FEDGP: BUFFER-BASED GRADIENT PROJECTION FOR CONTINUAL FEDERATED LEARNING

Shenghong Dai [1] **Yicong Chen** [1] **Jy-yong Sohn** [2] **S M Iftekharul Alam** [3] **Ravikumar Balakrishnan** [3]
Suman Banerjee [4] **Nageen Himayat** [3] **Kangwook Lee** [1]

ABSTRACT

Continual Federated Learning (CFL) has garnered significant attention in recent years due to its potential in real-world scenarios where multiple clients (e.g., mobile phones, autonomous vehicles) continuously observe data in a dynamic environment. However, CFL suffers from catastrophic forgetting, where the model forgets previously learned knowledge in favor of new data. To address this issue, we propose a novel method called buffer-based Gradient Projection (FedGP) that mitigates catastrophic forgetting by replaying local buffer samples and using aggregated buffer gradients to preserve the previously learned knowledge across clients. Our method can be combined with a variety of existing continual learning methods, and boost their performance in the CFL setup. Our approach is evaluated on both standard benchmark datasets and a realistic streaming decentralized automotive dataset generated using CARLA and OpenFL.

1 INTRODUCTION

Federated Learning (FL) is a machine learning technique that facilitates collaborative model training among multiple users while preserving data decentralization for enhanced privacy and efficient communication. However, the current FL benchmarks (Caldas et al., 2018; He et al., 2020) assume a static data distribution across clients throughout the training process, which fails to accurately capture the dynamic nature of real-world data. This discrepancy calls for continual federated learning (CFL), the integration of FL with continual learning (CL) (Shmelkov et al., 2017; Chaudhry et al., 2018a; Thrun, 1995; Aljundi et al., 2017; Chen & Liu, 2018; Aljundi et al., 2018). CL is a vital consideration that enables models to continuously learn from incoming data, improving their accuracy and adaptability over time.

The biggest challenge in CL (and thus in CFL) is *catastrophic forgetting*, where the model gradually shifts its focus towards new data and unintentionally discards previously acquired knowledge. Initial attempts to mitigate catastrophic forgetting in CFL incorporated existing CL solutions at each client of FL, such as replaying previous task data or penalizing the updates of weights that are crucial for preserving the knowledge from earlier tasks. However, recent works (Ma et al.; Yoon et al., 2021) have observed that this naive approach cannot fully mitigate the problem due to two reasons: (i) small-scale devices participating in FL only have limited buffer size to store the data from previous tasks, (ii) the fact that data distributions are not identical across clients in FL makes the problem much more challenging. Moreover, various existing methods developed for CFL (Yoon et al., 2021; Ma et al.; Venkatesha et al., 2022) have limitations in that they require explicit task boundaries. Mitigating catastrophic forgetting in practical scenarios where fixed task boundaries are absent throughout the training process, known as *general continual learning* (Buzzega et al., 2020), remains an important open question.

This paper introduces a novel method called buffer-based Gradient Projection (FedGP), which overcomes these existing challenges of CFL. Our approach, illustrated in Fig. 1, involves two key components:

1. Client k uses its buffer M^k to avoid catastrophic forgetting on the local training examples for previous tasks. This ensures the model retains information obtained *locally*.

2. Client k computes the gradient g^k_{ref} of the global model with respect to its buffer data, and updates its local model in the next round such that its direction is not conflicting with g_{ref} which is computed as the average of the buffered gradients g^k_{ref}. This allows each client to maintain *global* information obtained from other clients.

[1]Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, USA [2]Department of Applied Statistics, Yonsei University, Seoul, Korea [3]Intel Labs, Hillsboro, OR USA [4]Department of Computer Sciences, University of Wisconsin-Madison, Madison, USA. Correspondence to: Shenghong Dai <sdai37@wisc.edu>.

Proceedings of the 6^{th} MLSys Conference, Miami Beach, FL, USA, 2023.

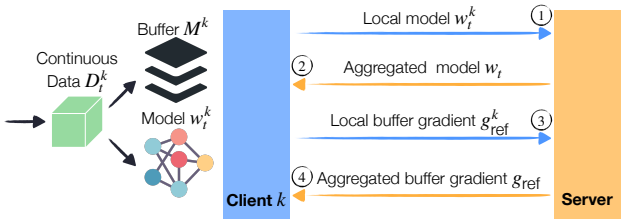


Figure 1. An overview of FedGP method proposed for continual federated learning. At each round t, client k receives data D_t^k and trains a local model w_t^k. To address catastrophic forgetting, a portion of the incoming data is stored in a buffer $\mathcal{M}^k$. Given the aggregated model w_t provided by the central server, each client computes the gradient with respect to w_t using its buffer data $\mathcal{M}^k$. The server averages the local buffer gradients from all clients to obtain a global buffer gradient g_{ref}. This global buffer gradient guides the local model update for each client k in the subsequent round.

We conducted experiments to test the efficacy of our method in improving the performance of existing methods developed for CL and CFL. We tested on various benchmarks datasets (rotated-MNIST, permuted-MNIST, sequential-CIFAR10, sequential-CIFAR100) and a realistic streaming automotive dataset (CARLA (Dosovitskiy et al., 2017)) using OpenFL framework (Dai et al., 2023; Reina et al., 2021). We observed that our method significantly increases the accuracy averaged over previous tasks and decreases the amount of forgetting when the data for distinct tasks are consecutively loaded. We consistently observed such improvement for (i) different incremental learning (IL) setups including domain-IL, class-IL, and task-IL, and for (ii) various system settings when we change the local buffer size, communication frequency, number of users, or whether the clients' model update is synchronous or asynchronous.

2 Related Work

Prior work related to our paper falls into three categories: Continual Learning (CL), Federated Learning (FL), and Continual Federated Learning (CFL). Further details are in Appendix E.

CL addresses the problem of learning multiple tasks consecutively using a single model. Catastrophic forgetting (McCloskey & Cohen, 1989; Ratcliff, 1990; French, 1999), where a classifier trained for a current task performs poorly on previous tasks, is a major challenge. Existing approaches can be categorized into regularization-based (Kirkpatrick et al., 2017; Zenke et al., 2017; Chaudhry et al., 2018a; Li & Hoiem, 2017), architecture-based (Rusu et al., 2016; Yoon et al., 2017; Mallya & Lazebnik, 2018; Serra et al., 2018; Fernando et al., 2017), and replay-based methods (Ratcliff, 1990; Robins, 1995; Rebuffi et al., 2017; Shin et al., 2017; Aljundi et al., 2019; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018b). FedGP is a replay-based method that alleviates forgetting by reusing a portion of data from previous tasks.

FL enables collaborative training of a model with improved data privacy (Kairouz et al., 2021; Lim et al., 2020; Zhao et al., 2018; Konečnỳ et al., 2016). FedAvg (McMahan et al., 2017) is a widely used FL algorithm, but most existing methods (Li et al., 2020; Shoham et al., 2019; Karimireddy et al., 2020; Li et al., 2019; Mohri et al., 2019) assume static data distribution over time, ignoring temporal dynamics.

CFL tackles the problem of learning multiple consecutive tasks in the FL setup. FedProx (Li et al., 2020) and FedCurv (Shoham et al., 2019) aim to preserve previously learned tasks, while FedWeIT (Yoon et al., 2021) and NetTailor (Venkatesha et al., 2022) prevent interference between irrelevant tasks. Other methods including CFeD (Ma et al.) and FedCL (Yao & Sun, 2020), and GLFC (Dong et al., 2022) use surrogate datasets, importance weights, or class-aware techniques to distill the knowledge obtained from previous tasks. However, existing CFL methods suffer from several limitations, *e.g.*, not scalable as the number of tasks increases (Yoon et al., 2021; Venkatesha et al., 2022), requiring a surrogate dataset (Ma et al.) or additional communication overhead (Yao & Sun, 2020), and not applicable to general continual setting that does not have fixed task boundaries.

3 Preliminaries

We focus on finding a single classifier f (having model parameter w) that performs well on T consecutive tasks. We assume that at time slot $t \in [T]$, the classifier is only allowed to be trained for task t, where we define $[N] \coloneqq \{1, \cdots, N\}$ for a positive integer N. We assume the feature-label pair (x_t, y_t) for task t is drawn from an unknown distribution D_t. The optimization problem for CL at time $\tau \in [T]$ is written as

$$\arg\min_{w} \sum_{t=1}^{\tau} \mathbb{E}_{(x_t,y_t)\sim D_t} \left[\ell\left(y_t, f\left(x_t, w\right)\right)\right], \qquad (1)$$

where ℓ is the loss function, and $f(x_t, w)$ is the output of classifier f with parameter w, for input x_t. We consider a practical scenario where we do not have enough storage to save all the data seen for the previous task ($t < \tau$); instead, we employ a replay buffer $\mathcal{M}$ that selectively stores a subset of data. We use the buffer data as a proxy to summarize past samples and refine the model updates. We constrain the model updates in a way that the loss for the data in buffer $\mathcal{M}$ is not increased. Given the model $w_{\tau-1}$ trained on previous tasks, the constrained optimization problem at time $\tau \in [T]$

is represented as:

$$\begin{aligned} &\arg\min_{w} \mathbb{E}_{(x_\tau,y_\tau)\sim D_\tau}\left[\ell\left(y_\tau, f\left(x_\tau, w\right)\right)\right] \\ &\quad \text{s.t.} \quad \ell\left(y_b, f\left(x_b, w\right)\right) \leq \ell\left(y_b, f\left(x_b, w_{\tau-1}\right)\right) \end{aligned} \tag{2}$$

where (x_b, y_b) is sampled from replay buffer $\mathcal{M}$. The optimization problem in (2) can be rephrased for various CL methods as below. First, some methods including DER (Buzzega et al., 2020) use regularization technique to find the model parameter w that minimizes the loss with respect to the local replay buffer $\mathcal{M}$ as well as current samples. For a given regularization coefficient γ, the optimization problem for CL with replay buffers at time $\tau \in [T]$ is:

$$\begin{aligned} &\arg\min_{w} \mathbb{E}_{(x_\tau,y_\tau)\sim D_\tau}\left[\ell\left(y_\tau, f(x_\tau, w)\right)\right] \\ &\qquad + \gamma \sum_{(x_b,y_b)\in\mathcal{M}} \ell\left(y_b, f(x_b, w)\right). \end{aligned} \tag{3}$$

Second, some other methods, including A-GEM (Chaudhry et al., 2018b), interpret the constraints of Eq. 2 in terms of the gradients with respect to the current/buffer data. To be specific, the constraint promotes the alignment of the gradient with respect to the current data (x_τ, y_τ) and that for the buffer data $(x_b, y_b) \in \mathcal{M}$. This optimization problem at time $\tau \in [T]$ is:

$$\begin{aligned} &\arg\min_{w} \mathbb{E}_{(x_\tau,y_\tau)\sim D_\tau}\left[\ell\left(y_\tau, f\left(x_\tau, w\right)\right)\right] \\ &\text{s.t.} \quad \left\langle \frac{\partial \ell\left(y_\tau, f\left(x_\tau, w\right)\right)}{\partial w}, \frac{\partial \ell\left(y_b, f\left(x_b, w\right)\right)}{\partial w} \right\rangle \geq 0 \end{aligned}$$

For the continual *federated* learning (CFL) setup where the data is owned by K clients, we use the superscript $k \in [K]$ to denote each client, *i.e.*, client k samples the data from D_t^k at time t and employs a local replay buffer $\mathcal{M}^k$. In the case of using FedAvg (McMahan et al., 2017), each round of the CFL is operated as follows. First, each client $k \in [K]$ performs multiple rounds of local training on D_t^k with the assistance of replay buffer $\mathcal{M}^k$. Second, once the local training is completed, each client sends the model updates to the central server. Finally, the central server aggregates the model updates and transmits them back to clients.

4 FedGP

We propose our method FedGP, which boosts up the performance of existing CL methods, under the FL setup. Our approach, FedGP, draws inspiration from the A-GEM algorithm (Chaudhry et al., 2018b), which projects the gradient with respect to its own historical data. Building upon this idea, we utilize the global buffer gradient, which is the average buffer gradient across all clients, as a reference to project the current gradient. This allows us to take advantage of the collective experience of multiple clients and mitigate the risk of forgetting previously learned knowledge in FL scenarios.

Algorithm 1 FedGP

Initialize random w_0, and set $\mathcal{M}^k = \{\}$, $g_{\text{ref}} = \text{None}$
for each task $t = 1$ **to** T **do**
  $w_t^k \leftarrow \texttt{ClientUpdate}(t, w_{t-1}, g_{\text{ref}})$, $\forall$ client k
  $w_t \leftarrow \frac{1}{K}\sum_{k=1}^{K} w_t^k$
  $g_{\text{ref}}^k \leftarrow \texttt{ComputeBufferGrad}(w_t, \mathcal{M}^k)$, $\forall$ client k
  $g_{\text{ref}} \leftarrow \frac{1}{K}\sum_{k=1}^{K} g_{\text{ref}}^k$
end for
Return w_T, the final global model

Algorithm 2 $\texttt{ClientUpdate}(t, w, g_{\text{ref}})$ at client k

Input: Task index t, model w, buffer gradient g_{ref}
Load the dataset $\mathcal{D}_t^k$, local buffer $\mathcal{M}^k$
$n \leftarrow 0$
for $(x, y) \in \mathcal{D}_t^k$ **do**
  $g_c = \nabla_w \left[\ell(y, f(x, w))\right]$
  $(x_b, y_b) \leftarrow$ a sample from $\mathcal{M}^k$
  $g \leftarrow \texttt{RefineGradient}(g_c, (x_b, y_b))$
  $\tilde{g} \leftarrow g - \text{proj}_{g_{\text{ref}}} g \cdot \mathbf{1}(g_{\text{ref}}^\top g \leq 0)$
  $w \leftarrow w - \alpha\tilde{g}$ for some learning rate α
  $\mathcal{M}^k \leftarrow \texttt{ReservoirSampling}(\mathcal{M}^k, (x, y), n)$
  $n \leftarrow n + 1$
end for
Return w to server

Note that FedGP can be combined with any CL methods (e.g., DER (Buzzega et al., 2020) and A-GEM (Chaudhry et al., 2018b)) where a memory buffer stores a subset of data sampled for old tasks, which is used to let the model maintain good performance on both new task and old tasks. The local buffer at client k is denoted by $\mathcal{M}^k$. As the continuous data is loaded to the client, it keeps updating the buffer so that $\mathcal{M}^k$ becomes a good representative of old tasks. As illustrated in Fig. 1, our method contains the process of sharing information (the model and the buffer gradient) between the server and each client $k \in [K]$. Algorithm 1 provides the overview of our method. For each new task $t \in [T]$, the server first aggregates the local models w_t^k from client $k \in [K]$, getting a global model w_t. Afterwards, the server aggregates the local buffer gradient g_{ref}^k (the gradient computed on the global model w_t with respect to the local buffer $\mathcal{M}^k$) from client $k \in [K]$ to obtain a global buffer gradient g_{ref}. Note that here we have two functions used at the client side, `ClientUpdate` and `ComputeBufferGrad`, which are given in Algorithm 2 and 3, respectively.

`ClientUpdate` shows how client k updates its local model for task t. The client first loads the global model w and the global buffer gradient g_{ref} which are received from the server in the current round. It also loads the lo-

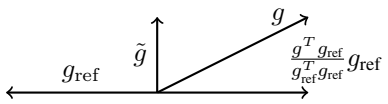


Figure 2. Illustration of the gradient projection in Eq. 4. If the angle between the gradient update g and global buffer gradient (considered as a reference) g_{ref} is larger than 90°, we project g on g_{ref} to minimize the interference and merely update along the directions of $\tilde{g}$ that is orthogonal to g_{ref}.

cal buffer $\mathcal{M}^k$ storing a subset of samples for previous tasks, and the data $\mathcal{D}_t^k$ for the current task. For each new data $(x, y) \in \mathcal{D}_t^k$, the client computes the gradient $g_c = \nabla_w l(y, f(x, w))$ for the model w. To mitigate the catastrophic forgetting issue in local training, various existing CL methods are designed, and we inherit these ideas while combining with FedGP method. Existing ideas refine the gradient g_c using data (x_b, y_b) (for previous tasks) sampled from the local buffer $\mathcal{M}^k$. We refer to such refinement process as `RefineGradient`. Depending on which CL method we are combining with FedGP, the function `RefineGradient` is defined differently. In the Appendix D, we added the details of `RefineGradient` when FedGP is combined with DER and A-GEM, respectively.

After getting the refined gradient g, the client compares the direction of g with the direction of the global buffer gradient g_{ref} received from the server. When the angle between g and g_{ref} is greater than 90°, it implies that following the direction g will improve the performance on the current task, at the cost of performing worse on previous tasks. To retain the knowledge on the previous tasks, we do the following: whenever g and g_{ref} are having a negative inner product, we project the gradient g onto the global buffer gradient (which can be considered as a reference) g_{ref} and remove this component from g, *i.e.*, define

$$\tilde{g} = g - \frac{g^T g_{\text{ref}}}{g_{\text{ref}}^T g_{\text{ref}}} g_{\text{ref}}, \tag{4}$$

following the idea suggested in (Chaudhry et al., 2018b). As illustrated in Fig. 2, this projection helps prevent the model updates along the direction that is harming the performance on previous tasks.

The client updates its local model w by applying the gradient descent step with the updated gradient $\tilde{g}$. Finally, the client updates the contents of the buffer $\mathcal{M}^k$ by using the reservoir sampling (Vitter, 1985) written in Algorithm 4 in the Appendix. Note that the reservoir sampling selects a random sample of $|\mathcal{M}^k|$ elements from a local input stream, while ensuring that each element has an equal probability of being included in the sample. One of the advantages of this method is that it does not require any prior knowledge of the size of the data stream.

Once the updated local models $\{w_t^k\}_{k=1}^K$ are transmitted to the server, the global model w_t is updated on the server side, and transmitted to each client. Then, each client k computes the local buffer gradient (*i.e.*, the gradient of the model w_t with respect to the samples in the local buffer $\mathcal{M}^k$) as shown in Algorithm 3 in the Appendix.

After each client computes the local buffer gradient g_{ref}^k, the server aggregates them to update the global buffer gradient g_{ref}. Repeating the above process for T tasks, the server finds the final global model w_T as shown in Algorithm 1. Note that here we described the algorithm for the case when we update the model w_t and the buffer gradient g_{ref} once per task. This can be directly extended to a general case when we repeat such updates for R communication rounds per task.

In summary, FedGP refines the local gradient in a way that it has a positive correlation with the average gradient computed from the buffer across all clients. To accomplish this, each local client calculates the gradient of the global model on its local buffer. Afterwards, the server computes the average of the clients' buffer gradients and transmits it back to the clients as a reference vector g_{ref}, which provides the information on the previous tasks across clients.

5 Experiments

In this section, we assess the efficacy of our method FedGP, and compare it with baseline CL/CFL methods with non-IID data across clients. To evaluate these methods, we conduct experiments on image classification tasks for benchmark datasets including rotated-MNIST (Lopez-Paz & Ranzato, 2017), permuted-MNIST (Goodfellow et al., 2013), sequential-CIFAR10, and sequential-CIFAR100 (Lopez-Paz & Ranzato, 2017) datasets, as well as an object detection task on streaming CARLA dataset. (Dai et al., 2023; Dosovitskiy et al., 2017). We also explore FedGP on NLP tasks in Appendix A. All experiments are averaged across five runs, each with a different seed. For further details and additional results, please refer to Appendix B.

5.1 Classification

5.1.1 Settings

Evaluation Datasets. We evaluate our approach on three CL scenarios: domain incremental learning (domain-IL), class incremental learning (class-IL), and task incremental learning (task-IL). For domain-IL, the data distribution of each class changes across different tasks. For example, we use the rotated-MNIST and permuted-MNIST datasets for domain-IL, where each task rotates the training digits by a random angle or applies a random permutation. We create $T = 10$ tasks for our experiments.

For class-IL and task-IL, we partition the set of classes into disjoint subsets and assign each subset to a particular

Table 1. Average classification accuracy Acc_T (%) on standard benchmark datasets at the final task T. The results, averaged over 5 random seeds, demonstrate the benefits of our proposed method in combination with other baselines. A buffer size of 200 is utilized whenever methods require it.

	rotated-MNIST (*Domain-IL*)		**sequential-CIFAR10** (*Class-IL*)		**sequential-CIFAR10** (*Task-IL*)	
	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP
FedAvg (McMahan et al., 2017)	$68.02^{\pm 3.1}$	$79.46^{\pm 4.1}$ (↑**11.44**)	$17.44^{\pm 1.3}$	$18.02^{\pm 0.6}$ (↑**0.58**)	$70.58^{\pm 4.0}$	$80.83^{\pm 2.0}$ (↑**10.25**)
FedCurv (Shoham et al., 2019)	$68.21^{\pm 2.6}$	$80.53^{\pm 4.3}$ (↑**12.32**)	$17.36^{\pm 0.7}$	$17.86^{\pm 0.5}$ (↑**0.50**)	$67.77^{\pm 1.4}$	$81.28^{\pm 1.1}$ (↑**13.51**)
FedProx (Li et al., 2020)	$67.79^{\pm 3.2}$	$78.74^{\pm 4.1}$ (↑**10.95**)	$16.67^{\pm 2.7}$	$17.97^{\pm 0.8}$ (↑**1.30**)	$69.57^{\pm 6.5}$	$81.23^{\pm 1.3}$ (↑**11.66**)
A-GEM (Chaudhry et al., 2018b)	$68.34^{\pm 5.6}$	$74.74^{\pm 2.3}$ (↑**6.40**)	$17.82^{\pm 0.9}$	$19.44^{\pm 0.9}$ (↑**1.62**)	$77.14^{\pm 3.1}$	$83.16^{\pm 1.6}$ (↑**6.02**)
DER (Buzzega et al., 2020)	$57.73^{\pm 3.6}$	$87.13^{\pm 1.1}$(↑**29.40**)	$18.44^{\pm 3.7}$	$30.94^{\pm 3.8}$ (↑**12.50**)	$69.34^{\pm 3.2}$	$77.99^{\pm 0.8}$ (↑**8.65**)
Ideal	$90.00^{\pm 2.7}$		$72.84^{\pm 1.4}$		$92.70^{\pm 0.5}$	
	permuted-MNIST (*Domain-IL*)		**sequential-CIFAR100** (*Class-IL*)		**sequential-CIFAR100** (*Task-IL*)	
FedAvg	$25.92^{\pm 2.1}$	$34.23^{\pm 2.7}$ (↑**8.31**)	$8.76^{\pm 0.1}$	$17.08^{\pm 1.8}$ (↑**8.32**)	$47.74^{\pm 1.2}$	$74.71^{\pm 0.9}$ (↑**26.97**)
FedCurv	$26.00^{\pm 2.4}$	$35.21^{\pm 5.1}$ (↑**9.21**)	$8.92^{\pm 0.1}$	$16.67^{\pm 0.9}$ (↑**7.76**)	$49.14^{\pm 1.6}$	$74.64^{\pm 0.7}$ (↑**25.49**)
FedProx	$25.92^{\pm 2.5}$	$35.60^{\pm 4.7}$ (↑**9.68**)	$8.75^{\pm 0.2}$	$16.92^{\pm 1.4}$ (↑**8.17**)	$47.05^{\pm 3.2}$	$73.95^{\pm 0.8}$ (↑**26.89**)
A-GEM	$33.43^{\pm 1.4}$	$39.09^{\pm 3.5}$ (↑**5.66**)	$8.90^{\pm 0.1}$	$19.53^{\pm 1.3}$ (↑**10.63**)	$63.84^{\pm 0.8}$	$74.84^{\pm 0.5}$ (↑**11.00**)
DER	$19.79^{\pm 1.7}$	$43.43^{\pm 0.9}$ (↑**23.64**)	$13.32^{\pm 1.6}$	$22.96^{\pm 3.6}$ (↑**9.64**)	$57.71^{\pm 1.2}$	$65.57^{\pm 1.9}$ (↑**7.86**)
Ideal	$61.16^{\pm 4.8}$		$67.09^{\pm 0.6}$		$90.06^{\pm 0.3}$	

task. For instance, in our image classification experiments for class-IL and task-IL, we divide the CIFAR-100 dataset (with $C = 100$ classes) into $T = 10$ subsets, each of which contains the samples for $C/T = 10$ classes. Each task $t \in [T]$ is defined as the classification of images from each subset $t \in [T]$. The difference between class-IL and task-IL is that in the task-IL setup, we assume the task identity t is given at inference time. That is, the model f predicts among the $C/T = 10$ classes corresponding to task t. The class-IL and task-IL settings for CIFAR-10 are defined by splitting the CIFAR-10 dataset into $T = 5$ tasks, with each task having two unique classes.

In the FL setup, we assume that the data distribution is non-IID across the different clients. Once we define the data for each task, we split it among K clients in a non-IID manner. For the rotated-MNIST or permuted-MNIST dataset, each client receives samples for two MNIST digits. To create a sequential-CIFAR10 or sequential-CIFAR100 dataset, we use the Latent Dirichlet Allocation (LDA) partition algorithm. This algorithm partitions the dataset among multiple clients by assigning samples of each class to different clients based on the probability distribution $p \sim \mathrm{Dir}(\alpha)$ (Hsu et al., 2019).

Architecture and Hyperparameters. For the rotated-MNIST and permuted-MNIST dataset, we use a simple CNN architecture used in (McMahan et al., 2017), and split the dataset into $K = 10$ clients. Each client performs local training for $E = 1$ epoch, and we set the number of communication rounds as $R = 20$ for each task. For the sequential-CIFAR10 and sequential-CIFAR100 dataset, we use a ResNet18 architecture, and divide the dataset into $K = 10$ clients. Each client trains for $E = 5$ epochs, and uses $R = 20$ rounds of communication for each task. We use stochastic gradient descent with a learning rate of 0.01 and 0.1, respectively, for the MNIST and CIFAR datasets during local training.

Baselines. We evaluate FedAvg with two replay-based methods applied to clients, A-GEM (Chaudhry et al., 2018b) and DER (Buzzega et al., 2020), and two CFL methods, FedCurv (Shoham et al., 2019) and FedProx (Li et al., 2020). A-GEM ensures the alignment between the gradient of the model for the buffer data and that for the incoming data, while DER utilizes the network output logits for distilling from past experiences. FedCurv prevents updating weights that are important for past tasks, and FedProx adds a proximal weight to constrain the local model from deviating too far from the global model. We also compare with the ideal scenario where the clients have enough memory to store all incoming data from previous tasks and train all tasks jointly; we denote this method as *Ideal*, the performance of which is a valid upper bound for all CFL methods. Finally, we compare with a naive method that trains only on the current task without considering the performance on previous tasks; this method is nothing but FedAvg, the performance of which is a valid lower bound for all CFL methods. Unless explicitly mentioned, we use a buffer size of 200 in the subsequent sections of this paper.

Performance Metrics. We assess the performance of the global model on the test dataset, which is a union of the test data for all previous tasks. The average accuracy (measured after training on task t) is denoted as $\mathrm{Acc}_t = \frac{1}{t}\sum_{i=1}^{t} a_{t,i}$, where $a_{t,i}$ is evaluated on task i af-

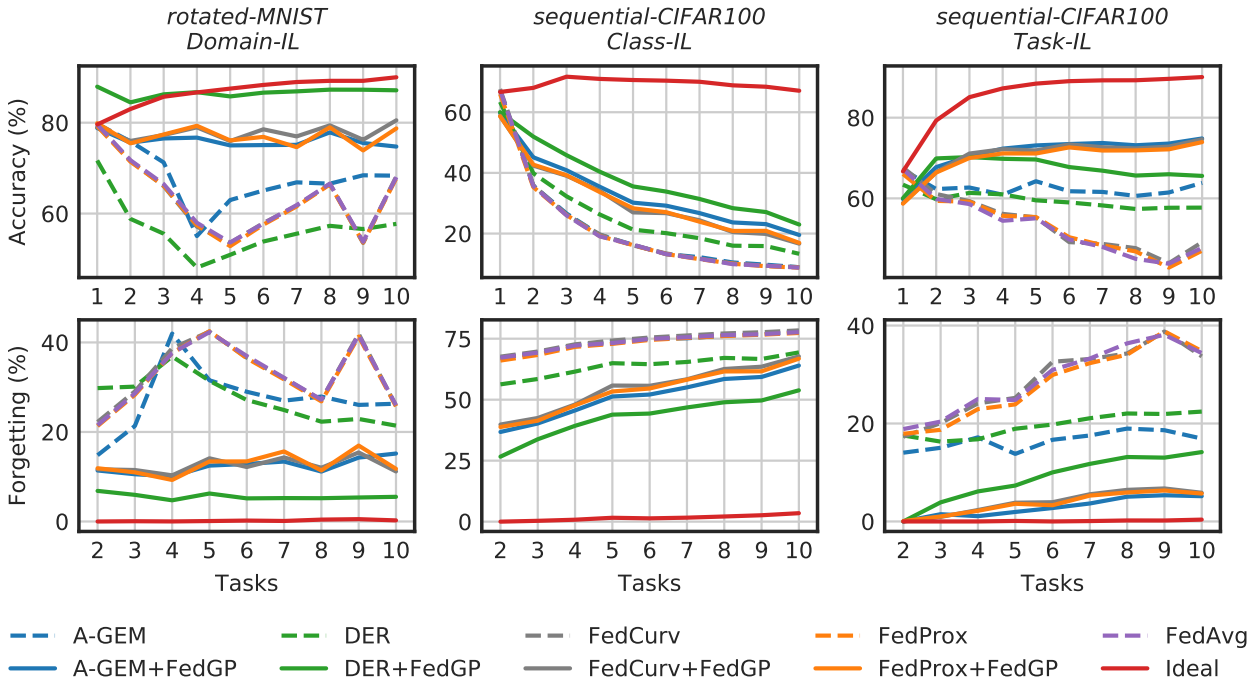


Figure 3. Evaluating accuracy (↑) and forgetting (↓) in multiple datasets with and without FedGP using a buffer size of 200. The solid lines indicate the results obtained either with our method or with the upper bound, while the dotted lines represent the results obtained without our method. The results show a significant improvement in accuracy as well as reduced forgetting for all settings.

ter training up to task t. Additionally, we measure a performance metric called *forgetting*, which is defined as the difference between the best accuracy obtained throughout the training and the current accuracy. This metric measures the model's ability to retain knowledge of previous tasks while learning new ones. The forgetting at task t is defined as: $\mathrm{Fgt}_t = \frac{1}{t-1}\sum_{i=1}^{t-1} \max_{j=1,\cdots,t-1}(a_{j,i} - a_{t,i})$.

5.1.2 Results

Accuracy & Forgetting. Table 1 presents the average accuracy Acc_T of various methods on image classification benchmark datasets measured at the final task T. For each experimental setting, we compare the performance of an existing method with/without FedGP. We observe that in all datasets, existing CL methods (A-GEM, DER) and CFL methods (FedCurv and FedProx) are not guaranteed to achieve higher accuracy than FedAvg, which is not designed for continual learning. However, when these baselines are combined with FedGP, the accuracy is consistently improved for all datasets and all baselines. For example, on the rotated-MNIST dataset in the domain-IL scenario, applying our method on DER improves accuracy by 29.4% (from 57.73% to 87.13%). Interestingly, FedGP with buffer size of 200 achieves the performance close to the ideal scenario of infinite buffer size in certain cases. In rotated-MNIST, our method combined with DER achieves 87.13% accuracy, whereas the ideal scenario with infinite memory achieves 90% accuracy. Similar results are obtained for forgetting performance Fgt_T given in Table 7 in Appendix.

Fig. 3 depicts the average accuracy Acc_t measured at task $t = 1, 2, \cdots, 10$ and the average forgetting Fgt_t measured at task $t = 2, 3, \cdots, 10$. The results for permuted-MNIST and sequential-CIFAR10 are included in Appendix B for the sake of brevity. The accuracy of FedAvg rapidly drops as different tasks are given to the model, as expected. FedCurv and FedProx perform similarly to FedAvg, while A-GEM and DER partially alleviate forgetting, resulting in higher accuracies and reduced forgetting compared to FedAvg. Combining these baselines with FedGP lead to significant performance improvements, which allows the solid lines in the accuracy plot consistently remain at the top. For example, for the experiment on task-IL for sequential-CIFAR100, the accuracy measured at task 5 (denoted by Acc_5) is 55.37% for FedProx, while 71.12% for FedProx+FedGP. These results demonstrate that FedGP effectively mitigates forgetting and enhances existing methods in CFL.

Table 2. Impact of the buffer size on Acc_T (%)

		rotated-MNIST (*Domain-IL*)		**sequential-CIFAR100** (*Class-IL*)		**sequential-CIFAR100** (*Task-IL*)	
Buffer Size	Methods	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP
200	A-GEM	$68.34^{\pm 5.6}$	$74.74^{\pm 2.3}$ (↑**6.40**)	$8.90^{\pm 0.1}$	$19.53^{\pm 1.3}$ (↑**10.63**)	$63.84^{\pm 0.8}$	$74.84^{\pm 0.5}$ (↑**11.00**)
500	A-GEM	$70.18^{\pm 8.7}$	$78.74^{\pm 3.2}$ (↑**8.56**)	$8.87^{\pm 0.1}$	$25.89^{\pm 0.9}$ (↑**17.02**)	$64.38^{\pm 1.4}$	$79.35^{\pm 0.5}$ (↑**14.97**)
5120	A-GEM	$69.97^{\pm 3.2}$	$79.17^{\pm 4.3}$ (↑**9.20**)	$8.85^{\pm 0.1}$	$33.30^{\pm 2.5}$ (↑**24.45**)	$64.99^{\pm 1.5}$	$84.52^{\pm 0.3}$ (↑**19.53**)
200	DER	$57.73^{\pm 3.6}$	$87.13^{\pm 1.1}$ (↑**29.40**)	$13.32^{\pm 1.6}$	$22.96^{\pm 3.6}$ (↑**9.64**)	$57.71^{\pm 1.2}$	$65.57^{\pm 1.9}$ (↑**7.86**)
500	DER	$60.00^{\pm 7.2}$	$88.83^{\pm 1.6}$ (↑**28.83**)	$15.44^{\pm 1.5}$	$34.87^{\pm 1.7}$ (↑**19.43**)	$60.79^{\pm 1.2}$	$73.53^{\pm 1.1}$ (↑**12.74**)
5120	DER	$58.63^{\pm 3.9}$	$89.46^{\pm 1.2}$ (↑**30.83**)	$18.89^{\pm 1.0}$	$45.76^{\pm 3.8}$ (↑**26.87**)	$62.77^{\pm 1.5}$	$83.41^{\pm 1.3}$ (↑**20.64**)

Table 3. Effect of communication on Acc_T (%). Our method outperforms baselines when utilizing equivalent communication overhead.

Methods	**R-MNIST** *Domain-IL*	**P-MNIST** *Domain-IL*	**S-CIFAR10** *Class-IL*	**S-CIFAR10** *Task-IL*	**S-CIFAR100** *Class-IL*	**S-CIFAR100** *Task-IL*
A-GEM	$68.34^{\pm 5.6}$	$33.43^{\pm 1.4}$	$17.82^{\pm 0.9}$	$77.14^{\pm 3.1}$	$8.90^{\pm 0.1}$	$63.84^{\pm 0.8}$
A-GEM w/ FedGP (2× comm overhead)	$74.74^{\pm 2.3}$	$39.09^{\pm 3.5}$	$\mathbf{19.44^{\pm 0.9}}$	$\mathbf{83.16^{\pm 1.6}}$	$19.53^{\pm 1.3}$	$74.84^{\pm 0.5}$
A-GEM w/ FedGP (equalized comm overhead)	$\mathbf{78.19^{\pm 2.7}}$	$\mathbf{40.72^{\pm 7.2}}$	$18.55^{\pm 2.5}$	$81.36^{\pm 4.4}$	$\mathbf{21.08^{\pm 1.6}}$	$\mathbf{74.95^{\pm 0.7}}$
DER	$57.73^{\pm 3.6}$	$19.79^{\pm 1.7}$	$18.44^{\pm 3.7}$	$69.34^{\pm 3.2}$	$13.32^{\pm 1.6}$	$57.71^{\pm 1.2}$
DER w/ FedGP (2× comm overhead)	$87.13^{\pm 1.1}$	$\mathbf{43.43^{\pm 0.9}}$	$\mathbf{30.94^{\pm 3.8}}$	$\mathbf{77.99^{\pm 0.8}}$	$22.96^{\pm 3.6}$	$65.57^{\pm 1.9}$
DER w/ FedGP (equalized comm overhead)	$\mathbf{88.64^{\pm 1.4}}$	$41.20^{\pm 3.0}$	$21.51^{\pm 2.8}$	$75.34^{\pm 1.9}$	$\mathbf{24.84^{\pm 1.6}}$	$\mathbf{69.00^{\pm 1.1}}$

Effect of Buffer size. Table 2 reports the performances of baseline CL methods (A-GEM and DER) with/without FedGP for different buffer sizes, ranging from 200 to 5120. For all different datasets and all IL settings, increasing the buffer size further improves the advantage of applying FedGP, by providing more data for replay and mitigating forgetting. Comparing the result of Table 2 with the result of Table 1 reporting the performance of ideal scenario with infinite memory (Ideal method), one can confirm that the performance of our method (with limited memory) nearly achieves the performance of ideal scenario (with infinite memory) in some cases. For the example of Task-IL on sequential-CIFAR100, our method combined with A-GEM has 84.52% accuracy, which is close to the 90.06% achieved by the ideal method.

Effect of communication frequency. Note that compared with baseline methods, FedGP has extra communication overhead for transmitting the buffer gradient from each client to the server. This means that the required amount of communication is doubled for FedGP. To compare baselines and FedGP under the same communication overhead constraint, we consider reducing the communication frequency of FedGP by a factor of two, while keeping the computation unchanged. Table 3 compares three methods: (i) the baseline, (ii) the baseline combined with FedGP, and (iii) the baseline combined with FedGP with the communication frequency reduced by half. Note that (i) and (iii) has the same amount of communication overhead, while (ii) is having 2x communication overhead compared with (i). We report the results for two baseline methods: A-GEM and DER. One can confirm that FedGP (with equalized communication overhead) improves the accuracy of each baseline without additional communication resources. For example, applying FedGP on the domain-IL setting for rotated-MNIST dataset improves the accuracy over 30% (from 57.73% to 88.64%), without additional overhead.

Table 4. Acc_T (%) for asynchronous task boundaries on the sequential-CIFAR100 dataset.

Methods	*Class-IL*	*Task-IL*
FedAvg	$16.22^{\pm 1.2}$	$59.04^{\pm 1.7}$
A-GEM	$16.92^{\pm 1.0}$	$69.41^{\pm 1.3}$
A-GEM+FedGP	$30.74^{\pm 1.5}$	$\mathbf{77.70^{\pm 0.4}}$
DER	$31.95^{\pm 2.6}$	$68.28^{\pm 1.5}$
DER+FedGP	$\mathbf{36.29^{\pm 1.0}}$	$72.02^{\pm 0.7}$
Ideal	$69.01^{\pm 0.5}$	$90.26^{\pm 0.2}$

Asynchronous task boundaries. In our previous experiments, we assumed synchronous task boundaries where clients finish tasks at the same time. However, in many real-world scenarios, different clients finish each task asynchronously. Motivated by this practical setting, we ran experiments on asynchronous task boundary setting where a client may finish a task faster than others and move on to the next task earlier. Table 4 shows the accuracy of each method averaged over T tasks, under the asynchronous setting. Similar to the synchronous case, FedGP improves the accuracy of baseline methods including A-GEM and DER. It is worth noting that various methods have a better performance in

Table 5. The Acc_T (%) performance measured when we have $K = 20$ users. Similar to the results for $K = 10$ in Table 1, our method improves the performance of baselines.

	rotated-MNIST (*Domain-IL*)		**sequential-CIFAR10** (*Class-IL*)		**sequential-CIFAR10** (*Task-IL*)	
	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP
FedAvg	$62.45^{\pm 8.5}$	$76.01^{\pm 4.6}$ (↑**13.56**)	$16.44^{\pm 1.4}$	$15.82^{\pm 1.7}$ (↓**0.62**)	$68.18^{\pm 5.3}$	$73.45^{\pm 4.3}$ (↑**5.27**)
FedCurv	$62.57^{\pm 8.3}$	$76.46^{\pm 4.1}$ (↑**13.89**)	$17.31^{\pm 0.6}$	$14.64^{\pm 3.1}$ (↓**2.67**)	$67.33^{\pm 3.3}$	$70.31^{\pm 3.7}$ (↑**2.98**)
FedProx	$62.14^{\pm 8.6}$	$75.84^{\pm 4.4}$ (↑**13.70**)	$16.37^{\pm 1.1}$	$16.15^{\pm 1.3}$ (↓**0.22**)	$66.24^{\pm 1.4}$	$74.79^{\pm 3.9}$ (↑**8.55**)
A-GEM	$67.66^{\pm 8.0}$	$78.10^{\pm 3.6}$ (↑**10.44**)	$16.15^{\pm 1.9}$	$17.36^{\pm 0.8}$ (↑**1.21**)	$72.39^{\pm 3.4}$	$80.61^{\pm 2.6}$ (↑**8.22**)
DER	$57.33^{\pm 3.2}$	$87.84^{\pm 1.5}$ (↑**30.51**)	$17.13^{\pm 2.3}$	$19.18^{\pm 3.7}$ (↑**2.05**)	$70.82^{\pm 1.9}$	$77.04^{\pm 2.5}$ (↑**6.22**)
Ideal	$91.50^{\pm 0.7}$		$71.67^{\pm 0.9}$		$91.94^{\pm 0.5}$	
	permuted-MNIST (*Domain-IL*)		**sequential-CIFAR100** (*Class-IL*)		**sequential-CIFAR100** (*Task-IL*)	
FedAvg	$20.26^{\pm 1.6}$	$20.67^{\pm 4.7}$ (↑**0.41**)	$8.61^{\pm 0.1}$	$17.47^{\pm 1.1}$ (↑**8.86**)	$50.00^{\pm 1.6}$	$76.29^{\pm 0.8}$ (↑**26.29**)
FedCurv	$20.25^{\pm 1.9}$	$23.30^{\pm 5.7}$ (↑**3.05**)	$8.93^{\pm 0.0}$	$19.42^{\pm 1.1}$(↑**10.49**)	$49.83^{\pm 1.4}$	$79.58^{\pm 0.6}$ (↑**29.75**)
FedProx	$20.19^{\pm 1.4}$	$23.78^{\pm 5.2}$ (↑**3.59**)	$8.88^{\pm 0.1}$	$18.86^{\pm 1.0}$ (↑**9.98**)	$50.86^{\pm 1.2}$	$78.19^{\pm 0.9}$ (↑**27.33**)
A-GEM	$24.43^{\pm 2.1}$	$23.29^{\pm 3.8}$ (↓**1.14**)	$8.62^{\pm 0.1}$	$19.58^{\pm 1.2}$ (↑**10.96**)	$63.02^{\pm 0.6}$	$76.23^{\pm 0.6}$ (↑**13.21**)
DER	$17.89^{\pm 1.3}$	$46.17^{\pm 3.0}$ (↑**28.28**)	$11.53^{\pm 0.5}$	$26.64^{\pm 2.8}$ (↑**15.11**)	$57.00^{\pm 1.4}$	$69.42^{\pm 1.0}$ (↑**12.42**)
Ideal	$65.05^{\pm 3.6}$		$67.51^{\pm 0.3}$		$90.40^{\pm 0.4}$	

the asynchronous setting (see Table 4) compared with the synchronous setting (see Table 1). This is because, in the asynchronous setting, some clients receive new tasks earlier than others, which allows the model to be exposed to more diverse data for each round, thus reducing the forgetting effect.

Effect of the number of users. While our previous experiments are conducted for cases with $K = 10$ clients, Table 5 shows the results for $K = 20$ clients. This results demonstrates that FedGP consistently improves the performance of baselines, across different number of clients.

5.2 Object Detection

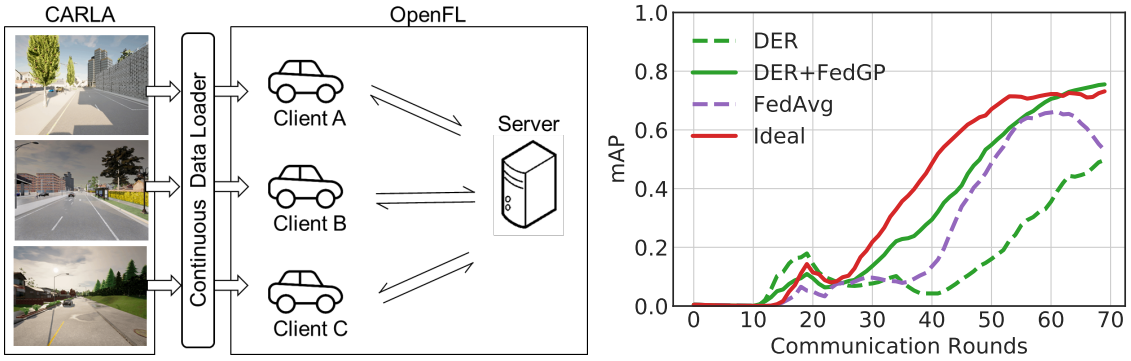
Here we test FedGP on realistic streaming data (Dai et al., 2023) which leverage two open source tools, an urban driving simulator (CARLA (Dosovitskiy et al., 2017)) and a FL framework (OpenFL (Reina et al., 2021)). As shown in Fig. 4a, CARLA provides OpenFL with a real-time collection of continuous streaming vehicle camera output data and automatic annotation about object detection. This streaming data capture the spatio-temporal dynamics of data generated from real-world applications. After loading data of vehicles from CARLA, OpenFL performs collaborative training over multiple clients.

We evaluate the solutions to the forgetting problem by spawning two vehicles in a virtual town. During the training of the tinyYOLO (Redmon & Farhadi, 2017) object detection model, we use a custom loss that combines classification, detection and confidence losses. In order to quantify the quality of the incremental model trained by various baselines, we report a common metric, namely, mean average precision (mAP). This metric assesses the correspondence between the detected bounding boxes and the ground truth, with higher scores indicating better performance. To calculate mAP, we analyze the prediction results obtained from pre-collected driving snippets of vehicular clients. These driving snippets are gathered by navigating the town over a duration of 3000 simulation seconds.

For those experiments on realistic CARLA streaming data, we compare the performances of Ideal, FedAvg, DER and DER+FedGP. The last two methods are equipped with buffer size of 200. We train for 70 communication rounds and each round continues for about 200 simulation seconds. The results are presented in Fig. 4b. Note that at communication round 60, one client gets on the highway, which incurs a domain shift. One can confirm that the performance of FedAvg degrades in such domain shift scenario, whereas DER and DER+FedGP maintain the accuracy. Moreover, FedGP nearly achieves the performance of the ideal scenario with infinite buffer size, demonstrating the effectiveness of our method.

6 Conclusion

In this paper, we present FedGP, a novel method of using buffer data for mitigating the catastrophic forgetting issues in continual federated learning. Specifically, we use the gradient projection method to prevent model updates that harm the performance on previous tasks. Our empirical results on benchmark datasets (rotated-MNIST, permuted-MNIST, sequential-CIFAR10 and sequential-CIFAR100) and on a realistic streaming automotive dataset (generated by the CARLA simulator and the OpenFL framework) show that FedGP improves the performance of existing continual federated learning methods.



(a) Framework for automotive data evaluation.

(b) Object detection performance comparison.

Figure 4. (a) The data loader continuously supplies data from CARLA camera outputs to individual FL clients. Each client trains on its local data and updates its buffer to retain old knowledge. (b) The result shows the object detection performance comparison between Ideal, FedAvg, DER, and DER+FedGP on a realistic CARLA dataset.

ACKNOWLEDGEMENTS

We'd like to thank the reviewers for their useful comments. This work is funded by Intel/NSF joint grant CNS-2003129.

REFERENCES

Aljundi, R., Chakravarty, P., and Tuytelaars, T. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3366–3375, 2017.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154, 2018.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V., and Talwalkar, A. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 532–547, 2018a.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018b.

Chen, Z. and Liu, B. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

Dai, S., Alam, S. M. I., Balakrishnan, R., Lee, K., and Suman Banerjee, N. H. Online federated learning based object detection across autonomous vehicles in a virtual world. *IEEE Consumer Communications & Networking Conference (CCNC) Demo*, 2023.

Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Dong, J., Wang, L., Fang, Z., Sun, G., Xu, S., Wang, X., and Zhu, Q. Federated class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10164–10173, 2022.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

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

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

Hsu, T.-M. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

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

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., and Miao, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3): 2031–2063, 2020.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Ma, Y., Xie, Z., Wang, J., Chen, K., and Shou, L. Continual federated learning based on knowledge distillation.

Mallya, A. and Lazebnik, S. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Mehta, S. V., Patil, D., Chandar, S., and Strubell, E. An empirical investigation of the role of pre-training in lifelong learning. *arXiv preprint arXiv:2112.09153*, 2021.

Mohri, M., Sivek, G., and Suresh, A. T. Agnostic federated learning. In *International Conference on Machine Learning*, pp. 4615–4625. PMLR, 2019.

Ratcliff, R. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Redmon, J. and Farhadi, A. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

Reina, G. A., Gruzdev, A., Foley, P., Perepelkina, O., Sharma, M., Davidyuk, I., Trushkin, I., Radionov, M., Mokrov, A., Agapov, D., et al. Openfl: An open-source framework for federated learning. *arXiv preprint arXiv:2105.06413*, 2021.

Ring, M. B. Child: A first step towards continual learning. In *Learning to learn*, pp. 261–292. Springer, 1998.

Robins, A. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Serra, J., Suris, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pp. 4548–4557. PMLR, 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Shmelkov, K., Schmid, C., and Alahari, K. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the IEEE international conference on computer vision*, pp. 3400–3409, 2017.

Shoham, N., Avidor, T., Keren, A., Israel, N., Benditkis, D., Mor-Yosef, L., and Zeitak, I. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

Thrun, S. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, 8, 1995.

Venkatesha, Y., Kim, Y., Park, H., Li, Y., and Panda, P. Addressing client drift in federated continual learning with adaptive optimization. *Available at SSRN 4188586*, 2022.

Vitter, J. S. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1): 37–57, 1985.

Yao, X. and Sun, L. Continual local training for better initialization of federated models. In *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 1736–1740. IEEE, 2020.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Yoon, J., Jeong, W., Lee, G., Yang, E., and Hwang, S. J. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pp. 12073–12086. PMLR, 2021.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995. PMLR, 2017.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

A NLP TASK

In addition to image classification, we also extended the evaluation of our method on text classification task (Mehta et al., 2021). For this purpose, we utilized the YahooQA (Zhang et al., 2015) dataset which comprises texts (questions and answers), and user-generated labels representing 10 different topics. Similar to the approach taken with the CIFAR10 dataset, we partitioned the YahooQA dataset into 5 tasks, where each task consisted of two distinct classes. Within each task, we used LDA to partition data across 10 clients in a non-IID manner. To conduct the experiment, we employed DistilBERT (Sanh et al., 2019) with linear classification layer. We freeze the DistilBERT model and only fine-tune the additional linear layer. The results of this experiment can be found in Table 6. Analyzing the table, we can observe that FedGP enhances the accuracy (Acc_T), particularly in class-IL scenarios.

B ADDITIONAL RESULTS

We present the complementary information to Table 1 in Table 7, illustrating the extent of Fgt_T observed across multiple benchmark datasets. Our method exhibits exceptional effectiveness in mitigating forgetting. Remarkably, it demonstrates consistent performance across all datasets and baselines, making it a versatile solution. As mentioned in Fig.3, we present the average accuracy (Acc_t) and average forgetting (Fgt_t) results for permuted-MNIST and sequential-CIFAR10 in Fig.5. This figure reinforces the main message conveyed by Fig. 3, highlighting the superior performance of FedGP in terms of higher accuracy and lower forgetting across tasks $t = 1$ to $t = 10$.

In the main body of our study, we examined the influence of different buffer sizes on the performance metric Acc_T, utilizing rotated-MNIST and sequential-CIFAR10 datasets. To further augment our analysis, we have included two additional datasets in Table 8, incorporating various buffer sizes. By evaluating Acc_T (where higher values indicate better performance), we discovered that our proposed method, referred to as FedGP, consistently enhances the average accuracy across these two datasets.

In line with the presentation of forgetting in Table 7, we present the forgetting analysis when the number of clients is set to 20 in Table 9. Notably, our method exhibits consistent and impressive performance across varying numbers of users. It consistently proves its effectiveness regardless of the specific user count, showcasing its robustness and reliability.

C ADDITIONAL ALGORITHMS

In this section, we present the pseudo-code for the `ComputeBufferGrad` algorithm (see Algorithm 3), which calculates the gradient of the model on buffer data. Additionally, we describe the `ReservoirSampling` algorithm (see Algorithm 4). In the initial phase, when the buffer is not yet full (i.e., $n \leq |\mathcal{M}^k|$), `ReservoirSampling` stores each new sample (x, y) in the buffer. After the buffer is full, the algorithm determines two things: (1) whether it should replace an element in the buffer with the new sample, and (2) which element in the buffer it will replace.

Algorithm 3 `ComputeBufferGrad`$(w, \mathcal{M}^k)$

Input: global model w, local buffer $\mathcal{M}^k$
$(x_1, y_1), \ldots, (x_m, y_m) \leftarrow m$ random samples from $\mathcal{M}^k$
$g = \frac{1}{m} \sum_{i=1}^{m} \nabla_w \left[\ell(y_i, f(x_i, w))\right]$
Return g to server

Algorithm 4 `ReservoirSampling`$(\mathcal{M}^k, (x, y), n)$ (Vitter, 1985) at client k

Input: local buffer $\mathcal{M}^k$, incoming data (x, y) and the number of observed samples n
if $n \leq |\mathcal{M}^k|$ **then**
  Add data (x, y) into local buffer $\mathcal{M}^k$
else
  $i \leftarrow \mathrm{Uniform}\{1, 2, \cdots, n\}$
  if $i \leq |\mathcal{M}^k|$ **then**
    $\mathcal{M}^k[i] \leftarrow (x, y)$
  end if
end if
Return $\mathcal{M}^k$, the updated local buffer

D CONTINUAL LEARNING METHODS WITH FEDGP

We provide detailed information regarding the operation (denoted as `RefineGradient`) in Algorithm 2, along with two examples of continual learning methods using FedGP.

Algorithm 5 incorporates Dark Experience Replay (DER) into the local update process on client $k \in [K]$. At time t, when the server sends the global model w_{t-1} to client k, the client calculates the output logits or pre-softmax response z. In addition, the client samples past data (x', y') and the corresponding logits z' from the buffer $\mathcal{M}^k$. To address forgetting, the regularization term considers the Euclidean distance between the sampled output logits and the current model's output logits on buffer data. The gradient g is then refined using this regularization term to minimize the discrepancy between the current and past output logits, thereby mitigating forgetting. The following steps are the same as in the main text.

Table 6. Average classification accuracy Acc_T (%) on text classification datasets at the final task T.

	sequential-YahooQA (*Class-IL*)		**sequential-YahooQA** (*Task-IL*)	
	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP
FedAvg	$17.86^{\pm 0.6}$	$30.67^{\pm 4.4}$(↑**12.81**)	$80.87^{\pm 1.2}$	$88.04^{\pm 1.4}$(↑**7.17**)
A-GEM	$20.86^{\pm 0.3}$	$47.02^{\pm 1.9}$(↑**26.16**)	$87.29^{\pm 1.3}$	$90.20^{\pm 0.2}$(↑**2.91**)
DER	$43.64^{\pm 2.1}$	$54.28^{\pm 1.3}$(↑**10.64**)	$89.57^{\pm 0.2}$	$90.48^{\pm 0.2}$(↑**0.91**)
Ideal	$66.11^{\pm 1.0}$		$91.60^{\pm 0.2}$	

Table 7. Average forgetting Fgt_T (%) (lower is better) on benchmark datasets at the final task T.

	rotated-MNIST (*Domain-IL*)		**sequential-CIFAR10** (*Class-IL*)		**sequential-CIFAR10** (*Task-IL*)	
	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP
FedAvg	$25.98^{\pm 3.2}$	$11.66^{\pm 2.7}$(↓**14.32**)	$80.69^{\pm 3.6}$	$78.62^{\pm 4.3}$(↓**2.07**)	$15.37^{\pm 4.8}$	$4.49^{\pm 1.9}$(↓**10.88**)
FedCurv	$25.80^{\pm 2.4}$	$11.18^{\pm 2.7}$(↓**14.62**)	$80.90^{\pm 6.6}$	$79.85^{\pm 3.9}$(↓**1.05**)	$19.37^{\pm 4.8}$	$4.77^{\pm 1.6}$(↓**14.60**)
FedProx	$25.74^{\pm 3.1}$	$11.76^{\pm 2.9}$(↓**13.98**)	$84.35^{\pm 2.4}$	$80.24^{\pm 2.5}$(↓**4.11**)	$18.24^{\pm 4.9}$	$4.17^{\pm 1.0}$(↓**14.07**)
A-GEM	$26.30^{\pm 5.7}$	$15.18^{\pm 2.4}$(↓**11.12**)	$82.18^{\pm 6.6}$	$80.38^{\pm 2.5}$(↓**1.80**)	$10.00^{\pm 3.0}$	$4.15^{\pm 0.7}$(↓**5.85**)
DER	$21.42^{\pm 4.0}$	$5.51^{\pm 1.2}$(↓**15.91**)	$60.98^{\pm 14.6}$	$47.88^{\pm 7.2}$(↓**13.10**)	$6.34^{\pm 4.9}$	$2.73^{\pm 1.3}$(↓**3.61**)
Ideal	$0.25^{\pm 0.2}$		$9.05^{\pm 1.4}$		$0.10^{\pm 0.1}$	
	permuted-MNIST (*Domain-IL*)		**sequential-CIFAR100** (*Class-IL*)		**sequential-CIFAR100** (*Task-IL*)	
FedAvg	$43.47^{\pm 5.3}$	$21.40^{\pm 4.9}$(↓**22.07**)	$77.69^{\pm 0.5}$	$67.02^{\pm 2.3}$(↓**10.67**)	$34.38^{\pm 1.6}$	$5.39^{\pm 0.8}$(↓**28.99**)
FedCurv	$42.88^{\pm 5.0}$	$22.85^{\pm 3.5}$(↓**20.03**)	$78.40^{\pm 0.9}$	$67.75^{\pm 0.8}$(↓**10.65**)	$33.71^{\pm 2.2}$	$5.86^{\pm 0.7}$(↓**27.85**)
FedProx	$42.59^{\pm 5.6}$	$20.77^{\pm 5.6}$(↓**21.82**)	$77.35^{\pm 0.4}$	$66.81^{\pm 2.2}$(↓**10.54**)	$34.79^{\pm 3.6}$	$5.69^{\pm 0.9}$(↓**29.10**)
A-GEM	$35.61^{\pm 5.3}$	$24.05^{\pm 2.4}$(↓**11.56**)	$77.97^{\pm 0.7}$	$63.99^{\pm 2.0}$(↓**13.98**)	$16.92^{\pm 1.1}$	$5.16^{\pm 0.5}$(↓**11.76**)
DER	$45.33^{\pm 5.0}$	$34.71^{\pm 5.0}$(↓**10.62**)	$69.37^{\pm 1.7}$	$53.84^{\pm 6.7}$(↓**15.53**)	$22.43^{\pm 0.7}$	$14.16^{\pm 1.7}$(↓**8.27**)
Ideal	$3.45^{\pm 0.7}$		$3.47^{\pm 0.4}$		$0.38^{\pm 0.1}$	

Table 8. Impact of the buffer size on Acc_T (%)

		permuted-MNIST (*Domain-IL*)		**sequential-CIFAR10** (*Class-IL*)		**sequential-CIFAR10** (*Task-IL*)	
Buffer Size	Methods	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP
200	A-GEM	$33.43^{\pm 1.4}$	$39.09^{\pm 3.5}$ (↑**5.66**)	$17.82^{\pm 0.9}$	$19.44^{\pm 0.9}$ (↑**1.62**)	$77.14^{\pm 3.1}$	$83.16^{\pm 1.6}$ (↑**6.02**)
500	A-GEM	$33.35^{\pm 1.0}$	$42.45^{\pm 6.9}$ (↑**9.10**)	$18.39^{\pm 0.2}$	$20.34^{\pm 0.6}$ (↑**1.95**)	$78.43^{\pm 3.0}$	$85.95^{\pm 0.6}$ (↑**7.52**)
5120	A-GEM	$32.72^{\pm 1.4}$	$40.07^{\pm 2.5}$ (↑**7.35**)	$16.41^{\pm 2.6}$	$20.64^{\pm 2.2}$ (↑**4.23**)	$73.89^{\pm 3.3}$	$86.82^{\pm 1.5}$ (↑**12.93**)
200	DER	$19.79^{\pm 1.7}$	$43.43^{\pm 0.9}$ (↑**23.64**)	$18.44^{\pm 3.7}$	$30.94^{\pm 3.8}$ (↑**12.50**)	$69.34^{\pm 3.2}$	$77.99^{\pm 0.8}$ (↑**8.65**)
500	DER	$19.17^{\pm 1.6}$	$43.38^{\pm 2.4}$ (↑**24.21**)	$20.81^{\pm 3.6}$	$29.78^{\pm 4.3}$ (↑**8.97**)	$71.17^{\pm 1.5}$	$74.98^{\pm 3.5}$ (↑**3.81**)
5120	DER	$18.57^{\pm 1.4}$	$44.68^{\pm 2.4}$ (↑**26.11**)	$34.75^{\pm 2.2}$	$42.38^{\pm 4.5}$ (↑**7.63**)	$78.22^{\pm 2.3}$	$81.94^{\pm 1.7}$ (↑**3.72**)

Table 9. The Fgt_T (%) (lower is better) performance measured when we have $K = 20$ users.

	rotated-MNIST (*Domain-IL*)		**sequential-CIFAR10** (*Class-IL*)		**sequential-CIFAR10** (*Task-IL*)	
	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP	w/o FedGP	w/ FedGP
FedAvg	$31.00^{\pm 9.5}$	$13.45^{\pm 3.6}$ (↓**17.55**)	$82.62^{\pm 3.1}$	$73.39^{\pm 4.5}$ (↓**9.23**)	$17.93^{\pm 2.7}$	$6.14^{\pm 4.9}$ (↓**11.79**)
FedCurv	$30.73^{\pm 9.3}$	$12.97^{\pm 3.8}$ (↓**17.76**)	$79.55^{\pm 3.8}$	$75.38^{\pm 5.3}$ (↓**4.17**)	$18.19^{\pm 3.0}$	$9.14^{\pm 3.1}$ (↓**9.05**)
FedProx	$31.04^{\pm 9.7}$	$13.31^{\pm 3.4}$ (↓**17.73**)	$82.94^{\pm 1.1}$	$78.67^{\pm 4.2}$ (↓**4.27**)	$20.60^{\pm 2.6}$	$8.52^{\pm 3.0}$ (↓**12.08**)
A-GEM	$25.22^{\pm 8.8}$	$11.02^{\pm 3.0}$ (↓**14.20**)	$82.39^{\pm 2.4}$	$80.25^{\pm 4.1}$ (↓**2.14**)	$12.29^{\pm 2.2}$	$4.00^{\pm 2.4}$ (↓**8.29**)
DER	$28.93^{\pm 6.6}$	$5.18^{\pm 1.1}$ (↓**23.75**)	$55.10^{\pm 9.8}$	$60.90^{\pm 3.8}$ (↑**5.80**)	$3.20^{\pm 1.6}$	$2.71^{\pm 1.7}$ (↓**0.49**)
Ideal	$0.15^{\pm 0.1}$		$9.15^{\pm 1.3}$		$0.34^{\pm 0.4}$	
	permuted-MNIST (*Domain-IL*)		**sequential-CIFAR100** (*Class-IL*)		**sequential-CIFAR100** (*Task-IL*)	
FedAvg	$24.27^{\pm 5.2}$	$8.67^{\pm 7.0}$ (↓**15.60**)	$73.05^{\pm 0.5}$	$62.71^{\pm 0.9}$ (↓**10.34**)	$27.07^{\pm 1.7}$	$2.48^{\pm 0.7}$ (↓**24.59**)
FedCurv	$24.02^{\pm 5.4}$	$8.10^{\pm 5.4}$ (↓**15.92**)	$80.07^{\pm 0.5}$	$68.58^{\pm 1.1}$ (↓**11.49**)	$34.63^{\pm 1.7}$	$3.48^{\pm 0.6}$ (↓**31.15**)
FedProx	$23.01^{\pm 5.7}$	$5.93^{\pm 5.1}$ (↓**17.08**)	$79.46^{\pm 0.5}$	$68.40^{\pm 0.9}$ (↓**11.06**)	$32.82^{\pm 1.4}$	$4.13^{\pm 0.7}$ (↓**28.69**)
A-GEM	$22.12^{\pm 4.9}$	$9.45^{\pm 5.4}$ (↓**12.67**)	$72.97^{\pm 1.1}$	$60.27^{\pm 1.3}$ (↓**12.70**)	$12.54^{\pm 1.3}$	$2.66^{\pm 0.2}$ (↓**9.88**)
DER	$32.26^{\pm 1.1}$	$27.30^{\pm 4.2}$ (↓**4.96**)	$67.07^{\pm 0.8}$	$47.74^{\pm 3.8}$ (↓**19.33**)	$19.78^{\pm 1.7}$	$8.67^{\pm 1.4}$ (↓**11.11**)
Ideal	$1.35^{\pm 0.4}$		$1.66^{\pm 0.3}$		$0.14^{\pm 0.1}$	

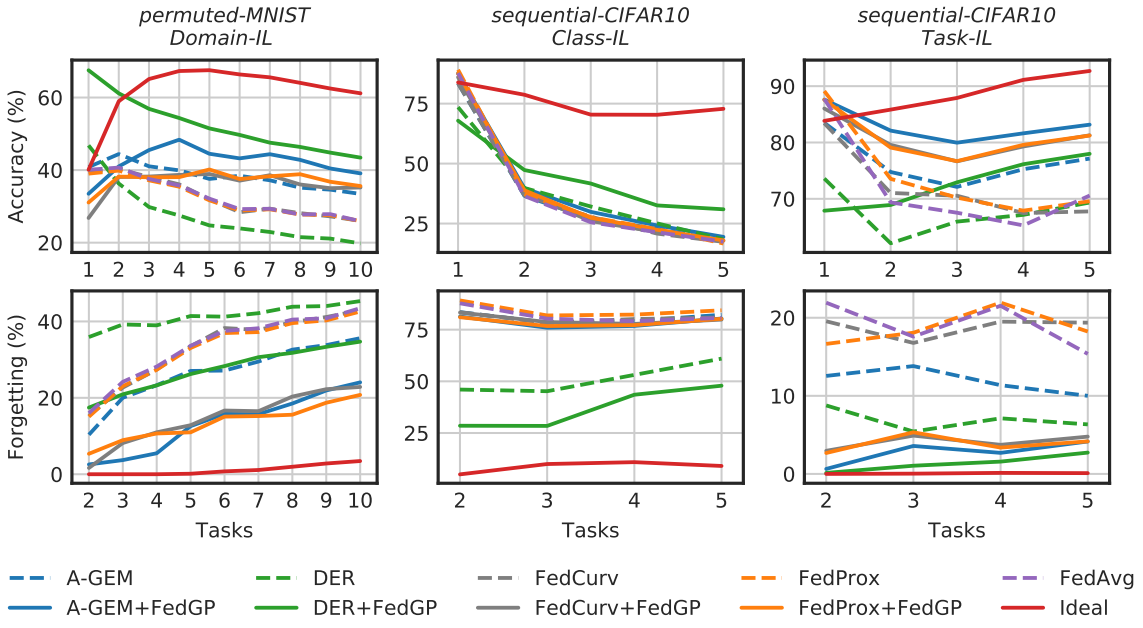


Figure 5. Evaluating accuracy and forgetting in multiple datasets with and without FedGP using a buffer size of 200. The solid lines indicate the results obtained with our method or upper bound, while the dotted lines represent the results obtained without our method. The experimental results demonstrate a significant improvement in accuracy for all settings.

Algorithm 6 applies Gradient Projection twice. First, the client computes the gradient g_c with respect to the new data from $\mathcal{D}_t^k$. After replaying previous samples (x', y') stored in the local buffer $\mathcal{M}^k$, the client computes the gradient g_b with respect to this buffered data. If these gradients differ significantly in terms of their direction, the client projects g_c onto g_b to remove interference.

E ADDITIONAL RELATED WORK

We summarize the prior works that are related to our paper, which are categorized as continual learning, federated learning, and continual federated learning.

E.1 Continual Learning (CL)

CL is a problem of learning multiple different tasks consecutively using a single model (Ring, 1998; Delange et al., 2021). For example, when the tasks are classification problems, CL focuses on the scenario when a classifier is trained for one task in the first phase, and then trained for another task in the second phase, and so on. In general, the data loaded at the current phase has a different distribution compared to the data at the previous phases, known as domain distribution shift or class distribution shift. Unfortunately, since the learner has a limited amount of memory to store data, the classifier is only allowed to access the data for the current task, not for the previous tasks. In such a setting, catastrophic forgetting (McCloskey & Cohen, 1989; Ratcliff, 1990; French, 1999) is a notorious problem, where a classifier that performs well for the task from the current round does not perform well on the tasks from the previous rounds. There has been extensive work to address this issue and can be divided into three major categories: regularization-based methods, architecture-based methods and replay-based methods.

Regularization-based methods Some CL methods add a regularization term in the loss used for the model update; they penalize the updates on weights that are important for previous tasks. EWC (Kirkpatrick et al., 2017), SI (Zenke et al., 2017), Riemannian Walk (Chaudhry et al., 2018a) are methods within this category. EWC uses Fisher information matrix to evaluate the importance of parameters for previous tasks. Besides, LwF (Li & Hoiem, 2017) leverages knowledge distillation to preserve outputs on previous tasks while learning the current task.

Architecture-based methods A class of CL methods assigns a subset of model parameters to each task, so that different tasks are learned by different parameters. This class of methods is also known as parameter isolation methods. Some methods including PNN (Rusu et al., 2016) and DEN (Yoon et al., 2017) uses dynamic architectures where the architecture changes dynamically as the number of tasks increases. These methods have issues where the number of required parameters grows linearly with the number of tasks. To tackle this issue, fixed network are used in the recent methods including PackNet (Mallya & Lazebnik, 2018), HAT (Serra et al., 2018) and PathNet (Fernando et al., 2017).

Replay-based methods To avoid catastrophic forgetting, a class of CL methods employs a replay buffer to save a small portion of the data seen in previous tasks and reuse it in the training of subsequent tasks.One of the early works in this area is ER (Ratcliff, 1990; Robins, 1995). More recent work, such as iCaRL (Rebuffi et al., 2017) stores exemplars of data from previous tasks and adds distillation loss for old exemplars to mitigate the forgetting issue. Deep Generative Replay (Shin et al., 2017) retains the memories of the previous tasks by loading the synthetic data generated by GANs without replaying the actual data for the previous tasks. GSS (Aljundi et al., 2019) optimally selects data for replay buffer by maximizing the diversity of samples in terms of the gradient in the parameter space. GEM (Lopez-Paz & Ranzato, 2017) and its variant A-GEM (Chaudhry et al., 2018b) leverage an episodic memory that stores part of seen samples for each task to prevent forgetting old knowledge.

General continual learning Prior works on CL often rely on the information about the task boundaries. For example, some replay-based methods perform specific steps specifically at task boundaries, some regularization-based methods store network responses at these boundaries; architecture-based methods update the model architecture after one task is finished. However, when dealing with streaming data in practical settings, task boundaries are not clearly defined. This scenario, where sequential tasks are learned continuously without explicit knowledge of task boundaries, is referred to as *general continual learning* (Buzzega et al., 2020; Aljundi et al., 2019; Chaudhry et al., 2018b). To address general continual learning, replay-based methods can utilize reservoir sampling (Vitter, 1985), which allows sampling throughout the training rather than relying on task boundaries. In our work, we specifically focus on general continual learning with reservoir sampling, particularly in the context of federated learning setups.

E.2 Federated Learning (FL)

There is a rapidly increasing level of interest in FL from both industry and academia, especially due to its benefits on enabling multiple users to collaboratively train a model with improved data privacy (Kairouz et al., 2021; Lim et al., 2020; Zhao et al., 2018; Konečnỳ et al., 2016). FedAvg (McMahan et al., 2017) is a widely used algorithm in FL where each round consists of three steps: first, each client updates its local model using its data and transmits the updated local model to the server; second, the central server aggregates the updated local models and updates the global model in the direction of the average of local updates; third, the global

Algorithm 5 DER `ClientUpdate`(t, w, g_{ref}) at client k

Input: Task index t, model w, buffer gradient g_{ref}
Load the dataset $\mathcal{D}_t^k$, local buffer $\mathcal{M}^k$
$n \leftarrow 0$
for $(x, y) \in \mathcal{D}_t^k$ **do**
  $z \leftarrow h(x, w)$ where $f(x, w) \coloneqq \sigma\left(h(x, w)\right)$
  $(x', z', y') \leftarrow \mathcal{M}^k$
  $\ell_{\text{reg}} \leftarrow \lambda \left\| z' - h(x', w) \right\|_2^2$
  $g = \nabla_w \left[\ell(y, f(x, w)) + \ell_{\text{reg}}\right]$
  $\tilde{g} \leftarrow g - \text{proj}_{g_{\text{ref}}} g \cdot \mathbf{1}(g_{\text{ref}}^\top g \leq 0)$
  $w \leftarrow w - \alpha \tilde{g}$ for some learning rate α
  `ReservoirSampling`$(\mathcal{M}^k, (x, z, y), n)$
  $n \leftarrow n + 1$
end for
Return w to server

Algorithm 6 A-GEM `ClientUpdate`(t, w, g_{ref}) at client k

Input: Task index t, model w, buffer gradient g_{ref}
Load the dataset $\mathcal{D}_t^k$, local buffer $\mathcal{M}^k$
$n \leftarrow 0$
for $(x, y) \in \mathcal{D}_t^k$ **do**
  $g_c = \nabla_w \left[\ell(y, f(x, w))\right]$
  $(x', y') \leftarrow \mathcal{M}^k$
  $g_b = \nabla_w \left[\ell(y', f(x', w))\right]$
  $g \leftarrow g_c - \text{proj}_{g_b} g_c \cdot \mathbf{1}(g_b^\top g_c \leq 0)$
  $\tilde{g} \leftarrow g - \text{proj}_{g_{\text{ref}}} g \cdot \mathbf{1}(g_{\text{ref}}^\top g \leq 0)$
  $w \leftarrow w - \alpha \tilde{g}$ for some learning rate α
  `ReservoirSampling`$(\mathcal{M}^k, (x, y), n)$
  $n \leftarrow n + 1$
end for
Return w to server

model is broadcasted to each client and the local model is redefined as the global model; we repeat these three steps for multiple communication rounds. Variants of FedAvg were suggested in recent years (Li et al., 2020; Shoham et al., 2019; Karimireddy et al., 2020; Li et al., 2019; Mohri et al., 2019), but most existing works assume that the data distribution is static over time, which fails to capture the temporal dynamics of real-world data.

E.3 Continual Federated Learning (CFL)

CFL is a problem of learning multiple consecutive tasks in the FL setup. Several CFL methods have been proposed in the literature. FedProx (Li et al., 2020) adds a proximal term to limit the impact of variable local updates, while FedCurv (Shoham et al., 2019) adds a penalty term using the diagonal of the Fisher information matrix to protect the important parameters for each task. Both methods aim to preserve previously learned tasks while training new ones. Although FedProx (Li et al., 2020) and FedCurv (Shoham et al., 2019) are effective approaches to mitigate forgetting in CFL, they have been shown to achieve suboptimal performance when applied in a naive manner. Other approaches, such as FedWeIT (Yoon et al., 2021) and NetTailor (Venkatesha et al., 2022), have attempted to prevent interference between irrelevant tasks by decomposing network parameters or using a dynamic architecture approach. However, these methods may not be practical for a large number of tasks as the number of parameters scale linearly and require a clear understanding of all task identities or boundaries in advance, which may not be feasible in real-world scenarios. CFeD (Ma et al.) and FedCL (Yao & Sun, 2020) utilize surrogate datasets or global importance weights to distill learned knowledge and constrain local model updates, respectively. However, these methods require extra effort to generate or collect auxiliary data and may consume extra communications or storage overhead. GLFC (Dong et al., 2022) is another approach that uses a class-aware gradient compensation loss and a class-semantic relation distillation loss to overcome catastrophic forgetting, but it only considers class-incremental learning scenarios.

Unlike previous methods, our approach does not require explicit task boundaries, making it more practical for real-world applications. Our method achieves this by aligning the gradients of the current model with those of the global buffer, which contains past experiences from multiple clients. By leveraging this collective experience, FedGP can effectively mitigate forgetting of previously learned knowledge in FL scenarios with continuous data and real-world temporal dynamics.