Mitigating Catastrophic Forgetting in Online Continual Learning by Modeling Previous Task Interrelations via Pareto Optimization

Yichen Wu [1 2 * †] Hong Wang [3 *] Peilin Zhao [2] Yefeng Zheng [3] Ying Wei [1 4] Long-Kai Huang [2]

Abstract

Catastrophic forgetting remains a core challenge in continual learning (CL), where the models struggle to retain previous knowledge when learning new tasks. While existing replay-based CL methods have been proposed to tackle this challenge by utilizing a memory buffer to store data from previous tasks, they generally overlook the interdependence between previously learned tasks and fail to encapsulate the optimally integrated knowledge in previous tasks, leading to sub-optimal performance of the previous tasks. Against this issue, we first reformulate replay-based CL methods as a unified hierarchical gradient aggregation framework. We then incorporate the Pareto optimization to capture the interrelationship among previously learned tasks and design a Pareto-Optimized CL algorithm (POCL), which effectively enhances the overall performance of past tasks while ensuring the performance of the current task. To further stabilize the gradients of different tasks, we carefully devise a hyper-gradient-based implementation manner for POCL. Comprehensive empirical results demonstrate that the proposed POCL outperforms current state-of-the-art CL methods across multiple datasets and different settings.

1. Introduction

An ideal intelligent system should possess the ability to incrementally learn, swiftly adapting to environmental changes while retaining previously acquired knowledge. Despite the remarkable performance of current deep neural networks (DNNs) on specific tasks, they still encounter challenges when adapting to streaming tasks. One critical issue is catastrophic forgetting, where acquiring knowledge on a new task leads to a significant decline in performance on previously learned tasks. To alleviate this issue, numerous algorithms have been proposed in continual learning (CL) area, aiming to enhance the incremental learning ability of DNNs on streaming tasks (Lopez-Paz & Ranzato, 2017; Serra et al., 2018; Gupta et al., 2020; Guo et al., 2020; Arani et al., 2022; Wang et al., 2023; Chrysakis & Moens, 2023).

Replay-based methods currently stand as straightforward yet effective research line in the field of CL. The objective of these methods is to emulate joint training, the upper bound of CL, which trains all the tasks simultaneously and optimizes the model $\boldsymbol{\theta}_t$ in the update direction $\boldsymbol{u}$ by fully exploiting the gradient of every task (as illustrated in Fig. 1). However, in the absence of full data from previous tasks, it becomes crucial to approximate their gradients. To this end, replay-based methods typically employ a small memory buffer $\mathcal{M}$ to store examples from previously encountered tasks in order to approximately compute the gradient of every previous task $\boldsymbol{g}_i^m$. Based on the strategies for manipulating the gradients of previous tasks $\boldsymbol{g}_i^m$, replay-based methods can be broadly categorized into experience replay methods (Rolnick et al., 2019; Buzzega et al., 2020; Arani et al., 2022; Caccia et al., 2022; Wang et al., 2023) and gradient alignment methods (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; Guo et al., 2020; Riemer et al., 2019; Gupta et al., 2020). Specifically, experience replay methods directly combine every approximated gradient of the previous task $\boldsymbol{g}_i^m$ with the gradient of the current task to help update model $\boldsymbol{\theta}_t$. As illustrated in Fig. 1(a), the gradient update direction $\boldsymbol{u}$ is derived by simply assigning equal weight to every $\boldsymbol{g}_i^m$. For gradient alignment methods, they favor weighting the gradients of previous tasks that conflict with the gradient of the current task, while disregarding the others. As exemplified in Fig. 1(b), the update direction $\boldsymbol{u}$ takes into account $\boldsymbol{g}_1^m$ that conflicts with the gradient of the current task $\boldsymbol{g}_3$, indicated by the negative inner product $\langle \boldsymbol{g}_3, \boldsymbol{g}_1^m \rangle$, while ignoring $\boldsymbol{g}_2^m$ that is not in conflict with $\boldsymbol{g}_3$.

[*]Equal contribution [†]Work was done when the author interned in Tencent AI Lab [1]City University of Hong Kong, Hong Kong SAR [2]Tencent AI Lab, Shenzhen, China [3]Tencent Youtu Lab, Shenzhen, China [4]Nanyang Technological University, Singapore. Correspondence to: Ying Wei <ying.wei@ntu.edu.sg>, Yichen Wu <wuyichen.am97@gmail.com>, Long-Kai Huang <hlongkai@gmail.com>.

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

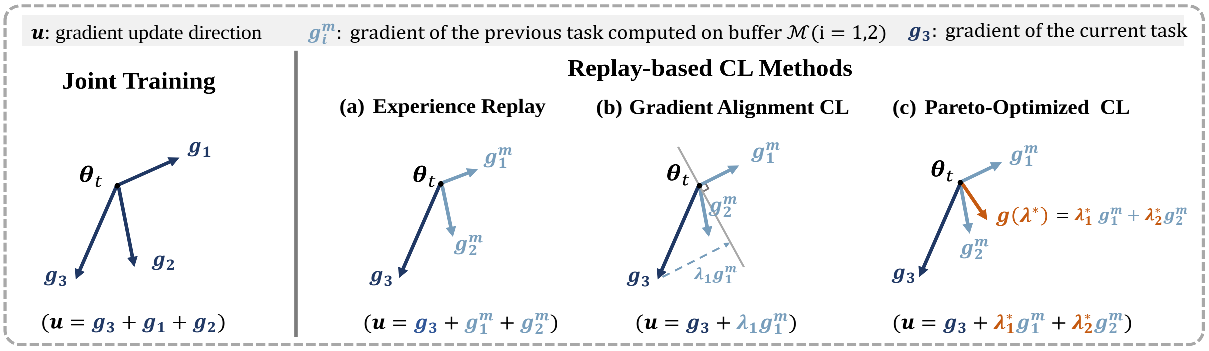

$\boldsymbol{u}$: gradient update direction
g_i^m: gradient of the previous task computed on buffer $\mathcal{M}$ (i = 1,2)
g_3: gradient of the current task
Joint Training
Replay-based CL Methods
(a) Experience Replay
(b) Gradient Alignment CL
(c) Pareto-Optimized CL
θ_t
g_1
g_2
g_3
g_1^m
g_2^m
$\lambda_1 g_1^m$
$g(\lambda^*) = \lambda_1^* g_1^m + \lambda_2^* g_2^m$
$(u = g_3 + g_1 + g_2)$
$(u = g_3 + g_1^m + g_2^m)$
$(u = g_3 + \lambda_1 g_1^m)$
$(u = g_3 + \lambda_1^* g_1^m + \lambda_2^* g_2^m)$

Figure 1. Gradient update $\boldsymbol{u}$ of different methods in the perspective of gradient weighting under a simplified scenario with only three tasks. Here $\boldsymbol{\theta}_t$ means the model parameters at the t-th streaming learning step, and the weights λ_i^* in (c) are obtained by our proposed Pareto-Optimized CL (POCL) algorithm, which flexibly models the gradients of previous tasks $\boldsymbol{g}_i^m$ and fully accounts for their interrelationships (see Sec. 3). In Sec. 2, we revisit these methods under the general scenario with more tasks.

Albeit achieving promising performance, for the current replay-based CL methods, the aforementioned weighting strategies on the gradient of previously seen tasks $\boldsymbol{g}_i^m$ lack enough flexibility and they do not adequately explore the intricate interrelationships among previous tasks. For instance, if the gradients of two previous tasks conflict with each other, simply merging them would lead to mutual cancellation of the gradients, thus potentially losing important knowledge from the previous tasks. Moreover, as demonstrated in (Lin et al., 2019; Momma et al., 2022), seemingly unrelated tasks can exhibit significant dependencies, and it is possible to enhance the overall performance across all previously learned tasks by effectively leveraging the dependencies among diverse tasks. As seen, without fully modeling the interrelations among previous tasks, the existing replay-based methods fall short in encapsulating the optimally integrated knowledge of previous tasks. This shortfall hinders the maximization of catastrophic forgetting mitigation, leading to sub-optimal performance of previous tasks.

Against the aforementioned issue, we first revisit current replay-based CL methods from the gradient weighting perspective and mathematically formulate them as a hierarchical gradient aggregation framework for joint training emulation, which combines the gradient of the current task and the gradient-weighted sum of each previous task in the update direction. Based on these understandings, we then specifically model the interrelationships among previous tasks by incorporating Pareto optimization into CL and optimize the weight $\boldsymbol{\lambda}_i$ assigned to the gradient of each past task $\boldsymbol{g}_i^m$ to get the aggregated gradient of previous tasks $\boldsymbol{g}(\boldsymbol{\lambda}^*)$ (as illustrated in Fig. 1(c)) for appropriately enhancing the integrated knowledge of the previous tasks and thus reducing catastrophic forgetting. Moreover, we propose to employ hyper-gradient to effectively avoid the potential conflict between the gradient of the current task and the Pareto-optimized gradient of previous tasks $\boldsymbol{g}(\boldsymbol{\lambda}^*)$, as illustrated in Fig. 1(c) that $\langle \boldsymbol{g}_3, \boldsymbol{g}(\boldsymbol{\lambda}^*)\rangle$ is non-negative. In this manner, our proposed method, called Pareto-Optimized CL (POCL), can not only ensure the performance of the current task but also maximize the performance of previous tasks and minimize catastrophic forgetting. Our main contributions are summarized as follows:

1) New Perspective. We mathematically revisit the current various replay-based CL methods and formulate them as a unified hierarchical gradient aggregation framework.

2) Effective Algorithm. Within the established hierarchical gradient aggregation framework, we propose the POCL algorithm for an enhanced gradient update direction $\boldsymbol{u}$, which takes into account not only the performance of the current task but also the interdependence among the previously learned tasks for maximizing their overall performance.

3) Superior Performance. We perform comprehensive experiments under various settings using three widely used datasets, which validate the effectiveness of the proposed POCL algorithm. Additionally, we conduct thorough ablation studies to analyze the individual impact of each component within the POCL algorithm, further enhancing our understanding of its performance.

2. Revisiting CL from a Gradient Perspective

For convenience, we present a simplified scenario with only three tasks in Fig. 1. In this section, we delve into a general scenario and formally revisit different kinds of replay-based CL methods.

Let's assume there are N sequential tasks, denoted as $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_N\}$. During the t-th streaming learning step, the model parameters are represented as $\boldsymbol{\theta}_t$. The gradient of the current training task $\mathcal{T}_n$ is referred to as $\boldsymbol{g}_n$ and the gradient of the previous task $\mathcal{T}_i$ is denoted as $\boldsymbol{g}_i^m$, where $i = 1, 2, ..., n-1$. Here $\boldsymbol{g}_i^m$ is computed by using randomly selected samples from the memory buffer $\mathcal{M}$ which stores a subset of the previous tasks. From the view of gradient, the main goal of current replay-based continual learning methods is to search for the gradient update direction $\boldsymbol{u}$ for optimizing model $\boldsymbol{\theta}_t$ during the t-th streaming learning step.

2.1. Experience Replay

One class of replay-based CL methods (Rolnick et al., 2019; Buzzega et al., 2020; Arani et al., 2022; Caccia et al., 2022), represented by ER (Rolnick et al., 2019), involves drawing samples from the buffer $\mathcal{M}$ and integrating them with the data from the current training task to optimize the model. Correspondingly, the update direction $\boldsymbol{u}$ is obtained as:

$$\boldsymbol{u} = \boldsymbol{g}_n + \sum_{i=1}^{n-1} \boldsymbol{g}_i^m = \boldsymbol{g}_n + \boldsymbol{g}_1^m + \boldsymbol{g}_2^m + \cdots + \boldsymbol{g}_{n-1}^m. \quad (1)$$

2.2. Gradient Alignment Methods

Another prominent category within replay-based CL methods is gradient alignment (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; Riemer et al., 2019; Gupta et al., 2020), exemplified by GEM (Lopez-Paz & Ranzato, 2017). The primary objective of these methods is to project the gradient $\boldsymbol{g}_n$ of the current training task $\mathcal{T}_n$ in a direction that minimizes potential negative impacts on preceding tasks. Mathematically, the update direction $\boldsymbol{u}$ is optimized by solving the following problem (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018):

$$\max_{\boldsymbol{u}} -\frac{1}{2}\|\boldsymbol{g}_n - \boldsymbol{u}\|_2^2, \; s.t. \langle \boldsymbol{u}, \boldsymbol{g}_i^m \rangle \geq 0, i = 1, \ldots, n-1. \quad (2)$$

The primary objective is to minimize the discrepancy between the update direction $\boldsymbol{u}$ and the gradient of the current task $\boldsymbol{g}_n$. Additionally, the non-negative inner product constraints, $\langle \boldsymbol{u}, \boldsymbol{g}_i^m \rangle \geq 0$, are imposed to ensure that the updated gradient $\boldsymbol{u}$ does not conflict with $\boldsymbol{g}_i^m$ to prevent performance degradation of previous tasks.

To solve Eq. (2), we propose to adopt the Lagrange multiplier (Boyd & Vandenberghe, 2004) and convert the objective into an unconstrained form:

$$\max_{\boldsymbol{u}} \min_{\lambda_i \geq 0} -\frac{1}{2}\|\boldsymbol{g}_n - \boldsymbol{u}\|_2^2 + \textstyle\sum_{i=1}^{n-1} \lambda_i \langle \boldsymbol{u}, \boldsymbol{g}_i^m \rangle, \quad (3)$$

where λ_i is a non-negative penalty coefficient. Let $\mathbb{Q}^{n-1} = \{(\lambda_1, ..., \lambda_{n-1}) | \lambda_i \geq 0\}$ and $\boldsymbol{g}(\boldsymbol{\lambda}) = \sum_{i=1}^{n-1} \lambda_i \boldsymbol{g}_i^m$, where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_{n-1}]$. Eq. (3) can be equivalently written as:

$$\max_{\boldsymbol{u}} \min_{\boldsymbol{\lambda} \in \mathbb{Q}^{n-1}} -\frac{1}{2}\|\boldsymbol{g}_n - \boldsymbol{u}\|_2^2 + \langle \boldsymbol{u}, \boldsymbol{g}(\boldsymbol{\lambda}) \rangle. \quad (4)$$

From Eq. (4), we can know that the feasible domains of the maximum and the minimum optimization processes are both convex sets, and the objective function is convex w.r.t. the variable of the minimum operation $\boldsymbol{\lambda}$ and concave w.r.t. the variable of the maximum operation $\boldsymbol{u}$. According to the MiniMax theorem (Du & Pardalos, 1995) (also detailed in Appendix A.1), we can swap the order of the maximum and the minimum operation in Eq. (4) and then derive the following optimization problem:

$$\begin{aligned}
&\min_{\boldsymbol{\lambda} \in \mathbb{Q}^{n-1}} \max_{\boldsymbol{u}} -\frac{1}{2}\|\boldsymbol{g}_n - \boldsymbol{u}\|_2^2 + \langle \boldsymbol{u}, \boldsymbol{g}(\boldsymbol{\lambda}) \rangle \\
&\overset{(a)}{\Leftrightarrow} \min_{\boldsymbol{\lambda} \in \mathbb{Q}^{n-1}} \frac{1}{2}\|\boldsymbol{g}(\boldsymbol{\lambda})\|_2^2 + \langle \boldsymbol{g}_n, \boldsymbol{g}(\boldsymbol{\lambda}) \rangle \\
&\overset{(b)}{\Leftrightarrow} \min_{\boldsymbol{\lambda} \in \mathbb{Q}^{n-1}} \frac{1}{2}\|\boldsymbol{g}(\boldsymbol{\lambda})\|_2^2 + \langle \boldsymbol{g}_n, \boldsymbol{g}(\boldsymbol{\lambda}) \rangle + \frac{1}{2}\|\boldsymbol{g}_n\|_2^2 \\
&\Leftrightarrow \min_{\boldsymbol{\lambda} \in \mathbb{Q}^{n-1}} \frac{1}{2}\|\boldsymbol{g}(\boldsymbol{\lambda}) + \boldsymbol{g}_n\|_2^2,
\end{aligned} \quad (5)$$

where (a) holds since the solution $\boldsymbol{u}^*$ of the maximum problem can be directly obtained as $\boldsymbol{u}^* = \boldsymbol{g}_n + \boldsymbol{g}(\boldsymbol{\lambda})$, and (b) holds since the term $\frac{1}{2}\|\boldsymbol{g}_n\|_2^2$ is independent of the optimization variable $\boldsymbol{\lambda}$. Consequently, the final gradient update direction is $\boldsymbol{u}^* = \boldsymbol{g}_n + \boldsymbol{g}(\boldsymbol{\lambda}^*)$, where $\boldsymbol{\lambda}^*$ is obtained by solving the last minimization problem in Eq. (5). The objective of determining $\boldsymbol{\lambda}^*$ reveals that a larger λ_i^* will be assigned to gradients $\boldsymbol{g}_i^m$ that are in conflict with $\boldsymbol{g}_n$.

2.3. Hierarchical Gradient Aggregation Framework

From the aforementioned derivations in Eq. (1) and (5), it is easily concluded that for replay-based CL methods, the gradient update direction $\boldsymbol{u}$ can be generally reformulated under the gradient weighting framework as:

$$\boldsymbol{u} = \boldsymbol{g}_n + \boldsymbol{g}(\boldsymbol{\lambda}) = \boldsymbol{g}_n + \sum_{i=1}^{n-1} \lambda_i \boldsymbol{g}_i^m. \quad (6)$$

Observably, the update gradient direction $\boldsymbol{u}$ consists of two components, the gradient of the current task $\boldsymbol{g}_n$ and the aggregation of gradients $\boldsymbol{g}_i^m$ from past tasks. Given the memory buffer with a fixed size $|\mathcal{M}|$, as the number of training tasks increases, the number of samples per task from past tasks stored at the buffer $\mathcal{M}$ gradually decreases. Consequently, compared to the gradient $\boldsymbol{g}_n$ of the current task $\mathcal{T}_n$, the importance of the gradient $\boldsymbol{g}_i^m$ progressively diminishes during the model's update process. Besides, for gradient alignment methods, as explained in Eq. (5), they always apply a fixed weight to $\boldsymbol{g}_n$ but varying weights λ_i to different $\boldsymbol{g}_i^m$ which are optimized based on their extent of conflicting with $\boldsymbol{g}_n$. These observations collectively suggest that replay-based CL methods can be viewed as a form of **hierarchical gradient aggregation framework** that prioritizes the gradient of the current training task $\boldsymbol{g}_n$ and the aggregation of gradients $\boldsymbol{g}_i^m$ of previous tasks primarily serve a regularization role in preventing forgetting.

Based on this understanding, the main task of searching for $\boldsymbol{u}$ can be reframed as the assignment of different weights λ_i to different $\boldsymbol{g}_i^m$ of previous tasks with a fixed weight on the prioritized gradient of the current task $\boldsymbol{g}_n$. From Eqs. (1) and (5), we have that experience replay methods choose the uniform weighting strategy, while gradient alignment methods focus more on past tasks that significantly conflict with the current task's gradient. However, these existing weighting strategies exhibit limited flexibility and insufficiently explore the intricate relationships among past tasks.

3. Pareto-Optimized Continual Learning

In this section, following the aforementioned hierarchical gradient aggregation framework, we aim at fully modeling the interrelationships among previous tasks and propose a Pareto-optimized CL algorithm for assigning more flexible and nearly optimal weights λ_i in Eq. (6).

3.1. Model Optimization

As discussed in (Lin et al., 2019; Momma et al., 2022), seemingly unrelated tasks can exhibit significant dependencies. Therefore, it is possible to enhance the overall performance across all previously learned tasks by effectively leveraging the dependencies among diverse tasks. To this end, we propose to model the relationships of past tasks based on the Pareto optimality concept (Pareto, 1964) to flexibly optimize the weighting schemes $\boldsymbol{g}(\boldsymbol{\lambda})$ of previous tasks in Eq. (6) in order to enhance the overall performance across all previously learned tasks. Here the Pareto optimality (Pareto, 1964) represents a solution where no action can improve the performance of one task without adversely affecting the performance of other tasks. This indicates that the model has reached an optimal state, where simultaneous performance improvement of different tasks is no longer possible.

Specifically, by considering $\boldsymbol{v}$ as the overall to-be-estimated gradient direction aggregated from all the previously learned tasks, our goal is to guarantee that the direction $\boldsymbol{v}$ can benefit all the tasks learned so far. This can be achieved by maximizing the minimum inner product of the previous gradient $\boldsymbol{g}_i^m$ and $\boldsymbol{v}$, where $i \in \{1, 2, ..., n-1\}$. Mathematically, this can be formulated as:

$$\max_{\boldsymbol{v}} \min_{1 \le i \le n-1} \langle \boldsymbol{g}_i^m, \boldsymbol{v} \rangle - \frac{1}{2}\|\boldsymbol{v}\|_2^2, \tag{7}$$

where the second term in the objective function is to constrain the gradient norm to avoid infinity.

Define $\mathbb{P}^{n-1} = \{(\lambda_1, ..., \lambda_{n-1}) | \lambda_i \ge 0, \sum_{i=1}^{n-1} \lambda_i = 1\}$ and $\boldsymbol{g}(\boldsymbol{\lambda}) = \sum_{i=1}^{n-1} \lambda_i \boldsymbol{g}_i^m$. For the first term in Eq. (7), we can prove that

$$\min_{1 \le i \le n-1} \langle \boldsymbol{g}_i^m, \boldsymbol{v} \rangle \Leftrightarrow \min_{\boldsymbol{\lambda} \in \mathbb{P}^{n-1}} \langle \boldsymbol{g}(\boldsymbol{\lambda}), \boldsymbol{v} \rangle, \tag{8}$$

Algorithm 1 Frank-Wolfe Algorithm for Solving Eq. (10)

```
Input: Initialization $\boldsymbol{\lambda} = [\frac{1}{n-1}, \dots, \frac{1}{n-1}]$
Output: The coefficient vector $\boldsymbol{\lambda}$ of previous tasks
1: Precompute $\boldsymbol{D} = \boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}$, where $\boldsymbol{G} = [\boldsymbol{g}_1^{\mathrm{T}}, \dots, \boldsymbol{g}_{n-1}^{\mathrm{T}}]$
2: repeat
3:     $\boldsymbol{\alpha} = \mathrm{argmin}_{\boldsymbol{\alpha} \in \{\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{1}=1, \boldsymbol{\alpha} \succeq \mathbf{0}\}} \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\lambda}$
4:     $L = \boldsymbol{\lambda} + \eta(\boldsymbol{\alpha} - \boldsymbol{\lambda})$
5:     $\eta = \mathrm{argmin}_{\eta \in [0,1]}\ L^{\mathrm{T}}\boldsymbol{D}L$
6:     $\boldsymbol{\lambda} \leftarrow (1-\eta)\,\boldsymbol{\lambda} + \eta\boldsymbol{\alpha}$
7: until $\eta \sim 0$ or Reaching the maximum iteration number
```

based on the two inequalities: 1) Since $0 \le \lambda_i \le 1$ and $\boldsymbol{\lambda} \in \mathbb{P}^{n-1}$, it always holds that $\langle \boldsymbol{g}(\boldsymbol{\lambda}), \boldsymbol{v} \rangle \ge \min_{1 \le i \le n-1} \langle \boldsymbol{g}_i^m, \boldsymbol{v} \rangle$, so we have $\min_{\boldsymbol{\lambda} \in \mathbb{P}^{n-1}} \langle \boldsymbol{g}(\boldsymbol{\lambda}), \boldsymbol{v} \rangle \ge \min_{1 \le i \le n-1} \langle \boldsymbol{g}_i^m, \boldsymbol{g}(\boldsymbol{\lambda}) \rangle$; 2) Since $\langle \boldsymbol{g}_i^m, \boldsymbol{v} \rangle$ is a special case of $\langle \boldsymbol{g}(\boldsymbol{\lambda}), \boldsymbol{v} \rangle$, we can deduce that $\min_{\boldsymbol{\lambda} \in \mathbb{P}^{n-1}} \langle \boldsymbol{g}(\boldsymbol{\lambda}), \boldsymbol{v} \rangle \le \min_{1 \le i \le n-1} \langle \boldsymbol{g}_i^m, \boldsymbol{v} \rangle$.

By substituting Eq. (8) into Eq. (7), we can obtain that

$$\max_{\boldsymbol{v}} \min_{\boldsymbol{\lambda} \in \mathbb{P}^{n-1}} \langle \boldsymbol{g}(\boldsymbol{\lambda}), \boldsymbol{v} \rangle - \frac{1}{2}\|\boldsymbol{v}\|_2^2. \tag{9}$$

According to the MiniMax theorem (Appendix A.1), we can swap the order of minimum and maximum operations in Eq. (9). Then we can get the solution of the maximum problem as $\boldsymbol{v}^* = \boldsymbol{g}(\boldsymbol{\lambda})$ and then obtain the $\boldsymbol{\lambda}^*$ by solving the following minimization problem:

$$\min_{\boldsymbol{\lambda} \in \mathbb{P}^{n-1}} \frac{1}{2}\|\boldsymbol{g}(\boldsymbol{\lambda})\|_2^2 \Leftrightarrow \min_{\boldsymbol{\lambda} \in \mathbb{P}^{n-1}} \frac{1}{2}\|\textstyle\sum_{i=1}^{n-1} \lambda_i \boldsymbol{g}_i^m\|_2^2. \tag{10}$$

This minimization problem can be solved utilizing the Frank-Wolfe algorithm (Frank et al., 1956; Jaggi, 2013; Sener & Koltun, 2018) as presented in Alg. 1. Then, we compute the optimized gradient update direction as:

$$\boldsymbol{u}^* = \boldsymbol{g}_n + \boldsymbol{v}^* = \boldsymbol{g}_n + \boldsymbol{g}(\boldsymbol{\lambda}^*) = \boldsymbol{g}_n + \textstyle\sum_{i=1}^{n-1} \lambda_i^* \boldsymbol{g}_i^m. \tag{11}$$

3.2. Gradient Implementation

To avoid the potential conflict between the overall update direction $\boldsymbol{v}$ for previous tasks and the gradient of the current task $\boldsymbol{g}_n$ as much as possible, we propose to employ the hyper-gradient $\boldsymbol{g}_j^{\mathrm{HD}}$ ($j \in \{1, 2, \dots, n\}$) to implement $\boldsymbol{g}_i^m$ ($i \in \{1, 2, \dots, n-1\}$) in Eq. (7) and $\boldsymbol{g}_n$. Concretely, at the t-th training step, the hyper-gradient $\boldsymbol{g}_j^{\mathrm{HD}}$, is calculated via the following bi-level iterative form as (Riemer et al., 2019; Gupta et al., 2020):

$$\boldsymbol{g}_j^{\mathrm{HD}} = \frac{\partial \mathcal{L}(f_{\tilde{\boldsymbol{\theta}}_{t+1}}(x^m), y^m)}{\partial \boldsymbol{\theta}_t}, \quad \text{where } \tilde{\boldsymbol{\theta}}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(f_{\theta_t}(x_j), y_j), \tag{12}$$

where $f(\cdot)$ denotes the model with parameter $\boldsymbol{\theta}$; (x^m, y^m) means the samples drawn from the buffer $\mathcal{M}$ which stores

Algorithm 2 The Entire Algorithm Implementation for the Proposed POCL

```
Input: At the t-th streaming training step, current training task $\mathcal{T}_n$, memory buffer $\mathcal{M}$, learning rates $\alpha$ and $\beta$, network parameter $\boldsymbol{\theta}_t$ for classification
Output: $\boldsymbol{\theta}_{t+1}$
1: Sample from memory buffer: $(x^m, y^m) \sim \mathcal{M}$
2: Sample from memory buffer for previous task $\mathcal{T}_i$: $(x_i, y_i) \sim \mathcal{M}_i, i \in \{1, 2, ..., n-1\}, \mathcal{M}_i \in \mathcal{M}$
3: Sample for the current task: $(x_n, y_n) \sim \mathcal{T}_n$
4: Compute the hyper-gradient $\boldsymbol{g}_j^{\mathrm{HD}}, j \in \{1, \ldots, n\}$ based on Eq. (12)
5: Compute the Pareto optimal weights $\boldsymbol{\lambda}^*$ based on Alg. 1 and get $\boldsymbol{g}(\boldsymbol{\lambda}^*) = \sum_{i=1}^{n-1} \lambda_i^* \boldsymbol{g}_i^{\mathrm{HD}}$
6: Compute the update direction: $\boldsymbol{u}^* = \boldsymbol{g}_n^{\mathrm{HD}} + \boldsymbol{g}(\boldsymbol{\lambda}^*)$
7: Update network parameters: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \beta \boldsymbol{u}^*$
8: Update the memory buffer $\mathcal{M}$ with $(x_n, y_n)$ following (Buzzega et al., 2020)
```

examples of all seen tasks $\mathcal{T}_{[1:n]}$; (x_j, y_j) are samples belonging to the task $\mathcal{T}_j$ which are drawn from $\mathcal{M}$ (when $j = 1, \ldots, n-1$) or $\mathcal{T}_n$ (when $j = n$); $\mathcal{L}(\cdot)$ is the cross-entropy loss function; $\tilde{\boldsymbol{\theta}}_{t+1}$ denotes the auxiliary model parameter to help compute $\boldsymbol{g}_j^{\mathrm{HD}}$. As seen, the inner-level loss $\mathcal{L}(f_{\theta_t}(x_j), y_j)$ is related to the specific task $\mathcal{T}_j$, while the outer-level objective is related to all the seen tasks $\mathcal{T}_{[1:n]}$.

Let $\mathcal{L}_{\theta_t}^m \triangleq \mathcal{L}(f_{\tilde{\boldsymbol{\theta}}_{t+1}}(x^m), y^m)$ and $\mathcal{L}_{\theta_t}^j \triangleq \mathcal{L}(f_{\theta_t}(x_j), y_j)$. Then we can transform $\boldsymbol{g}_j^{\mathrm{HD}}$ into the following expression:

$$\boldsymbol{g}_j^{\mathrm{HD}} \approx \frac{\partial(\mathcal{L}_{\theta_t}^m - \alpha \nabla_{\theta_t} \mathcal{L}_{\theta_t}^m \cdot \nabla_{\theta_t} \mathcal{L}_{\theta_t}^j)}{\partial \theta_t} \triangleq \frac{\partial \mathcal{L}^{\mathrm{HD}}}{\partial \theta_t}, \quad (13)$$

where $\mathcal{L}^{\mathrm{HD}} \triangleq \mathcal{L}_{\theta_t}^m - \alpha \nabla_{\theta_t} \mathcal{L}_{\theta_t}^m \cdot \nabla_{\theta_t} \mathcal{L}_{\theta_t}^j$. Please refer to Appendix A.2 for the detailed proof of Eq. (13).

As shown in Eq. (13), the hyper-gradient $\boldsymbol{g}_j^{\mathrm{HD}}$ aims to minimize the loss $\mathcal{L}^{\mathrm{HD}}$. Observing the second item of $\mathcal{L}^{\mathrm{HD}}$, it is easily known that the optimization goal is to promote the alignment between the gradient of the specific task $\nabla_{\theta_t} \mathcal{L}_{\theta_t}^j$ and the overall gradient on all the seen tasks $\nabla_{\theta_t} \mathcal{L}_{\theta_t}^m$ (Riemer et al., 2019; Gupta et al., 2020). Since the gradient of every specific task is aligned with $\nabla_{\theta_t} \mathcal{L}_{\theta_t}^m$, there is a small probability that different $\boldsymbol{g}_j^{\mathrm{HD}}$ $(j = 1, \ldots, n)$ is conflicting with each other. When replacing $\boldsymbol{g}_i^m$ and $\boldsymbol{g}_n$ in Eq. (6) with the hyper-gradient $\boldsymbol{g}_j^{\mathrm{HD}}$ in Eq. (12), according to Eq. (11), the ultimate Pareto-optimized gradient update direction is $\boldsymbol{u}^* = \boldsymbol{g}_n^{\mathrm{HD}} + \boldsymbol{v}^* = \boldsymbol{g}_n^{\mathrm{HD}} + \sum_{i=1}^{n-1} \lambda_i^* \boldsymbol{g}_i^{\mathrm{HD}}$. As seen, $\boldsymbol{v}^*$ is a weighted combination of different $\boldsymbol{g}_i^{\mathrm{HD}}$ $(i = 1, \ldots, n-1)$, there is a high probability that $\boldsymbol{g}_n^{\mathrm{HD}}$ and $\boldsymbol{v}^*$ would not conflict with each other. Consequently, the final gradient update direction $\boldsymbol{u}^*$ would be beneficial for both the current and previously learned tasks. The advantage of the proposed hyper-gradient-based implementation will be validated in Table 5. In consequence, we propose the Pareto-Optimized CL (POCL) algorithm with the hyper-gradient implementation. The complete algorithm is outlined in Alg. 2. Please note that in Fig. 1(c), we omit the hyper-gradient for the sake of readability.

4. Experiments

In this section, we conduct comprehensive experiments to evaluate the effectiveness of the proposed POCL based on diverse benchmark datasets and different CL settings. Besides, we provide a series of ablation studies to analyze and evaluate the specific role of each component in our method.

4.1. Evaluation Protocol

Benchmark Datasets. Following (Buzzega et al., 2020; Arani et al., 2022; Wang et al., 2023), we select three widely-used datasets with varying complexity for the subsequent CL experiments, *i.e.*, Split CIFAR-10, Split CIFAR-100, and Split TinyImageNet (Buzzega et al., 2020). Split CIFAR-10 and Split CIFAR-100 are derived from the CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009), respectively. For Split CIFAR-10, the number of tasks N is 5 and each task contains 2 classes. For Split CIFAR-100, N is 20 and each task is composed of 5 classes. Split Tiny-ImageNet is divided into 20 tasks, each containing 10 classes. More details about datasets are included in Appendix C.

Implementation Details. Consistent to (Chrysakis & Moens, 2023; Lopez-Paz & Ranzato, 2017), we utilize the widely-adopted Reduced ResNet-18 (He et al., 2016) as the network backbone architecture to implement the proposed POCL method. During the training, the stochastic gradient descent (SGD) optimizer is used for optimizing the model and the batch size is set as 32. For the experiments on different datasets and various CL settings, the learning rates α and β in Alg. 2 are fixed as 0.03 and the sampling batch size for memory buffer $\mathcal{M}$ is 300. Please note that all the comparison experiments are executed under the online class incremental setting, where each task is trained for only one epoch and the task identity is not provided during inference.

Baselines. For comprehensive comparisons, we adopt various types of state-of-the-art CL approaches. These include the regularization-based method EWC (Huszár, 2017); a range of gradient alignment methods such as AGEM (Chaudhry et al., 2018), GEM (Lopez-Paz & Ranzato, 2017), MER (Riemer et al., 2019), and La-MAML (Gupta et al., 2020); and experience replay methods like ER (Rolnick et al., 2019), DER (Buzzega et al., 2020), DER++ (Buzzega et al., 2020), CLSER (Arani et al., 2022), ER-ACE (Caccia et al., 2022) and CBA (Wang et al., 2023).

Evaluation Metrics. To fairly and comprehensively validate the effectiveness of our proposed methods, we adopt

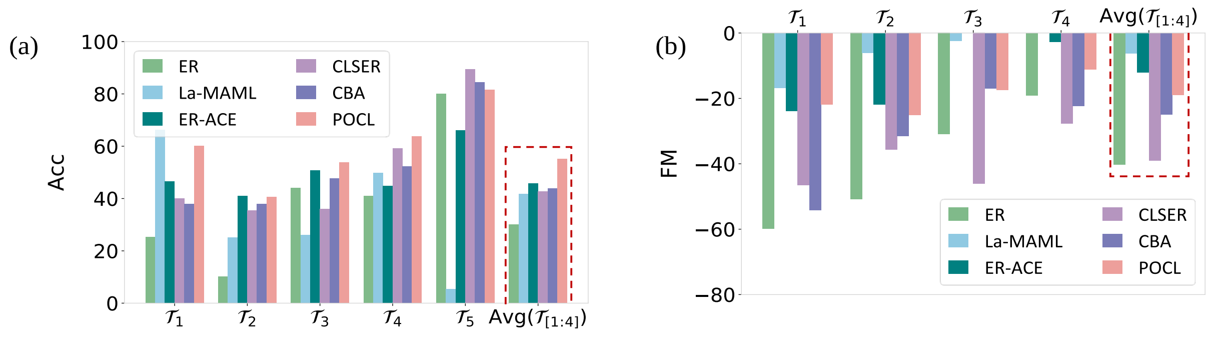


Figure 2. (a) accuracy (Acc) and (b) forgetting measure (FM) of each task achieved by different competitive CL methods on Split CIFAR-10 with $N = 5$ and buffer size $|\mathcal{M}|$ as 1k. Here $\mathcal{T}_{[1:4]}$ denotes the previously learned tasks and $\mathcal{T}_5$ is the current task, and Avg($\mathcal{T}_{[1:4]}$) means the average performance of past tasks $\mathcal{T}_{[1:4]}$.

Table 1. Average accuracy of previous tasks $\mathcal{T}_{[1:n-1]}$ and all the tasks $\mathcal{T}_{[1:n]}$, respectively, on the Split CIFAR10 with $|\mathcal{M}|$ =1k and $N = 5$ with an increasing number of all the seen tasks n ($n = 2, 3, 4, 5$).

	$n=2$		$n=3$		$n=4$		$n=5$	
Methods	Avg ($\mathcal{T}_1$)	Avg($\mathcal{T}_{[1:2]}$)	Avg($\mathcal{T}_{[1:2]}$)	Avg($\mathcal{T}_{[1:3]}$)	Avg($\mathcal{T}_{[1:3]}$)	Avg($\mathcal{T}_{[1:4]}$)	Avg($\mathcal{T}_{[1:4]}$)	Avg($\mathcal{T}_{[1:5]}$)
ER	71.22±7.40	66.16±2.29	32.31±6.78	46.51±3.85	35.52±6.41	41.95±7.25	32.51±6.15	42.04±4.33
La-MAML	79.82±3.56	53.14±3.11	51.51±1.06	37.62±2.15	40.53±2.37	38.45±2.68	43.49±1.72	35.89±1.35
ER-ACE	73.50±7.03	66.68±4.07	**66.69±1.37**	53.84±2.84	46.66±6.36	47.58±6.07	47.42±5.38	51.17±3.44
CLSER	71.47±2.06	71.28±0.53	41.57±5.24	55.12±3.51	39.38±3.03	51.25±2.30	43.90±2.01	53.06±1.58
CBA	79.65±3.58	73.21±0.91	52.70±3.85	57.95±1.46	47.18±3.86	53.56±1.12	44.00±3.51	52.08±0.15
POCL	**80.86±4.62**	**73.34±0.61**	55.19±5.13	**60.56±3.22**	**49.94±2.56**	**56.23±1.71**	**52.71±3.34**	**58.50±2.43**

several representative evaluation metrics for quantitative comparisons, including Average Accuracy, Forgetting Measure (Lopez-Paz & Ranzato, 2017), and Anytime Average Accuracy (Caccia et al., 2022). The higher these indicators, the better the performance. Specifically,

- **Average Accuracy (Acc)**: It represents the average accuracy on all the previously seen tasks after completing the model training on N tasks $\mathcal{T}_{[1:N]}$, computed as $\mathrm{Acc} = \mathrm{Acc}_N = \frac{1}{N}\sum_{i=1}^{N} a_{i,N}$, where $a_{i,j} (i \leq j)$ is the accuracy of the task $\mathcal{T}_i$ after the training on the task $\mathcal{T}_j$.
- **Forgetting Measure (FM)**: This metric reflects the degree of forgetting that occurs in a model during sequential training. Concretely, it computes the average decrease from the best accuracy to the final accuracy after training on $\mathcal{T}_N$ across all N tasks. This is denoted as $\mathrm{FM} = \frac{1}{N}\sum_{i=1}^{N}(a_{i,N} - a_i^*)$, where a_i^* is the best accuracy of the task $\mathcal{T}_i$ achieved during training.
- **Anytime Average Accuracy (AAA)**: Different from Acc, this indicator quantifies the classification performance of the model throughout the entire learning process. Specifically, its definition is $\mathrm{AAA} = \frac{1}{N}\sum_{j=1}^{N}\mathrm{Acc_j} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{j}\sum_{i=1}^{j} a_{i,j}\right)$.

4.2. Experimental Results

Direct verification about POCL. To better understand the working mechanism of our proposed POCL, based on Split CIFAR-10, we first provide direct model verification. Specifically, in Fig. 2(a), for the representative comparison methods, we provide the test accuracy of model on every task $\mathcal{T}_i$ $(i = 1, \ldots, 5)$ after finishing the sequential training on all 5 tasks and the average accuracy on all the previously learned four tasks as Avg($\mathcal{T}_{[1:4]}$). The average accuracy on all the previously learned tasks after learning a new task is a direct measure of the extent to which the algorithm has preserved knowledge from previous tasks after the update on a new task. From the results in Fig. 2(a), we can find that: 1) Our proposed POCL is obviously superior to other CL methods on Avg($\mathcal{T}_{[1:4]}$). This finely substantiates the potential of the proposed POCL algorithm to boost the entire performance of all the previously learned tasks, which complies with our design motivations; 2) On the newly learned $\mathcal{T}_5$, the proposed POCL also exhibits superior performance. Consequently, POCL generally performs competitively on all five tasks. Table 1 reports the average accuracy on previously learned tasks Avg($\mathcal{T}_{[1:n-1]}$) and the overall average accuracy on all the tasks Avg($\mathcal{T}_{[1:n]}$) with the increasing of the number of all the seen tasks n. It is clearly observed that POCL effectively boosts the performance of previous tasks during sequential learning and finely achieves the obvious overall performance improvement over all the seen tasks.

Fig. 2(b) depicts the forgetting measure (FM) of different methods on every task. For Avg($\mathcal{T}_{1\sim4}$), POCL averagely obtains a higher FM score and outperforms ER, CLSER,

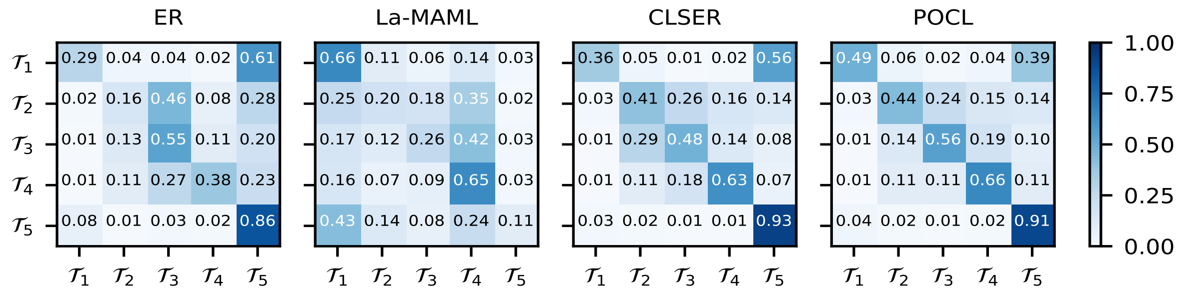


Figure 3. Task-based confusion matrix of various gradient-alignment-based CL methods on the Split CIFAR-10 dataset with $|\mathcal{M}|$=1k.

Table 2. Performance comparison on benchmark datasets under different memory buffer sizes $|\mathcal{M}|$.'-' indicates the implementation is both highly time-consuming and unstable. The full table with 95% confidence interval is in Appendix D.

Method	Split CIFAR-10 ($N=5$)				Split CIFAR-100 ($N=20$)				Split TinyImageNet ($N=20$)			
	$\lvert\mathcal{M}\rvert$ = 0.6k		$\lvert\mathcal{M}\rvert$ = 1k		$\lvert\mathcal{M}\rvert$ = 1k		$\lvert\mathcal{M}\rvert$ = 5k		$\lvert\mathcal{M}\rvert$= 2k		$\lvert\mathcal{M}\rvert$ = 5k	
	AAA	Acc	AAA	Acc	AAA	Acc	AAA	Acc	AAA	Acc	AAA	Acc
SGD	34.04	16.68	34.04	16.68	9.67	3.24	9.67	3.24	7.63	2.17	7.63	2.17
EWC (Huszár, 2017)	36.51	18.37	36.51	18.37	9.87	2.77	9.87	2.77	7.96	2.43	7.96	2.43
GEM (Lopez-Paz & Ranzato, 2017)	37.78	18.84	37.00	18.73	13.43	6.04	13.71	6.46	10.17	3.70	10.27	3.81
AGEM (Chaudhry et al., 2018)	37.67	18.51	37.62	18.03	10.61	3.75	10.80	3.52	7.66	2.33	7.79	2.40
ER (Rolnick et al., 2019)	54.68	39.43	54.91	42.04	17.86	11.89	20.67	14.87	16.27	10.52	16.10	11.89
MER (Riemer et al., 2019)	45.39	24.42	50.99	36.15	–	–	–	–	–	–	–	–
La-MAML (Gupta et al., 2020)	47.89	30.53	46.08	35.89	17.37	10.03	19.05	12.57	15.81	8.02	16.32	9.24
DER (Buzzega et al., 2020)	49.06	25.80	48.25	23.90	10.96	3.71	10.47	3.68	8.04	2.46	7.65	2.04
DER++ (Buzzega et al., 2020)	57.17	47.03	61.01	50.31	17.30	8.72	17.08	8.98	12.42	5.57	11.93	5.26
ER-ACE (Caccia et al., 2022)	52.27	46.04	57.42	51.17	24.02	15.46	24.93	20.58	20.57	**13.23**	21.16	17.22
CLSER (Arani et al., 2022)	61.64	50.36	63.27	53.06	22.58	15.68	23.25	16.42	18.50	10.03	18.88	11.61
CBA (Wang et al., 2023)	63.41	51.47	65.47	52.08	22.46	15.54	23.07	15.87	18.79	11.43	18.98	10.98
POCL (Ours)	**63.62**	**53.42**	**66.23**	**58.50**	**26.68**	**16.54**	**36.34**	**33.36**	**21.56**	12.69	**25.48**	**19.40**

and CBA, which demonstrates the favorable capability in alleviating forgetting. Please note that instead of giving higher priority to the current task as in other comparison methods, ER-ACE and La-MAML both over-emphasize preserving the performance of a past task without fully exploring the potential of the current task, which leads to a quite low a_i^* and in turn an extremely high FM score, but a quite low accuracy on $\mathcal{T}_5$ as shown in Fig. 2(a). It is unfair to directly compare the FM with these two methods which have different design focuses from other baselines. This analysis is consistent with (Caccia et al., 2022).

Moreover, we provide the task-based confusion matrix in Fig. 3 to investigate the proficiency of the proposed POCL and other replay-based methods in managing task interrelationships. It is easily understood that a method that can balance task interrelationships and optimize the overall performance of all the tasks should, at a minimum, distinguish between tasks. By comparing the diagonal elements representing the correct classification probability of the task identity, we can find that in general, the proposed POCL exhibits a higher accuracy on all five tasks, and better balances every task, which comprehensively validates its effectiveness in capturing the interdependence among different tasks. More results are presented in Appendix D.

Performance comparison with AAA and Acc. Table 2 reports the average AAA and Acc of different CL methods on Split CIFAR-10, Split CIFAR-100, and Split-TinyImageNet, under different memory buffer size $|\mathcal{M}|$. As seen, with the increase of the buffer size $|\mathcal{M}|$, the performance of previous tasks can be better maintained and then almost all the comparing methods present an upward trend. Besides, from Split CIFAR-10 to Split CIFAR-100 to Split-TinyImageNet, as the task difficulty increases, all the approaches basically show a downward trend. However, our proposed POCL always achieves higher AAA and Acc scores, which almost consistently outperform other baselines. This indicates that POCL not only performs well on the final trained model but also maintains a sustained advantage throughout the entire streaming training process, which is crucial in CL.

Performance comparison with FM. Table 3 compares the performance of different CL methods in mitigating forgetting and lists the average FM. As $|\mathcal{M}|$ increases, the FMs of other comparative methods improve very little, or even decrease. However, our proposed POCL always shows a significant performance improvement on different datasets.

Table 3. The Forgetting Measure (FM) with 95% confidence interval on benchmark datasets under different memory buffer size $|\mathcal{M}|$. All the reported results are averagely computed over 5 repetitions.

Method	Split CIFAR-10 ($N=5$)		Split CIFAR-100 ($N=20$)		Split TinyImageNet ($N=20$)	
	$\|\mathcal{M}\|$ = 0.6k	$\|\mathcal{M}\|$ = 1k	$\|\mathcal{M}\|$ = 1k	$\|\mathcal{M}\|$ = 5k	$\|\mathcal{M}\|$ = 2k	$\|\mathcal{M}\|$ = 5k
SGD	-61.01 $\pm$3.30	-61.01 $\pm$3.30	-54.24 $\pm$1.20	-54.24 $\pm$1.20	-43.58 $\pm$0.58	-43.58 $\pm$0.58
EWC (Huszár, 2017)	-64.21 $\pm$0.86	-64.21 $\pm$0.86	-56.55 $\pm$1.30	-56.55 $\pm$1.30	-44.30 $\pm$1.70	-44.30 $\pm$1.70
GEM (Lopez-Paz & Ranzato, 2017)	-58.09 $\pm$3.90	-57.64 $\pm$2.70	-45.83 $\pm$0.69	-50.61 $\pm$1.90	-42.30 $\pm$0.11	-42.71 $\pm$0.03
AGEM (Chaudhry et al., 2018)	-66.17 $\pm$1.61	-66.61 $\pm$1.60	-60.51 $\pm$0.34	-60.89 $\pm$0.37	-45.61 $\pm$0.04	-45.51 $\pm$0.27
ER (Rolnick et al., 2019)	-35.68 $\pm$3.20	-32.16 $\pm$5.10	-46.25 $\pm$0.47	-48.54 $\pm$1.10	-37.90 $\pm$0.28	-37.43 $\pm$0.81
MER (Riemer et al., 2019)	-32.78 $\pm$0.81	-23.33 $\pm$8.06	–	–	–	–
DER (Buzzega et al., 2020)	-54.60 $\pm$2.80	-55.22 $\pm$1.70	-60.58 $\pm$0.38	-59.94 $\pm$1.40	-47.11 $\pm$0.65	-46.27 $\pm$0.85
DER++ (Buzzega et al., 2020)	-24.00 $\pm$1.30	-28.01 $\pm$1.31	-58.61 $\pm$0.60	-61.23 $\pm$0.19	-47.13 $\pm$2.00	-47.65 $\pm$0.76
CLSER (Arani et al., 2022)	-29.03 $\pm$3.70	-31.38 $\pm$1.70	-45.63 $\pm$0.62	-50.71 $\pm$0.86	-44.45 $\pm$0.32	-44.21 $\pm$0.14
CBA (Wang et al., 2023)	-27.15 $\pm$4.70	-27.31 $\pm$3.70	-44.30 $\pm$1.53	-50.16 $\pm$2.76	-38.17 $\pm$1.56	-40.53 $\pm$1.34
POCL (Ours)	**-23.27** $\pm$**2.50**	**-16.85** $\pm$**2.80**	**-37.61** $\pm$0.63	**-17.55** $\pm$**0.36**	**-31.63** $\pm$**1.60**	**-20.22** $\pm$**1.20**

Table 4. Performance on the Split CIFAR-10 with $|\mathcal{M}|$=1k under different types of imbalanced CL settings, *i.e.*, Normal and Reversed.

Methods	Normal		Reversed		Methods	Normal		Reversed	
	AAA	Acc	AAA	Acc		AAA	Acc	AAA	Acc
SGD	36.51 $\pm$0.40	16.38 $\pm$0.33	35.32 $\pm$0.87	17.74 $\pm$0.22	ER	52.28 $\pm$0.65	32.59 $\pm$1.86	46.78 $\pm$3.01	27.89 $\pm$2.46
EWC	38.95 $\pm$0.25	16.95 $\pm$0.31	37.82 $\pm$0.27	17.91 $\pm$0.19	DER	47.09 $\pm$1.33	16.89 $\pm$0.69	40.37 $\pm$1.54	18.12 $\pm$0.63
GEM	41.36 $\pm$0.44	18.03 $\pm$0.53	38.71 $\pm$1.08	18.24 $\pm$0.35	DER++	61.97 $\pm$0.63	44.04 $\pm$2.06	58.52 $\pm$0.87	39.43 $\pm$3.29
AGEM	38.33 $\pm$0.25	17.48 $\pm$0.52	36.54 $\pm$0.39	17.52 $\pm$0.32	ER-ACE	61.47 $\pm$1.42	44.12 $\pm$2.33	60.21 $\pm$0.17	48.16 $\pm$1.79
MER	54.61 $\pm$1.39	35.15 $\pm$1.01	52.24 $\pm$1.92	39.47 $\pm$1.77	CLSER	61.87 $\pm$0.41	48.04 $\pm$0.72	55.32 $\pm$1.57	42.38 $\pm$2.97
La-MAML	36.17 $\pm$1.25	28.99 $\pm$0.78	31.79 $\pm$2.03	31.68 $\pm$1.42	POCL	**66.87** $\pm$**1.92**	**54.82** $\pm$**1.55**	**64.45** $\pm$**1.38**	**58.79** $\pm$**2.66**

Table 5. Ablation study on the proposed POCL. Here HD is the abbreviation for hyper-gradient derived in Sec. 3.2.

HD	Pareto	Split CIFAR-10 ($\|\mathcal{M}\|$=1k)			Split CIFAR-100 ($\|\mathcal{M}\|$=5k)		
		AAA	Acc	FM	AAA	Acc	FM
✗	✗	36.80	19.45	-55.54	16.20	9.05	-54.91
✗	✓	39.03	21.12	-52.98	18.98	11.34	-52.67
✓	✗	63.95	54.57	-21.99	35.89	30.47	-18.84
✓	✓	**66.23**	**58.50**	**-16.85**	**36.34**	**33.36**	**-17.55**

Table 6. Performance on Split CIFAR-10 with 95% confidence interval on the smaller 3-layer DNN with $|\mathcal{M}|$=1k. The results are averagely computed over 5 runs.

Methods	Split-CIFAR10 ($\|\mathcal{M}\|$ =1k)		
	AAA	Acc	FM
SGD	34.96 $\pm$0.27	16.08 $\pm$0.10	-61.63 $\pm$0.06
EWC	35.12 $\pm$0.12	16.38 $\pm$0.08	-60.65 $\pm$0.24
ER	56.82 $\pm$0.83	39.05 $\pm$1.80	-35.83 $\pm$0.81
A-GEM	35.88 $\pm$0.09	14.15 $\pm$0.27	-61.90 $\pm$0.22
GEM	46.86 $\pm$0.68	28.02 $\pm$0.55	-43.39 $\pm$1.17
DER	50.31 $\pm$0.19	29.04 $\pm$0.22	-49.38 $\pm$0.19
DER++	56.99 $\pm$0.08	42.30 $\pm$0.18	-34.07 $\pm$0.24
MER	58.18 $\pm$0.34	35.14 $\pm$0.84	-34.37 $\pm$0.74
CLSER	60.25 $\pm$0.12	43.82 $\pm$0.25	-33.97 $\pm$0.04
POCL (Ours)	**61.09** $\pm$**0.62**	**46.14** $\pm$**0.43**	**-22.13** $\pm$**0.42**

The underlying reason is that the derived hyper-gradient weighting formulation makes POCL able to fully exploit the memory buffer for more accurate gradient alignment. For POCL, while the introduced Pareto optimization aims to help improve the performance of entire previous tasks (as verified in Table 2), from another perspective, this also avoids forgetting to a certain extent, thus helping POCL obtain higher FM scores. As discussed in the analysis for Fig. 2, to avoid confusion, we defer reporting the FM results for ER-ACE and La-MAML to Appendix D.

Performance comparisons on more realistic settings. To comprehensively evaluate the effectiveness of our proposed methods, based on Split CIFAR-10, we additionally execute the comparing experiments on two more realistic CL settings, including Normal class imbalanced CL and Reversed class imbalanced CL. Specifically, for the Normal setting, the number of samples possessed by each streaming task is in a decreasing order, while Reversed takes an increasing order. More details are included in Appendix D. The corresponding comparison results are reported in Table 4. As seen, even under these more challenging scenarios, our proposed POCL still shows superior performance.

Ablation study on each component of POCL. To evaluate the role of each component of POCL, we conduct an ablation study based on Split CIFAR-10 with buffer size $|\mathcal{M}|$=1k. Table 5 presents the performance of different variants of the proposed gradient weighting framework. From the results, we can see that 1) the introduction of the hypergardient indeed introduces large performance gains; 2) through hyper-gradient design, it unleashes the potential of the Pareto optimization and leads to significant performance improvement. The Pareto optimization and the hyper-gradient part are the two integral components of our proposed POCL.

More comparisons on different backbones. To explore the versatility of our method across different backbones, based on Split CIFAR-10 with $|\mathcal{M}|$=1k, we also incorporate a smaller three-layer convolutional network as an additional backbone for experimental analysis. The results are provided in Table 6. Compared to Table 2 and Table 3 with Reduced ResNet-18 as the backbone, although all the comparison methods show a relative performance degradation under the smaller backbone, POCL still surpasses other CL methods on all the evaluation metrics. The results demonstrate that the proposed POCL algorithm has excellent applicability across various backbone models.

5. Related Work

Continual Learning Categories. In the field of continual learning, various approaches have been proposed to address catastrophic forgetting in recent years. These existing approaches can be broadly categorized into three classes: regularization-based, parameter-isolation-based, and replay-based methods. Specifically, regularization-based methods (Kirkpatrick et al., 2017; Huszár, 2017; Ritter et al., 2018; Zenke et al., 2017; Yang et al., 2019) intend to design different regularization techniques to preserve important parameters for previously learned tasks. Parameter-isolation-based approaches (Serra et al., 2018; Mallya & Lazebnik, 2018; Fernando et al., 2017; Aljundi et al., 2017; Ardywibowo et al., 2022; Wang et al., 2022) focus on isolating task-specific parameters to prevent interference between tasks. Replay-based methods (Rolnick et al., 2019; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; Aljundi et al., 2019; Buzzega et al., 2020; Arani et al., 2022; Wang et al., 2023; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; Guo et al., 2020; Riemer et al., 2019; Wu et al., 2024) hope to preserve previous knowledge by utilizing a small memory buffer $\mathcal{M}$, which retains samples from past tasks.

Replay-based Methods. According to different update strategies for samples in the memory buffer $\mathcal{M}$, replay-based methods can be further classified into gradient alignment methods and experience replay methods. Specifically, gradient alignment methods represented by GEM (Lopez-Paz & Ranzato, 2017) utilize a small memory buffer to store samples from previous tasks and aim to find a gradient update direction $\boldsymbol{u}$ adhering to two main constraints: 1) the to-be-estimated $\boldsymbol{u}$ should be as close as possible to the gradient of the current task $\boldsymbol{g}_n$ for improving the performance on the current task; 2) the inner product $\langle \boldsymbol{u}, \boldsymbol{g}_i \rangle$ between the to-be-updated gradient $\boldsymbol{u}$ and the gradient of every past task $\boldsymbol{g}_i$, where $1 \leq i < n$, should be non-negative to prevent adverse effect on the performance of past task, thereby alleviating forgetting. To accelerate the optimization process of GEM, instead of computing the individual gradient of each previous task, AGEM (Chaudhry et al., 2018) and MEGA (Guo et al., 2020) have proposed to compute an average past gradient $\boldsymbol{g}_{avg}$ to execute the aforementioned inner-product constraint. The average past gradient $\boldsymbol{g}_{avg}$ is calculated using examples randomly sampled from the memory buffer. Similarly, MER (Riemer et al., 2019) and La-MAML (Gupta et al., 2020) attempt to align the gradient of the current task $\boldsymbol{g}_n$ with the average gradient $\boldsymbol{g}_{avg}$ via a bi-level optimization procedure. For experience replay methods, they usually draw examples from the buffer $\mathcal{M}$ and directly utilize them to update the model. For example, Der++ (Buzzega et al., 2020) utilizes both the examples and the predicted logits of previous tasks, CLSER (Arani et al., 2022) employs a dual memory that maintains short-term and long-term semantic memories, and CBA (Wang et al., 2023) proposes to correct the recency bias through a proposed bias attractor with the help of previous examples in $\mathcal{M}$. In this paper, we revisit current replay-based methods from the gradient perspective and then propose a novel Pareto-Optimized CL algorithm. Compared to the existing replay-based methods, our method additionally takes into account the interrelationships among past tasks and achieves an overall superior performance.

6. Conclusion

In this paper, for replay-based CL methods, we have first mathematically formulated them as a hierarchical gradient aggregation framework that combines the gradient of the current task and the gradient-weighted sum of each previous task in the update direction. Then we have proposed the Pareto-Optimized CL (POCL) by modeling the interrelationships among previous tasks to enhance the integrated knowledge of previous tasks, thereby reducing catastrophic forgetting. Comprehensive experiments across various datasets and settings have finely substantiated the superiority of the proposed POCL as well as its good applicability beyond the current state-of-the-art continual learning methods.

Acknowledgement

We thank all the anonymous reviewers for their constructive suggestions on improving this paper. This research was sponsored by the Young Elite Scientists Sponsorship Program by CAST 2023QNRC001.

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 of which we feel must be specifically highlighted here.

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., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Arani, E., Sarfraz, F., and Zonooz, B. Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations*, 2022.

Ardywibowo, R., Huo, Z., Wang, Z., Mortazavi, B. J., Huang, S., and Qian, X. Varigrow: Variational architecture growing for task-agnostic continual learning based on bayesian novelty. In *International Conference on Machine Learning*, pp. 865–877. PMLR, 2022.

Boyd, S. P. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

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.

Caccia, L., Aljundi, R., Asadi, N., Tuytelaars, T., Pineau, J., and Belilovsky, E. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2018.

Chen, J., Nguyen, T., Gorur, D., and Chaudhry, A. Is forgetting less a good inductive bias for forward transfer? In *The Eleventh International Conference on Learning Representations*, 2022.

Chrysakis, A. and Moens, M.-F. Online bias correction for task-free continual learning. In *International Conference on Learning Representations*, 2023.

Du, D.-Z. and Pardalos, P. M. *Minimax and applications*, volume 4. Springer Science & Business Media, 1995.

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.

Frank, M., Wolfe, P., et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2): 95–110, 1956.

Guo, Y., Liu, M., Yang, T., and Rosing, T. Improved schemes for episodic memory-based lifelong learning. *Advances in Neural Information Processing Systems*, 33: 1023–1035, 2020.

Gupta, G., Yadav, K., and Paull, L. Look-ahead meta learning for continual learning. *Advances in Neural Information Processing Systems*, 33:11588–11598, 2020.

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.

Huszár, F. On quadratic penalties in elastic weight consolidation. *arXiv preprint arXiv:1712.03847*, 2017.

Jaggi, M. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pp. 427–435. PMLR, 2013.

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.

Lin, X., Zhen, H.-L., Li, Z., Zhang, Q.-F., and Kwong, S. Pareto multi-task learning. *Advances in Neural Information Processing Systems*, 32, 2019.

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

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.

Momma, M., Dong, C., and Liu, J. A multi-objective/multi-task learning framework induced by pareto stationarity. In *International Conference on Machine Learning*, pp. 15895–15907. PMLR, 2022.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint* *arXiv:1803.02999*, 2018.

Pareto, V. *Cours d'économie politique*, volume 1. Librairie Droz, 1964.

Pinelis, I. Optimum bounds for the distributions of martingales in banach spaces. *The Annals of Probability*, pp. 1679–1706, 1994.

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 *International Conference on Learning Representations*, 2019.

Ritter, H., Botev, A., and Barber, D. Online structured laplace approximations for overcoming catastrophic forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

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.

Wang, L., Zhang, X., Li, Q., Zhu, J., and Zhong, Y. Coscl: Cooperation of small continual learners is stronger than a big one. In *European Conference on Computer Vision*, pp. 254–271. Springer, 2022.

Wang, Q., Wang, R., Wu, Y., Jia, X., and Meng, D. Cba: Improving online continual learning via continual bias adaptor. *International Conference on Computer Vision*, 2023.

Wu, Y., Huang, L.-K., Wang, R., Meng, D., and Wei, Y. Meta continual learning revisited: Implicitly enhancing online hessian approximation via variance reduction. In *The Twelfth International Conference on Learning Representations*, 2024.

Yang, Y., Zhou, D.-W., Zhan, D.-C., Xiong, H., and Jiang, Y. Adaptive deep models for incremental learning: Considering capacity scalability and sustainability. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 74–82, 2019.

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

A. Theoretical Analysis

A.1. The MiniMax Theorem

MiniMax Theorem. (Du & Pardalos, 1995). This theorem answers the question of in which situation the order of minimization and maximization operations can be swapped. Concretely, let $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$ be two compact convex sets. Suppose $f : X \times Y \rightarrow \mathbb{R}$ is a continuous function that is concave-convex, i.e., $f(x, \cdot) : Y \rightarrow \mathbb{R}$ is convex for fixed x and $f(\cdot, y) : X \rightarrow \mathbb{R}$ is concave for the fixed y. Then we can get that

$$\max_{x\in X} \min_{y\in Y} f(x, y) = \min_{y\in Y} \max_{x\in X} f(x, y). \tag{1}$$

As discussed in Section 2, the optimization problem described by Eq. (4) satisfies the above requirements. Therefore, we can interchange its optimization order and obtain the gradient weighting optimization problem in Eq. (5).

A.2. Proof of the Derivation from Eq. (12) to Eq. (13)

In the first-order meta-learning algorithm Reptile (Nichol et al., 2018), it shows that one-step hyper-gradient (i.e., $\boldsymbol{g}_j^{\text{HD}}$ in Eq. (12)) is of the form:

$$\begin{aligned}
\boldsymbol{g}_j^{\text{HD}} &= \frac{\partial \mathcal{L}(f_{\tilde{\boldsymbol{\theta}}_{t+1}}(x^m), y^m)}{\partial \boldsymbol{\theta}_t} \\
&= \frac{\partial \mathcal{L}_{\theta_t}^m}{\partial \theta_t} - \alpha \frac{\partial^2 \mathcal{L}_{\theta_t}^m}{\partial \theta_t^2} \frac{\partial \mathcal{L}_{\theta_t}^j}{\partial \theta_t} - \alpha \frac{\partial^2 \mathcal{L}_{\theta_t}^j}{\partial \theta_t^2} \frac{\partial \mathcal{L}_{\theta_t}^m}{\partial \theta_t} + \mathcal{O}(\alpha^2) \\
&\approx \frac{\partial \mathcal{L}_{\theta_t}^m}{\partial \theta_t} - \alpha \frac{\partial}{\partial \theta_t} (\frac{\partial \mathcal{L}_{\theta_t}^j}{\partial \theta_t} \times \frac{\partial \mathcal{L}_{\theta_t}^m}{\partial \theta_t}) \\
&\approx \frac{\partial (\mathcal{L}_{\theta_t}^m - \alpha \nabla_{\theta_t} \mathcal{L}_{\theta_t}^m \cdot \nabla_{\theta_t} \mathcal{L}_{\theta_t}^j)}{\partial \theta_t},
\end{aligned} \tag{2}$$

where $\mathcal{L}_{\theta_t}^m \triangleq \mathcal{L}(f_{\tilde{\boldsymbol{\theta}}_{t+1}}(x^m), y^m)$ and $\mathcal{L}_{\theta_t}^j \triangleq \mathcal{L}(f_{\theta_t}(x_j), y_j)$. In this way, we have finished the derivations for Eq. (13).

A.3. Convergence Analysis

To gain a deeper understanding of the proposed POCL, we also provide its regret bound, as illustrated in Theorem 1. Let the notation $F_t(\theta) \triangleq \sum_{i=1}^{B_t} \mathcal{L}(f(x_i; \theta), y_i)$ and $F(\theta) = \sum_{t=1}^{T} F_t(\theta)$, where B_t means the samples number of the t-th training batch. There are four requiring assumptions as follows:

Assumption 1. *The compact convex set $\mathcal{C} \subseteq \mathbb{R}^d$ has diameter D, i.e., $\forall \theta, \theta^{'} \in \mathcal{C}$, $\|\theta - \theta^{'}\| \leq D$.*

Assumption 2. *Let $\boldsymbol{g}_t$ denote the t-th update gradient, there exisits $\sigma^2 < \infty$ such that $\|\nabla F_t(\theta) - \nabla F(\theta)\|^2 \leq \sigma^2$.*

Assumption 3. *The difference of $F_t(\theta)$ and $F(\theta)$ is bounded over the constraint set $\mathcal{C}$, i.e., $\forall \theta \in \mathcal{C}, t \in \{1, ..., T\}$, there exists $M^2 < \infty$ such that with probability 1,*

$$\|F_t(\theta) - F(\theta)\|^2 \leq M^2$$

Assumption 4. *The gradient $\nabla F_t(\theta)$ is unbiased and is L-Lipschitz continuous over the constraint set $\mathcal{C}$, i.e.,*

$$\|\nabla F_t(\theta) - \nabla F_t(\theta^{'})\| \leq L\|\theta - \theta^{'}\|, \forall \theta, \theta^{'} \in \mathcal{C}$$

Theorem 1. *Let T denote the training iteration. Suppose all the above four assumptions are satisfied, then with probability at least $1 - \delta$ for any $\delta \in (0, 1)$, we can get,*

$$R(T) \leq (\log T + 1)(F(\theta_1) - F(\theta^*)) + \sigma^2 D \log T + \frac{LD^2(\log T + 1)^2}{2} + 4M\sqrt{T \log(8/\delta)}.$$

Proof. Then we will prove the regret bound $R(T)$ of POCL as follows,

$$R(T) \triangleq \sum_{t=1}^{T} F_t(\theta_t) - \min_\theta \sum_{\mathrm{t}=1}^{\mathrm{T}} \mathrm{F}_\mathrm{t}(\theta). \tag{3}$$

Let $\theta^* = \mathrm{argmin}_\theta \sum_{\mathrm{t}=1}^{\mathrm{T}} \mathrm{F}_\mathrm{t}(\theta)$, then we can get,

$$\begin{aligned} R(T) &= \sum_{t=1}^{T} F_t(\theta_t) - \mathrm{min}_\theta \sum_{\mathrm{t}=1}^{\mathrm{T}} \mathrm{F}_\mathrm{t}(\theta) \\ &= \sum_{t=1}^{T} F_t(\theta_t) - \sum_{t=1}^{T} F_t(\theta^*) \\ &= \sum_{t=1}^{T} [F_t(\theta_t) - F_t(\theta^*)]. \end{aligned} \tag{4}$$

Then, through calculating the term $F_t(\theta_t) - F_t(\theta^*)$, we can get the regret bound. We first define a sequence $s_t = \tilde{F}_t(\theta_t) - F_t(\theta^*) - (F(\theta_t) - F(\theta^*)), t = 1, ..., T$. It can be observed that $\mathbb{E}[s_t \mid \mathcal{F}_{t-1}] = 0$, where $\mathcal{F}_{t-1}$ is the σ-algebra generated by $\{F_1, \epsilon_1, ..., F_{t-1}, \epsilon_{t-1}\}$. It means that $\{s_t\}_{t=1}^T$ is a martingale difference sequence. According to Assumption 4, we can derive that,

$$\|s_t\| = \|F_t(\theta_t) - F_t(\theta^*) - (F(\theta_t) - F(\theta^*)\| \leq 2M.$$

Based on the Theorem 3.5 in (Pinelis, 1994), we can get

$$\mathbb{P}\left(\|\sum_{t=1}^{T} s_t\| \geq \lambda\right) \leq 4\exp\left(-\frac{\lambda^2}{16TM^2}\right),$$

where $\lambda > 0$. By setting $\lambda = 4M\sqrt{Tlog(8/\delta)}$, we have with probability at least $1 - \delta/2$,

$$\sum_{t=1}^{T} s_t = \sum_{t=1}^{T} (F_t(\theta_t) - F_t(\theta^*)) - \sum_{t=1}^{T} (F(\theta_t) - F(\theta^*)) \leq 4M\sqrt{Tlog(8/\delta)}.$$

Combining this term with Eq. (4), we can get,

$$R(T) \leq \sum_{t=1}^{T} (F(\theta_t) - F(\theta^*)) + 4M\sqrt{Tlog(8/\delta)}.$$

Let $\beta_t = \frac{1}{t}$ denote the learning rate, $\boldsymbol{\epsilon}_t = \boldsymbol{g}_t - \nabla F(\theta_t)$ then we can get,

$$\begin{aligned} F(\theta_t) - F(\theta^*) &= (1 - \beta_t)(F(\theta_{t-1}) - F(\theta^*)) + \alpha_t D\|\boldsymbol{\epsilon}_{t-1}\|^2 + \frac{LD^2\beta_t^2}{2} \\ &\leq \prod_{\tau=1}^{t-1}(1 - \beta_\tau)(F(\theta_1) - F(\theta^*)) + \sum_{\tau=1}^{t-1}\beta_\tau(D\|\boldsymbol{\epsilon}_\tau\|^2 + \frac{LD^2\beta_\tau}{2}) \prod_{k=\tau+1}^{t-1}(1 - \beta_k) \\ &\leq \frac{1}{t}(F(\theta_1) - F(\theta^*)) + \frac{D}{t}\sum_{\tau=1}^{t-1}\|\boldsymbol{\epsilon}_\tau\|^2 + \frac{LD^2\log t}{2t}. \end{aligned}$$

$$\begin{aligned} \sum_{t=1}^{T} F(\theta_t) - F(\theta^*) &= \sum_{t=1}^{T}\frac{1}{t}(F(\theta_1) - F(\theta^*)) + \sum_{t=1}^{T}\frac{D}{t}\sum_{\tau=1}^{t-1}\|\boldsymbol{\epsilon}_\tau\|^2 + \sum_{t=1}^{T}\frac{LD^2\log t}{2t} \\ &\leq (\log T + 1)(F(\theta_1) - F(\theta^*)) + \sigma^2 D\log T + \frac{LD^2(\log T + 1)^2}{2}. \end{aligned}$$

Therefore, the final regret bound is,

$$R(T) \leq (\log T + 1)(F(\theta_1) - F(\theta^*)) + \sigma^2 D\log T + \frac{LD^2(\log T + 1)^2}{2} + 4M\sqrt{T\log(8/\delta)}.$$

□

B. Analysis of the Gradient Aggregation in Eq. (5)

By utilizing the MiniMax theorem in Appendix A.1, we have transformed the gradient alignment method into a gradient weighting problem, as shown in Eq. (5). The resulting optimization direction can be expressed as $\boldsymbol{u}^* = \boldsymbol{g}_n + \boldsymbol{g}(\boldsymbol{\lambda}^*)$, where $\boldsymbol{\lambda}^*$ is solved to minimize the objective function $\frac{1}{2}\|\boldsymbol{g}(\boldsymbol{\lambda}) + \boldsymbol{g}_n\|_2^2$. To gain a deeper understanding of the optimization direction $\boldsymbol{u}^*$, we conduct an analysis of simple cases (i.e., $n = 2, 3$) in two-dimensional scenarios. These simplified cases allow us to examine the derived hierarchical gradient aggregation framework more straightforwardly, facilitating a clear comparison with other gradient alignment methods.

Comparing with Other Replay-based Methods. As depicted in Fig. 4 where $n = 2$, there are two different situations. According to our derived optimization objective as shown in Eq. (5), our goal is to get λ_1 such that $\|\lambda_1 \boldsymbol{g}_1^m + \boldsymbol{g}_2\|_2^2$ as small as possible. Concretely, in the situation (a) where $\langle \boldsymbol{g}_2, \boldsymbol{g}_1^m \rangle \geq 0$, there is no interference between the two directions, leading to the final optimization direction $\boldsymbol{u}^*$ being $\boldsymbol{u}^* = 0 \cdot \boldsymbol{g}_1^m + \boldsymbol{g}_2 = \boldsymbol{g}_2$. In contrast, in situation (b), where $\langle \boldsymbol{g}_2, \boldsymbol{g}_1^m \rangle < 0$, interference between the two directions is observed. In this case, the final optimization direction is $\boldsymbol{u}^* = \boldsymbol{g}_2 - \frac{\langle \boldsymbol{g}_2, \boldsymbol{g}_1^m \rangle}{\|\boldsymbol{g}_1\|^2} \boldsymbol{g}_1^m$. It is worth noting that the solutions are exactly the same as those obtained from AGEM (Chaudhry et al., 2018), which implies that AGEM can be regarded as a special case within our hierarchical gradient aggregation framework. Additionally, La-MAML (Gupta et al., 2020), a simplified version of MER (Riemer et al., 2019), has derived an equivalent form of its bi-level optimization goal that coincides with optimizing the CL objective of AGEM. This equivalence demonstrates that La-MAML can also be regarded as a specific instance. Similarly, MEGA (Guo et al., 2020) just manually assigns weights to gradients of different tasks. Therefore, all these replay-based methods can be analyzed within this framework.

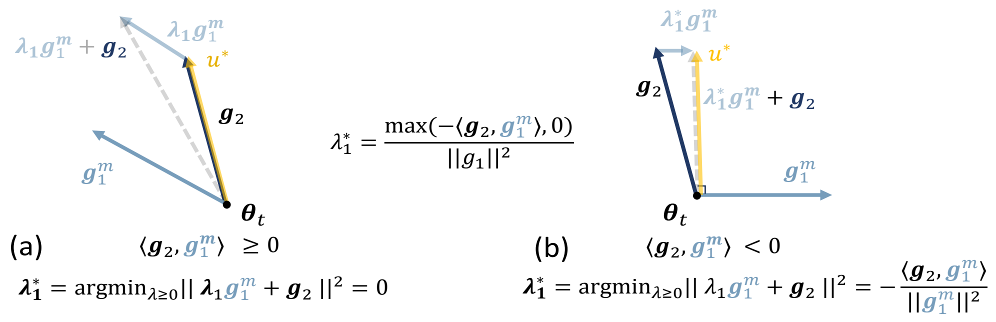

$\lambda_1 \boldsymbol{g}_1^m + \boldsymbol{g}_2$
$\lambda_1 \boldsymbol{g}_1^m$
u^*
$\boldsymbol{g}_2$
$\boldsymbol{g}_1^m$
$\boldsymbol{\theta}_t$
(a) $\langle \boldsymbol{g}_2, \boldsymbol{g}_1^m \rangle \geq 0$
$\boldsymbol{\lambda}_1^* = \text{argmin}_{\lambda \geq 0} \| \boldsymbol{\lambda}_1 \boldsymbol{g}_1^m + \boldsymbol{g}_2 \|^2 = 0$
$\lambda_1^* = \frac{\max(-\langle \boldsymbol{g}_2, \boldsymbol{g}_1^m \rangle, 0)}{\|\boldsymbol{g}_1\|^2}$
$\lambda_1^* \boldsymbol{g}_1^m$
u^*
$\boldsymbol{g}_2$
$\lambda_1^* \boldsymbol{g}_1^m + \boldsymbol{g}_2$
$\boldsymbol{g}_1^m$
$\boldsymbol{\theta}_t$
(b) $\langle \boldsymbol{g}_2, \boldsymbol{g}_1^m \rangle < 0$
$\boldsymbol{\lambda}_1^* = \text{argmin}_{\lambda \geq 0} \| \lambda_1 \boldsymbol{g}_1^m + \boldsymbol{g}_2 \|^2 = -\frac{\langle \boldsymbol{g}_2, \boldsymbol{g}_1^m \rangle}{\|\boldsymbol{g}_1^m\|^2}$

Figure 4. Illustration of the final determined optimization direction for our derived Eq. (5) with two Tasks (i.e., n=2) in the simplest two-dimensional context.

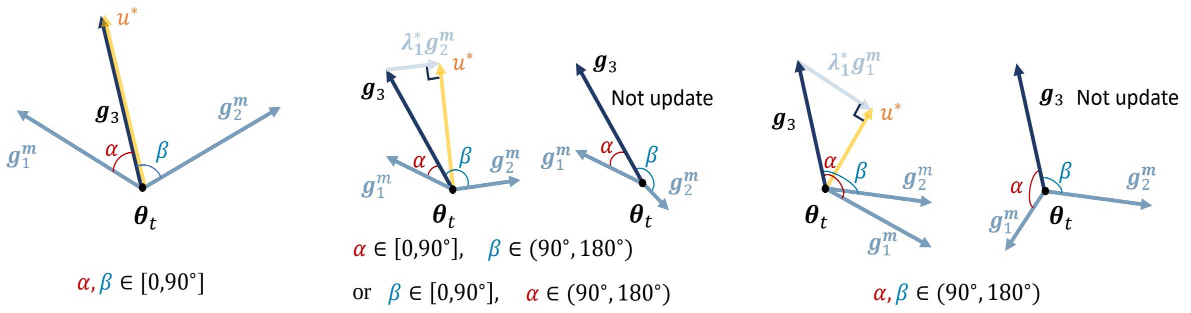

u^*
$\boldsymbol{g}_3$
$\boldsymbol{g}_1^m$
$\boldsymbol{g}_2^m$
α
β
$\boldsymbol{\theta}_t$
$\alpha, \beta \in [0,90°]$
$\lambda_1^* \boldsymbol{g}_2^m$
u^*
$\boldsymbol{g}_3$
Not update
$\alpha \in [0,90°], \quad \beta \in (90°, 180°)$
or $\beta \in [0,90°], \quad \alpha \in (90°, 180°)$
$\lambda_1^* \boldsymbol{g}_1^m$
$\boldsymbol{g}_3$ Not update
$\alpha, \beta \in (90°, 180°)$

Figure 5. Illustration of the final optimization direction determined our derived Eq. (5) with three Tasks (i.e., $n = 3$) in the simplest two-dimensional context.

C. More Details of the Experiments

Benchmark Datasets. Both CIFAR-10 and CIFAR-100 datasets (Krizhevsky et al., 2009) consist of 50,000 training images and 10,000 testing images, all with a size of 32× 32 pixels. However, they differ in the number of classes: CIFAR-10 contains 10 classes, while CIFAR-100 includes 100 classes. Following (Chen et al., 2022), the CIFAR-10 and CIFAR-100 are evenly split into 5 tasks and 20 tasks respectively, denoted as Split CIFAR-10 and Split CIFAR-100. Similarly, the

TinyImageNet dataset (Buzzega et al., 2020), which contains 200 classes and 100,000 images with a size of 64×64 pixels, is divided into 20 tasks, with each task containing 10 classes.

Details of the Comparison Methods. In this study, we compare the proposed POCL method with other gradient alignment methods as well as other representative CL methods. Here is a brief introduction to these comparison methods:

Gradient Alignment Methods:

- **GEM** (Lopez-Paz & Ranzato, 2017). This method manipulates the gradient of the current task in a way that ensures the final update direction satisfies the non-negative inner-product constraints w.r.t. the gradients of previous tasks.
- **AGEM** (Chaudhry et al., 2018). Since GEM solves its constrained optimization problem through quadratic programming, which is time-consuming, AGEM opts to simplify the problem by merging all previous tasks into one. This enables the derivation of a closed-form solution.
- **MEGA** (Guo et al., 2020). This method proposes to manually assign weights to gradients of different tasks and then update the model in the weighted direction.
- **MER** (Riemer et al., 2019). This approach utilizes replay examples to align the gradients of previous tasks and the current task to maximize the transfer from previous tasks and minimize interference.
- **La-MAML** (Gupta et al., 2020). This method simplifies the optimization process of MER and shares a similar gradient alignment objective.

Experience Replay Methods:

- **ER** (Rolnick et al., 2019). It constructs a memory buffer to store samples of previous tasks and utilize them during the sequential training.
- **DER++** (Buzzega et al., 2020). Building upon the foundation of ER, DER++ stores the predicted logits of each example. This strategy is employed to apply a distillation loss during the training process.
- **CLSER** (Arani et al., 2022). On the basis of ER, CLSER employs the dual memory which maintains short-term and long-term semantic memories.
- **ER-ACE** (Caccia et al., 2022). This method introduces a novel training approach that primarily focuses on previous tasks, aiming to avoid abrupt feature changes associated with these tasks.
- **CBA** (Wang et al., 2023). It proposes to correct the recency bias through a proposed bias attractor with the help of previous examples in $\mathcal{M}$.

D. Other Experimental Results

Full Table with Confidence Interval. In order to better present the results, we provide the mean performance of AAA, Acc, and FM metrics, along with their 95% confidence intervals, specifically for each dataset, as shown in Table 7, Table 8, and Table 9. The results clearly demonstrate that the proposed POCL method consistently achieves the highest average performance in terms of AAA and Acc across all three datasets under different buffer sizes. With respect to the FM metric, aside from ER-ACE and La-MAML, methods that inherently have an extremely high a_i^* as analyzed in Sec.4, our method also outperforms other methods, achieving the best FM performance.

Other Analysis Results. To better validate the effectiveness of the proposed POCL, we also provide the accuracy of each task on Split CIFAR-100 and Split TinyImageNet, as shown in Fig. 6. It is evident from the results that our method, on average, covers a large area, thereby validating that our proposed POCL can indeed sustain the performance of previous tasks and effectively mitigate forgetting. Moreover, we also provide the confusion matrix of other CL methods, including ER, DER++, CLSER, and ER-ACE. By comparing with Fig. 3, it can be seen that all the four methods cannot distinguish $\mathcal{T}_2$ and $\mathcal{T}_3$, while the proposed POCL exhibits much better performance. This further confirms that our proposed POCL, by introducing Pareto optimization to mitigate catastrophic forgetting, can effectively capture and model the relationships among past tasks.

Time Complexity Analysis. To analyze the time complexity of the proposed POCL algorithm, we conducted a comparison of training times on the Split CIFAR10 dataset with other CL methods that also utilize hyper-gradients, namely MER (Riemer et al., 2019) and La-MAML (Gupta et al., 2020). The results clearly indicate that the proposed POCL algorithm significantly reduces training time compared to MER (Riemer et al., 2019), while achieving comparable training costs to La-MAML.

Table 7. Performance of all the comparing methods on Split CIFAR-10 under different memory buffer size $|\mathcal{M}|$. All the reported results are averagely computed over 5 repetitions.

Method	Split CIFAR-10 ($N = 5$)					
	$\|\mathcal{M}\|$ = 0.6k			$\|\mathcal{M}\|$ = 1k		
	AAA	Acc	FM	AAA	Acc	FM
SGD	34.04±3.10	16.68±0.27	-61.01±3.30	34.04±3.10	16.68±0.27	-61.01±3.30
EWC (Huszár, 2017)	36.51±0.75	18.37±0.30	-64.21±0.86	36.51±0.75	18.37±0.30	-64.21±0.86
GEM (Lopez-Paz & Ranzato, 2017)	37.78±1.98	18.84±1.17	-58.09±3.90	37.00±0.47	18.73±0.76	-57.64±2.70
AGEM (Chaudhry et al., 2018)	37.67±1.76	18.51±0.07	-66.17±1.61	37.62±1.53	18.03±0.43	-66.61±1.60
ER (Rolnick et al., 2019)	54.68±5.11	39.43±3.79	-35.68±3.20	54.91±1.65	42.04±4.33	-32.16±5.10
MER (Riemer et al., 2019)	45.36±1.48	24.42±0.42	-32.78±0.81	50.99±2.75	36.15±1.21	-23.33±8.06
La-MAML (Gupta et al., 2020)	47.89±1.45	30.53±3.73	-10.09±0.52	46.08±3.45	35.89±1.35	-5.03±0.86
DER (Buzzega et al., 2020)	49.06±1.30	25.80±0.51	-54.60±2.80	48.25±1.95	23.90±0.47	-55.22±1.70
DER++ (Buzzega et al., 2020)	57.17±1.08	47.03±1.03	-24.00±1.30	61.01±1.55	50.31±1.50	-28.01±1.31
ER-ACE (Caccia et al., 2022)	52.27±0.97	46.04±3.75	-9.12±2.94	57.42±0.99	51.17±3.44	-9.68±2.90
CLSER (Arani et al., 2022)	61.64±1.62	50.36±4.06	-29.03±3.70	63.27±0.71	53.06±1.58	-31.38±1.70
CBA (Wang et al., 2023)	63.41±1.67	53.42±1.37	-27.15±4.70	65.47±0.77	52.08±0.15	-27.31±3.70
POCL	**63.62±2.65**	**53.42±1.17**	-23.27±2.50	**66.23±1.73**	**58.50±2.43**	-16.85±2.80

Table 8. Performance of all the comparing methods on Split CIFAR-100 under different memory buffer size $|\mathcal{M}|$. All the reported results are averagely computed over 5 repetitions.

Method	Split CIFAR-100 ($N = 20$)					
	$\|\mathcal{M}\|$ = 1k			$\|\mathcal{M}\|$ = 5k		
	AAA	Acc	FM	AAA	Acc	FM
SGD	9.67±0.18	3.24±0.12	-54.24±1.20	9.67±0.18	3.24±0.12	-54.24±1.20
EWC (Huszár, 2017)	9.87±0.28	2.77±0.27	-56.55±1.30	9.87±0.28	2.77±0.27	-56.55±1.30
GEM (Lopez-Paz & Ranzato, 2017)	13.43±0.04	6.04±0.52	-45.83±0.69	13.71±0.23	6.46±0.93	-50.61±1.90
AGEM (Chaudhry et al., 2018)	10.61±0.09	3.75±0.18	-60.51±0.34	10.80±0.15	3.52±0.13	-60.89±0.37
ER (Rolnick et al., 2019)	17.86±0.26	11.89±0.27	-46.25±0.47	20.67±0.68	14.87±0.60	-48.54±1.10
La-MAML (Gupta et al., 2020)	17.37±0.59	10.03±0.25	-5.95±0.01	19.05±0.54	12.57±0.34	-6.45±0.33
DER (Buzzega et al., 2020)	10.96±0.20	3.71±0.11	-60.58±0.38	10.47±0.32	3.68±0.06	-59.94±1.40
DER++ (Buzzega et al., 2020)	17.30±0.30	8.72±0.52	-58.61±0.60	17.08±0.79	8.98±1.05	-61.23±0.19
ER-ACE (Caccia et al., 2022)	24.02±0.22	15.46±1.05	-12.01±0.81	24.93±1.95	20.58±0.36	-9.84±1.00
CLSER (Arani et al., 2022)	22.58±0.42	15.68±0.62	-45.63±0.62	23.25±1.34	16.42±1.52	-50.71±0.86
CBA (Wang et al., 2023)	22.46±0.53	15.54±0.44	-44.30±1.53	23.07±0.58	15.87±0.75	-50.16±2.76
POCL	**26.68±0.21**	**16.54±0.34**	-37.61±0.63	**36.34±1.24**	**33.36±1.92**	-17.55±0.36

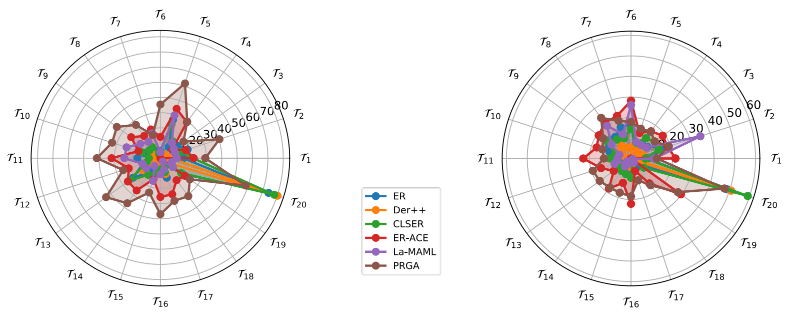
Moreover, considering the substantial improvements in both AAA and Acc metrics, the proposed POCL algorithm demonstrates its superiority in terms of both training efficiency and performance on the Split CIFAR-10 dataset.

Table 9. Performance of all the comparing methods on Split TinyImageNet under different memory buffer size $|\mathcal{M}|$. All the reported results are averagely computed over 5 repetitions.

Method	Split TinyImageNet ($N = 20$)					
	$\|\mathcal{M}\|$ = 2k			$\|\mathcal{M}\|$ = 5k		
	AAA	Acc	FM	AAA	Acc	FM
SGD	7.63 ±0.24	2.17 ±0.07	-43.58 ±0.58	7.63 ±0.24	2.17 ±0.07	-43.58 ±0.58
EWC (Huszár, 2017)	7.96 ±0.12	2.43 ±0.13	-44.30 ±1.70	7.96 ±0.12	2.43 ±0.13	-44.30 ±1.70
GEM (Lopez-Paz & Ranzato, 2017)	10.17 ±0.28	3.70 ±0.44	-42.30 ±0.11	10.27 ±0.07	3.81 ±0.15	-42.71 ±0.03
AGEM (Chaudhry et al., 2018)	7.66 ±0.35	2.33 ±0.18	-45.61 ±0.04	7.79 ±0.03	2.40 ±0.22	-45.51 ±0.27
ER (Rolnick et al., 2019)	16.27 ±0.10	10.52 ±0.59	-37.90 ±0.28	16.10 ±0.24	11.89 ±0.63	-37.43 ±0.81
La-MAML (Gupta et al., 2020)	15.81 ±0.41	8.02 ±0.10	-3.76 ±0.11	16.32 ±0.49	9.24 ±0.34	-4.38 ±0.50
DER (Buzzega et al., 2020)	8.04 ±0.20	2.46 ±0.06	-47.11 ±0.65	7.65 ±0.11	2.04 ±0.15	-46.27 ±0.85
DER++ (Buzzega et al., 2020)	12.42 ±0.34	5.57 ±0.11	-47.13 ±2.00	11.93 ±0.34	5.26 ±0.21	-47.65 ±0.76
ER-ACE (Caccia et al., 2022)	20.57 ±0.49	13.23 ±1.13	-8.42 ±1.25	21.16 ±0.14	17.22 ±0.54	-6.56 ±0.64
CLSER (Arani et al., 2022)	18.50 ±0.08	10.03 ±0.22	-44.45 ±0.32	18.88 ±0.59	11.61 ±0.19	-44.21 ±0.14
CBA (Wang et al., 2023)	18.79 ±0.05	11.43 ±0.67	-38.17 ±1.56	18.98 ±0.85	10.98 ±0.14	-40.53 ±1.34
POCL	**21.56 ±0.44**	**12.69** ±0.47	-31.63 ±1.60	**25.48** ±0.67	**19.40** ±0.92	-20.22 ±1.20

Table 10. Training time comparison of different CL methods on Seq-CIFAR10 under the online setting (i.e., the training epoch of each task is set as one.)

Method	La-MAML	MER	POCL
AAA/Acc	46.08%/35.89%	50.99%/36.92%	**66.23%/58.50%**
Training Time (s)	750.61	20697.70	1677.10



(a) Split CIFAR-100 (b) Split TinyImageNet

Figure 6. Radar charts illustrating each task accuracy on (a) Split CIFAR-100 dataset and (b) Split TinyImageNet with $|\mathcal{M}|$=5k.