An Effective Dynamic Gradient Calibration Method for Continual Learning

Weichen Lin [*1] **Jiaxiang Chen** [*1] **Ruomin Huang** [2] **Hu Ding** [3]

Abstract

Continual learning (CL) is a fundamental topic in machine learning, where the goal is to train a model with continuously incoming data and tasks. Due to the memory limit, we cannot store all the historical data, and therefore confront the "catastrophic forgetting" problem, i.e., the performance on the previous tasks can substantially decrease because of the missing information in the latter period. Though a number of elegant methods have been proposed, the catastrophic forgetting phenomenon still cannot be well avoided in practice. In this paper, we study the problem from the gradient perspective, where our aim is to develop an effective algorithm to calibrate the gradient in each updating step of the model; namely, our goal is to guide the model to be updated in the right direction under the situation that a large amount of historical data are unavailable. Our idea is partly inspired by the seminal stochastic variance reduction methods (e.g., SVRG and SAGA) for reducing the variance of gradient estimation in stochastic gradient descent algorithms. Another benefit is that our approach can be used as a general tool, which is able to be incorporated with several existing popular CL methods to achieve better performance. We also conduct a set of experiments on several benchmark datasets to evaluate the performance in practice.

1. Introduction

In the past years, Deep Neural Networks (DNNs) demonstrate remarkable performance for many different tasks in artificial intelligence, such as image generation (Ho et al., 2020; Goodfellow et al., 2014a), classification (Liu et al., 2021; He et al., 2016), and pattern recognition (Bai et al., 2021; Zhu et al., 2016). Usually we assume that the whole training data is stored in our facility and the DNN models can be trained offline by using Stochastic Gradient Descent (SGD) algorithms (Bottou et al., 1991; 2018). However, real-world applications often require us to consider training lifelong models, where the tasks and data are accumulated in a streaming fashion (Van de Ven & Tolias, 2019; Parisi et al., 2019). For example, with the popularity of smart devices, a large amount of new data is generated every day. A model needs to make full use of these new data to improve its performance while keeping old knowledge from being forgotten. Those applications motivate us to study the problem of *continual learning (CL)* (Kirkpatrick et al., 2017; Li & Hoiem, 2017), where its goal is to develop effective method for gleaning insights from current data while retaining information from prior training data.

A significant challenge that CL encounters is "*catastrophic forgetting*" (Kirkpatrick et al., 2017; McCloskey & Cohen, 1989; Goodfellow et al., 2014b), wherein the exclusive focus on the current set of examples could result in a dramatic deterioration in the performance on previously learned data. This phenomenon is primarily attributed to limited storage and computational resources during the training process; otherwise, one could directly train the model from scratch using all the saved data. To address this issue, we need to develop efficient algorithm for training neural networks from a continuous stream of non-i.i.d. samples, with the goal of mitigating catastrophic forgetting while effectively managing computational costs and memory footprint.

A number of elegant CL methods have been proposed to alleviate the catastrophic forgetting issue (Wang et al., 2023; De Lange et al., 2021; Mai et al., 2022). One representative CL approach is referred to as "Experience Replay (ER)" (Ratcliff, 1990; Chaudhry et al., 2019b), which has shown promising performance in several continual learning scenarios (Prabhu et al., 2023; Arani et al., 2022; Farquhar & Gal, 2018). Roughly speaking, the ER method utilizes reservoir sampling (Vitter, 1985) to maintain historical data in the buffer, then extract new incoming training data with random samplings for learning the current task. Though the intuition is simple, the ER method currently is one of the most popular CL approaches that incurs moderate computational and storage demands. Moreover, several recently proposed

[*]Equal contribution [1]School of Data Science, University of Science and Technology of China, Anhui, China. [2]Duke University [3]School of Computer Science and Technology, University of Science and Technology of China, Anhui, China. Correspondence to: Hu Ding <huding@ustc.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

approaches suggest that the ER method can be combined with knowledge distillation to further improve the performance; for example, the methods of DER/DER++ (Buzzega et al., 2020) and X-DER (Boschini et al., 2022) preserve previous training samples alongside their logits in the model as the additional prior knowledge. Besides the ER methods, there are also several other types of CL techniques proposed in recent years, and please refer to Section 1.2 for a detailed introduction.

1.1. Our Main Ideas and Contributions

Though existing CL methods can alleviate the catastrophic forgetting issue from various aspects, the practical performances in some scenarios are still not quite satisfying (Yu et al., 2023; Tiwari et al., 2022; Ghunaim et al., 2023). In this paper, we study the continual learning problem from the gradient perspective, and the rationality behind is as follows. In essence, an approach for avoiding the catastrophic forgetting issue in CL, e.g., the replay mechanisms or the regularization strategies, ultimately manifests its influence on the gradient directions during model updating (Wang et al., 2023). If all the historical data are available, one could compute the gradient by using the stochastic gradient descent method and obviously the catastrophic forgetting phenomenon cannot happen. The previous methodologies aim to approximate the gradient by preserving additional information and incorporating it as a constraint to model updates, thereby retaining historical knowledge. However, the replay-based methods in practice are often limited by storage capacity, which leads to a substantial loss of historical data information and inaccurate estimation of historical gradients (Yan et al., 2021). Therefore, our goal is to develop a more accurate gradient calibration algorithm in each step of the continual learning procedure, which can directly enhance the training quality.

We are aware of several existing CL methods that also take into account of the gradients (Liu & Liu, 2022; Tiwari et al., 2022; Farajtabar et al., 2020), but our idea proposed here is fundamentally different. We draw inspiration from the seminal *stochastic variance reduction* methods (e.g., SVRG from Johnson & Zhang (2013) and SAGA from Defazio et al. (2014)), which are originally designed to reduce the gradient variance so that the estimated gradient can closely align with the true full gradient over the entire dataset (including the current and historical data). These variance reduction methods have been extensively studied in the line of the research on stochastic gradient descent method (Jin et al., 2019; Babanezhad Harikandeh et al., 2015; Lei et al., 2017); their key idea is to leverage the additional saved full gradient information to calibrate the gradient in the current training step, which leads to significantly reduced gradient variance comparing with the standard SGD method. This intuition also inspires us to handle the CL problem. In a standard SGD method, the variance between the obtained gradient and the full gradient is due to the "batch size limit" (if the batch size has no bound, we can simply compute the full gradient). Recall that the challenge of CL is due to the "buffer size limit", which impedes the use of full historical data (this is similar with the dilemma encountered by SGD with the "batch size limit"). So an interesting question is

Can the calibration idea for "batch size limit" be modified to handle "buffer size limit"? Specifically, is it possible to develop an effecitve method to compute a SVRG (or SAGA)-like calibration for the gradient in CL scenarios?

Obviously, it is challenging to directly implement the SVRG or SAGA algorithms in continual learning because of the missing historical data. Note that Frostig et al. (2015) proposed a streaming SVRG (SSVRG) method that realizes the SVRG method within a given fixed buffer, but unfortunately it does not perform quite well in the CL scenarios (as shown in our experimental section). One possible reason is that SSVRG can only leverage information within the buffer and fails to utilize all historical information.

In this paper, we aim to apply the intuition of SVRG to handle the "buffer size limit" in CL, and our contributions can be summarized as follows:

- First, we propose a novel two-level dynamic algorithm, named **Dynamic Gradient Calibration (DGC)**, to maintain a gradient calibration in continual learning. DGC can effectively tackle the storage limit and leverage historical data to calibrate the gradient of the model at each current stage. Moreover, our theoretical analysis shows that our DGC based CL algorithm can achieve a linear convergence rate.

- Second, our method can be conveniently integrated with most existing reservoir sampling-based continual learning approaches (e.g., ER (Ratcliff, 1990), DER/DER++ (Buzzega et al., 2020), XDER (Boschini et al., 2022), and Dynamic ER (Yan et al., 2021)), where this hybrid framework can induce a significant enhancement to the overall model performance. Note that storing the gradient calibrator can cause an extra memory footprint; so a key question is whether such an extra memory footprint can yield a larger marginal benefit than simply taking more samples to fill the extra memory? In Fig 1, we illustrate a brief example to answer this question in the affirmative; more detailed evaluations are shown in the experimental part.

- Finally, we conduct a set of comprehensive experiments on the popular datasets S-CIFAR10, S-CIFAR100 and S-TinyImageNet; the experimental results suggest that our method can improve the final Average Incremental Accuracy (FAIA) in several CL

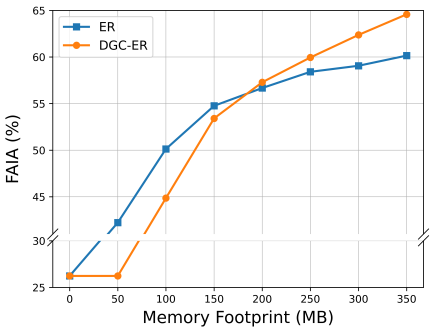

Figure 1. Performance comparison of the ER method and our proposed DGC-ER method with increasing storage memory size on CIFAR100. The x-axis denotes the actual memory footprint (MB) and the y-axis denotes the final Average Incremental Accuracy (FAIA) (Hou et al., 2019; Douillard et al., 2020) (the formal definition is shown in Section 4). The curves in the figure indicate that although ER outperforms DGC-ER at low MB, as MB increases, the dynamic gradient calibration method can achieve a larger marginal benefit than simply increasing the sample size of ER.

scenarios by more than 6%. Moreover, our improvement for a larger number of tasks is more significant than that for a smaller number. Furthermore, our adoption of the SVRG-inspired DGC calibration method leads to enhanced stability in minimizing the loss function throughout the parameter optimization process.

1.2. Related Work

We briefly overview existing important continual learning approaches (except for the ones mentioned before). We also refer the reader to the recent surveys (Wang et al., 2023; De Lange et al., 2021; Mai et al., 2022) for more details.

A large number of CL methods are replay based, where they often keep a part of previous data through approaches like reservoir sampling (Chaudhry et al., 2019a; Riemer et al., 2019). Several more advanced data selection strategies focus on optimizing the factors like the sample diversity of parameter gradients or the similarity to previous gradients on passed data, e.g., GSS (Aljundi et al., 2019b) and GCR (Tiwari et al., 2022). Experience replay can be effectively combined with knowledge distillation. For example, Hu et al. (2021) proposed to distill colliding effects from the features for new coming tasks, and ICARL (Rebuffi et al., 2017) proposed to take account of the data representation trained on old data. MOCA (Yu et al., 2023) improves replay-based methods by diversifying representations in the space. Another replay-based approach is based on generative replay, which obtains replay data by generative models (Shin et al., 2017; Gao & Liu, 2023; Wu et al., 2018).

Another way for solving continual learning is through some deliberately designed optimization procedures. For example, the methods GEM (Lopez-Paz & Ranzato, 2017), AGEM (Chaudhry et al., 2019a), and MER (Riemer et al., 2019) restrict parameter updates to align with the experience replay direction, and thereby preserve the previous input and gradient space with old training samples. Different from saving old training samples, Farajtabar et al. (2020) proposed to adapt parameter updates in the orthogonal direction of the previously saved gradient. The method AOP (Guo et al., 2022) projects the gradient in the direction orthogonal to the subspace spanned by all previous task inputs, therefore it only keeps an orthogonal projector rather than storing previous data.

To mitigate the problem of forgetting, we can also augment the model capacity for learning new tasks. Xu & Zhu (2018) tried to enhance model performance by employing meta-learning techniques when dynamically extending the model. The method ANCL (Kim et al., 2023) proposes to utilize an auxiliary network to achieve a trade-off between plasticity and stability. Dynamic ER (Yan et al., 2021) introduces a novel two-stage learning method that employs a dynamically expandable representation for learning knowledge incrementally.

2. Preliminaries

We consider the task of training a soft-classification function $\boldsymbol{f(\cdot;\theta)}$: $\boldsymbol{\mathcal{X} \to \mathcal{Y}}$, where $\mathcal{X}$ and $\mathcal{Y}$ respectively represent the space of data and the set of labels, and θ is the parameter to optimize. Without loss of generality, we assume $\mathcal{Y} = \{1, 2, \cdots, K\}$. So $f(\cdot;\theta)$ maps each $x \in \mathcal{X}$ to some $f(x;\theta) \in \mathbb{R}^K$. To find an appropriate θ, the classification function f is usually equipped with a loss function $\ell(f(x;\theta), y)$, which is differentiable for the variables x and θ (e.g., cross-entropy loss). To simplify the notation, we use $\ell(x, y, \theta)$ to denote $\ell(f(x;\theta), y)$. Given a set of data $P = \{(x^i, y^i) \mid 1 \le i \le n\} = \mathcal{X}_P \times \mathcal{Y}_P \subset \mathcal{X} \times \mathcal{Y}$, the training process is to find a θ such that the empirical risk of $\ell(x, y, \theta)$, i.e., $\sum_{i=1}^n \ell(x^i, y^i, \theta)$, is minimized. We define the full gradient of $\ell(x, y, \theta)$ on P as

$$\mathcal{G}(P, \theta) \triangleq \nabla_\theta \ell(\mathcal{X}_P, \mathcal{Y}_P, \theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \ell(x^i, y^i, \theta). \quad (1)$$

2.1. Continual Learning Models

In this paper, we focus on two popular CL models: *Class-Incremental Learning (**CIL**)* (Hsu et al., 2018) and *Task-Free Continual Learning (**TFCL**)* (Aljundi et al., 2019a).

In the setting of CIL, the training tasks come in a sequence $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_T\}$ with disjoint label space; each time spot $t \in \{1, 2, \cdots, T\}$ corresponds to the task $\mathcal{T}_t$ with a training dataset $\{(x_t^i, y_t^i) \mid 1 \le i \le n_t\}$. With a slight abuse of notations, we also use $\mathcal{T}_t$ to denote its training dataset. Also, we use $\mathcal{Y}_t$ to denote the corresponding set of labels $\{y_t^i \mid$

$1 \leq i \leq n_t\}$. Although the training data for individual tasks $\mathcal{T}_t$ is independently and identically distributed (i.i.d.), it is worth noting that the overall task stream $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_T\}$ does not adhere to the i.i.d. assumption due to the evolving label space over time. We define the overall risk under this CL setting at the current time spot t as follows:

$$\ell_{CL}^t(\theta) \triangleq \frac{1}{t}\sum_{c=1}^{t} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{T}_c}[\ell(x, y, \theta)]. \tag{2}$$

The setting of TFCL is similar to CIL, where the major difference is that the task identities are not provided in neither the training nor testing procedures. So the TFCL setting is more challenging than CIL because the algorithm is unaware of task changes and the current task identity. In the main part of our paper, we present our results in CIL; in our supplement, we explain how to extend our results to TFCL.

2.2. Variance Reduction Methods

A comprehensive introduction to variance reduction methods is provided in (Gower et al., 2020). Here we particularly introduce one representative variance reduction method **SVRG** (Johnson & Zhang, 2013) , which is closely related to our proposed CL approach.

The high-level idea of SVRG is to construct a calibration term to reduce the variance in the gradient estimate. The complete optimization process can be segmented into a sequence of stages. We denote the training data as $P = \{(x^i, y^i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$, and denote the parameter at the beginning of each stage as $\tilde{\theta}$. The key part of SVRG is to minimize the variance in SGD optimization by computing an additional term $\tilde{\mu}$:

$$\tilde{\mu} \triangleq \mathcal{G}(P, \tilde{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \ell(x^i, y^i, \tilde{\theta}). \tag{3}$$

In each stage, SVRG applies the standard SGD with the term $\tilde{\mu}$ to update the parameter θ: randomly sample an index i_k from $\{1, 2, ..., n\}$, and let

$$v^k = \nabla_\theta \ell(x^i, y^i, \theta^k) - (\nabla_\theta \ell(x^{i_k}, y^{i_k}, \tilde{\theta}) - \tilde{\mu}), \tag{4}$$

$$\theta^{k+1} = \theta^k - \eta v^k, \tag{5}$$

where $(x^i, y^i) \in P$ is the sampled training data, θ^k represents the parameter at the k-th step of SGD, and η is the learning rate. Since $\mathbb{E}[\nabla_\theta \ell(x^{i_k}, y^{i_k}, \tilde{\theta^k})] = \tilde{\mu}$, v^k is an unbiased estimate of the gradient $\mathcal{G}(P, \theta^k)$. Subsequently, the term "$\nabla_\theta \ell(x^{i_k}, y^{i_k}, \tilde{\theta}) - \tilde{\mu}$" in Eq (4) can be regarded as a **calibrator** to reduce the variance of gradient estimation and achieve a linear convergence rate (Johnson & Zhang, 2013), which is faster than directly using $\nabla_\theta \ell(x^i, y^i, \theta^k)$.

3. Our Proposed Method

In this section, we propose the Dynamic Gradient Calibration (DGC) approach which maintains a gradient calibration during the learning process. A highlight of DGC is that it utilizes the whole historical information to obtain a more precise gradient estimation, and consequently relieves the negative impact of catastrophic forgetting. In Section 3.1, we introduce ER and analyze the obstacle if we directly combine ER with SVRG. In Section 3.2, we present our DGC method for addressing the issues discussed in Section 3.1. In Section 3.3, we explain how to integrate DGC with other CL techniques.

3.1. Experience Replay Revisited and SVRG

First, we overview the classical ER (Ratcliff, 1990; Chaudhry et al., 2019b) method as a baseline for the CIL setting. ER employs the reservoir sampling algorithm to dynamically manage a **buffer** (denoted as $\boldsymbol{\mathcal{M}_t}$) at time t, which serves to store historical data. At each time spot t (and assume the current updating step number of the optimization is k), ER updates the model parameter θ_t^k following the standard gradient descent method:

$$\theta_t^{k+1} = \theta_t^k - \eta \cdot v_t^k, \tag{6}$$

where η is the learning rate and v_t^k is the calculated gradient. If not using any replay strategy, v_t^k is usually calculated on a randomly sampled training data $(x_t^k, y_t^k) \in \mathcal{T}_t$. It is easy to see that this simple strategy can cause the forgetting issue for shifting data stream since it does not contain any information from the previous data. Hence the classical ER algorithm takes a random sample $(\bar{x}_t^k, \bar{y}_t^k)$ from the aforementioned buffer $\mathcal{M}_t$ (who contains a subset of historical data via reservoir sampling), and computes the gradient:

$$v_t^k = \frac{1}{t}\nabla_\theta \ell(x_t^k, y_t^k, \theta_t^k) + \frac{t-1}{t}\nabla_\theta \ell(\bar{x}_t^k, \bar{y}_t^k, \theta_t^k). \tag{7}$$

Remark 3.1. **(1)** For simplicity, we assume that the data sets of all the tasks have the same size. So the obtained v_t^k in (7) is an unbiased estimation of the full gradient $\mathcal{G}(\bigcup_{c=1}^t \mathcal{T}_c, \theta)$ at the current time spot t. If they have different sizes, we can simply replace the coefficients "$\frac{1}{t}$" and "$\frac{t-1}{t}$" by "$\frac{|\mathcal{T}_t|}{\sum_{c=1}^t |\mathcal{T}_c|}$" and "$\frac{\sum_{c=1}^{t-1} |\mathcal{T}_c|}{\sum_{c=1}^t |\mathcal{T}_c|}$", respectively. **(2)** Also, we assume that the batch sizes of the random samples from $\mathcal{T}_t$ and $\mathcal{M}_t$ are both "1", i.e., we only take single item (x_t^k, y_t^k) and $(\bar{x}_t^k, \bar{y}_t^k)$ from each of them. Actually, we can also take larger batch sizes and then the Eq (7) can be modified correspondingly by taking their average gradients.

Now we attempt to apply the SVRG method to Eq (7). Our objective is to identify a more accurate unbiased estimate of the gradient at the current time spot t so as to determine the updating direction. At first glance, one possible solution is

to adapt the streaming SVRG method (Frostig et al., 2015) to the CL scenario. We treat $\mathcal{M}_t$ as the static data set in our memory, and apply the SVRG technique to calibrate the gradient "$\nabla_\theta \ell(x_t^k, y_t^k, \theta_t^k)$" and "$\nabla_\theta \ell(\bar{x}_t^k, \bar{y}_t^k, \theta_t^k)$" in Eq (7). Similar to the procedure introduced in Section 2.2, we denote the parameter at the beginning of the current stage s as $\tilde{\theta}_{t,s}$. For simplicity, when we consider the update within stage s, we just use $\tilde{\theta}_t$ to denote $\tilde{\theta}_{t,s}$. Similar with Eq (3), we define the terms

$$\tilde{\mu} \triangleq \mathcal{G}(\mathcal{M}_t, \tilde{\theta}_t), \quad \tilde{v} \triangleq \mathcal{G}(\mathcal{T}_t, \tilde{\theta}_t). \tag{8}$$

Then we can calibrate the gradient by a randomly sampled $(\dot{x}_t, \dot{y}_t)$ from $\mathcal{T}_t$ and $(\ddot{x}_t, \ddot{y}_t)$ from $\mathcal{M}_t$ (which serves as the similar role of (x^{i_k}, y^{i_k}) in (4)); the new form of v_t^k becomes

$$v_t^k = \frac{1}{t}\left(\nabla_\theta \ell(x_t, y_t, \theta_t^k) - \nabla_\theta \ell(\dot{x}_t, \dot{y}_t, \tilde{\theta}_t) + \tilde{v}\right) + \frac{t-1}{t}\left(\nabla_\theta \ell(\bar{x}_t, \bar{y}_t, \theta_t^k) - \nabla_\theta \ell(\ddot{x}_t, \ddot{y}_t, \tilde{\theta}_t) + \tilde{\mu}\right). \tag{9}$$

Obviously, if the buffer $\mathcal{M}_t$ contains the whole historical data (denote by $\mathcal{T}_{[1:t)} = \bigcup_{c=1}^{t-1} \mathcal{T}_c$), the above approach is exactly the standard SVRG. However, because $\mathcal{M}_t$ only takes a small subset of $\mathcal{T}_{[1:t)}$, this approach still cannot avoid information loss for the previous tasks. In next section, we propose a novel two-level dynamic algorithm to record more useful information from $\mathcal{T}_{[1:t)}$, and thereby reduce the information loss induced by $\mathcal{M}_t$. We also take the approach of Eq (9) as a baseline in Section 4 to illustrate the advantage of our proposed approach.

3.2. Dynamic Gradient Calibration

To tackle the issue discussed in Section 3.1, we propose a novel two-level update approach "Dynamic Gradient Calibration (DGC)" to maintain our calibration term. Our focus is designing a method to incrementally update an unbiased estimation for $\mathcal{G}(\mathcal{T}_{[1:t)}, \theta_t^k)$. To illustrate our idea clearly, we decompose our analysis to two levels: **(1)** update the parameter during the training within each time spot t; **(2)** update the parameter at the transition from time spot t to $t+1$ (i.e., the moment that the task $\mathcal{T}_t$ has just been completed and the task $\mathcal{T}_{t+1}$ is just coming).

(1) How to update the parameter during the training within each time spot t. We follow the setting of the streaming SVRG as discussed in Section 3.1: the training process at the current time spot t is divided into a sequence of stages; the model parameter at the beginning of each stage is recorded as $\tilde{\theta}_t$. To illustrate our idea for calibrating the gradient v_t^k in Eq (9), we begin by considering an "imaginary" approach: we let

$$v_t^k = \frac{1}{t}\left(\nabla_\theta \ell(x_t, y_t, \theta_t^k) - \nabla_\theta \ell(\dot{x}_t, \dot{y}_t, \tilde{\theta}_t) + \mathcal{G}(\mathcal{T}_t, \tilde{\theta}_t)\right) + \frac{t-1}{t}\Gamma, \tag{10}$$

where

$$\Gamma = \nabla_\theta \ell(\bar{x}_t, \bar{y}_t, \theta_t^k) - \underbrace{\left(\nabla_\theta \ell(\ddot{x}_t, \ddot{y}_t, \tilde{\theta}_t) - \mathcal{G}(\mathcal{T}_{[1:t)}, \tilde{\theta}_t)\right)}_{\text{Calibration from the previous parameter } \tilde{\theta}_t}.$$

Different from Eq (9), we compute v_t^k based on the full historical data $\mathcal{T}_{[1:t)}$, which follows the same manner of SVRG. However, a major obstacle here is that we cannot obtain the exact Γ since $\mathcal{T}_{[1:t)}$ is not available. This motivates us to design a relaxed form of (10). We define a surrogate function to approximate Γ, which can be computed through recursion. Suppose each training stage has $m \geq 1$ steps, then we define

$$\Gamma_{\text{DGC}}(t,k) = \nabla_\theta \ell(\bar{x}_t, \bar{y}_t, \theta_t^k) - \left(\nabla_\theta \ell(\ddot{x}_t, \ddot{y}_t, \tilde{\theta}_t) - \Gamma'_{\text{DGC}}(t)\right), \tag{11}$$

$$\Gamma'_{\text{DGC}}(t) = \Gamma_{\text{DGC}}(t, m+1), \tag{12}$$

in the stage. Note that the term "$\nabla_\theta \ell(\ddot{x}_t, \ddot{y}_t, \tilde{\theta}_t)$" in (11) can be computed by the previous parameter $\tilde{\theta}_t$ during training, so we do not need to store it in buffer. For the initial $t = 1$ case (i.e., when we just encounter the first task), we can directly set $\Gamma'_{\text{DGC}}(1) = \vec{0}$. We update the function $\Gamma'_{\text{DGC}}(t)$ at the end of each training stage in (12), and use the function $\Gamma_{\text{DGC}}(t,k)$ to approximate Γ in (10). Comparing with the original formulation of Γ in (10), we only replace the term "$\mathcal{G}(\mathcal{T}_{[1:t)}, \tilde{\theta}_t)$" by "$\Gamma'_{\text{DGC}}(t)$". Also, we have the following lemma to support this replacement. The detailed proof of lemma 3.2 is provided in our supplement.

Lemma 3.2.

$$\mathbb{E}\left[\Gamma'_{DGC}(t)\right] = \mathcal{G}(\mathcal{T}_{[1:t)}, \tilde{\theta}_t) \tag{13}$$

We utilize the term "$\nabla_\theta \ell(\ddot{x}_t, \ddot{y}_t, \tilde{\theta}_t) - \Gamma'_{\text{DGC}}(t)$" of (11) as the calibrator for each updating step, thereby preserving the unbiased nature of the gradient estimator and reducing the variance of gradient estimation.

(2) How to update the parameter at the transition from time spot t to $t+1$. At the end of time spot t, we update the recorded $\tilde{\theta}_t$ to θ_t^{m+1}, and the data $\mathcal{T}_t$ from time t should be integrated into the historical data. In this context, it is essential to update the calibrated gradient accordingly:

$$\Gamma'_{\text{DGC}}(t+1) = \frac{1}{t}\left((t-1)\cdot\Gamma'_{\text{DGC}}(t) + \mathcal{G}(\mathcal{T}_t, \tilde{\theta}_t)\right). \tag{14}$$

The complete algorithm is presented in Algorithm 1. Compared with the conventional reservoir sampling based approaches, we only require the additional storage for keeping

$\Gamma'_{\text{DGC}}(t)$, and so that the gradient $\Gamma_{\text{DGC}}(t,k)$ in (11) can be effectively updated by the recursion. Moreover, our method can also conveniently adapt to TFCL where the task boundaries are not predetermined. In such a setting, we can simply treat each batch data (during the SGD) as a "micro" task at the time point and then update the gradient estimation via Eq (14). The detailed algorithm for TFCL is placed in our appendix.

Similar to the theoretical analysis of SVRG (Johnson & Zhang, 2013), under mild assumptions, the optimization procedure of our DGC method at each time spot t can also achieve linear convergence. We denote the optimal parameter at time spot t as $\theta_* \triangleq \arg\min_\theta \ell^t_{\text{CL}}(\theta)$. Then we have the following theorem.

Theorem 3.3. *Assume that $f(x;\theta)$ is L-smooth and γ-strongly convex; the parameters $m \geq \frac{10L^2}{\gamma^2}$ and $\eta = \frac{\gamma}{10L}$. Then we have a linear convergence in expectation for the DGC procedure at time t:*

$$\mathbb{E}\Big[\Big\|\tilde{\theta}_{t,s+1} - \theta_*\Big\|_2^2\Big] \leq \frac{1}{2^s}\mathbb{E}\Big[\Big\|\tilde{\theta}_{t,1} - \theta_*\Big\|_2^2\Big]$$

where $\tilde{\theta}_{t,s}$ represents the initialization parameter at the beginning of the s-th stage at time spot t.

The proof of Theorem 3.3 is provided in appendix. This theorem indicates that the gradient calibrated by our DGC method shares the similar advantages with SVRG. For instance, when updating each task $\mathcal{T}_t$, the loss function has a smoother decrease (we validate this property in Section 4.3).

3.3. Combine DGC with Other CL Methods

Our proposed DGC approach can be also efficiently combined with other CL methods. As discussed in Section 1, a number of popular CL methods rely on the reservoir sampling technique to preserve historical data in buffer (Buzzega et al., 2020; Boschini et al., 2022; Riemer et al., 2019). For these methods, such as DER and XDER, we can conveniently combine the conventional batch gradient descent with the DGC calibrated gradient estimator $\Gamma_{\text{DGC}}(t,k)$ defined in Section 3.2, so as to obtain a more precise gradient estimator with reduced variance based on Eq (10):

$$\begin{aligned} v_t^k \;\; &= \tfrac{1}{t}\Big(\nabla_\theta \ell(x_t, y_t, \theta_t^k) - \nabla_\theta(\ddot{x}_t, \ddot{y}_t, \tilde{\theta}_t) + \mathcal{G}(\mathcal{T}_t, \tilde{\theta}_t)\Big) + \\ &\quad \tfrac{t-1}{t}\Big[\alpha\Gamma_{\text{DGC}}(t,k) + (1-\alpha)\nabla_\theta \ell(\bar{x}_t, \bar{y}_t, \theta_t^k)\Big], \end{aligned} \tag{15}$$

where α is a given parameter to control the proportion of the two unbiased estimations $\Gamma_{\text{DGC}}(t,k)$ and $\nabla_\theta \ell(\bar{x}_t, \bar{y}_t, \theta_t^k)$ of the gradient $\mathcal{G}(\mathcal{T}_{[1:t)}, \theta_t^k)$. According to the theoretical analysis in SSVRG (Frostig et al., 2015), the selection of α should be related to $1/L$, where L is the smoothness coefficient of the model $f(x;\theta)$, i.e.,

$$L = \max_{\theta_1,\theta_2 \in \Theta} \frac{|\nabla_\theta f(x;\theta_1) - \nabla_\theta f(x;\theta_2)|}{|\theta_1 - \theta_2|}, \tag{16}$$

where Θ represents the parameter space. The experimental study on the impact of α is placed in our supplement. In Section 4, we show that the amalgamation of the CL method and our DGC calibration procedure can yield a more precise update direction, and consequently enhance the ultimate model performance.

Algorithm 1 DGC procedure

Input: Data stream $\{\mathcal{T}_1, \mathcal{T}_2, \cdots \mathcal{T}_T\}$, update steps m, update stages S, batch size b, and learning rate η.
Output: Trained model parameter $\tilde{\theta}_T$
Initialize model parameters $\tilde{\theta}_0$, $\Gamma'_{\text{DGC}}(1) = \vec{0}$
Initialize buffer $\mathcal{M}_1 = \emptyset$
for $t = 1, 2, \ldots, T$ **do**
  $\tilde{\theta}_t \leftarrow \tilde{\theta}_{t-1}$
  for $s = 1, 2, \ldots, S$ **do**
    $\theta_t^1 \leftarrow \tilde{\theta}_t$
    for $k = 1, 2, \ldots, m$ **do**
      Take a uniform sample X_t of size b from $\mathcal{T}_t$
      Take a uniform sample X'_t of size b from $\mathcal{M}_t$
      Calculate $\Gamma_{\text{DGC}}(t,k)$ with $\Gamma'_{\text{DGC}}(t)$ according to (11)
      Calculate v_t^k with $\Gamma_{\text{DGC}}(t,k)$ according to (10)
      / Calculate the calibrated gradient */*
      $\theta_t^{k+1} \leftarrow \theta_t^k - \eta \cdot v_t^k$
    end for
    Update $\Gamma'_{\text{DGC}}(t)$ according to (11) and (12)
    / Update $\Gamma'_{DGC}(t)$ from $\Gamma_{DGC}(t, m+1)$ */*
    $\tilde{\theta}_t \leftarrow \theta_t^{m+1}$
  end for
  $\mathcal{M}_{t+1} \leftarrow$ MemoryUpdate($\mathcal{T}_t$, $\mathcal{M}_t$)
  / Reservoir sampling */*
  Calculate and store $\Gamma'_{\text{DGC}}(t+1)$ according to (14)
  / Update $\Gamma'_{DGC}(t+1)$ from $\Gamma'_{DGC}(t)$ */*
end for

4. Experiments

We conduct the experiments to compare with various baseline methods across different datasets. We consider both the CIL and TFCL models.

Datasets We carry out the experiments on three widely employed datasets S(Split)-CIFAR10, S-CIFAR100 (Krizhevsky et al., 2009), and S-TinyImageNet (Le & Yang, 2015). S-CIFAR10 is the split dataset by partitioning CIFAR10 into 5 tasks, each containing two categories; similarly, S-CIFAR100 and S-TinyImageNet are the datasets by respectively partitioning CIFAR100 and TinyImageNet

Datasets	**S-CIFAR10**		**S-CIFAR100**		**S-TinyImageNet**	
Size of Buffer	500	2000	500	2000	2000	5000
AOP	$66.73_{\pm 0.60}$		$42.73_{\pm 0.62}$		$21.40_{\pm 0.17}$	
AGEM	$45.42_{\pm 0.81}$	$45.58_{\pm 0.64}$	$26.11_{\pm 0.09}$	$26.13_{\pm 0.07}$	$22.41_{\pm 0.11}$	$21.98_{\pm 0.36}$
SSVRG	$49.02_{\pm 4.09}$	$58.68_{\pm 3.77}$	$27.74_{\pm 2.45}$	$39.09_{\pm 5.53}$	$14.39_{\pm 1.47}$	$15.88_{\pm 2.37}$
MOCA	$81.01_{\pm 0.97}$	$85.06_{\pm 0.51}$	$54.14_{\pm 0.43}$	$59.29_{\pm 2.97}$	$34.74_{\pm 10.08}$	$38.86_{\pm 8.58}$
GSS	$68.81_{\pm 0.98}$	$76.08_{\pm 1.35}$	$33.72_{\pm 0.22}$	$38.54_{\pm 0.39}$	$31.38_{\pm 0.11}$	$34.31_{\pm 0.22}$
GCR	$\underline{82.31}_{\pm 0.43}$	$86.35_{\pm 0.48}$	$53.43_{\pm 2.15}$	$63.18_{\pm 2.26}$	$48.94_{\pm 0.44}$	$54.60_{\pm 0.43}$
HAL	$58.06_{\pm 1.90}$	$69.53_{\pm 2.55}$	$24.85_{\pm 0.91}$	$28.05_{\pm 1.90}$	$18.66_{\pm 1.02}$	$21.45_{\pm 0.91}$
ICARL	$65.89_{\pm 2.74}$	$75.94_{\pm 0.84}$	$60.58_{\pm 0.50}$	$64.03_{\pm 0.41}$	$43.53_{\pm 0.21}$	$44.52_{\pm 0.31}$
ER	$74.19_{\pm 0.85}$	$84.27_{\pm 0.57}$	$42.34_{\pm 0.83}$	$55.48_{\pm 1.52}$	$39.23_{\pm 0.16}$	$45.47_{\pm 0.44}$
DGC-ER	$76.09_{\pm 0.62}$	$\underline{86.42}_{\pm 0.58}$	$44.46_{\pm 1.07}$	$59.55_{\pm 0.97}$	$41.38_{\pm 0.52}$	$47.40_{\pm 0.45}$
DER++	$59.66_{\pm 1.32}$	$66.81_{\pm 0.19}$	$47.03_{\pm 0.55}$	$55.22_{\pm 0.54}$	$32.20_{\pm 0.75}$	$40.89_{\pm 0.37}$
DGC-DER++	$62.92_{\pm 0.90}$	$67.43_{\pm 0.25}$	$49.59_{\pm 1.06}$	$57.05_{\pm 0.67}$	$33.67_{\pm 0.73}$	$41.76_{\pm 0.53}$
XDER	$70.12_{\pm 0.68}$	$70.35_{\pm 0.63}$	$61.45_{\pm 0.50}$	$66.51_{\pm 0.42}$	$52.45_{\pm 0.92}$	$55.12_{\pm 0.22}$
DGC-XDER	$72.34_{\pm 1.08}$	$72.41_{\pm 1.05}$	$\underline{62.70}_{\pm 0.44}$	$\underline{67.59}_{\pm 0.18}$	$\underline{53.50}_{\pm 0.25}$	$\underline{55.94}_{\pm 0.19}$
DYNAMIC ER	$79.65_{\pm 0.86}$	$83.30_{\pm 0.93}$	$61.92_{\pm 2.75}$	$64.57_{\pm 2.02}$	$54.88_{\pm 1.64}$	$56.70_{\pm 0.73}$
DGC-DYNAMIC ER	$\mathbf{84.23}_{\pm 1.62}$	$\mathbf{89.90}_{\pm 0.93}$	$\mathbf{63.33}_{\pm 1.26}$	$\mathbf{70.70}_{\pm 1.31}$	$\mathbf{58.10}_{\pm 1.06}$	$\mathbf{58.23}_{\pm 0.84}$

Table 1. The FAIA $\pm$ standard error(%) in CIL. The methods combined with DGC are colored in gray. The best results are highlighted in bold, and the best results except Dynamic ER and DGC-Dynamic ER are underlined. Since AOP does not store previous data during training, it only has one numerical result per dataset in the table (without specifying the buffer size).

Datasets	**S-CIFAR100**	
	5 Tasks	20 Tasks
AOP	$43.31_{\pm 0.44}$	$40.99_{\pm 0.36}$
AGEM	$38.67_{\pm 0.14}$	$16.21_{\pm 0.06}$
SSVRG	$46.00_{\pm 4.60}$	$33.22_{\pm 4.10}$
MOCA	$66.58_{\pm 0.14}$	$22.05_{\pm 2.56}$
GSS	$50.43_{\pm 0.43}$	$25.93_{\pm 0.11}$
GCR	$67.20_{\pm 0.38}$	$51.57_{\pm 1.63}$
HAL	$35.05_{\pm 0.68}$	$27.16_{\pm 1.51}$
ICARL	$67.45_{\pm 0.27}$	$55.77_{\pm 0.57}$
ER	$60.85_{\pm 0.60}$	$52.16_{\pm 0.90}$
DGC-ER	$62.13_{\pm 0.33}$	$54.86_{\pm 1.22}$
DER++	$56.27_{\pm 0.27}$	$53.77_{\pm 1.56}$
DGC-DER++	$57.45_{\pm 1.04}$	$57.71_{\pm 0.35}$
XDER	$67.14_{\pm 0.38}$	$60.45_{\pm 0.46}$
DGC-XDER	$\mathbf{67.56}_{\pm 0.79}$	$\mathbf{63.53}_{\pm 0.48}$

Table 2. The FAIA $\pm$ standard error(%) with different number of tasks. DGC methods are colored in gray. The best results are highlighted in bold.

into 10 tasks, each containing 10 (S-CIFAR100) and 20 (S-TinyImageNet) categories.

Baseline methods We consider the following baselines. **(1) Replay-based methods:** ER (Chaudhry et al., 2019b), DER++ (Buzzega et al., 2020), XDER (Boschini et al., 2022), MOCA (Yu et al., 2023), GSS (Aljundi et al., 2019b), GCR (Tiwari et al., 2022), HAL (Chaudhry et al., 2021), and ICARL (Rebuffi et al., 2017). **(2) Optimization-based methods:** AGEM (Chaudhry et al., 2019a), AOP (Guo et al., 2022), and SSVRG (Frostig et al., 2015). **(3) Dynamic architecture method:** Dynamic ER (Yan et al., 2021). We integrate DGC with ER, DER++, XDER, and Dynamic ER, and assess their performances in CIL. For TFCL, we consider its combination with ER and DER++. For convenience, we use "DGC-Y" to denote the combination of DGC with a CL method "Y". For example, DGC-ER denotes the method combining ER and DGC methods.

Evaluation metrics We employ the *Average Accuracy (AA)* (Chaudhry et al., 2018; Mirzadeh et al., 2020) and the *final Average Incremental Accuracy (FAIA)* (Hou et al., 2019; Douillard et al., 2020) to assess the performance. These two metrics are both widely used for continual learning. Let $a_{k,j} \in [0,1](k \geq j)$ denote the classification accuracy evaluated on the testing set of the task $\mathcal{T}_j$ after learning $\mathcal{T}_k$. The value AA at time spot i is defined as $\text{AA}_i \triangleq \frac{1}{i}\sum_{j=1}^{i} a_{i,j}$. In particular, we name the value AA_T as the *final Average Accuracy (FAA)*. The final AIA is defined as $\text{FAIA} \triangleq \frac{1}{T}\sum_{i=1}^{T} \text{AA}_i$. We also use *Final Forgetting (FF)* from (Chaudhry et al., 2018) to measure the forgetting of the model throughout the learning process (the formal definition of FF and its numerical results are placed in the supplement). Each instance of our experiments is repeated by 10 times.

4.1. Results in CIL

Hyper-parameters selection In our implementation, we fixed the values of epoch and batch size, which implies that the total number of optimization steps (i.e., the value $s \times m$)

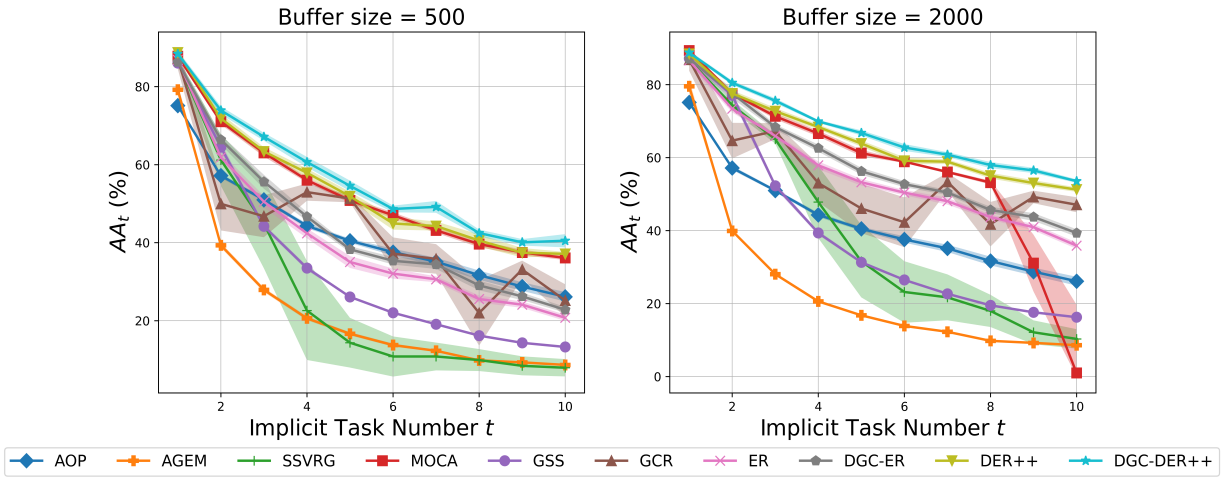

Figure 2. The averaged AA_t over implicit task number t in TFCL on S-CIFAR100.

is also fixed. In our experiments, we set the value $m = 200$, so the value of s in Algorithm 1 is also determined.

Except for the Dynamic ER (which is dynamically expanded), all other testing methods have constant storage limits. The results shown in Table 1 reveal that our DGC method can bring improvements to the combined methods on the testing benchmarks. For the sake of clarity, we also underline the best-performing method except Dynamic ER and DGC-Dynamic ER in Table 1. Among the methods with constant storage limits, DGC-ER achieves the best results on S-CIFAR10 when the buffer size is 2000, and DGC-XDER achieves the best results on S-CIFAR100 and S-TinyImageNet. It is worth noting that our DGC method can also be conveniently integrated with existing advanced dynamic expansion representation techniques, such as Dynamic ER, which demonstrates the improvements to certain extent, e.g., it achieves an improvement more than 6% on S-CIFAR10/100 with buffer size 2000. We also record the training time of these baseline methods in our supplement.

In the subsequent experiment, we investigate the performance of DGC compared to other baseline methods with varying the number of tasks. Throughout the experiment, we maintain a constant buffer size of 2000. As outlined in Table 2, our results demonstrate that DGC can bring certain improvements to ER, DER++, and XDER with setting the number of tasks to be 5 and 20. Moreover, similar to the previous research in (Boschini et al., 2022), the results shown in Table 2 also imply that increasing the number of tasks could exacerbate the catastrophic forgetting issue. This phenomenon occurs because the model faces a reduced volume of data on each specific task, and thereby necessitates the capability of retaining the information of historical data to guide the model updates. As can be seen, the improvement obtained by DGC for the case of 20 tasks usually is more significant compared with the case of 5 tasks. For example, DGC-DER++ achieves a 3.94% improvement with 20 tasks versus a 1.18% improvement with 5 tasks, while DGC-XDER exhibits a 3.08% improvement with 20 tasks and only 0.42% with 5 tasks. These results highlight the advantage of DGC for mitigating catastrophic forgetting by effectively utilizing historical data through gradient-based calibration.

4.2. Results in TFCL

We then conduct the experiments in TFCL. The curves shown in Figure 2 depict the average AA_t evolutions with varying the "implicit" task number t on S-CIFAR100. We call it "implicit" since the value t is not given during the training, which means that the design of the algorithm cannot rely on task boundaries or task identities. Therefore, we only compare those baseline methods that are applicable to TFCL model. The results suggest that DGC can bring improvements to ER and DER++ on AA_t for almost all the ts; in particular, DGC-DER++ achieves the best performance and also with small variances. Through approximating the full gradient, the GCR and SSVRG methods can relieve the catastrophic forgetting issue to a certain extent, but they still suffer from the issue of storage limit, which affects their effectiveness for estimating the full gradient. Consequently, these methods exhibit performance downgrade and larger variance in Figure 2. Comparing with them, the approaches integrated with DGC illustrate more consistent and stable improvements. More detailed results for TFCL are available in our supplement.

4.3. Smoothness of Training with DGC

Since our proposed DGC approach stems from the variance reduction method of SGD, a natural question is whether

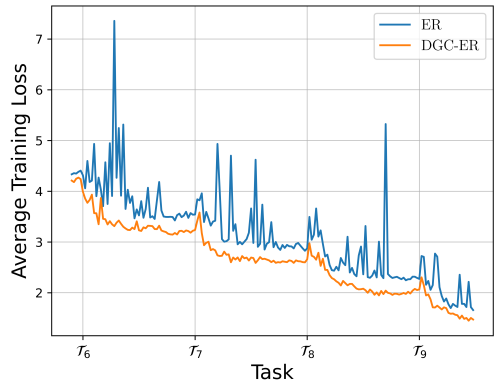

Figure 3. The partial trajectories of loss on S-CIFAR100.

it also improves the smoothness during the training process. We compare the loss trajectories on the entire training dataset before and after implementing DGC. From Figure 3 we can see that the classical ER method has erratic fluctuations in loss during the training process; this could cause some practical problems in a real-world CL scenario, e.g., we may need to pay more effort to carefully adjust the learning rate and determine the stopping condition. In contrast, the DGC method has a smoother reduction in loss, and ultimately yields lower loss values.

5. Conclusion

In this paper, we revisit the experience replay method and aim to utilize historical information to derive a more accurate gradient for alleviating catastrophic forgetting. Inspired by the variance reduction methods for SGD, we introduce a new approach "DGC" to dynamically manage a gradient calibration term in CL training. Our approach can be conveniently integrated with several existing continual learning methods, contributing to a substantial improvement in both CIL and TFCL. Moreover, the improved stability of training loss reduction can also ease our practical implementation.

Acknowledgements

We want to thank the anonymous reviewers and Xianglu Wang for their helpful comments. The research of this work was supported in part by the National Key Research and Development Program of China 2021YFA1000900, the National Natural Science Foundation of China 62272432, and the Natural Science Foundation of Anhui Province 2208085MF163.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019a.

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

Arani, E., Sarfraz, F., and Zonooz, B. Learning fast, learning slow: A general continual learning method based on complementary learning system. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Babanezhad Harikandeh, R., Ahmed, M. O., Virani, A., Schmidt, M., Konečnỳ, J., and Sallinen, S. Stopwasting my gradients: Practical svrg. *Advances in Neural Information Processing Systems*, 28, 2015.

Bai, X., Wang, X., Liu, X., Liu, Q., Song, J., Sebe, N., and Kim, B. Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments. *Pattern Recognition*, 120:108102, 2021.

Boschini, M., Bonicelli, L., Buzzega, P., Porrello, A., and Calderara, S. Class-incremental continual learning into the extended der-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5497–5512, 2022.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

Bottou, L. et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

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.

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, 2018.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations, ICLR*

2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019a.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019b.

Chaudhry, A., Gordo, A., Dokania, P., Torr, P., and Lopez-Paz, D. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6993–7001, 2021.

De Lange, 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*, 44(7):3366–3385, 2021.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

Douillard, A., Cord, M., Ollion, C., Robert, T., and Valle, E. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pp. 86–102. Springer, 2020.

Farajtabar, M., Azizan, N., Mott, A., and Li, A. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3762–3773. PMLR, 2020.

Farquhar, S. and Gal, Y. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

Frostig, R., Ge, R., Kakade, S. M., and Sidford, A. Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory*, pp. 728–763. PMLR, 2015.

Gao, R. and Liu, W. DDGR: continual learning with deep diffusion-based generative replay. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10744–10763. PMLR, 2023.

Ghunaim, Y., Bibi, A., Alhamoud, K., Alfarra, M., Hammoud, H. A. A. K., Prabhu, A., Torr, P. H. S., and Ghanem, B. Real-time evaluation in online continual learning: A new hope. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 11888–11897. IEEE, 2023.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014a.

Goodfellow, I. J., Mirza, M., Da, X., Courville, A. C., and Bengio, Y. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014b.

Gower, R. M., Schmidt, M., Bach, F., and Richtárik, P. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11):1968–1983, 2020.

Guo, Y., Hu, W., Zhao, D., and Liu, B. Adaptive orthogonal projection for batch and online continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6783–6791, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 831–839, 2019.

Hsu, Y.-C., Liu, Y.-C., Ramasamy, A., and Kira, Z. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

Hu, X., Tang, K., Miao, C., Hua, X.-S., and Zhang, H. Distilling causal effect of data in class-incremental learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 3957–3966, 2021.

Jin, H., Lin, D., and Zhang, Z. Towards better generalization: Bp-svrg in training deep neural networks. *arXiv preprint arXiv:1908.06395*, 2019.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

Kim, S., Noci, L., Orvieto, A., and Hofmann, T. Achieving a better stability-plasticity trade-off via auxiliary networks in continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 11930–11939. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01148.

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.

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

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Lei, L., Ju, C., Chen, J., and Jordan, M. I. Non-convex finite-sum optimization via scsg methods. *Advances in Neural Information Processing Systems*, 30, 2017.

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

Liu, H. and Liu, H. Continual learning with recursive gradient optimization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

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

Mai, Z., Li, R., Jeong, J., Quispe, D., Kim, H., and Sanner, S. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

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.

Mirzadeh, S. I., Farajtabar, M., Gorur, D., Pascanu, R., and Ghasemzadeh, H. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*, 2020.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Prabhu, A., Hammoud, H. A. A. K., Dokania, P. K., Torr, P. H. S., Lim, S., Ghanem, B., and Bibi, A. Computationally budgeted continual learning: What does matter? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 3698–3707. IEEE, 2023. doi: 10.1109/CVPR52729.2023.00360.

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.

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. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

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

Tiwari, R., Killamsetty, K., Iyer, R., and Shenoy, P. Gcr: Gradient coreset based replay buffer selection for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 99–108, 2022.

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

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

Wang, L., Zhang, X., Su, H., and Zhu, J. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*, 2023.

Wu, C., Herranz, L., Liu, X., Van De Weijer, J., Raducanu, B., et al. Memory replay gans: Learning to generate new categories without forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

Xu, J. and Zhu, Z. Reinforced continual learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Yan, S., Xie, J., and He, X. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3014–3023, 2021.

Yu, L., Hu, T., Hong, L., Liu, Z., Weller, A., and Liu, W. Continual learning by modeling intra-class variation. *Trans. Mach. Learn. Res.*, 2023, 2023.

Zhu, Y., Yao, C., and Bai, X. Scene text detection and recognition: recent advances and future trends. *Frontiers Comput. Sci.*, 10(1):19–36, 2016.

A. Omitted Proofs

Lemma A.1 (Lemma 3.2)**.**

$$\mathbb{E}\left[\Gamma'_{DGC}(t)\right] = \mathcal{G}(\mathcal{T}_{[1:t)}, \tilde{\theta}_t). \tag{17}$$

Proof. We prove it by induction. When $t = 1$, both the two-side terms are 0, so the equation holds. Assume the equation holds when $t = t_1$, and we consider the change of $\Gamma'_{\text{DGC}}(t)$ at the time spot t_1.

Firstly, $\Gamma'_{\text{DGC}}(t_1)$ is updated at the end of every stage. After the update, $\Gamma'_{\text{DGC}}(t_1) = \Gamma_{\text{DGC}}(t_1, m+1)$. Meanwhile, we have

$$\begin{aligned}
&\mathbb{E}[\Gamma_{\text{DGC}}(t_1, m+1)] \\
&= \mathbb{E}[\nabla_\theta \ell(\bar{x}_{t_1}, \bar{y}_{t_1}, \theta_{t_1}^{m+1}) - (\nabla_\theta \ell(\ddot{x}_{t_1}, \ddot{y}_{t_1}, \tilde{\theta}_{t_1}) - \Gamma'_{\text{DGC}}(t_1))] \\
&= \mathcal{G}(\mathcal{T}_{[1:t_1)}, \theta_{t_1}^{m+1}) - \mathcal{G}(\mathcal{T}_{[1:t_1)}, \tilde{\theta}_{t_1}) + \mathcal{G}(\mathcal{T}_{[1:t_1)}, \tilde{\theta}_{t_1}) \\
&= \mathcal{G}(\mathcal{T}_{[1:t_1)}, \theta_{t_1}^{m+1}) \\
&= \mathcal{G}(\mathcal{T}_{[1:t_1)}, \tilde{\theta}_{t_1}),
\end{aligned}$$

where the first equation comes from Eq (11) and the second equation is based on the inductive hypothesis.

Secondly, in the update from time spot t to $t+1$, we have

$$\begin{aligned}
&\mathbb{E}[\Gamma'_{\text{DGC}}(t_1+1)] \\
&\quad = \mathbb{E}[\frac{1}{t_1}((t_1-1)\cdot \Gamma_{\text{DGC}}(t_1, m+1) + \mathcal{G}(\mathcal{T}_{t_1}, \tilde{\theta}_{t_1}))] \\
&\quad = \frac{1}{t_1}((t_1-1)\cdot \mathcal{G}(\mathcal{T}_{[1:t_1)}, \tilde{\theta}_{t_1}) + \mathcal{G}(\mathcal{T}_{t_1}, \tilde{\theta}_{t_1})) \\
&\quad = \mathcal{G}(\mathcal{T}_{[1:t_1)}, \tilde{\theta}_{t_1+1})
\end{aligned}$$

based on Eq (14). This implies that our conclusion also holds when $t = t_1 + 1$. □

Theorem A.2 (Theorem 3.3)**.** *Assume that $f(x;\theta)$ is L-smooth and γ-strongly convex; the parameters $m \geq \frac{10L^2}{\gamma^2}$ and $\eta = \frac{\gamma}{10L}$. Then we have a linear convergence in expectation for the DGC procedure at time t:*

$$\mathbb{E}\left[\|\tilde{\theta}_{t,s+1} - \theta_*\|_2^2\right] \leq \frac{1}{2^s}\mathbb{E}\left[\|\tilde{\theta}_{t,1} - \theta_*\|_2^2\right]$$

where $\tilde{\theta}_{t,s}$ represents the initialization parameter at the beginning of the s-th stage at time spot t.

Our proof is inspired by the idea of (Johnson & Zhang, 2013). To simplify the notations, we define

$$v_t^k = \nabla\psi_{x_t}\left(\theta_t^k\right) - \nabla\psi_{x_t}(\tilde{\theta}_t) + \tilde{\mu}$$

through combining Eq (10) and (11) (we replace "Γ" in Eq (10) by (11)), where

$$\begin{aligned}
\nabla\psi_{x_t}\left(\theta_t^k\right) &\triangleq \frac{1}{t}\nabla_\theta \ell(x_t, y_t, \theta_t^k) + \frac{t-1}{t}\nabla_\theta \ell(\bar{x}, \bar{y}, \theta_t^k), \\
\nabla\psi_{x_t}(\tilde{\theta}_t) &\triangleq \frac{1}{t}\nabla_\theta \ell(\dot{x}_t, \dot{y}_t, \tilde{\theta}_t) + \frac{t-1}{t}\nabla_\theta \ell(\ddot{x}, \ddot{y}, \tilde{\theta}_t), \\
\tilde{\mu} &\triangleq \frac{1}{t}\mathcal{G}(\mathcal{T}_t, \tilde{\theta}_t) + \frac{t-1}{t}\Gamma'_{\text{DGC}}(t).
\end{aligned}$$

Before proving Theorem 3.3, we provide the following key lemmas first.

Lemma A.3.

$$\mathbb{E}[\left\|v_t^k\right\|_2^2] \leq 2L\left[\mathbb{E}[||\theta_t^k - \theta_*||_2^2] + \mathbb{E}[||\tilde{\theta}_t - \theta_*||_2^2]\right].$$

Proof.

$$
\begin{aligned}
&\mathbb{E}[\left\|v_t^k\right\|_2^2] \\
\leq & 2\mathbb{E}\left[\|\nabla\psi_{x_t}\left(\theta_t^k\right)-\nabla\psi_{x_t}\left(\theta_*\right)\|_2^2\right] \\
&+2\mathbb{E}\left[\|\nabla\psi_{x_t}(\tilde{\theta}_t)-\nabla\psi_{x_t}\left(\theta_*\right)-\nabla\ell_{\mathrm{CL}}^t(\tilde{\theta}_t)\|_2^2\right] \\
= & 2\mathbb{E}\left[\|\nabla\psi_{x_t}\left(\theta_t^k\right)-\nabla\psi_{x_t}\left(\theta_*\right)\|_2^2\right] \\
&+2\mathbb{E}[\|\nabla\psi_{x_t}(\tilde{\theta}_t)-\nabla\psi_{x_t}\left(\theta_*\right) \\
&-\mathbb{E}[\nabla\psi_{x_t}(\tilde{\theta}_t)-\nabla\psi_{x_t}\left(\theta_*\right)]\|_2^2] \\
\leq & 2\mathbb{E}[\left\|\nabla\psi_{x_t}\left(\theta_t^k\right)-\nabla\psi_{x_t}\left(\theta_*\right)\right\|_2^2] \\
&+2\mathbb{E}[\left\|\nabla\psi_{x_t}(\tilde{\theta}_t)-\nabla\psi_{x_t}\left(\theta_*\right)\right\|_2^2] \\
\leq & 2L\left[\mathbb{E}[||\theta_t^k-\theta_*||_2^2]+\mathbb{E}[||\tilde{\theta}_t-\theta_*||_2^2]\right],
\end{aligned}
\tag{18}
$$

where the first inequality comes from the fact $\|a+b\|_2^2 \leq 2\|a\|_2^2+2\|b\|_2^2$ for any vectors a, b and Lemma A.1, the second inequality is based on the fact $\mathbb{E}[\|\xi-\mathbb{E}\xi\|_2^2]=\mathbb{E}[\|\xi\|_2^2]-\|\mathbb{E}[\xi]\|_2^2 \leq \mathbb{E}[\|\xi\|_2^2]$ for any random vector ξ, and the third inequality is based on L-smooth. □

Lemma A.4.

$$
\begin{aligned}
&\mathbb{E}[\left\|\theta_t^k-\theta_*\right\|_2^2] \leq (1-2\eta\gamma+2L\eta^2)\mathbb{E}[||\theta_t^{k-1}-\theta_*||_2^2] \\
&+2L\eta^2\mathbb{E}[||\tilde{\theta}_t-\theta_*||_2^2].
\end{aligned}
\tag{19}
$$

Proof. Note that $\mathbb{E}[v_t^k]=\mathbb{E}[\nabla\psi_{x_t}(\theta_t^k)]=\nabla\ell_{\mathrm{CL}}^t\left(\theta_t^{k-1}\right)$, and so we have

$$
\begin{aligned}
&\mathbb{E}[\left\|\theta_t^k-\theta_*\right\|_2^2] \\
=&\left\|\theta_t^{k-1}-\theta_*\right\|_2^2-2\eta\left(\theta_t^{k-1}-\theta_*\right)^\top\mathbb{E}[v_t^{k-1}]+\eta^2\mathbb{E}[\left\|v_t^{k-1}\right\|_2^2] \\
\leq&\left\|\theta_t^{k-1}-\theta_*\right\|_2^2-2\eta\left(\theta_t^{k-1}-\theta_*\right)^\top\nabla\ell_{\mathrm{CL}}^t\left(\theta_t^{k-1}\right) \\
&+2L\eta^2\left[\mathbb{E}[||\theta_t^{k-1}-\theta_*||_2^2]+\mathbb{E}[||\tilde{\theta}_t-\theta_*||_2^2]\right] \\
\leq&\left\|\theta_t^{k-1}-\theta_*\right\|_2^2-2\eta\gamma||\theta_t^{k-1}-\theta_*||_2^2 \\
&+2L\eta^2\left[\mathbb{E}[||\theta_t^{k-1}-\theta_*||_2^2]+\mathbb{E}[||\tilde{\theta}_t-\theta_*||_2^2]\right],
\end{aligned}
$$

where the first inequality comes from Lemma A.3, and the second inequality is based on the γ-strong convexity of $\ell_{\mathrm{CL}}^t(\theta)$. We take the expectation on θ_t^{k-1} from the above inequality and then have

$$
\begin{aligned}
&\mathbb{E}[\left\|\theta_t^k-\theta_*\right\|_2^2] \\
&\leq\mathbb{E}[\left\|\theta_t^{k-1}-\theta_*\right\|_2^2]-2\eta\gamma\mathbb{E}[||\theta_t^{k-1}-\theta_*||_2^2] \\
&+2L\eta^2\left[\mathbb{E}[||\theta_t^{k-1}-\theta_*||_2^2]+\mathbb{E}[||\tilde{\theta}_t-\theta_*||_2^2]\right] \\
&=(1-2\eta\gamma+2L\eta^2)\mathbb{E}[||\theta_t^{k-1}-\theta_*||_2^2]+2L\eta^2\mathbb{E}[||\tilde{\theta}_t-\theta_*||_2^2].
\end{aligned}
$$

□

Proof of Theorem 3.3. We now proceed to prove the theorem. To simplify notation, we denote $\mathbb{E}\left\|\theta_t^k-\theta_*\right\|_2^2$ as p_k. Based on lemma A.4, we have

$$
p_k \leq \left(1-2\eta\gamma+2\eta^2L\right)p_{k-1}+4\eta^2Lp_{k-1}.
$$

By setting $\eta = \frac{\gamma}{10L}$ and $m \geq \frac{10L^2}{\gamma^2}$, we have

$$\begin{aligned} p_{m+1} &\leq \left(1 - \frac{\gamma^2}{10L^2}\right) p_m \leq \left(1 - \frac{\gamma^2}{10L^2}\right)^m p_1 \\ &\leq \exp\left(-\frac{\gamma^2}{10L^2} m\right) p_1 \leq \frac{1}{2} p_1, \end{aligned}$$

i.e

$$\mathbb{E}\left[\|\theta_t^{m+1} - \theta_*\|_2^2\right] \leq \frac{1}{2}\mathbb{E}\left[\|\theta_t^1 - \theta_*\|_2^2\right]. \tag{20}$$

Subsequently, we consider the update during different stage s, so that the initial parameter $\tilde{\theta}_{t,s}$ is θ_t^1 and updated parameter $\tilde{\theta}_{t,s+1} = \theta_t^{m+1}$ is selected after all of the updates have been completed. Then Eq (20) becomes

$$\mathbb{E}\left[\|\tilde{\theta}_{t,s+1} - \theta_*\|_2^2\right] \leq \frac{1}{2}\mathbb{E}\left[\|\tilde{\theta}_{t,s} - \theta_*\|_2^2\right]. \tag{21}$$

Through recursively applying the above equation from stage 1 to stage $s+1$, we have

$$\mathbb{E}\left[\|\tilde{\theta}_{t,s+1} - \theta_*\|_2^2\right] \leq \frac{1}{2^s}\mathbb{E}\left[\|\tilde{\theta}_{t,1} - \theta_*\|_2^2\right].$$

□

B. Detailed DGC procedure in TFCL

We consider the data streams $\{\mathcal{T}_1, \mathcal{T}_2, \cdots \mathcal{T}_T\}$ with each $\mathcal{T}_i$ being the batch training data in TFCL. Algorithm 2 shows how to implement the DGC procedure in this model.

Algorithm 2 DGC procedure in TFCL

Input: Data stream $\{\mathcal{T}_1, \mathcal{T}_2, \cdots \mathcal{T}_T\}$, update steps m, batch size b, and learning rate η.
Output: Trained model parameter θ_T
Initialize model parameters θ_0, $\Gamma'_{\text{DGC}}(1) = \vec{0}$, $\tilde{t} = 1$
Initialize buffer $\mathcal{M}_1 = \emptyset$, $\tilde{\theta} = \theta_0$
for $t = 1, 2, \ldots, T$ **do**
  $\theta_t \leftarrow \theta_{t-1}$
  Take a uniform sample X'_t of size b from M_t
  Calculate $\Gamma_{\text{DGC}}(t, 1)$ with $\Gamma'_{\text{DGC}}(\tilde{t})$ according to (11)
  Calculate v_t^1 with $\Gamma_{\text{DGC}}(t, 1)$ according to (10)
  / Calculate the calibrated gradient */*
  $\theta_t \leftarrow \theta_t - \eta \cdot v_t^1$
  if $(t-1) \mod m = 0$ **then**
    Update $\Gamma'_{\text{DGC}}(\tilde{t})$ according to (11) and (12)
    / Update $\Gamma'_{DGC}(\tilde{t})$ from $\Gamma_{DGC}(t, 1)$ */*
    $\tilde{t} \leftarrow t$, $\tilde{\theta} \leftarrow \theta_t$
  end if
  $\mathcal{M}_{t+1} \leftarrow$ MemoryUpdate($\mathcal{T}_t$, $\mathcal{M}_t$)
  / Reservoir sampling */*
  Calculate and store $\Gamma'_{\text{DGC}}(\tilde{t})$ according to (14)
  / Update $\Gamma'_{DGC}(\tilde{t})$ for new historical data */*
end for

Datasets	**S-CIFAR100**			
α	$1e^{-2}$	$1e^{-3}$	$1e^{-4}$	0
DGC-ER	$57.4_{\pm 2.14}$	$59.55_{\pm 0.97}$	$59.14_{\pm 0.91}$	$55.48_{\pm 1.52}$

Table 3. The FAIA $\pm$ standard error(%) with different selection of α.

Datasets	**S-CIFAR100**				
m	1	100	200	300	400
DGC-ER	$49.11_{\pm 5.30}$	$59.44_{\pm 0.95}$	$59.55_{\pm 0.97}$	$59.27_{\pm 0.65}$	$58.94_{\pm 0.98}$

Table 4. The FAIA $\pm$ standard error(%) with different selection of m.

C. Details for Experimental Setup

Architecture and hyperparameter Most of the comparison methods in our experiments use the implementation of Mammoth [1]. For the Dynamic ER (Yan et al., 2021) method we use the implementation of PyCIL [2], and for the GCR method we use the implementation of Google-research [3]. To be consistent with the selection of these open source frameworks, all the methods use ResNet18 (He et al., 2016) as the base network. All the networks are trained from scratch. For the hyperparameter selection of different methods, we directly use the original hyperparameters used in these open-source frameworks, which are obtained by using grid search on 10% of the training set.

Training details To maintain consistency, we utilize the default parameter settings and network architectures provided by the framework. As in previous studies (Tiwari et al., 2022; Yan et al., 2021; Buzzega et al., 2020), random crops and horizontal flips are used as data augmentation in all experiments. All methods that do not incorporate DGC are optimized using standard SGD. In our DGC calibrated method, we empirically set $m = 200$. To fairly compare the methods with constant storage limits, we uniformly train for 50 epochs in each task on S-CIFAR10 and S-CIFAR100, and 100 epochs in each task on S-TinyImageNet. The batch size is all set to be 32. According to the experimental description in the Dynamic ER method, we train the first task for 200 epochs on all datasets and train all subsequent tasks for 170 epochs; the batch size is set to be 128.

D. Other Experimental Results

Selection of $\boldsymbol{\alpha}$ We explore the impact of different α values to the performance of our DGC method. In the experiment, we fix the buffer size to 2000. Table 3 shows that DGC can improve ER under different α values. In all the experiments of Section 4, we fix the value of α to be $1e^{-3}$.

Selection of $\boldsymbol{m}$ As stated in Theorem 3.3, when m is sufficiently large, our DGC method has a linear convergence in expectation. The results in Table 4 show that when m is greater than 100, DGC-ER can significantly improve the performance of ER. Table 4 also suggests that our method is relatively robust to the selection of m.

FF and FAA results in CIL we use Final Forgetting (FF) (Chaudhry et al., 2018) to measure the forgetting of the model throughout the learning process. Following the previous work, it is defined as $\mathrm{FF} \triangleq \frac{1}{T-1}\sum_{j=1}^{T-1}\left(\max_{k\in\{1,\ldots,T-1\}} a_{k,j} - a_{T,j}\right)$. Table 5 and Table 6 show that DGC based methods can reduce forgetting while improving the performance in all the cases. For most instances, our DGC based methods achieve the best performance.

Detailed results in TFCL Table 7 shows that our DGC method can improve the performance in TFCL. The DGC-DER++ method achieves the best performance in all the cases.

Training Time We set $b = 500$ and conduct experiments on S-CIFAR100 to measure the training time of several algorithms on the first three tasks in the CIL scenario. Table 8 shows that our method and GCR have roughly the same training time, which is higher than AGEM but lower than other baselines. The results on the remaining tasks and other datasets are similar. Therefore, the increase caused by our method on training time is not significant, compared with most of the baselines.

[1] https://github.com/aimagelab/mammoth

[2] https://github.com/G-U-N/PyCIL

[3] https://github.com/google-research/google-research/tree/master/gradient_coresets_replay

Datasets	**S-CIFAR10**		**S-CIFAR100**		**S-TinyImageNet**	
Size of Buffer	500	2000	500	2000	2000	5000
AOP	$49.54_{\pm 1.44}$		$14.24_{\pm 2.18}$		$3.92_{\pm 0.19}$	
AGEM	$96.50_{\pm 1.29}$	$96.34_{\pm 0.87}$	$89.58_{\pm 0.14}$	$89.56_{\pm 0.22}$	$77.30_{\pm 0.22}$	$77.53_{\pm 0.39}$
SSVRG	$52.10_{\pm 7.83}$	$68.75_{\pm 11.86}$	$56.65_{\pm 5.21}$	$70.15_{\pm 8.61}$	$21.99_{\pm 2.02}$	$31.29_{\pm 3.92}$
MOCA	$12.97_{\pm 1.00}$	$7.22_{\pm 0.71}$	$28.65_{\pm 0.87}$	$58.08_{\pm 17.70}$	$41.61_{\pm 14.34}$	$32.04_{\pm 11.13}$
GSS	$64.81_{\pm 4.99}$	$47.27_{\pm 5.06}$	$83.43_{\pm 0.50}$	$79.52_{\pm 0.43}$	$72.75_{\pm 0.40}$	$69.97_{\pm 0.48}$
GCR	$24.18_{\pm 1.76}$	$13.34_{\pm 1.09}$	$58.03_{\pm 5.80}$	$34.41_{\pm 2.56}$	$48.47_{\pm 1.60}$	$39.34_{\pm 1.77}$
HAL	$62.86_{\pm 2.80}$	$36.65_{\pm 2.88}$	$55.22_{\pm 1.62}$	$47.69_{\pm 2.73}$	$43.72_{\pm 2.34}$	$38.86_{\pm 1.25}$
ICARL	$31.91_{\pm 2.25}$	$26.56_{\pm 1.19}$	$30.20_{\pm 0.40}$	$24.64_{\pm 0.33}$	$18.55_{\pm 0.45}$	$17.62_{\pm 0.61}$
ER	$44.28_{\pm 1.93}$	$22.96_{\pm 0.90}$	$74.29_{\pm 0.73}$	$54.60_{\pm 0.63}$	$65.74_{\pm 0.59}$	$54.27_{\pm 0.68}$
DGC-ER	$39.72_{\pm 1.91}$	$20.37_{\pm 1.03}$	$72.31_{\pm 0.85}$	$52.30_{\pm 1.09}$	$64.58_{\pm 0.52}$	$52.38_{\pm 0.68}$
DER++	$9.24_{\pm 3.01}$	$6.82_{\pm 0.60}$	$14.94_{\pm 2.23}$	$9.19_{\pm 1.09}$	$7.86_{\pm 3.24}$	$6.75_{\pm 0.84}$
DGC-DER++	$\mathbf{4.90}_{\pm 1.86}$	$\mathbf{3.99}_{\pm 1.01}$	$\mathbf{9.15}_{\pm 2.52}$	$\mathbf{5.79}_{\pm 0.85}$	$\mathbf{3.40}_{\pm 0.66}$	$\mathbf{4.59}_{\pm 0.83}$
XDER	$10.64_{\pm 0.70}$	$8.89_{\pm 0.33}$	$25.11_{\pm 0.79}$	$12.15_{\pm 0.26}$	$17.95_{\pm 0.55}$	$12.81_{\pm 0.31}$
DGC-XDER	$8.74_{\pm 0.98}$	$7.22_{\pm 0.97}$	$23.24_{\pm 0.65}$	$11.11_{\pm 0.37}$	$16.99_{\pm 0.74}$	$11.84_{\pm 0.47}$

Table 5. The FF $\pm$ standard error(%) in CIL. The methods combined with DGC are colored in gray. The best results are highlighted in bold.

Datasets	**S-CIFAR10**		**S-CIFAR100**		**S-TinyImageNet**	
Size of Buffer	500	2000	500	2000	2000	5000
AOP	$48.52_{\pm 1.06}$		$26.13_{\pm 1.22}$		$8.61_{\pm 0.02}$	
AGEM	$19.80_{\pm 0.29}$	$20.06_{\pm 0.38}$	$9.25_{\pm 0.08}$	$9.31_{\pm 0.06}$	$7.94_{\pm 0.06}$	$7.86_{\pm 0.11}$
SSVRG	$32.08_{\pm 6.97}$	$43.55_{\pm 11.38}$	$9.92_{\pm 2.22}$	$17.40_{\pm 5.35}$	$0.75_{\pm 0.09}$	$7.92_{\pm 2.22}$
MOCA	$70.34_{\pm 1.23}$	$78.68_{\pm 0.86}$	$37.13_{\pm 0.55}$	$9.35_{\pm 18.67}$	$0.50_{\pm 0.00}$	$0.50_{\pm 0.00}$
GSS	$44.56_{\pm 3.95}$	$56.46_{\pm 4.48}$	$13.22_{\pm 0.05}$	$16.12_{\pm 0.13}$	$11.74_{\pm 0.14}$	$13.45_{\pm 0.20}$
GCR	$\mathbf{72.62}_{\pm 0.76}$	$\mathbf{80.95}_{\pm 0.47}$	$28.66_{\pm 4.14}$	$48.89_{\pm 1.99}$	$29.01_{\pm 0.87}$	$36.45_{\pm 0.89}$
HAL	$39.41_{\pm 1.96}$	$58.34_{\pm 2.66}$	$8.46_{\pm 1.09}$	$11.91_{\pm 2.05}$	$5.43_{\pm 0.71}$	$8.00_{\pm 0.99}$
ICARL	$57.51_{\pm 2.90}$	$69.91_{\pm 0.68}$	$45.69_{\pm 0.53}$	$52.30_{\pm 0.47}$	$30.30_{\pm 0.44}$	$31.69_{\pm 0.32}$
ER	$61.08_{\pm 1.23}$	$76.82_{\pm 0.95}$	$21.18_{\pm 0.65}$	$36.24_{\pm 1.27}$	$18.39_{\pm 0.38}$	$26.71_{\pm 0.53}$
DGC-ER	$64.42_{\pm 1.35}$	$79.46_{\pm 0.61}$	$23.11_{\pm 0.76}$	$40.23_{\pm 1.05}$	$19.52_{\pm 0.55}$	$27.96_{\pm 0.54}$
DER++	$54.89_{\pm 1.42}$	$64.84_{\pm 0.40}$	$37.56_{\pm 1.03}$	$50.42_{\pm 0.68}$	$15.34_{\pm 3.53}$	$31.09_{\pm 0.59}$
DGC-DER++	$58.03_{\pm 1.10}$	$66.02_{\pm 0.51}$	$40.48_{\pm 1.14}$	$52.14_{\pm 0.64}$	$20.82_{\pm 0.91}$	$33.72_{\pm 1.63}$
XDER	$63.88_{\pm 1.43}$	$67.86_{\pm 0.84}$	$47.45_{\pm 0.65}$	$56.88_{\pm 0.53}$	$41.41_{\pm 0.36}$	$44.66_{\pm 0.09}$
DGC-XDER	$69.46_{\pm 1.82}$	$71.40_{\pm 1.69}$	$\mathbf{49.07}_{\pm 0.40}$	$\mathbf{57.97}_{\pm 0.31}$	$\mathbf{41.58}_{\pm 0.32}$	$\mathbf{44.88}_{\pm 0.40}$

Table 6. The FAA $\pm$ standard error(%) in CIL. The methods combined with DGC are colored in gray. The best results are highlighted in bold.

Datasets	**S-CIFAR10**	**S-CIFAR100**	**S-TinyImageNet**
Size of Buffer	500	2000	2000
AOP	$66.73_{\pm 0.60}$	$42.73_{\pm 0.62}$	$21.40_{\pm 0.17}$
AGEM	$50.24_{\pm 0.53}$	$23.55_{\pm 0.23}$	$18.95_{\pm 0.40}$
SSVRG	$49.02_{\pm 4.09}$	$39.09_{\pm 5.53}$	$14.39_{\pm 1.47}$
MOCA	$81.01_{\pm 0.97}$	$59.29_{\pm 2.97}$	$34.74_{\pm 10.08}$
GSS	$68.81_{\pm 0.98}$	$38.54_{\pm 0.39}$	$31.38_{\pm 0.11}$
GCR	$82.31_{\pm 0.43}$	$63.18_{\pm 2.16}$	$48.94_{\pm 0.44}$
ER	$73.04_{\pm 0.47}$	$55.46_{\pm 0.77}$	$37.44_{\pm 0.20}$
DGC-ER	$74.03_{\pm 0.55}$	$58.35_{\pm 0.81}$	$38.00_{\pm 0.34}$
DER++	$82.72_{\pm 0.17}$	$65.57_{\pm 0.42}$	$47.23_{\pm 1.22}$
DGC-DER++	$\mathbf{82.93}_{\pm 0.14}$	$\mathbf{66.57}_{\pm 0.20}$	$\mathbf{50.08}_{\pm 1.10}$

Table 7. The FAIA $\pm$ standard error(%) in TFCL. The methods combined with DGC are colored in gray. The best results are highlighted in bold.

Methods(s)	GSS	XDER	AOP	MOCA	SSVRG	GCR	DGC	AGEM	ER
T_1	4959.29	1083.86	1418.49	655.15	216.38	689.85	218.46	212.76	214.34
T_2	4281.39	2495.60	1412.48	1192.89	873.39	690.34	666.08	478.99	345.05
T_3	4158.51	2826.82	1415.24	1173.05	878.89	686.16	668.86	488.91	340.36

Table 8. Training time of the first three task in CIL on S-CIFAR100.