# Using Diffusion Models as Generative Replay in Continual Federated Learning – What will Happen?

Yongsheng Mei<sup>1\*</sup>Liangqi Yuan<sup>2\*</sup>Dong-Jun Han<sup>3</sup>Kevin S. Chan<sup>4</sup>Christopher G. Brinton<sup>2</sup>Tian Lan<sup>1</sup><sup>1</sup>The George Washington University<sup>2</sup>Purdue University<sup>3</sup>Yonsei University<sup>4</sup>Army Research Labysmei@email.gwu.edu, liangqiy@purdue.edu,

djh@yonsei.ac.kr, kevin.s.chan.civ@army.mil, cgb@purdue.edu, tlan@gwu.edu

Code: https://github.com/ysmei97/DCFL

#### Abstract

Federated learning (FL) has become a cornerstone in decentralized learning, where, in many scenarios, the incoming data distribution will change dynamically over time, introducing continuous learning (CL) problems. This continual federated learning (CFL) task presents unique challenges, particularly regarding catastrophic forgetting and non-IID input data. Existing solutions include using a replay buffer to store historical data or leveraging generative adversarial networks. Nevertheless, motivated by recent advancements in the diffusion model for generative tasks, this paper introduces DCFL, a novel framework tailored to address the challenges of CFL in dynamic distributed learning environments. Our approach harnesses the power of the conditional diffusion model to generate synthetic historical data at each local device during communication, effectively mitigating latent shifts in dynamic data distribution inputs. We provide the convergence bound for the proposed CFL framework and demonstrate its promising performance across multiple datasets, showcasing its effectiveness in tackling the complexities of CFL tasks.

# 1 Introduction

Federated learning (FL) has emerged as a prevalent decentralized learning paradigm, allowing training of a global model through interactions with distributed clients while maintaining the privacy of their local data [1, 2]. Most FL frameworks operate under the assumption that the client datasets at each client remain static throughout extensive learning cycles and iterations. However, this assumption does not align with the dynamic nature of real-world scenarios [3, 4]. The global model trained on such fixed datasets often fails to adapt effectively to the constantly evolving real world [5]. Furthermore, in real-world scenarios, clients often encounter new environments, objectives, and tasks – an aspect of adaptability that conventional FL frameworks have not yet fully addressed.

Continual learning (CL) methods were proposed to handle the phenomenon of catastrophic forgetting, where historical data may become inaccessible due to privacy regulations or storage constraints [6]. These methods were proposed that focus on developing systems capable of continuously learning from new tasks without erasing previously acquired knowledge. Classical CL scenarios can be broadly categorized into three types, ranging from task incremental learning (TIL) and domain incremental learning (DIL) to class incremental learning (CIL) [7]. However, these CL scenarios may face broader and more diverse challenges in the context of FL, as it is essential to consider cross-client scenarios and the non-IID data distribution among clients.

<sup>\*</sup>Equal contribution

We aim to address the challenges of continual federated learning (CFL) tasks in practical settings. Specifically, in CFL, learning is decentralized across multiple heterogeneous devices and is coordinated by a central server, where devices encounter new data and tasks over time. This poses challenges in handling both catastrophic forgetting issues induced by timely-shifted data distribution and non-IID problems in FL [8]. Recent approaches have been proposed to mitigate these challenges, such as leveraging replay memory to store historical data experienced by the model in the past [9] and utilizing generative adversarial networks (GANs) to generate historical data on each device to help remember past experiences [10]. Some methods utilize the global model on the server to train a generative model through knowledge distillation, but this approach leads to low-quality synthetic data and introduces additional noise [11, 12]. While storing real historical data [13] is useful for memory replay, it may not be feasible in cases where the data is available only for limited-time usage. Alternatively, FOT [14] performs global subspace extraction to identify features of previous tasks, aiming to prevent forgetting. However, FOT incurs higher communication costs between clients and servers due to the transfer of subspace information and orthogonal projections. GANs, as traditional generative models, on the other hand, involve learning two models (generator and discriminator) to reach a stable equilibrium, which can be difficult to train and sometimes susceptible to mode collapse [15] problems. Therefore, the powerful data generation capability demonstrated by the diffusion model [16] in various domains [17, 18, 19] makes it a strong candidate to replay data within the CFL context.

In this paper, we propose a novel framework DCFL that integrates CFL with conditional diffusion. At each local device, the embedded diffusion model serves to alleviate the impact of catastrophic forgetting by generating synthetic historical data. Since the diffusion model is not shared with anyone, our framework adheres to general privacy restrictions. Subsequently, the target models (i.e., models performing FL) are aggregated on the global server to obtain a generalized global model. We also provide a convergence analysis of DCFL by separately examining the convergence of the FL backbone, the data distribution shift, and the data generation convergence with the diffusion model. By combining these results, we demonstrate that the overall convergence of the system ultimately hinges on the performance of the introduced diffusion model, of which the bounded characteristic contributes to the system's convergence. DCFL has been tested on three CFL scenarios and four mainstream benchmark datasets, where DCFL significantly outperformed classical FL, classical CL, traditional generative model, and state-of-the-art (SOTA) baselines.

The main contributions of this paper are provided as follows:

- We introduce a novel CFL framework, termed DCFL, which eliminates the need for replay memory, enabling model learning for both local clients and the global server with dynamic data inputs. DCFL leverages the modern diffusion model to generate synthetic historical data based on previously observed data distributions (Section 3.1).
- We provide the convergence analysis for our DCFL framework. Our convergence result captures the bound of the FL model, the bound affected by the diffusion model, and the effect of data distribution shift between time steps (Section 3.2).
- We conduct extensive experiments using MNIST, FashionMNIST, CIFAR-10, and PACS datasets under three practical CFL environments. The results demonstrate that our DCFL framework improves upon the best baseline by 32.61% in the Class Incremental IID scenario, 15.16% in the Class Incremental Non-IID scenario, and 7.45% in the Domain Incremental scenario (Section 4).

To the best of our knowledge, our DCFL is the first work that successfully integrates diffusion models into continual federated learning, addressing its unique challenges with theoretical analysis.

# 2 Preliminary

#### 2.1 Continual Federated Learning Scenarios

Unlike classical CL setups, CFL tends toward greater diversity due to the presence of multiple clients. In the FL context, it requires (i) all clients to have the same model architecture, (ii) uniform consensus among clients (i.e., all clients agree on the definitions of labels), and (iii) the same task (i.e., the same global test set). Additionally, FL typically needs to address non-IID settings, where clients have different class distributions. Therefore, CFL introduces entirely distinct scenario setups, and so far there has been no unified approach in the literature.



Figure 1: **Three Continual Federated Learning Scenarios.** Class Incremental IID: Clients have an identical class distribution, with classes incrementing over time. Class Incremental Non-IID: Clients have a non-identical class distribution, with classes incrementing over time. Domain Incremental: Clients data domain changes over time.

We introduce three different CFL scenarios: 1) Class Incremental IID, 2) Class Incremental Non-IID, and 3) Domain Incremental settings, as shown in Figure 1. It is important to note that the IID setting here differs from the traditional IID setting, such as those considered in [1]. In conventional FL, the IID setting is modeled by distributing the whole training set uniformly at random across all clients, without considering dataset evolution over time. In contrast, our IID setting in the CFL setting ensures that all clients have identical class distribution at any given time but only with a subset of classes (e.g., only labels 0 and 1). We model the dataset evolution by letting the clients have different set of classes (e.g., labels 2 and 3) in the next time step.

#### 2.2 Federated Averaging

We take the vanilla FedAvg [1] model as the general framework of our design, where the objective function with model parameter  $\theta$  is defined as:

$$\min_{\theta} \left\{ F(\theta) \triangleq \sum_{k=1}^{K} p_k F_k(\theta) \right\},$$
s.t. 
$$\sum_{k=1}^{K} p_k = 1, \quad p_k \ge 0,$$
(1)

where K is the number of clients and  $p_k$  is the weight of the k-th client. The local model learning loss function  $F_k(\theta)$  is given by:

$$F_k(\theta) \triangleq \frac{1}{a_k} \sum_{j=1}^{a_k} f(\theta; X_{k,j}) \quad \text{where} \quad a_k = |X_k|, \tag{2}$$

where  $f(\cdot)$  is the loss function, and  $a_k$  is the number of training data in k-th client. Considering local clients in the t-th round, we have the aggregation of the global model as:

$$\theta_t \leftarrow \sum_{k=1}^{K} p_k \theta_t^k \quad \text{where} \quad p_k = \frac{|X_k|}{|X|},$$
(3)

where n represents the total number of samples across all clients. As in the FedAvg setup, the aggregation of models is performed by weighting each client's contribution according to the number of samples they possess.

#### 2.3 Denoising Diffusion Probabilistic Models

Basic DDPM [16] gradually adds random noise to the data over a series of time steps  $(x_0, \dots, x_N)$  in the forward process, where  $x_0 = x$ ,  $x_N = z$ . Specifically, the sample at each time step is sampled from a Gaussian distribution conditioned on the sample from the previous time step with predefined schedule  $\beta_{1:N}$ ,

$$c_n \sim F(\cdot | x_{n-1}) = \mathcal{N}(\sqrt{1 - \beta_n} x_{n-1}, \beta_n \boldsymbol{I}).$$
(4)

With equation (4), the sample at each step t can be expressed as a function of  $x_0$ :  $x_n = \sqrt{\alpha_n}x_0 + \sqrt{1 - \alpha_n}\epsilon$ , where  $\alpha_n = \prod_{s=0}^n (1 - \beta_s)$ ,  $\epsilon \sim \mathcal{N}(0, I)$  [20]. On the contrary,  $x_N$  can be converted

back to  $x_0$  step-wisely via the reverse denoising process:

$$x_{n-1} \sim G_{\omega}(\cdot | x_n) = \mathcal{N}(\mu_{\omega}(x_n, n), \Sigma_{\omega}(x_n, n)),$$
(5)

where  $\mu_{\omega}$  and  $\Sigma_{\omega}$  can be obtained from neural networks. The objective of learning this denoising function is to match the joint distributions of  $x_{0:N}$  in the forward and reverse processes. To optimize this objective, Ho *et al.* proposed a reformulation [16] by specifying the variance schedule  $\beta_{1:N}$  and fixing the reverse variance  $\Sigma_{\omega}(x_n, n)$  to be  $\beta_n I$ , which is:

$$\min_{\omega} \{ \mathcal{L}(\omega) \triangleq \mathbb{E}_{n \sim \mathcal{U}[1,N], x_0 \sim P_X(\cdot), \epsilon \sim \mathcal{N}(0, I)} [\lambda(n) \\ \| \epsilon - \epsilon_{\omega} (\sqrt{\alpha_n} x_0 + \sqrt{1 - \alpha_n} \epsilon, n) \|^2 ] \},$$
(6)

where  $\mathcal{L}(\cdot)$  is the loss function of the diffusion model and  $\lambda(n)$  is a positive weighting function usually set as 1 for all *n* to improve sample quality. Recently, the polynomial bounds on the convergence rate of the diffusion model have also been given in [21].

#### 3 Methodology

#### 3.1 Proposed DCFL Framework

We propose using a conditional diffusion model for replay in CFL, termed DCFL, as depicted in Figure 2. In the DCFL framework, each client possesses a target model  $\theta$  (for various FL tasks) and a diffusion model  $\omega$  for replaying data distributions from previous time periods. The server receives and aggregates only the target models from each client while remaining unaware of the clients' diffusion models.



Figure 2: **Proposed** DCFL **Framework.** Each client has a target model and a diffusion model, both trained on the same dataset, consisting of the previous time period's real and synthetic data. The target model is uploaded to the server for aggregation, while the diffusion model remains local to prevent privacy leakage. The trained diffusion model will generate synthetic data encompassing all previously acquired knowledge.

We utilize Algorithm 1 to outline the general workflow of DCFL. The complete version is provided in Appendix C. As we adopt FedAvg as the backbone for FL, both the local devices and the global server perform updates and aggregation in a manner similar to FedAvg, including training the target local model with gradient descent, as shown in the algorithm. Additionally, we incorporate the diffusion model (in line 3 of Algorithm 1) for each local device to address the latent input data distribution shift. This diffusion model is utilized to recover historical data experienced by generating synthetic data, thereby mitigating the issue of catastrophic forgetting. Before training the local target model, the diffusion model generates a portion of historical synthetic data mixed with current real data as input for learning. The diffusion model is also trained on the mixed dataset of real and synthetic data, which helps prevent catastrophic forgetting within the diffusion model itself. This process is repeated for each local client, and the learned model gradients are aggregated at the end of every communication round.

Algorithm 1 Proposed DCFL Framework - See Algorithm 2 for Complete Procedure **Input:** Communication rounds (T), client datasets ( $\mathcal{D}_t^k$ ), target model, loss function, and learning rate ( $\theta, F$ ,  $\eta_{\theta}$ ), diffusion model, loss function, and learning rate ( $\omega, \mathcal{L}, \eta_{\omega}$ ) **Output:** Generalized global model  $(\theta_T)$ Local update of the *k*-th client: 1: initialize  $\omega_0$ 2: for each round t = 1 : T do Obtain  $\mathcal{G}_{t-1}^k \leftarrow \omega_{t-1}^k (\mathcal{G}_{t-1,n}^k \sim \mathcal{N}(0, \boldsymbol{I}))$  via reverse process Combine real and synthetic data with a scale factor  $\delta: \mathcal{X}_t^k = \mathcal{D}_t^k \cup \delta \cdot \mathcal{G}_{t-1}^k$ 3: ▷ Generate synthetic data 4:  $\begin{aligned} \theta_t^k &\leftarrow \theta_{t-1}^k - \eta_\theta \nabla F_k(\theta_{t-1}^k; \mathcal{X}_t^k) \\ \text{repeat} \ \omega_t^k &\leftarrow \omega_{t-1}^k - \eta_\omega \nabla \mathcal{L}_k(\omega_{t-1}^k; \mathcal{X}_t^k) \text{ until converge} \end{aligned}$ 5: ▷ Train target model 6: ▷ Train diffusion model 7: end for 8: return  $\theta_t^k$  to server Global update of the server 1: initialize  $\theta_0$ 2: for each round t = 1 : T do for each client k = 1 : K in parallel do 3: 4:  $\theta_t^k \leftarrow k$ -th client's *local update* 5: end for  $\theta_t \leftarrow \sum_{k=1}^K p_k \theta_t^k$ 6: ▷ Aggregate target models 7: end for

#### 3.2 Convergence Bound of CFL Framework with Diffusion

To determine the convergence bound of the designed CFL model, we need to 1) find the convergence of the general FL framework with both original data and synthetic data, 2) verify the convergence of data generation in the integrated diffusion model, and 3) prove the convergence of the whole model regarding the latent input data distribution shift. To start with, we provide several standard assumptions that have been widely used in the FL literature [22, 23].

**Assumption 1.**  $F_1, \dots, F_K$  are all L-smooth: for all model parameters  $\theta_1$  and  $\theta_2$ ,  $F_k(\theta_1) \leq F_k(\theta_2) + (\theta_1 - \theta_2)^T \nabla F_k(\theta_2) + \frac{L}{2} \|\theta_1 - \theta_2\|_2^2$ .

**Assumption 2.**  $F_1, \dots, F_K$  are all  $\mu$ -strongly convex: for all model parameters  $\theta_1$  and  $\theta_2$ ,  $F_k(\theta_1) \ge F_k(\theta_2) + (\theta_1 - \theta_2)^T \nabla F_k(\theta_2) + \frac{\mu}{2} \|\theta_1 - \theta_2\|_2^2$ .

**Assumption 3.** Let  $\xi_t^k$  be sampled from the k-th device's local data uniformly at random. The variance of stochastic gradients in each device is bounded:  $\mathbb{E} \left\| \nabla F_k(\theta_t^k, \xi_t^k) - \nabla F_k(\theta_t^k) \right\|^2 \leq \sigma_k^2$  for  $k = 1, \dots, K$ .

**Assumption 4.** The expected squared norm of stochastic gradients is uniformly bounded, i.e.,  $\mathbb{E} \|\nabla F_k(\theta_t^k, \xi_t^k)\|^2 \leq G^2$  for all  $k = 1, \dots, K$  and  $t = 1, \dots, T-1$ .

The basic FedAvg model has been proven to converge to a global optimum in non-iid settings [22]. When all the devices participate in the aggregation step and the FedAvg algorithm terminates after T rounds, the following lemma will hold.

**Lemma 1** (FedAvg convergence bound [22]). Let Assumptions 1 to 4 hold and  $L, \mu, \sigma_k, G$  be defined therein. Choose  $\kappa = \frac{L}{\mu}, \gamma = \max\{8\kappa, E\}$  and the learning rate  $\eta_t = \frac{2}{\mu(\gamma+t)}$ . Then FedAvg with full device participation to the optimal  $F^*$  satisfies:

$$\mathbb{E}\left[F(\theta_T)\right] - F^* \le \frac{\kappa}{\gamma + T - 1} \left(\frac{2B}{\mu} + \frac{\mu\gamma}{2} \mathbb{E}\|\theta_1 - \theta^*\|^2\right),\tag{7}$$

where  $B = \sum_{k=1}^{K} p_k^2 \sigma_k^2 + 6L\Gamma + 8(E-1)^2 G^2$  and  $\Gamma = F^* - \sum_{k=1}^{K} p_k F_k^*$  measuring the degree of non-iid.

Proof. See Section 3.2 in [22].

Different from conventional FL, in the CL context, the data distribution will shift at every step, introducing the catastrophic forgetting issue. In our framework, we avoid this problem by leveraging the diffusion model to generate labeled synthetic data that can reflect the experienced historical real data without storing them physically. Therefore, at step t + 1, the input data  $\mathcal{X}_{t+1}$  is the combination of real data  $\mathcal{D}_{t+1}$  and synthetic data  $\mathcal{G}_t$  from the diffusion model recovering the data from the previous step, satisfying:

$$\mathcal{X}_{t+1} \equiv \mathcal{D}_{t+1} \cup \delta \cdot \mathcal{G}_t,\tag{8}$$

where  $\delta$  is an extra scale factor controlling the amount of generated synthetic data compared to real data to mitigate the negative impacts of inaccurate synthesis generations that cannot represent previous real inputs. For simplicity of the derivation, we omit this factor (set  $\delta = 1$ ) in this proof, while we use the sensitivity study regarding  $\delta$  in Appendix H.2 for a thorough discussion. Besides, according to equation (8), the sampled data distributions  $\xi_{t+1}^k$  regarding each part can be describe as follows:

$$\xi_{t+1}^k \sim \mathcal{X}_{t+1}, \quad \tilde{\xi}_{t+1}^k \sim \mathcal{D}_{t+1}, \quad \tilde{\tilde{\xi}}_{t+1}^k \sim \mathcal{G}_t, \tag{9}$$

From t to t + 1, the input data distribution is different as new data loaded at t + 1 has not been seen at t, and the synthetic data recovering the loaded data at t will deviate from the real data distribution. We capture such data distribution shift in a controllable range, measured by  $\Delta_t$ , which aligns with most learning scenarios. This character is depicted in the following assumption.

**Assumption 5.** In each continual learning step, the incoming data distribution shift is bounded and deviation of can be captured in a measurable range with  $\Delta_t$ , which is:  $D_{\text{KL}}(F_t^*(\theta_t^k; \xi_t^k) \parallel F_{t+1}^*(\theta_{t+1}^k; \tilde{\xi}_{t+1}^k)) \leq \Delta_t$ .

Based on the given assumption, we measure the distance between  $F_t^*$  and  $F_{t+1}^*$  with KL divergence:

**Theorem 1** (Data distribution deviation measurement). *The following equation can further bound the KL divergence:* 

$$D_{\mathrm{KL}}\left(F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \parallel F_{t+1}^{*}(\theta_{t+1}^{k};\xi_{t+1}^{k})\right) \\ \leq \frac{1}{2}\left[D_{\mathrm{KL}}\left(F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \parallel F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k})\right) + \Delta_{t}\right],$$
(10)

where  $\Delta_t$  is defined by the upper bound of incoming data distribution shift based on Assumption 5.

Proof. See Appendix A.

$$\Box$$

 $\square$ 

The KL divergence in the right-hand side of equation (10) indicates the distance between original real data from step t and generated synthetic data in step t + 1. We use the diffusion model as the generative model in our framework, where we aim to reconstruct the real data distribution from the previous step with newly generated synthetic data. Therefore, it is equivalent to measuring the reconstruction convergence of the incorporated diffusion model. Recently, such a convergence bound has been proposed in [21], summarized in the following lemma.

**Lemma 2** (Convergence of data generation via the diffusion model) [21]). Suppose there exists  $\kappa > 0$  controlling the diffusion step size  $\gamma$  such that for each m = 0, ..., M - 1, we have  $\gamma_m \leq \kappa \in \{1, N - n_{m+1}\}$  where N is the total step in the diffusion model. Then, the bound of the approximated reverse process for data generation is:

$$D_{\mathrm{KL}}\left(F_t^*(\theta_t^k;\xi_t^k) \parallel F_{t+1}^*(\theta_{t+1}^k;\tilde{\xi}_{t+1}^k)\right) \lesssim \epsilon_{\mathrm{score}}^2 + \kappa^2 dM + \kappa dN + d\exp(-2N),$$
(11)

where d is the dimension of the data and  $\epsilon_{\text{score}}$  denotes the maximum score approximation [24] error.

Proof. See Section 3 in [21].

Therefore, by integrating Lemma 1, 2 and Theorem 1, our CFL framework with the diffusion model DCFL can be proven bounded during the learning.

**Theorem 2** (Convergence of DCFL). *The convergence bound derived from Lemma 1, 2 and Theorem 1 is:* 

$$\mathbb{E}\left[F(\theta_T)\right] - F_T^* \leq \frac{\kappa}{\gamma + T - 1} \left(\frac{2B}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}\|\theta_1 - \theta^*\|^2\right) \\ + (1 - 2^{-T})(\epsilon_{\text{score}}^2 + \kappa^2 dM + \kappa dN \\ + d\exp(-2N)) + 2^{-T}\Delta_T.$$
(12)

Proof. See Appendix B.

The bound calculated in equation (12) offers a theoretical measurable convergence bound for our proposed continual federated learning model, which utilizes the diffusion model as a synthetic data generator. The convergence bound, as outlined in Theorem 2, comprises three components: the convergence bound of the federated learning model, the bound of the diffusion model, and the divergence of the data distribution. As the number of communication rounds approaches infinity  $(T \to \infty)$ , the first and third terms in equation (12) tend towards zero, rendering only the second term relevant. This suggests that the convergence ultimately hinges on the performance of the introduced diffusion model. However, according to Corollary 1 in [21], under specific conditions, the second term relies solely on the error  $\epsilon_{\text{score}}^2$ , signifying the final convergence of the entire system.

It is worth noting that the model experiences an inevitable distribution shift at each step. However, the last term in equation (12) decreases monotonically, indicating that the deviation  $2^{-t}\Delta_t$  regarding a particular round t is mitigated with each training step, as the diffusion model will contribute synthetic data for alleviating the data distribution shift.

### **4** Experiments

#### 4.1 Experimental Setup

**Datasets.** We adopt commonly used datasets, including **MNIST** [25], **Fashion-MNIST** [26], and **CIFAR-10** [27], for the Class Incremental IID and Class Incremental Non-IID CFL scenarios described in Figure 1. Our sampling method aligns with FedAvg's but is adapted for the CFL setting. Specifically, we uniformly split the datasets into 200 *shards* based on their classes, with each client accessing only 2 shards during any given *session* (i.e., a period of time). The distribution of data changes in subsequent sessions according to the different CFL scenario settings. We consider T = 100 communication rounds for preliminary experiments, with the clients' data distribution changing every 20 rounds, resulting in 5 sessions (S = 100/20 = 5). Additionally, we use the popular domain generalization dataset **PACS** [28] for the Domain Incremental CFL scenario. We consider each client to have all classes within any given domain, and the clients' data changes across the 4 domains in the sequence Sketch  $\rightarrow$  Cartoon  $\rightarrow$  Art Painting  $\rightarrow$  Photo (increasing the level of realism over time) [29]. Details of the datasets, scenario settings, and data preprocessing can be found in Appendix D.

**Implementation.** To reduce computational overhead, the target model is trained 20 times during any session, whereas the diffusion model is trained only once and generates synthetic data once. Apart from T = 100 communication rounds, we set the target model to undergo  $E_{\theta} = 5$  local epochs, while the diffusion model is trained for  $E_{\omega} = 100$  or  $E_{\omega} = 1000$  local epochs. Both models employ the Adam optimizer with a learning rate of 1e-4. We employ the same CNN architecture used in FedAvg for constructing the target model, consisting of two convolutional layers and two fully connected layers. The backbone of the diffusion model is a conditional UNet, structured with one convolutional layer and four-channel layers scaled to 64, 128, 128, and 256, respectively. Real labels (and the domain for PACS) are utilized as conditions to guide data synthesis by the diffusion model. We set the diffusion model to generate a number of synthetic samples equal to the number of real samples, the ratio  $\delta = 1$ . Details of implementation details can be found in Appendix E.

**Baselines.** We compare our DCFL with baselines with several frameworks: FL algorithms, FL with classical CL methods, and state-of-the-art (SOTA) CFL frameworks. Specifically, for FL, we compare with **FedAvg** [1] and **FedProx** [30]. For the integration of FL and CL, we implement **FedAvg+LwF** [31] and **FedAvg+EWC** [32]. We adopt **FedAvg+ACGAN** [33] as the integration of the FL and

generative model. For SOTA CFL frameworks, we consider **FedCIL** [34], **FOT** [14], **MFCL** [11], and **TARGET** [12]. Details of the baseline algorithms and settings can be found in Appendix F.

#### 4.2 Main Experimental Results

Figure 3 illustrates the performance comparison of DCFL against baselines across three CFL scenarios on three datasets. The results indicate that catastrophic forgetting significantly impacts the clients' ability to retain prior knowledge. In the Class-Incremental IID scenario, the class distributions among all clients change every T = 20 rounds, with each client generally aware of a subset (i.e., 20%) of all classes at any given time. The stepwise improvement in accuracy suggests effective retention and mastery of both current and previous knowledge by the model. For the Class Incremental Non-IID scenario, all clients are aware of all class information at any time, leading to convergent behavior across all frameworks. However, the replay functionality in our framework, which ensures that each client's target model is trained with multiple class information, facilitates faster convergence.



Figure 3: **Main Result** - Comparison of Model Convergence with Baselines. Refer to Figure 5 for the Domain Incremental scenario.

Table 1 shows that, compared to the baselines, our DCFL framework demonstrates significant advantages across all three datasets in both the Class Incremental IID and Class Incremental Non-IID scenarios. Specifically, the proposed approach achieves improvements of  $32.61 \pm 15.91\%$  in the Class Incremental IID scenario (compared to the best baseline, FedAvg+ACGAN),  $15.16 \pm 6.97\%$  in the Class Incremental Non-IID scenario (compared to the best baseline, FedAvg+ACGAN),  $15.16 \pm 6.97\%$  in the Domain Incremental scenario (compared to the best baseline, FedAvg+ACGAN). This shows that our approach is better at overcoming catastrophic forgetting than the baselines. Unlike FedAvg+ACGAN, which also relies on a generative model for replay, our use of a diffusion model generates higher-quality synthetic images, thereby avoiding noise contamination in the training dataset. Furthermore, the SOTA baselines suffer from their inherent limitations: FedCIL experiences unstable convergence due to multiple loss functions, while FOT demonstrates slow convergence, rendering it impractical. Meanwhile, both MFCL and TARGET produce low-quality synthetic images as the generative model is trained solely on the basis of knowledge distillation.

#### 4.3 Analysis of Scenarios

In the previous subsection, we compared the accuracy of DCFL with the baselines on the global test set. In this subsection, we discuss the performance within the scenarios. Specifically, we demonstrate the results using the standard CL evaluation method in Appendix G, considering only the classes or domains encountered so far, following the settings in CL. The Class Incremental Non-IID scenario is unsuitable for CL evaluation because it includes all classes at any given time.

**Class Incremental IID** is the most challenging CFL scenario because the server cannot aggregate global knowledge, as all clients only have the same subset of classes at any given time. Due to this characteristic, catastrophic forgetting is particularly severe. Figure 3 shows that most traditional

Scenario	Cla	ass Incremental	IID	Class	Domain Incremental		
Dataset	MNIST	FMNIST	CIFAR-10	MNIST	FMNIST	CIFAR-10	PACS
FedAvg [1]	19.77 ↓74.92	19.96 ↓52.54	18.74 19.39	82.99 15.15	71.19 16.60	42.89 14.27	29.96 18.06
FedProx [30]	19.78 474.91	19.96 \52.54	18.60 19.53	87.81 10.33	70.93 16.86	42.24 14.92	33.82 14.21
FedAvg+LwF [31]	19.77 174.92	19.96 ↓ 52.54	18.67 19.46	85.16 12.98	68.24 19.55	42.75 14.41	34.57 13.46
FedAvg+EWC [32]	19.78 474.91	19.95 \52.55	18.64 19.49	93.81 4.33	74.37 13.42	46.15 11.01	38.02 10.01
FedAvg+ACGAN [33]	42.15 \52.54	40.83 \31.67	24.52 13.61	92.31 \$5.83	70.70 17.09	34.60 \22.56	40.57 17.45
FedCIL [34]	46.36 48.33	38.17 \34.33	21.59 16.54	89.30 \$8.84	62.25 \25.54	25.80 \31.36	27.79 20.24
FOT [14]	32.18 462.51	27.47 45.03	13.47 124.66	78.97 19.17	63.82 23.97	37.27 19.89	39.42 8.60
MFCL [11]	21.14 73.55	19.81 52.69	18.74 19.39	76.02 22.12	64.08 23.71	37.98 19.18	36.16 11.86
TARGET [12]	20.55 ↓74.14	19.96 ↓52.54	18.62 ↓19.51	92.68 \ 5.46	68.96 ↓18.83	36.63 \20.53	30.72 ↓17.30
DCFL (Ours)	94.69	72.50	38.13	98.14	87.79	57.16	48.02

Table 1: Main Result - Comparison of Final Accuracy with Baselines.

↓ indicates the accuracy decrease of the baselines compared to our DCFL framework.

FMNIST refers to Fashion-MNIST.

baselines completely fail to retain previous knowledge because all knowledge appears only in a single session. In this scenario, replay-based approaches are the best. This is because generative models can replay very old knowledge, whereas other approaches struggle to retain even the knowledge from the previous session. Among these, the diffusion model's high-quality synthetic images avoid error propagation, creating a more positive and effective feedback loop than other generative models like ACGAN. CL analysis of the Class Incremental IID scenario can be found in Appendix G.1 and Tables 3 and 4.

**Class Incremental Non-IID** is the simplest (i.e., the least affected by catastrophic forgetting) CFL scenario because the server can aggregate a generalized global model based on the clients with diverse class distributions. So that the clients can continually learn global knowledge of other classes from other clients. As shown in Figure 3, the Class Incremental Non-IID scenario mirrors the convergence process of traditional FedAvg in a Non-IID setting. Even with changes in client data, FedAvg can eventually converge based on its proven methodology. However, our DCFL framework accelerates this process, providing more stability and faster convergence than the baselines. Benefiting from the diffusion model, clients can learn features of multiple classes simultaneously, transforming the Non-IID problem into an IID problem: as clients iterate through communication rounds, they gradually acquire information about global classes through synthetic datasets.

**Domain Incremental** is a moderately challenging task. Similar to Class Incremental IID, it requires retaining knowledge of all domains without forgetting, as clients will not revisit them. However, we ensure that all clients have access to all classes within each domain, which makes it easier for clients to learn global class knowledge. The diffusion model excels at generating images across different domains and classes, outperforming all baselines. Notably, for Domain Incremental, we consider both domain and class as conditions for synthetic data generation, which is crucial due to the significant differences between domains in PACS that need special consideration. CL analysis of the Domain Incremental scenario can be found in Appendix G.2 and Figure 5, and Table 5.

### 5 Related Work

**Federated Learning (FL)** has gained significant attention for its ability to train models across distributed data sources without centralizing data. A plethora of FL frameworks are designed for distributed learning, including asynchronous FL, decentralized FL, and hierarchical FL, among others, as well as numerous studies aimed at addressing the challenges of non-IID data and heterogeneity within FL [35, 36, 37, 38]. Some research has considered FL in dynamic contexts, such as varying communication conditions, client mobility status, client availability, and resource constraints [39, 40, 41, 42]. However, current research on the issue of dynamically varying client datasets (i.e., CFL) is not comprehensive, and existing solutions are limited to single CFL scenarios [43, 13, 44]. In this work, we enumerate three CFL scenarios and experimentally demonstrate that DCFL is a universal solution.

**Continual Learning (CL)** approaches mitigate catastrophic forgetting [45] issues while sequentially learning new tasks. Among these, some methods introduce a replay memory where the experienced data can be stored [46, 47]. Others include structure-based methods [48] and regularization-based approaches [49, 32] to reduce the forgetting. Recently, leveraging generative replay [50, 51, 10]

offers a promising avenue for preserving past knowledge while adapting to new tasks, thus addressing the evolving nature of learning scenarios without storing the past data. However, some traditional CL algorithms, such as Learning without Forgetting (LwF) [31], may not be applicable if there is no overlap between the previous and current data distributions. LwF typically assumes that new tasks share commonalities with previous tasks, enabling the model to maintain old knowledge while learning new information. This assumption breaks down when previous tasks' classes (e.g.,  $\{0, 1\}$ ) are completely different from those of the new tasks (e.g.,  $\{2, 3\}$ ). Therefore, our proposed diffusion model as replay has been demonstrated to be a universal solution for all different CFL scenarios.

**Diffusion Models** have showcased superior performance in generating detailed and diverse instances [52] and succeeded in many areas, including computer vision [53, 54] and reinforcement learning [55, 56]. There exist three main formulations of diffusion models: Score-based Generative Models (SGM) [57, 58], Denoised Diffusion Probabilistic Models (DDPM) [20, 16, 59], and Stochastic Differential Equations (Score SDE) [60, 61]. On this basis, [62] shows the superiority of the diffusion model compared to other generative models via training a classifier on noisy images and using gradients to guide the diffusion sampling process to the conditioning information, such as labels, by altering the noise prediction. Besides, [63] also shows the possibility of running conditional diffusion without an independent classifier. In this work, we exploit a conditional diffusion model in our CFL framework for historical data recovery.

# 6 Conclusion

In this paper, we introduce DCFL, a novel Continual Federated Learning (CFL) framework that incorporates diffusion models for synthetic historical data generation. The synthetic data generated by DCFL helps in retaining memory of previously encountered input data distributions and mitigates the impact of data distribution shifts during learning, thus avoiding latent catastrophic forgetting issues. Theoretical analyses are provided to support the convergence of our proposed framework. Furthermore, experimental results on multiple datasets showcase the effectiveness of DCFL in addressing FL tasks and mitigating the negative impact of input distribution shifts. Currently, as one limitation, we have not explored utilizing multimodality data, such as text prompts, for enhanced data synthesis in CFL tasks, which can be considered as future work.

### References

- 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*, pages 1273–1282, 2017.
- [2] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends*<sup>®</sup> in machine learning, 14(1–2):1–210, 2021.
- [3] Jie Ding, Eric Tramel, Anit Kumar Sahu, Shuang Wu, Salman Avestimehr, and Tao Zhang. Federated learning challenges and opportunities: An outlook. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8752–8756. IEEE, 2022.
- [4] Mario Chahoud, Hani Sami, Azzam Mourad, Safa Otoum, Hadi Otrok, Jamal Bentahar, and Mohsen Guizani. On-demand-fl: A dynamic and efficient multi-criteria federated learning client deployment scheme. *IEEE Internet of Things Journal*, 2023.
- [5] Yeongwoo Kim, Ezeddin Al Hakim, Johan Haraldson, Henrik Eriksson, José Mairton B da Silva, and Carlo Fischione. Dynamic clustering in federated learning. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.
- [6] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [7] Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.
- [8] Liangqi Yuan, Yunsheng Ma, Lu Su, and Ziran Wang. Peer-to-peer federated continual learning for naturalistic driving action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5249–5258, 2023.
- [9] 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*, pages 12073–12086. PMLR, 2021.
- [10] Daiqing Qi, Handong Zhao, and Sheng Li. Better generative replay for continual federated learning. In *The Eleventh International Conference on Learning Representations*, 2022.
- [11] 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.
- [12] 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*, pages 4782–4793, 2023.
- [13] 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, pages 10164–10173, 2022.
- [14] Yavuz Faruk Bakman, Duygu Nur Yaldiz, Yahya H Ezzeldin, and Salman Avestimehr. Federated orthogonal training: Mitigating global catastrophic forgetting in continual federated learning. arXiv preprint arXiv:2309.01289, 2023.
- [15] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. Advances in neural information processing systems, 30, 2017.
- [16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [17] Tomer Amit, Tal Shaharbany, Eliya Nachmani, and Lior Wolf. Segdiff: Image segmentation with diffusion probabilistic models. *arXiv preprint arXiv:2112.00390*, 2021.
- [18] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. Advances in Neural Information Processing Systems, 34:17981–17993, 2021.

- [19] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022.
- [20] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.
- [21] Joe Benton, Valentin De Bortoli, Arnaud Doucet, and George Deligiannidis. Linear convergence bounds for diffusion models via stochastic localization. arXiv preprint arXiv:2308.03686, 2023.
- [22] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of FedAvg on non-iid data. arXiv preprint arXiv:1907.02189, 2019.
- [23] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International conference on artificial intelligence and statistics*, pages 2021– 2031. PMLR, 2020.
- [24] Minshuo Chen, Kaixuan Huang, Tuo Zhao, and Mengdi Wang. Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data. In *International Conference on Machine Learning*, pages 4672–4712. PMLR, 2023.
- [25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [26] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.
- [27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [28] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.
- [29] Haiyan Zhao, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Does continual learning equally forget all parameters? In *International Conference on Machine Learning*, pages 42280–42303. PMLR, 2023.
- [30] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning* and systems, 2:429–450, 2020.
- [31] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.
- [32] 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. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- [33] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.
- [34] Daiqing Qi, Handong Zhao, and Sheng Li. Better generative replay for continual federated learning. *arXiv preprint arXiv:2302.13001*, 2023.
- [35] Kilian Pfeiffer, Martin Rapp, Ramin Khalili, and Jörg Henkel. Federated learning for computationally constrained heterogeneous devices: A survey. ACM Computing Surveys, 55(14s):1–27, 2023.
- [36] Liangqi Yuan, Ziran Wang, Lichao Sun, Philip S. Yu, and Christopher G. Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 11(21):34617 – 34638, May 2024.
- [37] Hanhan Zhou, Tian Lan, Guru Prasadh Venkataramani, and Wenbo Ding. Every parameter matters: Ensuring the convergence of federated learning with dynamic heterogeneous models reduction. Advances in Neural Information Processing Systems, 36, 2024.

- [38] Hanhan Zhou, Tian Lan, Guru Prasadh Venkataramani, and Wenbo Ding. Federated learning with online adaptive heterogeneous local models. In Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022), 2022.
- [39] Weijie Liu, Xiaoxi Zhang, Jingpu Duan, Carlee Joe-Wong, Zhi Zhou, and Xu Chen. Dynamite: Dynamic interplay of mini-batch size and aggregation frequency for federated learning with static and streaming dataset. *IEEE Transactions on Mobile Computing*, 2023.
- [40] Tinghao Zhang, Kwok-Yan Lam, Jun Zhao, and Jie Feng. Joint device scheduling and bandwidth allocation for federated learning over wireless networks. *IEEE Transactions on Wireless Communications*, 2023.
- [41] Liangqi Yuan, Dong-Jun Han, Su Wang, Devesh Upadhyay, and Christopher G Brinton. Communication-efficient multimodal federated learning: Joint modality and client selection. *arXiv preprint arXiv:2401.16685*, 2024.
- [42] Jing Qiao, Zuyuan Zhang, Sheng Yue, Yuan Yuan, Zhipeng Cai, Xiao Zhang, Ju Ren, and Dongxiao Yu. Br-defedrl: Byzantine-robust decentralized federated reinforcement learning with fast convergence and communication efficiency. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 141–150. IEEE, 2024.
- [43] Riccardo Volpi, Diane Larlus, and Grégory Rogez. Continual adaptation of visual representations via domain randomization and meta-learning. In *Proceedings of the IEEE/CVF Conference* on Computer Vision and Pattern Recognition, pages 4443–4453, 2021.
- [44] Jungwuk Park, Dong-Jun Han, Jinho Kim, Shiqiang Wang, Christopher Brinton, and Jaekyun Moon. Stablefdg: Style and attention based learning for federated domain generalization. *Advances in Neural Information Processing Systems*, 36, 2024.
- [45] Cuong V Nguyen, Alessandro Achille, Michael Lam, Tal Hassner, Vijay Mahadevan, and Stefano Soatto. Toward understanding catastrophic forgetting in continual learning. *arXiv* preprint arXiv:1908.01091, 2019.
- [46] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.
- [47] 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.
- [48] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.
- [49] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.
- [50] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.
- [51] Chenshen Wu, Luis Herranz, Xialei Liu, Joost Van De Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *Advances in neural information processing systems*, 31, 2018.
- [52] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. ACM Computing Surveys, 56(4):1–39, 2023.
- [53] Yongsheng Mei, Guru Venkataramani, and Tian Lan. Exploiting partial common information microstructure for multi-modal brain tumor segmentation. In *Workshop on Machine Learning for Multimodal Healthcare Data*, pages 64–85. Springer, 2023.
- [54] Lilian Weng. Diffusion models video generation. *lilianweng.github.io*, Apr 2024.
- [55] Zeyu Fang and Tian Lan. Learning from random demonstrations: Offline reinforcement learning with importance-sampled diffusion models. *arXiv preprint arXiv:2405.19878*, 2024.
- [56] Jiayu Chen, Bhargav Ganguly, Yang Xu, Yongsheng Mei, Tian Lan, and Vaneet Aggarwal. Deep generative models for offline policy learning: Tutorial, survey, and perspectives on future directions. *arXiv preprint arXiv:2402.13777*, 2024.

- [57] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.
- [58] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.
- [59] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.
- [60] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456, 2020.
- [61] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. Advances in neural information processing systems, 34:1415– 1428, 2021.
- [62] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- [63] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop* on Deep Generative Models and Downstream Applications, 2021.
- [64] Nipun Wijerathne Varuna Jayasiri. labml.ai annotated paper implementations, 2020.

# A Proof of Theorem 1

*Proof.* Given the KL divergence, we have:

$$D_{\mathrm{KL}}\left(F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \| F_{t+1}^{*}(\theta_{t+1}^{k};\xi_{t+1}^{k})\right) = \sum F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \log\left(\frac{F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k})}{F_{t+1}^{*}(\theta_{t+1}^{k};\xi_{t+1}^{k})}\right) = \sum F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \frac{1}{2} \log\left[\frac{F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k})}{(1+\delta)^{-1}\left(F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k}) + F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k})\right)}\right]^{2}$$

$$\leq \sum F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \frac{1}{2} \left(\log\frac{F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k})}{(1+\delta)^{-1}F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k})} + \log\frac{F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k})}{(1+\delta)^{-1}F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k})}\right) = \frac{1}{2} \left[ D_{\mathrm{KL}}\left(F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \| F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k})\right) + D_{\mathrm{KL}}\left(F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \| F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k})\right) \right]$$

$$\stackrel{(a)}{=} \frac{1}{2} \left[ D_{\mathrm{KL}}\left(F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \| F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k})\right) + \Delta_{t} \right],$$

$$(13)$$

where (a) adopts Assumption 5.

This concludes the proof.

# **B Proof of Theorem 2**

. .

.

*Proof.* Considering the generated data at step t will be the input data of step t + 1, we have the iterative measurement of the bound regarding this data distribution shift. According to Theorem 1 and equation (9), we have:

$$D_{\mathrm{KL}} \left( F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \parallel F_{t+1}^{*}(\theta_{t+1}^{k};\xi_{t+1}^{k}) \right) \\ \leq \frac{1}{2} \left[ D_{\mathrm{KL}} \left( F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \parallel F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k}) \right) + \Delta_{t} \right] \\ = \frac{1}{2} \left[ D_{\mathrm{KL}} \left( F_{t}^{*}(\theta_{t}^{k};\xi_{t}^{k}) \parallel F_{t+1}^{*}(\theta_{t+1}^{k};\tilde{\xi}_{t+1}^{k}) \right) + \frac{1}{2} \left[ D_{\mathrm{KL}} \left( F_{t+1}^{*}(\theta_{t+1}^{k};\xi_{t+1}^{k}) \parallel F_{t+2}^{*}(\theta_{t+2}^{k};\tilde{\xi}_{t+2}^{k}) \right) + \Delta_{t+1} \right] \right] \\ \cdots \\ \begin{pmatrix} a \\ \leq \left( 1 - \frac{1}{2^{T}} \right) \left( \epsilon_{\mathrm{score}}^{2} + \kappa^{2} dM + \kappa d\bar{T} + d \exp(-2\bar{T}) \right) + \frac{1}{2^{T}} \Delta_{T}, \end{cases}$$
(14)

where we further simplify the results at the last inequality (a) using the summation of the geometric series and Lemma 2.

We use the FedAvg model as the backbone for federated learning and the diffusion model to generate synthetic data for continual learning. The convergence of the system takes into account the inevitable shift in distribution, which can be measured using KL divergence. This distribution shift occurs at each iteration from t = 1 to T, nested as shown in equation (14). Initially, the convergence is determined by:

$$\mathbb{E}\left[F(\theta_T)\right] - F_T^* \leq \mathbb{E}\left[F(\theta_T)\right] + D_{\mathrm{KL}}(F_1^* \parallel F_2^*) - F_T^*.$$
(15)

Equation (14) has provided us with an upper bound about the introduced data distribution shift ranging from t = 1, ..., T. Considering this in the designed CFL system, the final convergence bound can be derived as:

$$\mathbb{E}\left[F(\theta_T)\right] - F_T^* \le A + D_{\mathrm{KL}}(F_1^* \parallel F_2^*)$$
  
$$\le A + \left(1 - \frac{1}{2^T}\right) \left(\epsilon_{\mathrm{score}}^2 + \kappa^2 dM + \kappa d\bar{T} + d\exp(-2\bar{T})\right) + \frac{1}{2^T} \Delta_T.$$
(16)

where  $A \triangleq \frac{\kappa}{\gamma + T - 1} \left( \frac{2B}{\mu} + \frac{\mu\gamma}{2} \mathbb{E} \| \theta_1 - \theta^* \|^2 \right)$  based on Lemma 1. This concludes the proof.  $\Box$ 

# C Complete algorithm

We present our complete algorithm combining FL with the diffusion model below as the complete extension of the Algorithm 1. As noted in Algorithm 2, we illustrate the alignment of the diffusion model with local clients. In practice, depending on the task's complexity, we may transition from an unguided diffusion model, such as a common DDPM diffusion model, to a conditioned diffusion model. While the former is sufficient for most simple generation workloads, the latter performs better for tasks requiring complicated synthetic unstructured data generation, such as images with delicate contents. To utilize the conditioned diffusion model, both the data and its label are passed as inputs for model training. This often necessitates thorough model training to achieve precise data generation, demanding significant computational resources. To address this challenge, we can leverage a pre-trained model trained on a large general dataset and fine-tune our diffusion model accordingly based on our purposes.

Algorithm 2 Proposed DCFL Framework Complete Procedure

**Input:** Communication rounds (*T*), client datasets ( $\mathcal{D}_t^k$ ), target model, loss function, and learning rate ( $\theta$ , *F*,  $\eta_{\theta}$ ), diffusion model, diffusion step, loss function, and learning rate ( $\omega$ , *N*,  $\mathcal{L}$ ,  $\eta_{\omega}$ ) **Output:** Generalized global model ( $\theta_T$ )

**Local update** of the *k*-th client:

1: initialize  $\omega_0$ 2: for each round t = 1 : T do if t = 0 then 3: Data includes current real data only:  $\mathcal{X}_0^k = \mathcal{D}_0^k$ 4: 5: else if t > 1 then  $\begin{array}{l} \triangleright \ \mathbf{Generate synthetic data} \\ \mathcal{G}_{t-1,n}^k \sim \mathcal{N}(0, \mathbf{I}) \\ \mathbf{for} \ n = N: 1 \ \mathbf{do} \\ \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \ \mathrm{if} \ t > 1 \ \mathrm{else} \ \mathbf{z} = \mathbf{0} \end{array}$ 6: 7: 8: 9:  $\mathcal{G}_{t-1,n-1}^{k} = \frac{1}{\sqrt{\alpha_n}} \left( \mathcal{G}_{t-1,n}^{k} - \frac{1-\alpha_n}{\sqrt{1-\alpha_n}} \epsilon_{\omega}(\mathcal{G}_{t-1,n}^{k}, \boldsymbol{y}, n) \right) + \sqrt{\beta_n} \boldsymbol{z}, \text{ given labels } \boldsymbol{y}$ 10: end for 11: Obtain synthetic data  $\mathcal{G}_{t-1}^k \equiv \mathcal{G}_{t-1,0}^k$ 12: Combine real and synthetic data with a scale factor  $\delta$ :  $\mathcal{X}_t^k = \mathcal{D}_t^k \cup \delta \cdot \{\mathcal{G}_{t-1}^k, y\}$ 13: 14: end if  $-\cdot - \cdot - - \uparrow$  Generating synthetic data  $\uparrow - \cdot - \cdot - - \downarrow$  Training models  $\downarrow - \cdot - \cdot - \cdot - \downarrow$ **repeat**  $E_{\theta}$  epochs 15:  $\begin{array}{c} \theta_t^k \leftarrow \theta_{t-1}^k - \eta_\theta \nabla F_k(\theta_{t-1}^k; \mathcal{X}_t^k) \\ \text{until } \theta \text{ converged} \end{array}$  $\triangleright$  Train target model 16: 17: repeat  $E_{\omega}$  epochs  $\omega_t^k \leftarrow \omega_{t-1}^k - \eta_{\omega} \nabla \mathcal{L}_k(\omega_{t-1}^k; \mathcal{X}_t^k, n)$ until  $\omega$  converged 18: 19: ▷ Train diffusion model 20: 21: end for 22: **return**  $\theta_t^k$  to server Global update of the server 1: initialize  $\theta_0$ 2: for each round t = 1 : T do 3: for each client k = 1 : K in parallel do  $\theta_t^k \leftarrow k$ -th client's *local update* 4: end for  $\theta_t \leftarrow \sum_{k=1}^{K} p_k \theta_t^k$ 5: 6: ▷ Aggregate target models 7: end for

# **D** Datasets

We use the following four datasets for our experiments, each applied to different CFL scenarios, as shown in Table 2.

Tuble 2. Troper des et Dutusets und Three et El Secharlos.								
Scenario	<b>Class Incremental IID</b>	Class Incremental Non-IID	Domain Incremental					
Dataset	MNIST Fashion-MNIST CIFAR-10	MNIST Fashion-MNIST CIFAR-10	PACS					
Attribute	$ \begin{array}{ l } \mbox{Client } 0{:}\{0,1\} \rightarrow \{2,3\} \\ \mbox{Client } 1{:}\{0,1\} \rightarrow \{2,3\} \end{array} $	$ \begin{array}{ c c } \mbox{Client } 0{:}\{0,1\} \rightarrow \{2,3\} \\ \mbox{Client } 1{:}\{2,3\} \rightarrow \{4,5\} \end{array} $	$ \begin{vmatrix} \text{Client 0: Domain } 0 \to 1 \\ \text{Client 1: Domain } 0 \to 1 \end{vmatrix} $					
Total Rounds T	100	100	80					
Session/Domain	5	5	4					
Number of Classes at Any Time	All Clients: 2 Each Client: 2	All Clients: 10 Each Client: 2	All Clients: 7 Each Client: 7					

Table 2: Properties of Datasets and Three CFL Scenarios.

### • Class-Incremental IID and Class-Incremental Non-IID.

- **MNIST** [25] is a large dataset for handwritten digit recognition, containing 70,000 grayscale images of size  $28 \times 28$  pixels, representing digits from 0 to 9. To accommodate the input dimensions of the UNet in our diffusion model, we resize the images to  $32 \times 32$  pixels. We split the MNIST dataset into 20 clients, with each client having 5 sessions and each session containing 2 classes. Therefore, each session in every client has 300 samples for any given class.
- Fashion-MNIST [26] is a large dataset for clothing recognition, similar in size and format to MNIST, with 70,000 grayscale images of size 28 × 28 pixels. It includes categories such as T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. Compared to MNIST, the images in Fashion-MNIST have more complex shapes and textures, increasing the difficulty of recognition. We adopt the same loading, partitioning, and preprocessing procedures as with MNIST.
- CIFAR-10 [27] is a large dataset for object recognition, consisting of 60,000 color images of size 32 × 32 pixels, each with three RGB color channels. The dataset includes objects such as Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck. Compared to MNIST and Fashion-MNIST, CIFAR-10 poses a greater challenge due to the complexity and diversity of backgrounds and objects. We split the CIFAR-10 dataset into 10 clients, with each client having 5 sessions and each session containing 2 classes. Therefore, each session in every client has 500 samples for any given class.

### • Domain-Incremental.

- PACS [28] is a widely used dataset for domain adaptation and generalization studies, featuring 4 significantly different visual domains: Photo, Art Painting, Cartoon, and Sketch. It contains a total of 9.991 color images of size 227x227 pixels, with each domain comprising: Photo (1,670 images), Art Painting (2,048 images), Cartoon (2,344 images), and Sketch (3,929 images). Each domain includes seven categories: Dog, Elephant, Giraffe, Guitar, Horse, House, Person. We use the PACS dataset for domain-incremental learning, ensuring that each client retains the same categories across different domains (the same as the IID setting in FedAvg). The entire PACS dataset is split into 80% training and 20% testing sets, with the training set further divided among clients according to the domain-incremental setting. We split the PACS dataset into 10 clients, with each client having 4 domains, each containing all classes. Consequently, the number of samples per domain in each client is approximately 312, 184, 161, and 131, respectively. Note that the sample sizes for each domain in PACS are different; and these are approximate values because we randomly split the training and testing sets in an 8:2 ratio, and different seeds result in varying sample sizes (we ensure that the number of samples per class is consistent within each client). These sample sizes include all classes, but the class distribution differs across domains, as shown in Figure 4.



Figure 4: **Data Distribution of PACS Dataset.** The figure shows the number of samples owned by each one single client.

# **E** Implementation Details

**Training Details.** For all experiments, both the target model and the diffusion model use the Adam optimizer with a learning rate of 1e-4 and a batch size of 32. For the target model, the training epochs are set to  $E_{\theta} = 5$ . For the diffusion model, the training epochs are set according to the complexity of the images:  $E_{\omega} = 100$  for MNIST and Fashion-MNIST, and  $E_{\omega} = 1000$  for CIFAR-10 and PACS. All models are implemented in PyTorch and trained on an NVIDIA A100 GPU with 40 GB of memory. Using MNIST in the Class Incremental IID scenario as an example, the execution time for a single client is approximately 0.5 seconds for training the target model, 180 seconds for training the diffusion model, and 200 seconds for generating synthetic samples.

**Target Model - CNN Architecture.** We use a CNN as the target model for MNIST, Fashion-MNIST, CIFAR-10, and PACS. The architecture consists of two convolutional layers, the first with 32 filters and the second with 64 filters, both using a kernel size of 5 and padding of 2. Each convolutional layer is followed by a ReLU activation and a max pooling layer with a pool size of 2. The output from the convolutional layers is then flattened and passed through a fully connected layer with 512 units, followed by another ReLU activation. The final layer is a fully connected layer with 10 units (or 7 units for PACS), corresponding to the number of classes, and a log-softmax activation function.

Diffusion Model - Conditional UNet Architecture. We use a conditional UNet as the backbone for the diffusion model in our framework. The architecture starts with an initial convolutional layer that projects the input image into a higher-dimensional space with 64 channels. It then processes the data through a series of downsampling and upsampling blocks. The downsampling path consists of four blocks with channel sizes of 64, 128, 128, and 256, respectively. Each block contains residual blocks and, in the deepest laver, attention mechanisms to enhance feature representation. Each residual block includes group normalization and dropout, ensuring stable training and preventing overfitting. In the middle of the network, a middle block contains both residual and attention mechanisms, maintaining the channel size at 256. The upsampling path mirrors the downsampling path but in reverse order, reducing the channel dimensions while merging features from corresponding downsampling layers through skip connections. This upsampling process includes upsampling layers to expand the spatial dimensions back to the original size. Finally, the model normalizes and activates the features before passing them through a final convolutional layer, which reduces the output to the desired number of image channels. We incorporate time and condition information (class for MNIST, Fashion-MNIST, and CIFAR-10; class and domain for PACS) into the model, where the condition dimension is set to 32 for MNIST, Fashion-MNIST, and CIFAR-10, and 64 for PACS. The original implementations of DDPM and UNet are sourced from labml\_nn library [64].

# **F** Baselines

We consider four types of baseline for comparison.

### • FL Algorithms.

- FedAvg [1] is the most representative and classic FL algorithm, where the server aggregates the received client models weighted by the number of client samples to obtain the global model.
- FedProx [30] is popular FL algorithm. Built on FedAvg, it introduces a proximal term in the loss function during local training to reduce the divergence between the local and global models, addressing client heterogeneity. We did not adapt this proximal term loss to the CFL scenario, but only compute the difference between the global model at round t 1 and the local model at round t. The proximal term loss function is  $\frac{\mu}{2} ||\theta_t^k \theta_{t-1}||^2$ . We set the weight parameter  $\mu = 1$  for the proximal term as in the original paper.

### • FL with Classical CL Methods.

- FedAvg+ LwF (Learning without Forgetting) [31] is a classic CL algorithm that utilizes knowledge distillation, requiring the new model to mimic the output of the old model on the same input when training on new tasks. We adapt LwF to the CFL scenario, where, in session s, the student model is guided by the teacher model from the last communication round of session s 1. In which the last communication round of the previous session can be calculated as  $t^* = \left\lfloor \frac{t}{T/S} \right\rfloor \times \frac{T}{S}$ , where T is the total number of rounds, and S is the number of sessions. The teacher model  $\theta_{t^*}$  guides all training in the next session. The LwF loss function is expressed as  $\mathcal{L}_{LwF} = \mathcal{L}_{task} + \lambda_{LwF} D_{KL} (\theta_{t^*}(\mathbf{x}) || \theta_t(\mathbf{x}))$ , where  $D_{KL}$  denotes the Kullback-Leibler (KL) divergence, and  $\lambda_{LwF}$  is a weighting factor used to balance the influence of previous and current tasks. We set  $\lambda_{LwF} = 1$ , consistent with the setting in [31].
- FedAvg+ EWC (Elastic Weight Consolidation) [32] is another classic CL algorithm that applies additional constraints to protect model parameters relevant to old tasks from significant updates. We adapt EWC to the CFL scenario similarly to LwF, using the model from the last communication round of the previous session  $\theta_{t^*}$  as the old task model to guide all training in the next session. The EWC loss function is  $\mathcal{L}_{\text{EWC}} = \mathcal{L}_{\text{task}} + \frac{\lambda_{\text{EWC}}}{2} \sum_i F_i^{t^*} (\theta_i^t \theta_i^{t^*})^2$ , where  $\lambda_{\text{EWC}}$  is the EWC penalty, and  $F_i$  are the values of the Fisher information matrix for the parameter  $\theta_i^{t^*}$ . We set  $\lambda_{\text{EWC}} = 400$ , consistent with the setting in [32].
- FL with Generative Model as Replay.
  - FedAvg+ ACGAN (Auxiliary Classifier Generative Adversarial Networks) [33] is a widely used conditional generative model. A traditional GAN consists of a generator that creates synthetic images and a discriminator that distinguishes between real and fake samples. ACGAN also extends this by requiring the discriminator to classify the samples, enabling conditional output. For the implementation of FedAvg+ACGAN, we replace the diffusion model in our DCFL framework with ACGAN while keeping the rest of the experimental settings identical, such as training epochs, learning rate, and the number of generated images.
- SOTA CFL Frameworks.
  - FedCIL [34] uses ACGAN for generating synthetic data while employing model consolidation to aggregate ACGANs from different clients and consistency enforcement to ensure that local training aligns better with the global model, thereby reducing training bias. However, FedCIL has some significant limitations, including (i) the server requires resources to train a server ACGAN, (ii) clients need to upload their ACGANs, causing privacy leakage, and (iii) it is not applicable to different tasks, such as object detection and semantic segmentation, since FedCIL only considers the discriminator in FL. In contrast, decoupling the generative model from the target model, as done in our DCFL framework and the baseline FedAvg+ACGAN, offers better scalability.
  - FOT [14] performs global principal subspace extraction to identify features critical to previous tasks, which are then protected during subsequent training to prevent forgetting. Subsequently, through the orthogonal projection aggregation method, when training new tasks, the server orthogonally projects model updates from the clients onto the orthogonal complement of the old tasks' subspace. This ensures that updates occur only in directions unrelated to the old

tasks, thereby minimizing the impact of learning new tasks on the performance of previous tasks. Although this approach does not require additional computation on the clients, it does necessitate additional computation on the server, such as orthogonal projection and subspace extraction. Moreover, FOT entails higher communication costs between clients and servers to transfer subspace information and orthogonal projections, raising concerns about potential privacy breaches.

- MFCL [11] trains a generative model on the server through knowledge distillation using the aggregated global model and then downloads this generative model to all clients to generate synthetic data and perform local training. MFCL optimizes four loss functions for training the generative model: cross entropy, diversity, batch statistics, and image prior. Although this is a data-free generative model training strategy, the quality of the generated models is suboptimal. While clients incur no additional computational overhead, the server requires substantial computational resources to train the generative model and incurs significant communication overhead to transmit the model.
- TARGET [12] trains a generative model on the server by knowledge distillation using the aggregated global model and then downloads the synthetic dataset to all clients for local training. TARGET optimizes the generative model using cross-entropy, KL loss with the student model, and batch normalization. The success of local training is heavily dependent on the quality of the synthetic dataset, which is difficult to guarantee. Additionally, the server incurs substantial computational overhead for training the generative model and significant communication overhead for transmitting the synthetic dataset.

# G Continual Learning Analysis of Scenario

### G.1 Class Incremental IID

We demonstrate the accuracy of the model on the *classes encountered so far* during the FL process as the classes increment. Specifically, in our setup, the accuracy for session 1 refers to the accuracy of the global target model in the communication round T = 20 on a test set that contains only classes  $\{0, 1\}$ . The accuracy for session 2 refers to the accuracy of the global target model in the communication round T = 40 on a test set that contains only classes  $\{0, 1, 2, 3\}$ , and so on. Session 5 represents the accuracy of the final global target model on the complete test set, which is the final accuracy shown in Table 1. This method is a popular way to illustrate Class Incremental CL scenarios, providing a fine-grained view of the model's learning performance at any given time. It is evident that the model cannot accurately classify classes it has never encountered, so showing the accuracy on unseen classes is not meaningful.

As shown in Tables 3 and 4, the accuracy varies across three datasets in the Class Incremental scenarios. For session 1, most of the frameworks are essentially vanilla FedAvg, and accuracy differences arise solely from the uncertainties in the training process. In session 2, the CL strategies of all frameworks start to take effect to avoid catastrophic forgetting. FedAvg, FedProx, FedAvg+LwF, and FedAvg+EWC fail to remember the previous data distribution, as they can only correctly classify the current classes  $\{2, 3\}$ , resulting in a subsequent accuracy drop of about 50%. By session 3, these frameworks can remember only the current 2 classes out of a total of 6 classes, leading to an accuracy of approximately  $\frac{2}{6}$ . In contrast, frameworks specifically designed for CFL problems perform significantly better, demonstrating their ability to overcome catastrophic forgetting to varying degrees. Comparatively, the accuracy declines faster for CIFAR-10, indicating that its images are rich in information and complex patterns, making them more prone to forgetting. Therefore, for the CIFAR-10 dataset, more samples per client and additional training epochs are necessary for the model to learn the representations of different classes better.

Note that we focus on the Class Incremental IID scenario because it only has a subset of classes at any given time or session, whereas Class Incremental Non-IID includes all classes. Therefore, the Class Incremental Non-IID scenario does not require fine-grained analysis of the accuracy on currently encountered classes and can be directly tested on the complete test set.

Method		MNIST - Session				Fashion-MNIST - Session				
	1	2	3	4	5	1	2	3	4	5
FedAvg [1]	100.00	49.07	31.07	24.90	19.77	99.40	49.00	33.33	25.00	19.96
FedProx [30]	100.00	49.03	31.07	24.77	19.78	99.45	49.10	33.33	25.00	19.96
FedAvg+LwF [31]	99.95	49.07	31.07	24.88	19.77	99.40	49.05	33.33	25.00	19.96
FedAvg+EWC [32]	99.95	49.07	31.32	26.46	19.78	99.55	49.27	33.35	25.00	19.95
FedAvg+ACGAN [33]	100.00	87.76	61.37	52.90	42.15	99.25	82.28	60.87	37.11	40.83
FedCIL [34]	99.84	81.78	73.38	66.99	46.36	98.57	66.51	54.59	35.95	38.17
FOT [14]	100.00	47.53	38.27	32.76	32.18	99.45	48.58	48.35	34.61	27.47
MFCL [11]	100.00	49.07	33.43	26.86	20.55	99.45	48.95	33.33	25.00	19.96
TARGET [12]	100.00	50.88	35.07	26.38	21.15	99.40	49.65	33.10	24.77	19.81
DCFL (Ours)	99.95	99.62	98.86	97.52	94.69	99.55	90.40	80.98	68.28	72.50

Table 3: Analysis of Scenario - Class Incremental IID. Tested on encountered classes.

Table 4: Analysis of Scenario - Class Incremental IID. Tested on encountered classes.

Method	CIFAR-10 - Session						
	1	2	3	4	5		
FedAvg [1]	94.55	41.70	29.32	24.01	18.74		
FedProx [30]	95.45	41.40	29.37	24.02	18.60		
FedAvg+LwF [31]	94.45	41.75	29.70	24.07	18.67		
FedAvg+EWC [32]	94.40	41.17	29.48	24.16	18.64		
FedAvg+ACGAN [33]	93.90	71.17	43.58	30.34	24.52		
FedCIL [34]	80.66	48.58	35.88	29.36	21.59		
FOT [14]	94.60	34.42	23.07	18.06	13.47		
MFCL [11]	94.80	41.05	28.97	23.95	18.62		
TARGET [12]	94.80	47.10	31.32	23.43	18.74		
DCFL (Ours)	94.55	76.70	54.68	40.75	38.13		

#### G.2 Domain Incremental

We show the accuracy of the model on the *domains encountered so far* during the FL process as the domains increment. Unlike in the Class Incremental IID scenario, where the model needs a thorough understanding of different classes for accurate classification, in the Domain Incremental scenario, the model can potentially classify categories even from unseen domains. For example, a model trained on the Cartoon domain might correctly classify images from the Photo domain. Figure 5 shows the changes in the model accuracy in the complete test dataset, where it can be observed that the model accuracy initially increases and then decreases. This is partly due to the nature of the dataset, where some domains (e.g., Art Painting) are not closely related to others. Additionally, the order of domains can lead to inconsistent accuracy variations. In this paper, we follow the setting from other literature, specifically Sketch  $\rightarrow$  Cartoon  $\rightarrow$  Art Painting  $\rightarrow$  Photo (increasing the level of realism over time) [29]. If we consider decreasing the level of realism over time, the accuracy variations would present a different case.

For the above reasons, we should also consider presenting and analyzing the model's performance in the Domain Incremental scenario by testing only on the currently encountered domain, as shown in Table 5. It is evident that for our DCFL framework, there is a significant drop in accuracy from session 2 to session 3. This indicates a substantial difference between the synthetic data generated for session 1 (Sketch) and session 2 (Cartoon) compared to session 3 (Art Painting), leading to confusion in the target model on these data. In contrast, the accuracy drop for baselines occurs sharply from session 1 to session 2, and then slows down in subsequent domain changes. This further confirms the significant differences between domains. However, our DCFL framework can mitigate the impact of these domain differences on the performance of the target model.



Figure 5: Main Result - Domain Incremental Scenario. Tested on the complete test set.

Method	PACS - Domain						
	1	2	3	4			
FedAvg [1]	63.36	45.82	33.21	29.96			
FedProx [30]	66.79	44.38	35.50	33.82			
FedAvg+LwF [31]	64.76	46.61	36.58	34.57			
FedAvg+EWC [32]	66.16	44.70	42.52	38.02			
FedAvg+ACGAN [33]	64.89	51.08	42.82	40.57			
FedCIL [34]	50.51	25.95	15.90	13.49			
FOT [14]	65.52	43.82	40.12	39.42			
MFCL [11]	66.79	53.34	41.25	36.16			
TARGET [12]	65.02	44.83	35.91	30.72			
DCFL (Ours)	69.59	64.70	52.55	48.02			

#### Table 5: Analysis of Scenario - Domain Incremental. Tested on encountered classes.

# H Sensitivity Study

### H.1 Effect of Number of Clients (Sample Size per Client)

For simulation datasets such as MNIST, Fashion-MNIST, and CIFAR-10, where increasing the number of clients results in a reduction in the data available to each client. In our setup, with a total of 5 sessions, when the number of clients is set to 20, the sample size per client aligns with that set in FedAvg. However, variations in client numbers change the sample size per client, affecting not only the training of the target model but also the training of the diffusion model. Consequently, we analyze the effect of different numbers of clients on accuracy, as illustrated in Figure 6. For synthetic datasets consistent with the main text, we ensure that the sample sizes of the synthetic datasets match those of the real datasets.

The results demonstrate that the DCFL framework performs better when there are fewer clients and each client has a larger sample size, which means the data are more concentrated. This improvement is rationalized by the fact that the diffusion model has access to more training data, resulting in more realistic and less noisy synthetic data. Moreover, since our setup ensures that the synthetic datasets have the same number of samples as the real datasets, fewer clients allow the diffusion model to generate more synthetic data, which in turn aids the training of the target model.



Figure 6: **Sensitivity Study - Number of Clients.** Using the MNIST dataset in the Class-Incremental IID scenario as an example: 5 clients - 2400 samples per session per client, 10 clients - 1200 samples per session per client, 20 clients - 600 samples per session per client, 30 clients - 400 samples per session per client, 50 clients - 240 samples per session per client. The setup with 20 clients corresponds to the configuration in the main text.

#### H.2 Effect of Number of Synthetic Samples

The number of synthetic samples generated needs to be balanced in real-world scenarios to maximize performance while minimizing computational overhead and memory usage. In the main text, we always set the number of synthetic samples equal to the number of real samples (i.e.,  $\delta = 1$ ) to ensure a balance between previous and current knowledge during target model training. This varies across different datasets; for example, in the MNIST dataset, the number of synthetic samples is 600, while in the CIFAR-10 dataset, it is 500. Although the diffusion model can generate an unlimited number of synthetic samples, it is evident that when there are too many synthetic samples, the target model may struggle to learn effective current knowledge. We analyze the effect of different number of synthetic samples on accuracy, as illustrated in Figure 7.

The results show that performance is optimal when the number of synthetic samples equals the number of real samples, i.e., with the ratio  $\delta = 1$ . Fewer synthetic samples cannot provide sufficient training quality for the target model, while an excess of synthetic samples causes the model to overly focus on previous knowledge at the expense of current knowledge. Additionally, the diffusion model's training dataset comprises the real data from the current session and the synthetic data from previous sessions. Therefore, if there are too many synthetic samples from previous sessions, they will continue to influence the diffusion model's training in the current session, preventing the diffusion model from adequately learning the real data of the current session.



Figure 7: Sensitivity Study - Number of Synthetic Samples. Using the MNIST dataset in the Class-Incremental IID scenario as an example. We maintain a setup of 20 clients and 600 real samples per session per client while varying the number of synthetic samples generated. The setup with  $\delta = 1$  corresponds to the configuration in the main text.

# **I** Broader Impacts

DCFL is a CFL framework designed to address the issue of catastrophic forgetting in dynamic FL scenarios. It has a wide range of societal impacts, promoting applications and deployments across multiple fields, but it also carries potential risks. Besides the typical privacy and security concerns faced in FL, including a generative model also inherits potential issues associated with generative models. Below, we outline the positive impacts of DCFL, its potential risks, and mitigation strategies.

### **Potential Positive Societal Impacts.**

- Clients Operate in Dynamic Environment. DCFL effectively supports multiple clients in everchanging environments. The targets may change continuously, such as personnel, vehicles, buildings, etc., or the environment itself may vary, such as day and night, seasonal changes, different weather conditions, etc. For example, in an intelligent transportation scenario, a static roadside unit captures photos of vehicles in various weather conditions and needs to remember the features of previous weather conditions while continuing to learn in the current weather.
- Clients Move Through Different Environments. DCFL effectively supports multiple dynamic clients performing tasks in different environments because it prevents clients from forgetting knowledge from previous environments. This is crucial as the targets in these different environments are likely to be similar. For example, an unmanned aerial vehicle (UAV) patrolling different environments, from towns to highways to forests, may encounter similar manifestations of potential hazards like fires.

### Potential Negative Societal Impacts.

- Malicious Attacks. Although clients in DCFL do not send the diffusion model to anyone, including the server or other clients, which minimizes the risk of privacy leakage, there is still a potential risk of replaying sensitive private data. If an attacker gains access to the diffusion model on a client, they could potentially replay all historical data. In contrast, if a client in vanilla FedAvg is compromised, only the current data are at risk. Given the broader temporal span and richness of the data that could be exposed, the former scenario is clearly more severe. Therefore, researchers should consider implementing privacy protection techniques for generative models when deploying DCFL to prevent attackers from extracting sensitive data from synthetic samples.
- **Misuse.** The diffusion model in DCFL also carries the risk of misuse. For example, some users might use sensitive data to train the diffusion model to circumvent regulatory scrutiny of stored data. Therefore, regulatory bodies need to implement stringent oversight and verification processes to ensure that generative models are not being used to bypass data compliance regulations.