2017; Kirkpatrick et al., 2017; Zenke et al., 2017), the isolation of architectural parameters (Mallya & Lazebnik, 2018; Yoon et al., 2017; Mallya et al., 2018), the use of storing prior data (Rebuffi et al., 2017; Chaudhry et al., 2019; Buzzega et al., 2020), and studies of data generation (He et al., 2018; Zhuang et al., 2022; 2023; 2024; Magistri et al., 2024). In CL, replay-based methods observe significant performance in accuracy and forgetting measures. However, privacy concerns in FL prevent data storage (Dong et al., 2022), making these methods inapplicable. An extensive alternative to address this issue is generative-based approaches.

These mitigation strategies become more crucial depending on the type of learning in CL. Specifically, there are three main types: Task-Incremental Learning (Task-IL), Domain-Incremental Learning (Domain-IL), and Class-Incremental Learning (Class-IL) (Van de Ven & Tolias, 2019). Each task is distinct in Task-IL and comes with its distribution during training and testing. In Domain-IL, the learning task does not change, while different domains or data distributions sequentially arrive during training. In Class-IL, each new task adds classes to what the model has to learn, continually expanding the amount of information the model needs to handle.

2.2 Federated Class-Incremental Learning

FCIL aims to train a model to learn new classes over time without forgetting previously learned classes while ensuring that data privacy is maintained across multiple decentralized devices. Several approaches exploit Knowledge Distillation (Hinton et al., 2015) to mitigate forgetting by appointing the global model's weight of the most recent task as a teacher. **Continual Federated Learning with Distillation (CFeD)** (Ma et al., 2022) constructs server and client-side knowledge distillation using a surrogate dataset, but this costs time and financial resources to collect enough data for this surrogate dataset. **Global-Local Forgetting Compensation (GLFC)** (Dong et al., 2022) relaxes this problem by training a proxy server globally to ease the imbalance issue between classes.

The above approaches yield extensive performance in past knowledge retention but cannot learn well on new tasks. To alleviate this issue, **Federated Class-Incremental Learning (FedCIL)** (Qi et al., 2023) trains generators at both client-side and server-side, as well as utilizing knowledge distillation to balance the stability-plasticity trade-off. However, this approach raised a privacy risk since information about client-side generative models is shared with the server. **Federated Class-Continual Learning via Exemplar-Free Distillation (TARGET)** (Zhang et al., 2023) and **Mimicking Federated Continual Learning (MFCL)** (Babakniya et al., 2024) relax this issue by training a data generator on the global-side and adding distilling regularizers to the client-side training to enhance overall performance. However, as shown in Figure 1, these methods still perform badly on old classes, leaving catastrophic forgetting mitigation a desirable goal.

3 Methodology

3.1 Preliminaries

There are c clients and a central server, denoted as $\{C_1, C_2, \ldots, C_c\}$ and S, respectively. We consider the Synchronous Federated Continual Learning setting (Yang et al., 2024) where all clients share the same task sequence $T = \{T_1, T_2, \ldots, T_n\}$. At task T_t, each client C_i has a private dataset $\mathcal{D}_i^t = \{\mathcal{X}_i^t, \mathcal{Y}^t\}$. During the first task, the global model θ_G^1 is obtained after aggregating local models $\left\{\theta_1^1, \theta_2^1, \ldots, \theta_{s_1}^1\right\}$ using conventional Deep Learning methods, where s_1 is a number of selected clients among all. From the task T_t, $t \geq 2$, the global model θ_G^{t-1} can distinguish the samples belonged to the classes set $\bigcup_{i=1}^{t-1} \mathcal{Y}_i$. The server then distributes its parameters back to the clients. Client C_i uses θ_G^{t-1} as an initial model to train on task T_t using its private dataset $\mathcal{D}_i^t$. The local model θ_i^t should perform well in classifying classes from the set $\bigcup_{i=1}^{t} \mathcal{Y}_i$. Finally, the server collects the local models from clients who participate in the process after each r_t communication round and obtains a new global model θ_G^t, which can identify classes from the set $\bigcup_{i=1}^{t} \mathcal{Y}_i$.

3.2 Overview

Several replay-based approaches (Rebuffi et al., 2017; Chaudhry et al., 2019; Buzzega et al., 2020) in the conventional CL achieve significant performance across all classes by storing a subset of samples from previous tasks. However, these methods are not viable in the FL setting due to privacy concerns (e.g., local hospitals cannot share data with the central server). One initial solution is using generative models, which can generate synthetic data for subsequent training, as demonstrated in earlier studies (Zhang et al., 2023; Babakniya et al., 2024). However, only generating synthetic examples causes

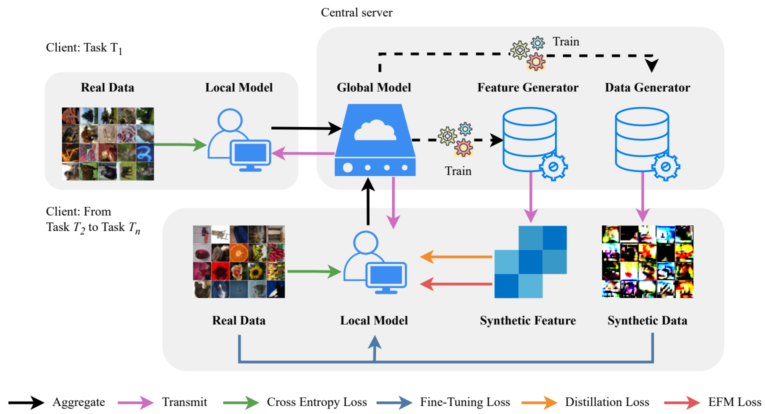


Figure 2: Illustration of the proposed framework. After completing one task, the server employs a data-free approach to train two generators. The clients then use two types of synthetic information from these generators to train their local models for retaining knowledge and learning new tasks well.

forgetting in previously learned classes, as shown in Figure 1. This is because synthetic images cannot fully reflect the knowledge from prior works (Liu et al., 2020). The authors also show that synthetic features can alleviate this problem by capturing the information held within the model's weights. Therefore, we also train a feature generator in addition to the data generator. Specifically, we propose a Federated Global Twin Generator (FedGTG), which can balance the stability-plasticity trade-off. This method has two main stages: (1) At the end of each task, the server trains a data generator and a feature generator to capture the information of all seen classes; (2) Clients receive generators from the server to create synthetic information, and obtaining global weights as initialization, which helps retain knowledge from previous tasks and learn the new task efficiently.

3.3 Server-side Data Generator

Since the server only has access to the global model's weights, we can only train the data generator using data-free methods, such as DeepInversion (Yin et al., 2020). Specifically, we have a generative model that takes a noise $z \sim \mathcal{N}(0, 1)$ as input and produces a synthetic example $\tilde{x}$ mirroring the dimensions of the original training input. This synthetic data should observe the following objectives.

After training task T_t, the synthetic data should be classified correctly by the global model θ_G^t and not be biased to any classes. With this aim, we employ a modified cross entropy classification loss between its assigned label z and the prediction of θ_G^t on $G_D^t(z)$, as follows:

$$\mathcal{L}_{\text{CE}} = \text{CE}_{\text{last}}\left(\text{argmax}\left(z\left[:, q\right]\right), \theta_G^t\left(\widetilde{x}\right)\right) + \lambda_{\text{current}} \text{CE}_{\text{current}}\left(\text{argmax}\left(z\left[:, q\right]\right), \theta_G^t\left(\widetilde{x}\right)\right), \tag{1}$$

where $\widetilde{x}$ is the output of $G_D^t(z)$; q is the total number of classes seen in the previous tasks, we just take q dimension for the noise; CE_{last} and $\text{CE}_{\text{current}}$ respectively are the Cross-Entropy Loss using the truncated outputs of $\theta_G^t(\widetilde{x})$ corresponding with last classes from task T_i, $i < t$, and current learned classes on task T_t, and λ_{current} is the temperature hyper-parameter.

Generating synthetic examples can easily be biased to a subset of classes. To maintain the diversity between classes, we utilize the Information Entropy (IE) Loss (Chen et al., 2019) as follows:

$$\mathcal{L}_{\text{IE}} = -\mathbf{H}_{\text{info}}\left(\frac{1}{\text{bs}}\sum_{i=1}^{\text{bs}} \theta_G^t\left(\widetilde{x}_i\right)\right), \text{bs: batch size}, \tag{2}$$

This loss measures the IE for samples of a batch. Maximizing this value can promote a more uniform and balanced output distribution from the generator across all classes.

To further improve the stability of generator training, we use Batch Normalization Loss (Smith et al., 2021) to make all Batch Normalization layers have the same statistics on synthetic images, as follows:

$$\mathcal{L}_{\text{batch}} = \frac{1}{L}\sum_{j=1}^{L} \mathbf{KL}\left(\mathcal{N}\left(\mu_j, \sigma_j^2\right) \parallel \mathcal{N}\left(\widetilde{\mu}_j, \widetilde{\sigma}_j^2\right)\right), \tag{3}$$

where L is the total number of Batch Normalization layers in the architecture of the global model. μ_j and σ_j^2 are the mean and variance stored in Batch Normalization layer j of the global model, $\widetilde{\mu}_j$ and $\widetilde{\sigma}_j$ are measured statistics of Batch Normalization layer j for the synthetic data.

Adjacent pixels in real images typically have values near one another. One typical method to promote similar patterns in the synthetic images is to add Image Prior Loss (Haroush et al., 2020). We can create the smoothed (blurred) version of an image by applying a Gaussian kernel and minimizing the distance of the original and Smooth $(\widetilde{x})$ as

$$\mathcal{L}_{\text{smooth}} = \|\widetilde{x} - \text{Smooth}\,(\widetilde{x})\|_2^2\,. \tag{4}$$

In summary, we can write the training objective of G_D as follows:

$$\min_{G_D} \mathcal{L}_{\text{CE}} + \lambda_{\text{IE}}\mathcal{L}_{\text{IE}} + \lambda_{\text{batch}}\mathcal{L}_{\text{batch}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}}, \tag{5}$$

where λ_{IE}, λ_{batch}, and λ_{smooth} are hyper-parameters of specific loss functions.

3.4 Server-side Feature Generator

As mentioned in Section 3.2, only synthetic images can exacerbate the catastrophic forgetting problem. (Liu et al., 2020) has shown that features can store better knowledge of previous tasks than data. Therefore, we train a feature generator that synthesizes features, capturing the knowledge within the feature space. Like the data generator, this generative model is trained only on the server side. The feature generator takes noise input $z \sim \mathcal{N}(0, 1)$ and produces synthetic features $\widetilde{f}$ that match the dimensions of the original features. These synthetic features must meet the following objectives:

After training task T_t, the generative feature should be classified correctly by the classifier H_G^t of the global model. Additionally, the synthetic features should not be biased to any classes. With this aim, we employ a temperature cross entropy classification loss between its assigned label z and the prediction of H_G^t on $G_D^t(z)$ as

$$\mathcal{L}_{\text{FCE}} = \text{CE}_{\text{last}}\left(\text{argmax}\,(z\,[:, q])\,, H_G^t\left(\widetilde{f}\right)\right) + \lambda_{\text{current}}\text{CE}_{\text{current}}\left(\text{argmax}\,(z\,[:, q])\,, H_G^t\left(\widetilde{f}\right)\right), \tag{6}$$

where $\widetilde{f}$ is the output of $G_F^t(z)$; q is the total number of classes seen in the previous tasks, we take q dimension for the noise; CE_{last} and $\text{CE}_{\text{current}}$ respectively are the Cross-Entropy Loss using the truncated outputs of $H_G^t\left(\widetilde{f}\right)$ corresponding with last classes from task T_i, $i < t$, and current learned classes on task T_t, and λ_{current} is the temperature hyper-parameter.

The generated features should not be biased to any subset of classes. Therefore, we propose the Feature Information Entropy Loss to make the synthetic feature have this quality, which is

$$\mathcal{L}_{\text{FIE}} = -\mathbf{H}_{\text{info}}\left(\frac{1}{\text{bs}}\sum_{i=1}^{\text{bs}} H_G^t\left(\widetilde{f}_i\right)\right), \text{bs: batch size}, \tag{7}$$

In summary, we can write the training objective of G_F as follows:

$$\min_{G_F} \mathcal{L}_{\text{FCE}} + \lambda_{\text{FIE}}\mathcal{L}_{\text{FIE}}, \tag{8}$$

where λ_{FIE} is the hyper-parameter of Feature Information Entropy Loss.

3.5 Client-side

For the first task, clients will carry out the traditional FL process after receiving the global model weights from the server. For each subsequent task, clients will use the data and feature generators provided by the server to produce synthetic information throughout the task. Note that the transmission of these generators from the server occurs only once per task. Specifically, from the second task onward, the local models need to learn the current task quickly, as well as retaining knowledge from previous tasks efficiently. Therefore, we divide the learning objectives into two parts, as follows:

To learn new tasks well, the model needs to learn the new information separately from the old classes. We compute the Cross-Entropy Loss using only the new classes' linear heads. Formally, we minimize:

$$\mathcal{L}_{\text{CE}} = \text{CE}\left(\theta^t\left(x \mid T_t\right), y\right), \tag{9}$$

where $\theta^t\left(x \mid T_t\right)$ is model's output and masking old classes before task T_t's linear heads.

To mitigate forgetting, previous approaches leverage knowledge distillation (Usmanova et al., 2021; Zhang et al., 2023). However, this can cause information loss in probability space due to squashing functions (Liu et al., 2018). Therefore, motivated by (Buzzega et al., 2020), we propose Synthetic Logits Distillation Loss, which matches the logits of the old and current linear heads. These classifiers take synthetic features as input instead of synthetic data since the feature stores more previous information. Formally, we optimize:

$$\mathcal{L}_{\text{logits}} = \left\|\theta_G^{t-1}\left(\widetilde{f}\right) - \theta^t\left(\widetilde{f}\right)\right\|, \tag{10}$$

where θ_G^{t-1} is the global model trained up to task T_{t-1}.

As shown in (Babakniya et al., 2024), when there is a sudden shift in the distribution of the input of the task sequence, biased features on previous tasks can output biased logits, hindering the ability to obtain new knowledge. To mitigate this shortcoming, we utilize only the extracted features of the data, i.e., clients freeze the feature extraction layers and update only the linear head (represented by H^t) for both real (x) and synthetic ($\widetilde{x}$) images. This Fine-tuning loss is formulated as

$$\mathcal{L}_{\text{FT}} = \text{CE}\left(H^t\left(\left[f, \widetilde{f}\right]\right), [y, \widetilde{y}]\right), \tag{11}$$

where f and $\widetilde{f}$ respectively are the extracted features of x and $\widetilde{x}$ after passing through the freezed feature extractor F^t of the local model, y and $\widetilde{y}$ is the hard label of x and $\widetilde{x}$.

Figure 1c shows that combining the above objectives reduces the model's performance across all classes. We contend that this happens because the feature directions are unconstrained, resulting in the total loss failing to converge. We then add additional loss to balance this problem, named Empirical Feature Matrix Loss (Magistri et al., 2024), which constrains directions in feature space most important for previous tasks. At the same time, it allows more plasticity in other directions when learning new tasks. In this work, we re-utilize the synthetic features to calculate the Empirical Feature Matrix E_{t-1} from the previous task T_{t-1}. We have,

$$\mathcal{L}_{\text{EFM}} = \left(F^t\left(x\right) - F_G^{t-1}\left(x\right)\right)^\top \left(\lambda_E E_{t-1} + \eta I\right)\left(F^t\left(x\right) - F_G^{t-1}\left(x\right)\right), \tag{12}$$

where F^t and F_G^{t-1} respectively are the feature extractor of the current model and the previous global model, η is the damping term to constrain features to stay in a specific region.

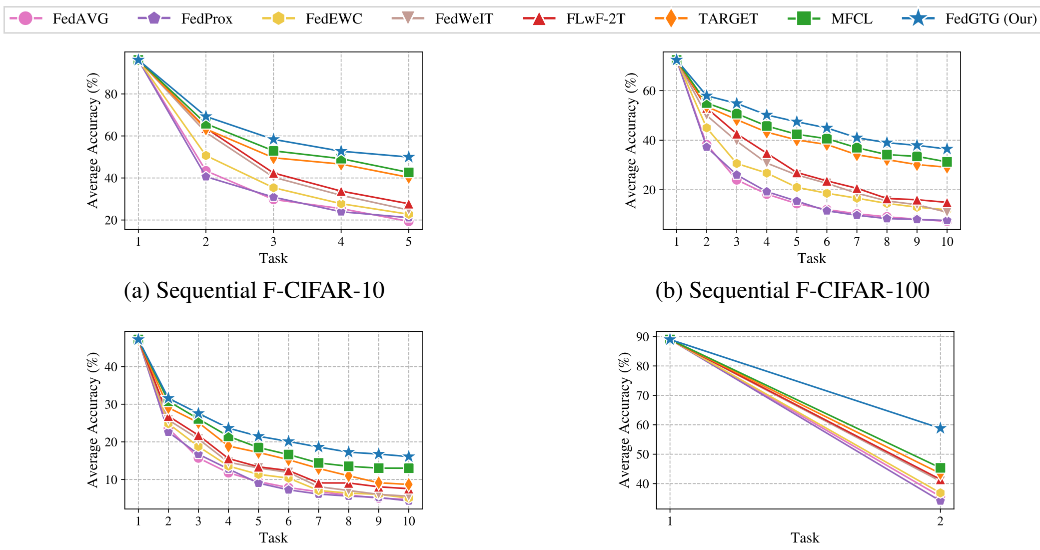
In summary, the final objective on the client side as

$$\min_{\theta^t} \mathcal{L}_{\text{CE}} + \lambda_{\text{logits}}\mathcal{L}_{\text{logits}} + \lambda_{\text{FT}}\mathcal{L}_{\text{FT}} + \lambda_{\text{EFM}}\mathcal{L}_{\text{EFM}}. \tag{13}$$

4 Experimental results

4.1 Experimental Setup

Datasets We perform our experiments on three widely-used benchmark datasets in FCIL (Dong et al., 2022; Zhang et al., 2023; Babakniya et al., 2024), which are the protocol versions of **CIFAR-10** (Krizhevsky et al., 2009), **CIFAR-100** (Krizhevsky et al., 2009), **tiny-ImageNet** (Yao & Miller, 2015), and we name it respectively are **Sequential F-CIFAR-10**, **Sequential F-CIFAR-100** and **Sequential F-tiny-ImageNet**. Moreover, FCIL is usually applied in the finance and healthcare industries (Wang et al., 2024), where the data distribution shifts significantly. We want to investigate the effectiveness of our work on this application. We introduce the protocol dataset named **HealthMNIST** to assess the domain shift scenario, which involved two distinct classification tasks: **Task 1** is the Colon Pathology Classification from PathMNIST (Yang et al., 2023) and **Task 2** is the Blood Cell Classification from BloodMNIST (Yang et al., 2023). The data preparation is explained later in Appendix B. We use Latent Dirichlet Allocation (LDA) (Reddi et al., 2020) with $\alpha = 1$ and $\alpha = 0.5$ to distribute the data of each task among clients. Additional experiments on **SuperImageNet** (Babakniya et al., 2024) are then provided in the Appendix D.3.



(a) Sequential F-CIFAR-10 (b) Sequential F-CIFAR-100

(c) Sequential F-tiny-ImageNet (d) HealthMNIST

Figure 3: AA per task of various algorithms on several benchmarks under the IID scenario.

FCIL Baselines We compare FedGTG with two conventional aggregating methods **FedAvg** (McMahan et al., 2017) and **FedProx** (Li et al., 2020), as well as three regularization-based FCIL methods, **FLwF-2T** (Usmanova et al., 2021) and the FCIL version of **FedWeIT** (Yoon et al., 2021) and **FedEWC** (Zhang et al., 2023), and two generative-based methods, **TARGET** (Zhang et al., 2023) and **MFCL** (Babakniya et al., 2024). The detailed description can be seen in Appendix B.

Models and Implementation Details In all experiments, we train a ResNet-18 (He et al., 2016) backbone using the SGD optimizer (Bottou, 1998). We train the model for 100 epochs per task on every dataset. We use FedAvg (McMahan et al., 2017) for aggregation. Additional implementation detailsare then provided in the Appendix B. We also conducted experiments on different architectures, including ResNet-34 and ResNet-50, which can be found in the Appendix D.2.

Evaluation Metrics We report the performance of the methods using two metrics: Average Incremental Accuracy and Average Forgetting. **Average Incremental Accuracy (AIA)** measures the average accuracy of the global model on all tasks after the training finishes. Forgetting (f_t) of task T_t is the difference between the model's best performance on task T_t and its accuracy after completed training. Consequently, **Average Forgetting (AF)** is the average of all f^t, from task T_1 to task T_{n-1}, at the end of task T_n. We report the averaged result over three different random initializations.

4.2 PERFORMANCE RESULTS

We present the performance of FedGTG and the baselines. Figure 3 shows the Average Accuracy of the model at each task in the training process. It can be seen that FedGTG achieves state-of-the-art performance in all settings. Specifically, our method observes better Average Accuracy on all later tasks. Table 1 reports both AIA results *(higher is better)* and AF results *(lower is better)* under the IID and Non-IID data distribution, respectively.

As expected, FedAvg and FedProx suffer the highest forgetting since they are not designed for FCIL. Compared to FedEWC, FedWeIT and FLwF-2T, the performance gap between it and our FedGTG is significant, indicating that regularization towards previous parameter sets is insufficient to avoid forgetting. Compared to the generative-replay methods TARGET and MFCL, our FedGTG achieves the least AF and the best AIA, showing that FedGTG can effectively retain knowledge and learn new tasks. Moreover, FedGTG performs extensively in the context of domain shift, which can retain knowledge from the first task of HealthMNIST, where MFCL and TARGET fail to do so.

Table 1: Performance of the different baselines in terms of AIA and AF for four datasets. $\alpha = 1$ is the IID scenario, and $\alpha = 0.5$ is the Non-IID scenario. [↑] higher is better, [↓] lower is better.

Method	CIFAR-10		CIFAR-100		tiny-ImageNet		HealthMNIST	
	AIA (↑)	AF (↓)	AIA (↑)	AF (↓)	AIA (↑)	AF (↓)	AIA (↑)	AF (↓)
				$\alpha = 1$				
FedAVG	42.82 ± 0.23	55.55 ± 0.58	21.39 ± 0.22	78.67 ± 0.83	13.80 ± 0.19	74.12 ± 0.81	62.31 ± 0.14	43.27 ± 0.75
FedProx	42.43 ± 0.32	56.15 ± 0.71	21.54 ± 0.32	78.12 ± 0.71	13.69 ± 0.21	75.16 ± 0.79	61.56 ± 0.24	43.27 ± 0.70
FedEWC	45.27 ± 0.17	49.46 ± 0.76	26.63 ± 0.29	62.17 ± 0.49	14.58 ± 0.15	58.00 ± 0.51	62.92 ± 0.16	40.42 ± 0.62
FedWeIT	50.46 ± 0.21	45.99 ± 0.59	30.19 ± 0.19	55.57 ± 0.48	16.02 ± 0.22	46.23 ± 0.77	64.91 ± 0.12	39.59 ± 0.52
FLwF-2T	52.74 ± 0.23	39.51 ± 0.59	32.19 ± 0.18	50.78 ± 0.63	17.18 ± 0.17	44.51 ± 0.67	65.22 ± 0.12	36.92 ± 0.69
TARGET	59.19 ± 0.16	17.23 ± 0.45	42.15 ± 0.13	26.45 ± 0.61	19.46 ± 0.25	20.17 ± 0.57	66.23 ± 0.22	36.41 ± 0.57
MFCL	61.34 ± 0.21	22.32 ± 0.52	45.07 ± 0.12	28.30 ± 0.78	21.47 ± 0.15	23.90 ± 0.58	67.18 ± 0.15	36.23 ± 0.52
FedGTG (ours)	$\mathbf{64.50 \pm 0.22}$	$\mathbf{13.14 \pm 0.67}$	$\mathbf{46.42 \pm 0.18}$	$\mathbf{18.66 \pm 0.76}$	$\mathbf{24.04 \pm 0.23}$	$\mathbf{16.18 \pm 0.62}$	$\mathbf{73.91 \pm 0.19}$	$\mathbf{19.15 \pm 0.60}$
				$\alpha = 0.5$				
FedAVG	40.92 ± 0.26	55.59 ± 0.58	20.66 ± 0.25	61.34 ± 0.72	11.82 ± 0.22	74.16 ± 0.68	59.93 ± 0.14	43.21 ± 0.85
FedProx	40.43 ± 0.32	55.15 ± 0.67	20.43 ± 0.22	62.73 ± 0.81	11.49 ± 0.21	75.01 ± 0.72	60.15 ± 0.24	43.21 ± 0.12
FedEWC	43.22 ± 0.17	50.70 ± 0.66	25.53 ± 0.18	59.17 ± 0.56	12.90 ± 0.15	60.93 ± 0.55	60.88 ± 0.22	42.45 ± 0.32
FedWeIT	48.11 ± 0.21	47.34 ± 0.46	28.89 ± 0.20	56.11 ± 0.63	14.55 ± 0.11	49.76 ± 0.49	61.05 ± 0.17	42.09 ± 0.55
FLwF-2T	50.23 ± 0.23	40.21 ± 0.51	30.25 ± 0.14	53.72 ± 0.66	16.14 ± 0.18	44.59 ± 0.67	63.11 ± 0.12	39.72 ± 0.59
TARGET	56.19 ± 0.19	19.45 ± 0.45	41.03 ± 0.13	28.23 ± 0.68	18.46 ± 0.25	22.23 ± 0.57	64.03 ± 0.23	37.41 ± 0.60
MFCL	56.65 ± 0.25	18.34 ± 0.59	42.07 ± 0.25	30.30 ± 0.59	21.42 ± 0.19	21.02 ± 0.58	65.11 ± 0.19	36.13 ± 0.52
FedGTG (ours)	$\mathbf{61.11 \pm 0.18}$	$\mathbf{13.12 \pm 0.37}$	$\mathbf{44.58 \pm 0.21}$	$\mathbf{20.89 \pm 0.76}$	$\mathbf{23.23 \pm 0.27}$	$\mathbf{15.18 \pm 0.69}$	$\mathbf{68.66 \pm 0.23}$	$\mathbf{25.15 \pm 0.71}$

4.3 Model Analysis

The majority of FCIL research concentrates on testing experiments on ideal benchmarks (Dong et al., 2022; Zhang et al., 2023; Babakniya et al., 2024), such as CIFAR (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009). This lacks analysis concerning real-world scenarios, such as the decision-making required in hospitals or the model's generalization to diverse environments. Therefore, in this section, we conducted experiments to analyze the robustness of FedGTG and three FCIL algorithms on corrupted environments, and the qualities of generalization (Chaudhari et al., 2019; Keskar et al., 2016) and achieve calibrated networks (Guo et al., 2017; Kull et al., 2019).

Robustness to natural corruptions. We evaluate our method and the recent FCIL methods on the **CIFAR-100-C** dataset. This dataset includes 18 augmentations of the original CIFAR-100, inspired by CIFAR-10-C (Hendrycks & Dietterich, 2019). Models are trained using standard CIFAR-100 with the same setting in Section 4.1 and tested on CIFAR-100-C. Figure 4 shows robustness to 09 different corruptions averaged over three different runs, the results of the rest augmentations are shown in Appendix D.4. Specifically, our approach achieves higher test accuracy on various corruptions, with an average improvement of 5% over MFCL and 8% over TARGET. Evidently, our method offers noticeable advantages in robustness against natural corruption.

Converging to flatter minima. Extensive CL algorithms (Bhat et al., 2022; Wang et al., 2023; Park et al., 2024) explore how well their methods generalize by examining their ability to converge to flat minima. In this part, we compare the flatness of the training minima of FLwF-2T, TARGET, and MFCL with FedGTG. As done in (Zhang et al., 2019), we consider the model at the end of training and add independent Gaussian noise with growing variance to each parameter. This allows us to evaluate its effect on the average loss $\sum_{t=1}^{n} \mathcal{L}_{CE}^{(T_t)}$ across all training examples. As shown in Figures 5a and 5b, MFCL, especially FLwF-2T, and TARGET, reveal higher sensitivity to perturbations than FedGTG. This concludes that FedGTG can achieve better generalization than previous methods.

Converging to a more calibrated network. Calibration measures how well a learner's prediction confidence matches its accuracy, with ideal outcomes reflecting true probabilities of correctness. In real-world applications, including weather forecasting (Bröcker, 2009) and econometric analysis (Gneiting et al., 2007), the calibrating ability of a model should be investigated. Figures 5c and 5d show the value of the Expected Calibration Error (ECE) (Naeini et al., 2015) across various FCIL methods after completing each task. It can be seen that FedGTG achieves a lower ECE than the others, proving that models trained with FedGTG are less over-confident and easier to interpret.

Robustness to different client sizes. We validate the effectiveness of FedGTG across different client sizes on the tiny-ImageNet dataset. We run experiments by varying the number of total clients

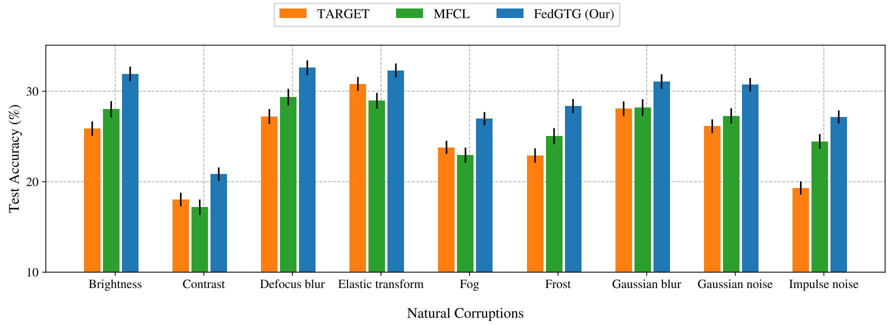
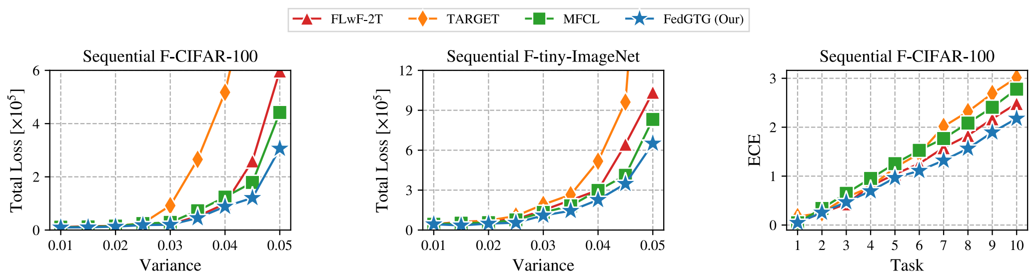
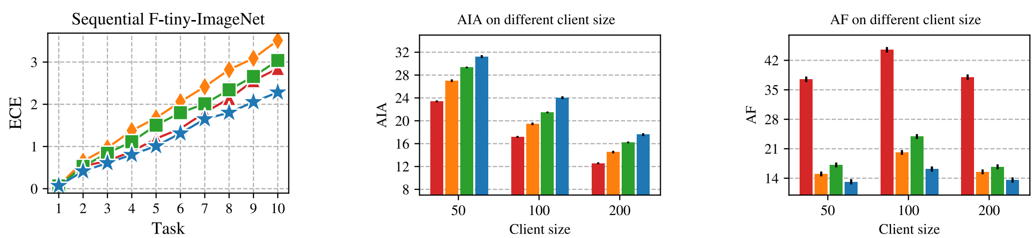


Figure 4: Robustness to natural corruptions.



(a) Perturbation [↓] (b) Perturbation [↓] (c) Calibration Error [↓]



(d) Calibration Error [↓] (e) Average Accuracy [↑] (f) Average Forgetting [↓]

Figure 5: Results for the model analysis. [↑] higher is better, [↓] lower is better *(best seen in color)*.

(maintaining a consistent participation rate of 0.1 per round), ranging from 50 to 200, and compare the results. Figures 5e and 5f demonstrate that our method still outperforms other approaches, achieving an accuracy 4% higher and a forgetting rate 6% lower compared to the next best method, MFCL.

5 Ablation Study

We highlight the significance of each loss within our framework and analyze both server and client contributions by sequentially removing components to observe their effects. Table 2 shows our results, where each row corresponds to the removal of a specific loss component, and the columns display the corresponding Average Accuracy ($\mathcal{A}^t$), for $1 \leq t \leq 10$, Average Incremental Accuracy ($\mathcal{A}$), and Average Forgetting ($\mathcal{F}$). Specifically, the performance of the model is influenced by generative models, as poorly trained ones result in low AIA and high AF compared to others. Nevertheless, the Fine-tuning Loss has the lowest AF because it did not learn tasks well (lowest AIA). The final two rows illustrate how the feature-constraining losses ($\mathcal{L}_{\text{logits}}$ and $\mathcal{L}_{\text{EFM}}$) impact the performance of the global model, where the decrease in accuracy demonstrates the importance of these two losses.

6 Discussion

Since two generative models are trained using the global model solely, the clients do not have to send their data to the server. Moreover, as shown in Appendix E, the visualization of synthetic images does

Table 2: Ablation study for FedGTG on Sequential F-CIFAR-100.

w/o Loss	$\mathcal{A}^1$	$\mathcal{A}^2$	$\mathcal{A}^3$	$\mathcal{A}^4$	$\mathcal{A}^5$	$\mathcal{A}^6$	$\mathcal{A}^7$	$\mathcal{A}^8$	$\mathcal{A}^9$	$\mathcal{A}^{10}$	$\mathcal{A}$	$\mathcal{F}$
$\mathcal{L}_{\text{IE}}$	72.40	56.60	46.96	38.22	33.18	30.73	28.50	25.05	23.98	22.37	33.95	37.31
$\mathcal{L}_{\text{batch}}$	72.40	55.62	48.67	41.25	33.48	30.08	27.10	22.98	22.48	21.41	33.67	41.42
$\mathcal{L}_{\text{smooth}}$	72.40	57.15	49.10	41.85	39.44	37.58	34.67	31.15	30.50	29.20	38.96	29.36
$\mathcal{L}_{\text{FIE}}$	72.40	56.35	47.96	37.22	34.18	32.61	29.82	26.17	24.98	23.03	34.70	33.42
$\mathcal{L}_{\text{FT}}$	72.40	41.80	34.67	29.25	22.68	17.60	15.17	13.06	12.59	12.09	22.10	11.66
$\mathcal{L}_{\text{logits}}$	72.40	57.15	49.77	43.05	39.36	37.72	35.67	33.19	32.73	31.62	40.03	25.35
$\mathcal{L}_{\text{EFM}}$	72.40	56.30	48.77	42.53	39.56	37.28	35.37	31.91	31.66	29.92	39.26	27.38
FedGTG (ours)	72.40	57.95	54.85	50.20	47.45	44.95	41.00	39.01	37.92	36.46	46.42	18.66

not replicate any real data, and therefore, it will preserve privacy. In addition, our framework does not affect the aggregation stage, allowing the integration of Secure Aggregation techniques (Kim et al., 2023; Kanchan et al., 2024). This ensures that when local updates are sent to the server for aggregation, they remain encrypted, which prevents the server from accessing the client's information.

Table 3: Total parameters sent from the server to the clients across FCIL algorithms.

Dataset/Method	**FedGTG (ours)**	**MFCL**	**TARGET**	**FLwF-2T**	**FedWeIT**	**FedEWC**	**FedAvg**	**FedProx**
CIFAR-10	20,996,877	19,696,397	19,696,397	11,272,458	11,272,458	11,272,458	11,272,458	11,272,458
CIFAR-100	20,949,681	19,649,201	19,649,201	11,225,262	11,225,262	11,225,262	11,225,262	11,225,262
tiny-ImageNet	21,416,607	19,853,471	19,853,471	11,281,692	11,281,692	11,281,692	11,281,692	11,281,692
HealthMNIST	20,949,681	19,649,201	19,649,201	11,225,262	11,225,262	11,225,262	11,225,262	11,225,262

Table 4: Training time in seconds of different algorithms trained on the CIFAR-100 dataset.

Training Time/Method	**FedGTG (ours)**	**MFCL**	**TARGET**	**FLwF-2T**	**FedWeIT**	**FedEWC**	**FedProx**	**FedAvg**
$t = 1$	1.2	1.2	1.2	1.2	1.2	1.2	1.8	1.2
$t > 1$	4.1	3.7	3.5	3.4	2.2	1.2	1.8	1.2

In our work, the clients need to accommodate two generative models and the global model's weights from the most recent task, which introduces higher storage requirements than previous methods. However, this transmission of generators takes place **only once per task**, representing a necessary cost to prevent catastrophic forgetting. Table 3 reports the total parameters transmitted from the server to the clients, serving as a measure of the communication overheads. We also calculate the amount of time in seconds that the clients need to complete one round, as shown in Table 4. The implementational details of training time are provided in Appendix D.1. While FedGTG requires more parameters and time for training than others, it delivers significant benefits. As shown in Table 1 and Figure 3, FedGTG outperforms others in AIA and AF. Moreover, the framework proves effective in handling complex scenarios such as domain shifts, making the added computational cost justifiable, particularly in applications like healthcare and finance, where data privacy and performance are crucial. We can see that although FedGTG introduces additional computational components, these are essential to achieving a balance between retaining knowledge and learning new tasks in FCIL, which other methods struggle with.

7 Conclusion

In this work, we alleviate the lack of stability of previous works in the FCIL setting by introducing a framework named FedGTG, both utilizing data and feature generative models trained by the server, eliminating the requirement for costly on-device memory for clients. Our experiments show that FedGTG is successful in reducing catastrophic forgetting and surpasses the current state-of-the-art methods. Moreover, By analyzing the robustness on natural images, testing the qualities of converging to flat minima and calibrated networks, and the performance, as well as the performance on the context of domain shifts between FCIL algorihms, we observe that our framework outperforms previous approaches, making FedGTG more applicable in real-world scenarios.

References

Rahaf Aljundi. Continual learning in neural networks. *arXiv preprint arXiv:1910.02718*, 2019.

Sara Babakniya, Zalan Fabian, Chaoyang He, Mahdi Soltanolkotabi, and Salman Avestimehr. A data-free approach to mitigate catastrophic forgetting in federated class incremental learning for vision tasks. *Advances in Neural Information Processing Systems*, 36, 2024.

Prashant Shivaram Bhat, Bahram Zonooz, and Elahe Arani. Task agnostic representation consolidation: a self-supervised based continual learning approach. In *Conference on Lifelong Learning Agents*, pp. 390–405. PMLR, 2022.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1: 374–388, 2019.

Léon Bottou. Online algorithms and stochastic approximations. *Online learning in neural networks*, 1998.

Jochen Bröcker. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 135(643):1512–1519, 2009.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3514–3522, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. doi: 10.1109/CVPR.2009.5206848.

Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. Federated class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10164–10173, 2022.

Jiahua Dong, Hongliu Li, Yang Cong, Gan Sun, Yulun Zhang, and Luc Van Gool. No one left behind: Real-world federated class-incremental learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Keval Doshi and Yasin Yilmaz. Federated learning-based driver activity recognition for edge devices. In *Proceedings of the IEEE/CVF Conference on computer Vision and Pattern Recognition*, pp. 3338–3346, 2022.

Ronald A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222:309–368, 1922.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 69(2): 243–268, 2007.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2020.

Chen He, Ruiping Wang, Shiguang Shan, and Xilin Chen. Exemplar-supported generative reproduction for class incremental learning. In *BMVC*, volume 1, pp. 2, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL `https://arxiv.org/abs/1503.02531`.

Sneha Kanchan, Jae Won Jang, Jun Yong Yoon, and Bong Jun Choi. Gsfedsec: Group signature-based secure aggregation for privacy preservation in federated learning. *Applied Sciences*, 14(17):7993, 2024.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Jien Kim, Gunryeong Park, Miseung Kim, and Soyoung Park. Cluster-based secure aggregation for federated learning. *Electronics*, 12(4):870, 2023.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. 114(13):3521–3526, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32, 2019.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Fedprox: A scalable approach to federated learning with heterogeneous data. In *Proceedings of the 2nd MLSys Conference*, 2020.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: A research platform for federated learning in natural language processing. *arXiv preprint arXiv:2104.08815*, 2021.

Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 226–227, 2020.

Xuan Liu, Xiaoguang Wang, and Stan Matwin. Improving the interpretability of deep neural networks with knowledge distillation. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 905–912. IEEE, 2018.

Yunyi Liu and Ruining Yang. Federated learning application on depression treatment robots (dtbot). In *2021 IEEE 13th International Conference on Computer Research and Development (ICCRD)*, pp. 121–124. IEEE, 2021.

Yuhang Ma, Zhongle Xie, Jue Wang, Ke Chen, and Lidan Shou. Continual federated learning based on knowledge distillation. In *IJCAI*, pp. 2182–2188, 2022.

Simone Magistri, Tomaso Trinci, Albin Soutif-Cormerais, Joost van de Weijer, and Andrew D Bagdanov. Elastic feature consolidation for cold start exemplar-free incremental learning. *arXiv preprint arXiv:2402.03917*, 2024.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. pp. 7765–7773, 2018.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. pp. 67–82, 2018.

H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2:2, 2016.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N Asokan, and Ahmad-Reza Sadeghi. Dïot: A federated self-learning anomaly detection system for iot. In *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*, pp. 756–767. IEEE, 2019.

Junyoung Park, Jin Kim, Hyeongjun Kwon, Ilhoon Yoon, and Kwanghoon Sohn. Layer-wise auto-weighting for non-stationary test-time adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1414–1423, 2024.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

Daiqing Qi, Handong Zhao, and Sheng Li. Better generative replay for continual federated learning. *arXiv preprint arXiv:2302.13001*, 2023.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9374–9384, 2021.

Anastasiia Usmanova, François Portet, Philippe Lalanda, and German Vega. A distillation-based approach integrating continual learning and federated learning for pervasive services. *arXiv preprint arXiv:2109.04197*, 2021.

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Lixu Wang, Yang Zhao, Jiahua Dong, Ating Yin, Qinbin Li, Xiao Wang, Dusit Niyato, and Qi Zhu. Federated learning with new knowledge: Fundamentals, advances, and futures. *arXiv preprint arXiv:2402.02268*, 2024.

Zhenyi Wang, Li Shen, Donglin Zhan, Qiuling Suo, Yanjun Zhu, Tiehang Duan, and Mingchen Gao. Metamix: Towards corruption-robust continual learning with temporally self-adaptive data transformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24521–24531, 2023.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.

Xin Yang, Hao Yu, Xin Gao, Hao Wang, Junbo Zhang, and Tianrui Li. Federated continual learning via knowledge fusion: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Leon Yao and John Miller. Tiny imagenet classification with convolutional neural networks. *CS 231N*, 2(5):8, 2015.

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pp. 12073–12086. PMLR, 2021.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pp. 3987–3995. PMLR, 2017.

Jie Zhang, Chen Chen, Weiming Zhuang, and Lingjuan Lyu. Target: Federated class-continual learning via exemplar-free distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4782–4793, 2023.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3713–3722, 2019.

Huiping Zhuang, Zhenyu Weng, Hongxin Wei, Renchunzi Xie, Kar-Ann Toh, and Zhiping Lin. Acil: Analytic class-incremental learning with absolute memorization and privacy protection. *Advances in Neural Information Processing Systems*, 35:11602–11614, 2022.

Huiping Zhuang, Zhenyu Weng, Run He, Zhiping Lin, and Ziqian Zeng. Gkeal: Gaussian kernel embedded analytic learning for few-shot class incremental task. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7746–7755, 2023.

Huiping Zhuang, Run He, Kai Tong, Ziqian Zeng, Cen Chen, and Zhiping Lin. Ds-al: A dual-stream analytic learning for exemplar-free class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17237–17244, 2024.

A FedGTG algorithm

Recall that there are n tasks $T_1, T_2, \ldots, T_n$. At task T_1, the system is trained using the conventional FedAvg algorithm for aggregating the weight from the clients in R communication rounds. At the end of every task, the server trains a data generator and a feature generator without using any information from the clients. From task T_2, these two generators are sent to the clients, which combine with modified objectives to both retain knowledge and learn new tasks well. We formalize our approach in Algorithm 1 in detail.

Algorithm 1 Federated Global Twin Generator

Input:
n tasks with n datasets $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$.
c clients with c local models θ, R communication rounds.
A global model θ_G, a data generator G_D and a feature generator G_F.
Procedure:
for $t = 1$ **to** T **do**
    for $r = 1$ **to** R **do** ▷ Each task is learned on several communication rounds
        Select k clients for training.
        if $r > 1$ or $t > 1$ **then**
            The server sends the global model's weight to selected clients.
            if $t > 1$ **then**
                The server sends the two generators, the global model's weight from the previous task and the Empirical Feature Matrix to selected clients.
            end if
        end if
        if $t = 1$ **then**
            Train local models $\theta_j^{(t,r)}$ conventionally. ▷ $1 \leq j \leq k$
        else

            Train local models $\theta_j^{(t,r)}$ using Algorithm 2.
        end if
        Aggregate local model updates to the server.
    end for
    Train the data generator and the feature generator.
    Calculate Empirical Feature Matrix E^t using synthetic features.
end for

Algorithm 2 Client-side: Continual Learning

Input:
Task T_t, $t \geq 2$ with the dataset $\mathcal{D}_t$ in round r has B batches.
The global model $\theta_G^{(t,r)}$, a data generator G_D^{t-1}, a feature generator G_F^{t-1}.
The freezed global weight $\theta_G^{(t-1)}$ and the Empirical Feature Matrix E^{t-1}.
Procedure:
Calculate the Current Cross-Entropy Loss $\mathcal{L}_{\text{CE}}$ using $\mathcal{D}_t$ and Equation 9.
Generate synthetic data $\mathcal{D}_S$ and synthetic features $\mathcal{F}_S$ having B batches each.
Calculate the Fine-tunig Loss $\mathcal{L}_{\text{FT}}$ using $\mathcal{D}_t$, $\mathcal{D}_S$ and Equation 11.
Calculate the Synthetic Logits Distillation Loss $\mathcal{L}_{\text{logits}}$ using $\mathcal{F}_S$ and Equation 10.
Calculate the EFM Loss $\mathcal{L}_{\text{EFM}}$ using $\mathcal{F}_S$ and Equation 12.
Optimize Equation 13.

B Experimental Setup

In this section, we detail the settings used in our experiments, including datasets, FCIL algorithms, and experimental setups.

Datasets We perform our experiments on theree widely-used benchmark datasets, including the FCIL version of CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009) and tiny-ImageNet (Yao & Miller, 2015):

- **Sequential F-CIFAR-10.** The CIFAR-10 dataset (Krizhevsky et al., 2009) consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. We split the training set into five disjoint subsets corresponding to 5 tasks.
- **Sequential F-CIFAR-100.** Sequential F-CIFAR-100 is constructed by dividing the original CIFAR-100 dataset (Krizhevsky et al., 2009), which contains 50,000 images belonging to 100 classes, into ten disjoint subsets corresponding to 10 tasks. In this way, each task has 5,000 images from 10 distinct categories, and each class has 500 images.
- **Sequential F-tiny-ImageNet.** Tiny-ImageNet (Yao & Miller, 2015) is a subset of ImageNet, containing 100,000 images of 200 real objects. We follow settings in (Babakniya et al., 2024) to form the Sequential F-tiny-ImageNet. In particular, we split the original dataset into ten non-overlapping subsets. We consider each subset as a task whose images are labeled by 20 different classes, and each class has 500 samples.

We also investigate the effectiveness of FedGTG in the context of domain shift. We introduce a protocol dataset named HealthMNIST:

- **HealthMNIST.** This dataset includes two distinct classification tasks: **Task 1** is the Colon Pathology Classifcation having data from PathMNIST (Yang et al., 2023). **Task 2** is the Blood Cell Classifcation from BloodMNIST (Yang et al., 2023). PathMNIST contains 107,180 samples of 9 classes, and BloodMNIST has 17,092 samples from 8 blood types. For both tasks, we select the first five classes from each dataset, with 500 samples per class for each task for training. Finally, the test set includes all test samples from these two datasets.

FCIL Baselines In addition to our FedGTG, we also include three regularization-based FCIL methods, **FLwF-2T** (Usmanova et al., 2021) and the FCIL version of **FedWeIT** (Yoon et al., 2021) and **FedEWC** (Zhang et al., 2023), and two generative-based methods, **TARGET** (Zhang et al., 2023) and **MFCL** (Babakniya et al., 2024). **FLwF-2T** utilize knowledge distillation both on the server side and client side to ease the catastrophic forgetting issue. **FedWeIT** maximizes the knowledge transfer between clients by storing previous tasks-adaptive parameters of clients. **FedEWC** is the FCIL version of EWC (Kirkpatrick et al., 2017), which uses Fisher Information Matrix (Fisher, 1922) as a regularizer to prevent forgetting. **TARGET** utilizes a global model to transfer knowledge from past tasks to the current task while also training a generator to generate synthetic data, mimicking the overall data distribution across clients. **MFCL** employs a generative model to create samples from previous distributions, which are then combined with training data to prevent catastrophic forgetting. Both of these data generation-based algorithms ensure privacy by training the generative model on the server after each task without client data retrieval.

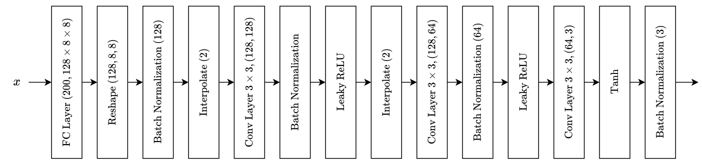
Implementation Details Table 5 shows settings and the hyper-parameter tuning for each dataset.

C GENERATIVE MODEL SETUP

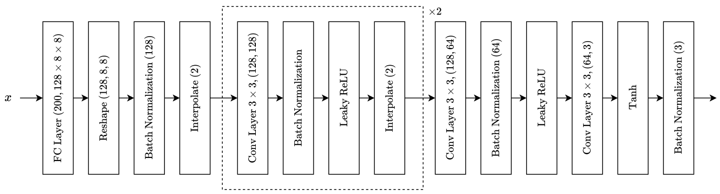
Data Generative Model Architecture Figure 6 presents the architecture of the data generative models used for the Sequential F-CIFAR-10, Sequential F-CIFAR-100, Sequential F-tiny-ImageNet, and HealthMNIST dataset. In all experiments, the global model is based on the ResNet-18 backbone.

Feature Generative Model Architecture The architecure of the feature generative models is illustrated in Figure 7, which employed for the Sequential F-CIFAR-10, Sequential F-CIFAR-100, Sequential F-tiny-ImageNet, and HealthMNIST dataset. As the outputs are feature vectors, only fully connected layers are needed.

Information generation To create synthetic data, clients sample i.i.d. noise, which is used to determine the classes through the application of the `argmax` function to the first q elements, where q represents the total number of classes observed. Since the noise is sampled i.i.d., each class has an equal probability of $\frac{1}{q}$ for sample generation.

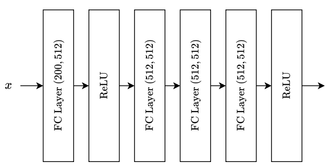


(a) Sequential F-CIFAR-10, Sequential F-CIFAR-100, and HealthMNIST.

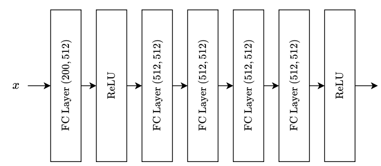


(b) Sequential F-tiny-ImageNet.

Figure 6: Architecture of the data generative model across three datasets.



(a) Sequential F-CIFAR-10, Sequential F-CIFAR-100, and HealthMNIST.



(b) Sequential F-tiny-ImageNet.

Figure 7: Architecture of the feature generative model across three datasets.

D ADDITIONAL RESULTS

D.1 TRAINING TIME COMPARISON

We use the amount of time in seconds that the clients need to complete one FL round to compare the training time between FCIL algorithms. The time is measured in rounds on our local GPU NVIDIA-A5000 and averaged between different clients. Table 4 summarizes the training time per round for all methods across various benchmarks. We can see that the increase in training time for FedGTG is comparable to MFCL and TARGET, with a moderate overhead. However, this cost is justified by the significant performance gains achieved, as demonstrated in Table 1 and Figures 3, 4, and 5 of the main paper. These results validate the efficiency and effectiveness of our approach despite the additional parameters required. We believe this balance of cost and performance underscores the practical value of FedGTG.

D.2 ROBUSTNESS ON VARIOUS ARCHITECTURES

We conducted additional experiments using ResNet34 and ResNet50 backbones to further validate the generality and robustness of our approach. Table 6 shows the results for all methods across various datasets under the Non-IID scenario. Specifically, FedGTG consistently outperforms other FCIL algorithms across different architectures, reinforcing its adaptability and effectiveness.

D.3 ROBUSTNESS ON CHALLENGING DATASET

Following the same setting from the work of Babakniya et al. (2024), we conducted experiments on the protocol version of ImageNet (Deng et al., 2009), named **SuperImageNet** which was created by superclassing the ImageNet dataset, thus greatly increasing the number of available samples for each class. Specifically, we conducted experiments on the SuperImageNet-L version, which consists of 7500 samples per class and 50 classes overall. The dataset was divided into 10 tasks, each of which contained 5 classes. The training process involves 300 clients, with 30 clients participating in each

Setting	Dataset	CIFAR-10	CIFAR-100	tiny-ImageNet	HealthMNIST
Experimental Setup	Image size	32 × 32	32 × 32	64 × 64	28 × 28
	Task number	5	10	10	2
	Classes per task	2	10	20	5
	Samples per task	5000	500	500	500
	Learning rate	0.1	0.1	0.1	0.1
	Weight decay	0.1	0.1	0.1	0.1
	Batch size	32	32	32	32
	Synthetic batch size	32	32	128	32
	Communication round	100	100	100	100
	Local epoch	10	10	10	10
Hyper-parameter tuning	λ_{IE}	1.0	1.0	1.0	1.0
	λ_{batch}	1.0	1.0	1.0	1.0
	λ_{smooth}	1.0	1.0	1.0	1.0
	λ_{FIE}	1.0	1.0	1.0	1.0
	λ_{current}	1.5	1.5	2.0	1.5
	λ_{FT}	1.0	1.0	1.0	1.0
	λ_{logits}	0.1	0.1	0.1	0.1
	λ_{EFM}	0.005	0.005	0.005	0.005
	λ_{E}	10.0	10.0	10.0	10.0
	η	0.1	0.1	0.1	0.1

Table 5: Detail settings across three datasets.

Table 6: Performance of the different algorithms in terms of AIA and AF for four datasets with different architectures. [↑] higher is better, [↓] lower is better.

Method	CIFAR-10		CIFAR-100		tiny-ImageNet		HealthMNIST	
	AIA (↑)	AF (↓)	AIA (↑)	AF (↓)	AIA (↑)	AF (↓)	AIA (↑)	AF (↓)
ResNet-34								
FedAvg	42.20	45.68	21.31	50.41	12.19	60.94	61.81	35.51
FedProx	41.70	45.32	21.07	51.55	11.85	61.64	62.03	35.51
FLwF-2T	51.80	33.04	31.20	44.15	16.65	36.64	65.09	32.64
TARGET	57.95	15.98	42.32	23.20	19.04	18.27	66.04	30.74
MFCL	58.42	15.07	43.39	24.9	22.09	17.27	67.15	29.69
FedGTG (ours)	**63.02**	**10.78**	**45.98**	**17.17**	**23.96**	**12.47**	**70.81**	**20.67**
ResNet-50								
FedAvg	36.79	41.21	18.57	45.47	10.63	54.98	53.88	32.03
FedProx	36.35	40.88	18.37	46.50	10.33	55.61	54.08	32.03
FLwF-2T	45.16	29.81	27.20	39.82	14.51	33.06	56.74	29.45
TARGET	50.52	14.42	36.89	20.93	16.60	16.48	57.56	27.73
MFCL	50.93	13.60	37.82	22.46	19.26	15.58	58.54	26.78
FedGTG (ours)	**54.94**	**9.73**	**40.08**	**15.49**	**20.88**	**11.25**	**61.73**	**18.64**

round. Table 7 shows the results of various FCIL algorithms on SuperImageNet. We can see that FedGTG still outperforms other FCIL methods in this dataset, showing its ability in the field.

D.4 Robustness to natural corruptions

In this section, we show additional results about the robustness of testing on natural images across our method and other FCIL methods. Figure 4 shows the last 09 augmentations of the CIFAR-100 dataset averaged over three different runs. Our approach still outperforms MFCL and TARGET in terms of test accuracy.

Table 7: Performance of the different algorithms training on the SuperImageNet dataset.

Metrics/Method	FedGTG (ours)	MFCL	TARGET	FLwF-2T	FedWeIT	FedEWC	FedProx	FedAvg
AIA (↑)	**36.19**	31.52	30.11	25.13	24.01	22.66	21.57	21.41
AF (↓)	**26.08**	32.33	31.56	41.23	44.36	49.67	57.99	59.23

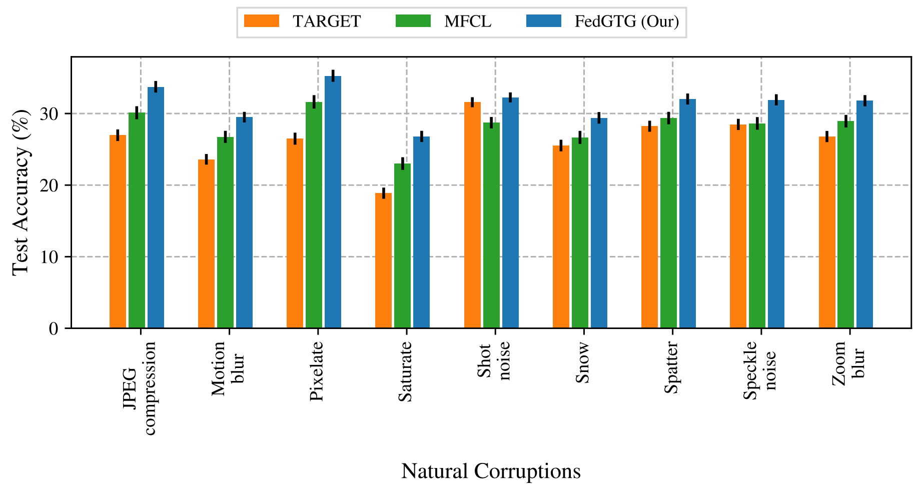


Figure 8: Robustness to natural corruptions.

E Data visualization

Figure 9 illustrates synthetic images of the CIFAR-100 dataset produced by the data generator; while these images retain specific characteristics of the original datasets, their altered shapes ensure privacy is maintained.

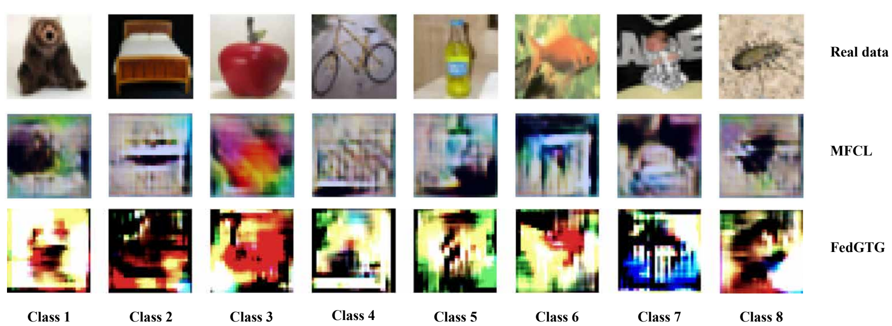


Figure 9: Synthetic images of CIFAR-100, generated by data generator from MFCL and FedGTG.

To visualize synthetic features, we employ Principal Component Analysis (Pearson, 1901) to reduce the dimensionality of the both real and synthetic features to the 2D space. As shown in Figure 10, synthetic features generated by FedGTG's feature generator align with real features, which shares the same decision boundary.

F Comparison between FedGTG and MFCL

In this section, we provide a detail comparison between our framework (FedGTG) and MFCL, which both use the data generation approach to alleviate the stability-plasticity trade-off, as follows:

- In FedGTG, we additionally train the feature generator to overcome the catastrophic forgetting that MFCL still suffers from.

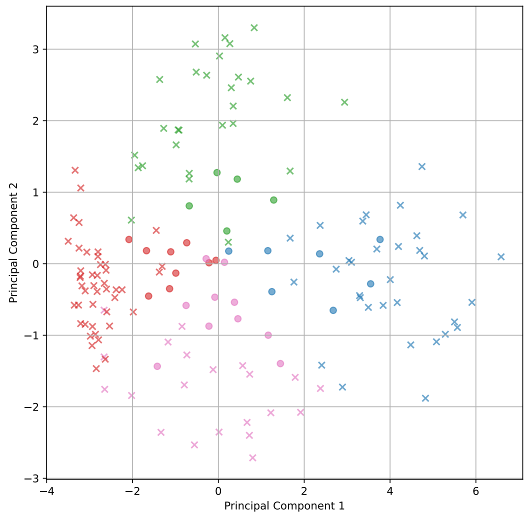


Figure 10: Synthetic features of CIFAR-100, generated by feature generator from FedGTG. × points and ○ denote the real and synthetic features, respectively.

- FedGTG is effectiveness in thedomain shift scenario, where MFCL witnesses a bad stability.
- FedGTG is more robust to natural images, which is easier to interpret in the real-world scenarios.
- Although there are more computational resources needed to complete the training process on FedGTG, the increasement is comparable with MFCL, and the performance gained is better, as shown in Table 1 and Figure 3.

G HYPER-PARAMETER SELECTION

Hyper-parameters can have a significant impact on how well algorithms work. While it is true that each loss term in FedGTG has an associated hyper-parameter, these parameters are carefully designed to allow fine-tuning for optimal balance between knowledge retention and adaptation to new classes. We offer basic hyper-parameter settings based on extensive experimental results to help users First, note that GANs are sensitive to hyper-parameters, we set the generative model's hyper-parameters to the same values as MFCL (Babakniya et al., 2024) for a fair comparison. Second, we modify on of the local side hyper-parameters to see a difference in accuracy. Our FedGTG results from testing several hyper-parameter settings on the CIFAR-100 dataset are shown in Table 8:

Table 8: Performance of different hyper-parameters for the CIFAR-100 dataset.

λ_{FT}	AIA (↑)	AF (↓)	λ_{logits}	AIA (↑)	AF (↓)	λ_{EFM}	AIA (↑)	AF (↓)
1.0	46.42	18.66	0.1	46.42	18.66	0.005	46.42	18.66
0.5	44.44	22.15	0.15	45.34	20.33	0.1	45.11	21.22
0.005	45.23	20.88	0.05	45.78	19.55	1	45.23	20.99