BufferGA: Buffer-based Gradient Adjustment for Continual Federated Learning

Anonymous Authors[1]

Abstract

Continual Federated Learning (CFL) has garnered significant attention in recent years due to its potential in real-world scenarios where clients (e.g., mobile phones, autonomous vehicles) continuously observe data in a dynamic environment. However, CFL, like the centralized continual learning paradigm, can suffer 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 Adjustment (BufferGA) that mitigates catastrophic forgetting by replaying local buffer samples and using aggregated buffer gradients to ensure positive knowledge transfer across clients. Our method can be integrated with a variety of existing continual learning methods, and further boost the performance in a CFL setup. Our approach is evaluated on benchmark datasets such as rotated-MNIST and split-CIFAR100, as well as on a realistic streaming decentralized automotive dataset generated using CARLA and OpenFL. Our experiments show that BufferGA outperforms state-of-the-art techniques on multiple datasets, providing a promising approach for continual federated learning in dynamic environments.

1 Introduction

Federated Learning (FL) is a machine learning technique that enables multiple users to train a model collaboratively while keeping their data decentralized for better privacy and efficient communication. However, current FL benchmarks (Caldas et al., 2018; Krizhevsky et al., 2009; Li et al., 2020; McMahan et al., 2017) assume a static distribution of data at each client throughout the training, which may not reflect the real-world scenario where data varies significantly across time. To address this challenge, continual learning is a crucial consideration, as it enables a model to learn continuously from incoming data, thereby improving its accuracy and adaptability. By incorporating continual learning into FL, we can create more robust and flexible models that can evolve with changing data distributions, ultimately leading to more effective and accurate machine learning systems.

Continual learning is also known as incremental learning (Shmelkov et al., 2017; Chaudhry et al., 2018a) or lifelong learning (Thrun, 1995; Aljundi et al., 2017; Chen & Liu, 2018; Aljundi et al., 2018) and has been extensively studied in the context of learning from sequential data streams. However, to the best of our knowledge, there has been limited exploration of continual learning in the context of federated learning. The ability of a model to continuously learn from incoming data and acquire new knowledge on-the-fly is crucial, especially in the context of federated learning. This is because data can vary significantly over time and across users, and the model needs to adapt to these changes in a seamless and efficient manner. Incorporating continual learning into federated learning can lead to more flexible, robust, and accurate models that can learn and adapt continuously, improving their performance and applicability in real-world scenarios.

Most existing continual learning scenarios assume that the data is divided into separate tasks, and the model needs to learn from each task sequentially. However, this approach may not be practical in scenarios where the data is continuously streamed without explicit task boundaries. In this work, we want to consider a more practical scenario where there are no fixed task boundaries throughout training. We refer to this type of continual learning without task boundaries as general continual learning, as described in (Buzzega et al., 2020). By taking this practical aspect into account, our proposed method can better handle real-world scenarios where data streams are continuous and dynamic, without relying on predefined task boundaries.

Similar to continual learning, a continual federated learning model may suffer from the problem of catastrophic forgetting, where the model may drift in favor of new data and forget previously learned knowledge. However, the model needs to adapt to new data while retaining its exist-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Machine Learning and Systems (MLSys) Conference. Do not distribute.

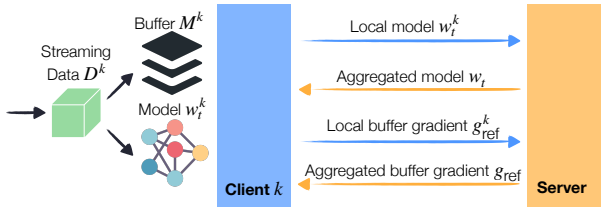

Figure 1. An overview of BufferGA method proposed for continual federated learning. At each time step t, each client k receives streaming data D_t^k and trains a local model w_t^k. To address catastrophic forgetting, a portion of the incoming data is stored in a buffer $\mathcal{M}^k$ for future use. Given the aggregated model w_t provided by the central server, each client computes the gradient of w_t with respect to their own buffer $\mathcal{M}^k$, denoted by g_{ref}^k. Each client then sends this gradient to the server, which aggregates the buffer gradients from all clients to obtain a global buffer gradient g_{ref}. This global buffer gradient is sent back to each client. In the next round of model updates, each client uses the aggregated buffer gradient g_{ref} to update their local model, in addition to their own local gradient computed on new data.

ing or prior knowledge, without relying on access to old data. Simply implementing solutions for continual learning in the context of federated learning may not fully address this problem. Therefore, new strategies need to be developed to address catastrophic forgetting in federated learning specifically.

In this paper, we propose a novel method Buffer-based Gradient Adjustment (BufferGA), as shown in Figure 1, that combines FL and continual learning without task boundaries. We leverage old memories from other clients to limit forgetting and mitigate cross-client interference. Our approach involves two key components: (i) Each collaborator trains on examples from the current task while using a replay buffer to avoid catastrophic forgetting on local historical data. This approach ensures that the model can adapt to new data while retaining relevant information from the past. (ii) The global model is used to compute gradient vectors on the buffer data, and the averaged global buffer gradient is considered as a reference for past knowledge across clients. This approach enables the model to learn from the collective experience of multiple clients and mitigate cross-client interference. By combining these two components, our proposed BufferGA method can effectively address the problem of catastrophic forgetting in federated learning. Our approach provides a practical solution for federated learning in real-world scenarios where data is continuously streaming without explicit task boundaries.

We evaluate our proposed method on rotated-MNIST, split-CIFAR100, and realistic streaming automotive dataset generated using the urban driving simulator, CARLA (Dosovitskiy et al., 2017), and the practical FL framework, OpenFL (Reina et al., 2021). Our experimental results show that the BufferGA method performs significantly better than existing state-of-the-art methods in terms of average accuracy and average forgetting. Our experimental results confirm the effectiveness and practicality of our approach and highlight its potential for real-world applications in federated learning scenarios with streaming data.

The remainder of this paper is organized as follows. We first summarize related work in Sec.2. Sec.3 provides the problem setup and preliminaries. In Sec.4, we propose our method, BufferGA, that combines FL and continual learning without task boundaries. Sec.5 presents our experiments and compares BufferGA with baselines. Finally, we conclude our paper in Sec. 6.

2 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.

2.1 Continual Learning

In the continual learning setup, we consider the scenarios when a classifier is facing multiple different tasks consecutively (Ring, 1998; Delange et al., 2021). In other words, 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. Continual learning considers the scenario when the classifier is only allowed to access the data for the current task, not for the previous task. 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: replay-based methods, regularization-based methods and architecture-based methods.

Replay-based methods To avoid catastrophic forgetting, a class of continual learning 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, such as ER and MER (Ratcliff, 1990; Robins, 1995; Riemer et al., 2018). More recent work, for example, iCaRL (Rebuffi et al., 2017) stores exemplars of previous tasks and adds distillation loss for old exemplars to limit forgetting. Deep Generative Replay (Shin et al., 2017) retains former memories by pseudo-data produced by GANs without replaying actual past data. 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. This category is also called rehearsal-based methods.

Regularization-based methods Some of continual learning methods add a regularization term in the loss used for 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 new task without old samples.

Architecture-based methods Another category of continual learning methods is called architecture-based methods, which assign a subset of model parameters to each task. These methods are also known as parameter isolation methods - they reduce the interference among different tasks. PNN (Rusu et al., 2016), DEN (Yoon et al., 2017) are dynamic architectures examples in this category. Note those dynamic architecture-based methods have issues with the amount of required memory as parameters linearly grow as the number of tasks increases. To tackle this issue, fixed network are used by PackNet (Mallya & Lazebnik, 2018), HAT (Serra et al., 2018) and PathNet (Fernando et al., 2017).

General continual learning Prior works on continual learning often rely on the information about the task boundaries. For example, some replay-based methods assign the buffer for each task, based on when a new task arrives; some regularization-based methods store network responses at the task boundaries; architecture-based methods update the model architecture after one task is finished. However, we do not have clear task boundaries in general continual learning (Buzzega et al., 2020; Aljundi et al., 2019; Chaudhry et al., 2018b) settings, *e.g.*, when we use streaming data. In such scenarios, replay-based methods can employ reservoir sampling (Vitter, 1985). Our focus is on general continual learning with reservoir sampling, especially under the federated learning setup.

2.2 Federated Learning

There is a rapidly increasing level of interest in Federated Learning 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 federated learning: at each round, each client runs multiple epochs of local model updates, and a central server updates the global model in the direction of the average of local updates. Many different 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) However, most existing works on federated learning assume that the data distribution is static over time, which fails to capture the temporal dynamics of real-world data.

2.3 Continual Federated Learning

Several continual federated learning methods have been proposed in the literature, including FedProx (Li et al., 2020) and FedCurv (Shoham et al., 2019), which are relatively simple methods. FedProx adds a proximal term to limit the impact of variable local updates, while FedCurv adds a penalty term using the diagonal of the Fisher information matrix to protect the important parameters for each task. However, these methods have only been tested on simple datasets, and may not be effective for more complex scenarios. 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 they 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.

To address these limitations, we propose a simple yet effective method, named BufferGA, that utilizes the global buffer gradient as a reference to prevent catastrophic forgetting in federated 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, BufferGA can effectively mitigate forgetting of previously learned knowledge in federated learning scenarios with continuous data and real-world temporal dynamics.

3 PRELIMINARIES

We focus on the continual learning setup of finding a single classifier f (having model parameter w) that performs well on all T tasks. We assume that at time slot $t \in [T]$, the classifier is only allowed to be trained for task t, where we define $[N] := \{1, \cdots, N\}$ 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 continual learning at time $\tau \in [T]$ is written as

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

where ℓ is the loss function, and $f(x_t, w)$ is the output of classifier f with parameter w, for a given 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 the summarization of past samples, and use the buffer to refine the model updates.

We constrain the model 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 task, this constrained optimization problem at time $\tau \in [T]$ is represented as:

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

where (x_b, y_b) is sampled from replay buffer $\mathcal{M}$.

The optimization problem in (2) can be rephrased for various continual learning 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 continual learning with replay buffers at time $\tau \in [T]$ is represented as:

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

Second, some other methods including A-GEM (Chaudhry et al., 2018b) interprete 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,

Algorithm 1 BufferGA

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
  $w_t \leftarrow \frac{1}{K}\sum_{k=1}^{K} w_t^k$ (Update Global Model)
  $g_{\text{ref}}^k \leftarrow \texttt{ComputeBufferGrad}(w_t, \mathcal{M}^k)$, $\forall$ client k
  $g_{\text{ref}} \leftarrow \frac{1}{K}\sum_{k=1}^{K} g_{\text{ref}}^k$ (Update Global Buffer Gradient)
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 $\mathcal{M}^k$
  $g \leftarrow \texttt{RefineGradient}(g_c, (x_b, y_b))$
  if $g^T g_{\text{ref}} \geq 0$ or g_{ref} is None **then**
    $\tilde{g} \leftarrow g$
  else
    $\tilde{g} \leftarrow g - \frac{g^T g_{\text{ref}}}{g_{\text{ref}}^T g_{\text{ref}}} g_{\text{ref}}$
  end if
  $w \leftarrow w - \alpha\tilde{g}$ 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

$$\begin{aligned} &\arg\min_{w} \mathbb{E}_{(x_\tau,y_\tau)\sim D_\tau} \left[\ell\left(y_\tau, f\left(x_\tau, w\right)\right)\right] \\ &\text{s.t. } \left\langle \frac{\partial \ell\left(y_\tau, f\left(x_\tau, w\right)\right)}{\partial w}, \frac{\partial \ell\left(y_b, f\left(x_b, w\right)\right)}{\partial w} \right\rangle \geq 0 \end{aligned} \tag{4}$$

For the federated continual learning 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 federated continual learning 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 BUFFERGA

We propose our method called BufferGA, which boosts up the performance of existing continual learning methods, un-

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
  $i \leftarrow \text{Uniform}\{1, 2, \cdots, n\}$
  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

der the federated learning setup. Our approach, BufferGA, 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 federated learning scenarios.

Note that BufferGA can be combined with any replay-based continual learning 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 a good performance on both new task and old tasks. The local buffer at client k is denoted by $\mathcal{M}^k$. As the streaming 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 $k \in [K]$, getting 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.

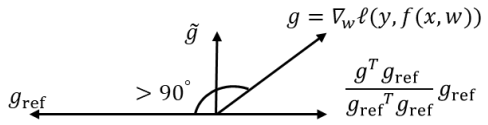

Figure 2. Illustration of the gradient projection in Eq. 5. If the angle between the gradient update g and global buffer gradient (considered as a reference) g_{ref} is larger than 90°, we project g on g_{ref} to minimize the interference and merely update along the directions of $\tilde{g}$ that is orthogonal to g_{ref}.

`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 after the training process for the previous task $t-1$. It also loads the local 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 a local training, various existing replay-based continual learning methods are designed, and we are inheriting these ideas while combining with BufferGA method. Existing ideas are refining the gradient g_c using the data (x_b, y_b) (for previous tasks) sampled from the local buffer $\mathcal{M}^k$. We encapsulate such refinement process as `RefineGradient`. Note that depending on which replay-based continual learning method we are combining with BufferGA, the function `RefineGradient` is defined differently. In the supplementary material, we added the details of `RefineGradient` when BufferGA 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

$$\tilde{g} = g - \frac{g^T g_{\text{ref}}}{g_{\text{ref}}^T g_{\text{ref}}} g_{\text{ref}}, \tag{5}$$

following the idea suggested in (Chaudhry et al., 2018b). As illustrated in Figure 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 initial phase where the buffer is not full (*i.e.*, $n \leq |\mathcal{M}^k|$), we store each new sample (x, y) in the buffer. If the buffer is full, we randomly decide (1) whether we replace an element in the buffer with the new sample, and (2) which element in the buffer we will replace. 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.

Once 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.

In summary, BufferGA retains learned knowledge by refining local gradients through correlation checks with the gradient computed on the average global buffer across all clients. To achieve this, we compute the gradients of each local buffer using the aggregated global model. Additionally, we compute the average gradient vector of the client buffer and transmit this as a reference g_{ref} of prior data information to all clients.

5 Experiments

In this section, we investigate the effectiveness of our method (BufferGA) and baseline methods in addressing continual learning tasks under the non-IID federated learning setting. We evaluate these methods on image classification tasks for the rotated-MNIST (Lopez-Paz & Ranzato, 2017) dataset and the split-CIFAR100 (Lopez-Paz & Ranzato, 2017) dataset, as well as an object detection task on the streaming CARLA data (Dai et al., 2023; Dosovitskiy et al., 2017).

5.1 Classification

Evaluation Datasets: We consider three scenarios of continual learning: (1) domain incremental learning (domain-IL), (2) task incremental learning (task-IL) and (3) class incremental learning (class-IL). For domain-IL, the data distribution of each class varies for different tasks. Our experiments for the domain-IL setup use the rotated-MNIST dataset (Lopez-Paz & Ranzato, 2017), where each task rotates the training digits by a random angle. We construct $T = 10$ tasks.

For task-IL and class-IL, we partition the set of classes into a disjoint subset, and define each task for each subset. For example, in our image classification experiments for the task-IL and class-IL setup, we split the CIFAR-100 dataset (having $C = 100$ 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 an image sampled from each subset $t \in [T]$. The only difference between task-IL and class-IL is the following: in the task-IL setup, we assume that the task identity t is provided at the inference time, *i.e.*, the model f predicts among $C/T = 10$ classes corresponding to task t.

In the federated learning setup, we assume the data distribution is non-IID among different clients. Once the data for each task is defined, we divide it for K clients in a non-IID manner. For rotated-MNIST dataset, each client gets samples for two MNIST digits. To create a split-CIFAR100 dataset, we use the Latent Dirichlet Allocation (LDA) partition algorithm. This approach partitions the dataset among multiple clients by assigning samples of each class c to different clients according to the probability distribution $p \sim \text{Dir}(\alpha)$.

Architecture and Hyperparameters: For rotated-MNIST dataset, we train a simple CNN, as outlined in (McMahan et al., 2017) and the dataset is divided into $K = 10$ clients. Each client performs local training of $E = 1$ epoch, and the number of communication rounds is set to $R = 20$ for each task. For split-CIFAR100 dataset, we train a ResNet18 and the dataset is divided into $K = 10$ clients. Each client trains for $E = 5$ epochs, and uses $R = 20$ rounds of communications for each task. The local training uses stochastic gradient descent with a learning rate of 0.1 and 0.01, respectively, for rotated-MNIST and split-CIFAR100.

Baselines: We compare BufferGA with two rehearsal-based methods, A-GEM (Chaudhry et al., 2018b) and DER (Buzzega et al., 2020), and two regularization-based methods, FedCurv (Shoham et al., 2019) and FedProx (Li et al., 2020). A-GEM ensures the alignment of the gradient of the model for the buffer data and that for the incoming data, while DER utilizes the network output logits for distilling 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

the incoming data for all tasks; we denote this method as *IDEAL*, the performance of which is a valid upper bound for all continual federated learning methods. Finally, we compare with a naive method which trains only on the current task without considering the performance on previous tasks; we denote this method as FedAvg, the performance of which is a valid lower bound for all continual federated learning methods.

Performance Metrics: We assess the performance of the global model on all previously seen tasks using their corresponding test sets. The test dataset is a union of the test data from all previous tasks. The average accuracy at task t is denoted as $Acc_t = \frac{1}{t}\sum_{i=1}^{t} a_{t,i}$, where $a_{t,i}$ is evaluated on task i after training up to task t. Additionally, forgetting is reported as a measure of the accuracy performance drop between the best accuracy obtained throughout training and the current accuracy, in order to assess the model's ability to retain knowledge of previous tasks while learning new ones. The forgetting at task t is defined as: $\frac{1}{t-1}\sum_{i=1}^{t-1} \max_{j=1}^{t-1}(a_{j,i} - a_{t,i})$. All experiments are averaged across five runs, each with a different seed.

Results: The results of experiments on rotated-MNIST and split-CIFAR100 datasets are presented in Figure 3, for three scenarios: domain-IL, class-IL, and Task-IL. The baselines considered in these experiments include A-GEM, DER, FedCurv, and FedProx, as well as upper-bound *IDEAL* and lower-bound FedAvg. The solid lines in the figure represent baselines with BufferGA or *IDEAL*, while the dotted lines represent those without BufferGA or FedAvg.

The results presented in Figure 3 demonstrate that existing regularization-based methods, such as FedCurv or FedProx, perform comparably to the FedAvg baseline in addressing the forgetting problem in Federated Learning (FL). On the other hand, using existing replay-based methods (A-GEM, DER) at the client level, without BufferGA, can alleviate forgetting to some extent in FL, resulting in higher accuracies and less forgetting compared to the baseline FedAvg method. These findings suggest that replay-based methods are more effective than regularization-based methods in preventing forgetting in FL.

Moreover, the addition of BufferGA to the existing methods resulted in significant improvements in their performance compared to the vanilla baselines. Specifically, the accuracies achieved by the methods combined with BufferGA surpassed those achieved by the baselines alone. For instance, in domain-IL, class-IL, and task-IL, BufferGA led to an improvement of up to 30.01%, 11.38%, and 16.87%, respectively. The forgetting metric also showed a substantial improvement, with BufferGA leading to a decrease of 21.85%, 22.29%, and 24.85%. These impressive results demonstrate that BufferGA effectively mitigates the forgetting problem in CFL and enhances the performance of existing methods.

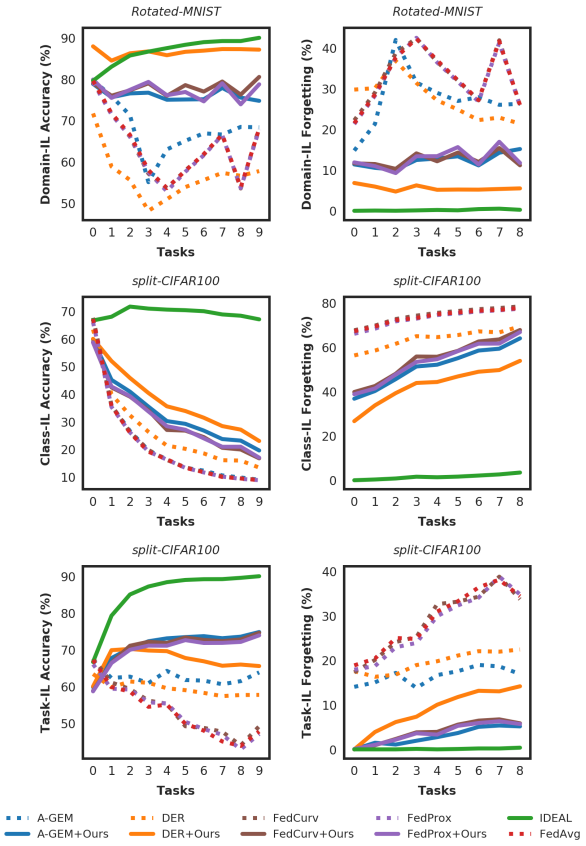

Figure 3. Evaluating accuracy and forgetting in rotated-MNIST and split-CIFAR100 with and without BufferGA using a buffer size of 200. The solid lines indicate the results obtained with our method or upper bound, while the dotted lines represent the results obtained without our method. The experimental results demonstrate a significant improvement in accuracy and reduction in forgetting for all settings.

We conducted an experiment to investigate the impact of buffer size on the performance of our proposed method. In this experiment, we used a larger buffer size of 5120. For vanilla replay-based baselines, namely A-GEM and DER, as shown by the blue and orange dotted lines in each subfigure. Compared to Figure 3, Figure 4 with a larger buffer size exhibits more consistent and smoother lines with smaller variances, slightly higher accuracy, and less forgetting. This indicates that a larger buffer size leads to improved performance of vanilla baseline methods as it better preserves the knowledge acquired by individual clients.

Moreover, by comparing the solid lines (with BufferGA) between the two figures, we observe a significant increase

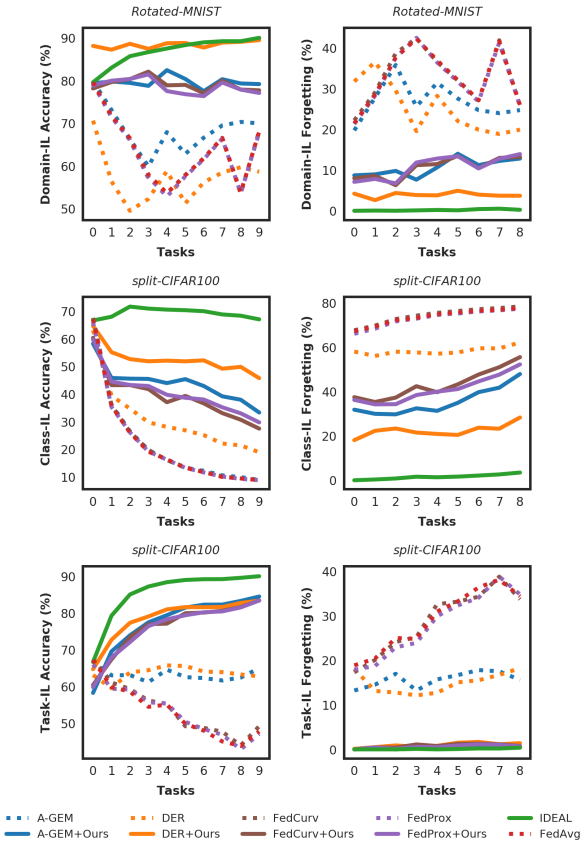

Figure 4. Comparison of rotated-MNIST and split-CIFAR100 with a larger buffer size of 5120 for each client. As the buffer size increases, we observe significant improvements in both accuracy and forgetting, highlighting the benefits of using a larger buffer size.

in class-IL and task-IL accuracy, and a decrease in forgetting. In particular, for task-IL, we found that a larger buffer size can almost eliminate forgetting, which demonstrates the benefits of BufferGA. The improved performance of BufferGA becomes more pronounced in the presence of a larger buffer, resulting in even better performance compared to the baseline methods.

5.2 Object Detection

For a comprehensive evaluation of methods, we implement methods 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 Figure 5, CARLA provides OpenFL with a real-time collection of continuous streaming vehicle camera output data and automatic annotation about object detection. This streaming data

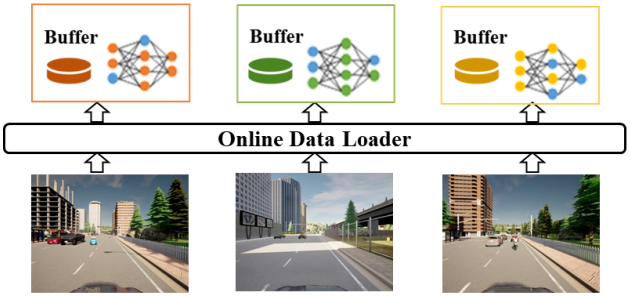

Figure 5. Online data loader provides continuous data from CARLA camera outputs to federated learning collaborators. Each collaborator trains on its local data and updates its buffer to attain old knowledge.

captures the spatio-temporal dynamics of data generated from real-world applications. After loading data of vehicles from CARLA, OpenFL performs collaborative training and communication.

We access the solutions to forgetting problems by spawning two vehicles in the same virtual world (Town 5) and repeat experiments by placing vehicles in different environments (Town 3 and Town 5). While training 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 mean average precision (mAP). We measure the mAP by analyzing the prediction results on pre-collected driving snippets from each individual vehicle client. Those driving snippets are pre-collected via traveling around the town for a long time (3000 simulation seconds).

For those experiments on realistic CARLA streaming data, we consider baselines, *IDEAL*, FedAvg, DER and DER+BufferGA. 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 Figure 6.

First, we observe an obvious catastrophic forgetting problem - the FedAvg curve suddenly goes down around the $60th$ round, considered as a degradation of the learned knowledge. We investigate the dataset and notice the client in Town 5 gets on the highway at $60th$ round, suggesting a domain shift occurred on this client at round 60. As can be seen in Figure 6, the results of DER with BufferGA outperform alternative methods.

In the case of clients being in the same town, our proposed method shows outstanding performance, achieving high

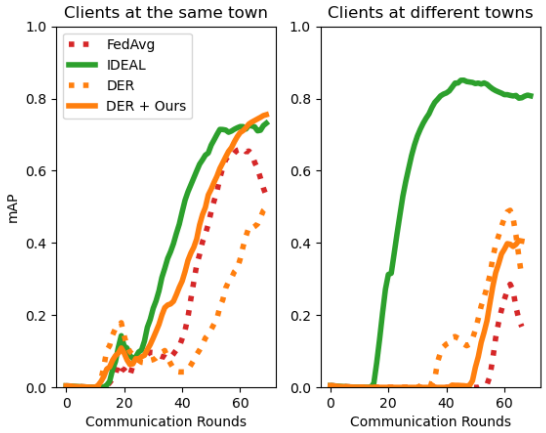

Figure 6. The object detection performance of *IDEAL*, FedAvg, DER and DER+BufferGA by spawning vehicles in the same (Town 5) or different towns (Town 3 and Town 5) respectively.

mAP detections that are close to the upper bound performance, where vehicles store all seen data. Moreover, the DER algorithm with BufferGA converges faster compared to the performance of the lower bound algorithm, FedAvg, and even the standard DER algorithm itself. On the other hand, when clients are in different towns, our proposed method, DER+BufferGA, outperforms FedAvg. Although BufferGA has a slightly lower mAP, it greatly reduces the catastrophic forgetting problem on top of the standard DER algorithm.

6 Conclusion

In this paper, we present BufferGA, a novel approach to continual federated learning that builds on buffer-based techniques and leverages aggregated buffer gradients across clients to mitigate forgetting. We evaluate the effectiveness of our approach through experiments conducted under various continual learning scenarios using benchmark datasets such as rotated-MNIST and split-CIFAR100, as well as a realistic streaming automotive dataset generated using the CARLA simulator and the Open FL framework. Our experimental results demonstrate that BufferGA outperforms existing state-of-the-art methods in terms of accuracy and forgetting. These findings highlight the potential of our method for real-world applications of continual federated learning.

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.

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.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

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.

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.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

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.

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.

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 CONTINUAL LEARNING METHODS WITH BUFFERGA

We provide detailed information with respect to the `RefineGradient`in Algorithm 2, including two examples of continual learning methods with BufferGA.

Algorithm 5 incorporates DER into the local update on client $k \in [K]$. After receiving globle model w_{t-1} from the server at time t, client k computes the output logits/pre-softmax response z. Note that $f(x, w) := \sigma(h(x, w))$ and we refer $h(x, w)$ as the model logits z. Though similar to other replay-based methods, client has more information as they sample past data (x', y') as well as logits z' from the buffer $\mathcal{M}^k$. Regularization term considers the Euclidean distance between corresponding output logits. Gradient g is refined with this Regularization. The following steps are the same as in the main text.

Algorithm 6 applied Gradient Adjustment twice. First, client computes gradient g_c w.r.t new data from $\mathcal{D}_t^k$. After replaying previous samples (x', y') in the local buffer $\mathcal{M}^k$, client computes the gradient g_b w.r.t this buffer data. If those gradients violate in terms of their direction, client projects g_c onto g_b in order to remove interference.

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

Input: task index t, current dataset $\mathcal{D}_t^k$, global model w, local buffer $\mathcal{M}^k$, global buffer gradient g_{ref}, learning rate α and number of seen samples $n = 0$
for $(x, y) \in \mathcal{D}_t^k$ **do**
  $z \leftarrow h(x, w)$ where $f(x, w) := \sigma(h(x, w))$
  $(x', z', y') \leftarrow \mathcal{M}^k$
  $\ell_{\text{reg}} \leftarrow \lambda \|z' - h(x', w)\|_2^2$
  $g = \nabla_w [\ell(y, f(x, w)) + \ell_{\text{reg}}]$
  if $g^T g_{\text{ref}} \geq 0$ or g_{ref} is None **then**
    $\tilde{g} \leftarrow g$
  else
    $\tilde{g} \leftarrow g - \frac{g^T g_{\text{ref}}}{g_{\text{ref}}^T g_{\text{ref}}} g_{\text{ref}}$
  end if
  $w \leftarrow w - \alpha \tilde{g}$
  $\mathcal{M}^k \leftarrow$ `Reservoir Sampling`$(\mathcal{M}^k, (x, z, y), n)$
  $n \leftarrow n + 1$
end for
Return w to server

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

Input: task index t, current dataset $\mathcal{D}_t^k$, global model w, local buffer $\mathcal{M}^k$, global buffer gradient g_{ref}, learning rate α and number of seen samples $n = 0$
for $(x, y) \in \mathcal{D}_t^k$ **do**
  $g_c = \nabla_w \left[\ell(y, f(x, w))\right]$
  $(x', y') \leftarrow \mathcal{M}^k$
  $g_b = \nabla_w \left[\ell(y', f(x', w))\right]$
  if $g_c^T g_b \geq 0$ **then**
    $g \leftarrow g_c$
  else
    $g \leftarrow g_c - \frac{g_c^T g_b}{g_b^T g_b} g_b$
  end if
  if $g^T g_{\text{ref}} \geq 0$ or g_{ref} is None **then**
    $\tilde{g} \leftarrow g$
  else
    $\tilde{g} \leftarrow g - \frac{g^T g_{\text{ref}}}{g_{\text{ref}}^T g_{\text{ref}}} g_{\text{ref}}$
  end if
  $w \leftarrow w - \alpha \tilde{g}$
  $\mathcal{M}^k \leftarrow$ `Reservoir Sampling`$(\mathcal{M}^k, (x, y), n)$
  $n \leftarrow n + 1$
end for
Return w to server