A Faster Path to Continual Learning

Wei Li[1] Hangjie Yuan[2] Zixiang Zhao[3] Borui Kang[4] Ziwei Liu[1,5] Tao Feng[6*]
[1]College of Computer Science, Sichuan University, China
[2]DAMO Academy, Alibaba Group, China
[3]Photogrammetry and Remote Sensing Lab, ETH Zürich, Switzerland
[4]School of Computer Science, Nanjing University, China
[5]College of Computing and Data Science, Nanyang Technological University, Singapore
[6]Department of Computer Science and Technology, Tsinghua University, China
{ymjiii98, fengtao.hi}@gmail.com, hj.yuan@zju.edu.cn

Abstract

Continual Learning (CL) aims to train neural networks on a dynamic stream of tasks without forgetting previously learned knowledge. Among optimization-based approaches, C-Flat has emerged as a promising solution due to its plug-and-play nature and its ability to encourage uniformly low-loss regions for both new and old tasks. However, C-Flat requires three additional gradient computations per iteration, imposing substantial overhead on the optimization process. In this work, we propose C-Flat Turbo, a faster yet stronger optimizer that significantly reduces the training cost. We show that the gradients associated with first-order flatness contain direction-invariant components relative to the proxy-model gradients, enabling us to skip redundant gradient computations in the perturbed ascent steps. Moreover, we observe that these flatness-promoting gradients progressively stabilize across tasks, which motivates a linear scheduling strategy with an adaptive trigger to allocate larger turbo steps for later tasks. Experiments show that C-Flat Turbo is 1.0× to 1.25× faster than C-Flat across a wide range of CL methods, while achieving comparable or even improved accuracy.

1. Introduction

In the open world, learning systems are expected to absorb new knowledge incrementally, a process known as continual learning (CL) [41, 54, 66]. A major obstacle in CL is catastrophic forgetting. To address this challenge, various approaches have been proposed [49, 57, 70], including memory-based, regularization-based, expansion-based, and pre-trained model (PTM)-based methods [13, 18, 37]. Among them, PTM-based approaches [25, 26, 52] have demonstrated particularly strong resistance to forgetting due to their superior generalization capabilities.

Beyond architectural and data-level strategies, the optimization process of CL has recently received increasing attention [23, 26, 27, 52]. A line of work shows that sharpness-aware minimization serves as a powerful mechanism for improving generalization and mitigating forgetting [14, 19, 64, 75]. For instance, Mehta et al. [39] provide theoretical and empirical evidence that optimizing for flat regions helps reduce forgetting. Following this insight, C-Flat [6] was recently proposed as a CL-friendly optimizer that improves stability across the joint knowledge space of new and old tasks [8, 47]. By explicitly seeking uniformly flat minima, C-Flat effectively alleviates forgetting caused by distribution and task shifts [30, 47, 75]. However, the flatness alignment mechanism of C-Flat incurs significant computational overhead. (i) Its zeroth-order sharpness perturbation requires an additional backward pass, doubling the cost [36, 61, 64]. (ii) Its first-order flatness term, based on Gradient Norm Aware Minimization (GAM) [61], involves calculating gradient norms at both a proxy model and its perturbed state, leading to two extra backward passes [75].

In this work, we observe that the orthogonal component of the first-order flatness gradient changes more gradually than both the empirical gradient and the zeroth-order sharpness term, similar to the behavior observed in LookSAM [36], which focuses only on zeroth-order sharpness. This stability inspires us to periodically skip redundant SAM and GAM computations and take shortcuts along these stable directions, thereby maintaining C-Flat's effectiveness while reducing computational cost. Moreover, we observe that both sharpness and flatness gradients decrease not only as training progresses within each task, but also across tasks, indicating an overall trend toward increased stability. Mo-

*Corresponding Author: Tao Feng (fengtao.hi@gmail.com)

tivated by this observation, (i) we introduce a stage-wise turbo-step scheduler that flexibly adjusts the shortcut frequency, and (ii) we integrate an adaptive policy that dynamically combines C-Flat with SGD for further efficiency gains.

Our technical contributions are three-fold:

(i) We identify a direction-invariant component in the first-order flatness gradients and propose C-Flat Turbo, an efficient variant of C-Flat that selectively takes shortcuts along these stable directions to reach flatter regions with substantially lower cost.

(ii) We reveal a stabilization trend in sharpness- and flatness-aware gradients during continual learning, and based on this, introduce a stage-wise linear turbo-step scheduler together with an adaptive triggering mechanism that dynamically regulates when C-Flat regularization should be applied.

(iii) Through extensive experiments, we demonstrate that C-Flat Turbo consistently matches or surpasses the performance of state-of-the-art CL methods, while being up to 1.25× faster than standard C-Flat.

2. Related Work

Continual learning. Along this line, continual learning can be broadly categorized into three groups [18, 54] as follows, *(i) Memory-based methods* maintain a limited memory budget to resist forgetting [22, 42, 43, 51]. Many efforts selectively store a few representative exemplars for rehearsal during CL. Apart from direct replay, some other efforts [8, 33, 34, 45, 50] use proxies of previous knowledge as memory to overcome forgetting, such as gradient-space bases and representative prototypes. *(ii) Regularization-based methods* are characterized by introducing favorable regularization terms to trade off new and old knowledge [7, 29, 32]. Common practices include weight [2, 28, 44], function [32, 40], and feature regularization [5, 16], which encourage these spaces to remain close to their original states. Like natural cognitive systems, this consolidation helps preserve parameters or representations that are important for previous tasks. *(iii) Expansion-based methods* aim to dynamically modularize network structures towards each task to tackle forgetting [67]. Methods in this group construct task-specific parameters or architecture (e.g., parameter allocation [1, 35], model decomposition/mixture, and modular network [58, 74]) to explicitly reduce inter-task interference [21, 46, 59]. In other words, these efforts shift the burden of storing raw data to the retention architecture [67].

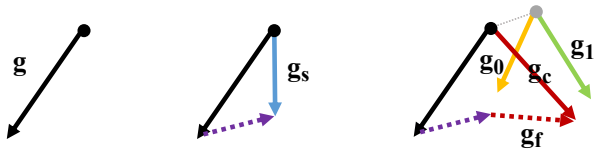
Continual learning using generalization. The strong generalizability of PTMs further advances the CL [70]. Following [70], we categorize these studies into three groups as follows, *(i) Prompt-based methods* leverage prompt learning to enable lightweight updates to PTM [23]. Many efforts [12, 25, 49, 52, 57] resort to various prompt selections, e.g., the way of prompt retrieval, task-special prompts, and prompt generation. Moreover, some work extends the concept of prompts to broader scenes [15, 26, 27], i.e., visual prompts, pre-trained vision-language model prompts, etc. Overall, these efforts strike a balance between the generalizability of PTM and the encoding of information from downstream tasks. *(ii) Representation-based methods* focus on building classifiers by leveraging the generalization capability of PTM [69]. Briefly, this line of work mainly focuses on calibrating classifiers and optimizing their internal relationships [38, 60, 71, 73]. For example, these methods concatenate backbone features to improve classifier representations, use random projections to remove class-wise correlations, and replay features to calibrate the classifier. *(iii) Model mixture-based methods* introduce a set of models and utilize various model-mixture approaches, such as model ensembling and model merging, to generate final outputs [17, 55, 56, 68]. In general, methods in this category integrate several PTMs with strong generalization capabilities to produce more robust results [70]. Yet, some of the drawbacks that arise are associated with these efforts, e.g. extra computational overhead and memory buffer.

Better and more efficient generalization in continual learning. In continual learning (CL), research on how flat minima [3, 14, 19] affect catastrophic forgetting is still in its early stages. A few studies have explored flatness evaluation during CL in specific scenarios [8, 47], such as tuning parameters within flat minima regions. Notably, C-Flat [6], a CL-tailored optimizer, introduces the concept of continual flatness to mitigate forgetting, opening a new avenue in CL research. While these efforts provide valuable insights into the role of flat minima in CL, they often face challenges in optimization efficiency, particularly due to the entrapment in random perturbations [10, 11, 36], especially when dealing with successive tasks. Moreover, most sharpness-aware CL methods directly inherit the full computational cost of SAM updates, leading to multiple additional backward passes compared to vanilla optimizers. This creates a tension between pursuing flatter minima and maintaining practical training efficiency, which is particularly critical in long task sequences and large-scale PTM-based CL. In this paper, we present an efficient version of C-Flat that improves the performance of CL while maintaining optimization efficiency.

3. Method

3.1. Rethinking the Mechanism of C-Flat

Let $\mathcal{S}^t = \{(\mathbf{x}_i^t, \mathbf{y}_i^t)\}_{i=1}^{n^t}$ denote the training set with n^t samples for task t, and let $\ell(f(\mathbf{x};\boldsymbol{\theta}), \mathbf{y})$ be the per-sample loss of the neural network f with parameters $\boldsymbol{\theta}$ on a data point $(\mathbf{x}, \mathbf{y})$. Continual Learning (CL) aims to learn a model f with parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ that minimizes the statistical risk across all tasks seen up to the current task T, under the



(a) SGD (b) SAM (c) C-Flat

Figure 1. Brief illustration of C-Flat [6]. (a) SGD optimizes along the negative direction of the gradients, $g = \nabla L(f(\theta^T))$. (b) SAM [14] computes the gradients g_s at an adversarially perturbed position $\theta + \rho \cdot g/\|g\|$, and then updates the original model parameters. (c) C-Flat [6] further calculates the first-order flatness gradient g_f, based on the perturbed parameters $\theta + \rho \cdot (g_s - g)/\|(g_s - g)\|$.

constraint of limited or no access to previous data $\mathcal{S}^t$ for $t < T$. Specifically, CL methods optimize the model parameters $\boldsymbol{\theta}_T$ during training on task T by minimizing the empirical loss over the available data: $\boldsymbol{\theta}_T = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{S}^T)$, where $\mathcal{L}(\boldsymbol{\theta}, \mathcal{S}^T) = \frac{1}{n^T} \sum_{i=1}^{n^T} \ell(f(\mathbf{x}_i^T; \boldsymbol{\theta}), \mathbf{y}_i^T)$. Depending on the specific CL methods, the training data may include a mixture of current task data $\mathcal{S}^T$ and additional data, such as stored exemplars or reconstructed samples from previous tasks. For simplicity, we omit $\mathcal{S}^T$ and denote $\boldsymbol{\theta}$ as $\boldsymbol{\theta}^T$ later.

As shown in Figure 1, the C-Flat optimizer [6] mitigates catastrophic forgetting by jointly optimizing for zeroth-order sharpness $\mathcal{R}_\rho^0(\boldsymbol{\theta})$ and first-order flatness $\mathcal{R}_\rho^1(\boldsymbol{\theta})$, encouraging the model to converge to parameter regions with uniformly low loss and curvature. The optimization objective on task T can be written as

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) + \mathcal{R}_\rho^0(\boldsymbol{\theta}) + \lambda \cdot \mathcal{R}_\rho^1(\boldsymbol{\theta}), \tag{1}$$

where $\lambda > 0$ is a balancing hyperparameter and $\rho > 0$ defines the neighborhood radius around the current parameter $\boldsymbol{\theta}$. Specifically, the zeroth-order term is defined as

$$\mathcal{R}_\rho^0(\boldsymbol{\theta}) = \max_{\|\boldsymbol{\epsilon}_0\| \le \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_0) - \mathcal{L}(\boldsymbol{\theta}), \tag{2}$$

which encourages uniformly low loss within a local neighborhood of $\boldsymbol{\theta}$. The first-order flatness term is given by

$$\mathcal{R}_\rho^1(\boldsymbol{\theta}) = \rho \cdot \max_{\|\boldsymbol{\epsilon}_1\| \le \rho} \left\| \nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1) \right\|, \tag{3}$$

which encourages low curvature of the loss landscape. Here, $\|\cdot\|$ denotes the ℓ_2 norm.

Propagation flow. Assuming that the loss function $\mathcal{L}$ is differentiable and bounded, and following the derivation of $\nabla R_\rho^0(\theta)$ in [14, 48, 61], we can approximate $\nabla R_\rho^1(\theta)$ as follows. More notations are provided in Appendix A.2.

$$\nabla \mathcal{R}_\rho^0(\boldsymbol{\theta}) = \nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_0^*) - \nabla \mathcal{L}(\boldsymbol{\theta}), \quad \boldsymbol{\epsilon}_0^* \approx \rho \cdot \frac{\nabla \mathcal{L}(\boldsymbol{\theta})}{\left\| \nabla \mathcal{L}(\boldsymbol{\theta}) \right\|}. \tag{4}$$

$$\begin{aligned} \nabla \mathcal{R}_\rho^1(\boldsymbol{\theta}) &\approx \rho \cdot \nabla \left\| \nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*) \right\| \\ &\approx \nabla \mathcal{L}\left(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^* + \rho \cdot \frac{\nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*)}{\left\| \nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*) \right\|} \right) - \nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*), \\ \boldsymbol{\epsilon}_1^* &\approx \rho \cdot (\boldsymbol{g}_s - \boldsymbol{g}) / (\|\boldsymbol{g}_s - \boldsymbol{g}\|). \end{aligned} \tag{5}$$

Notations. Here, we define several variations and terminology that will frequently be used later. Detailed descriptions can be found in Appendix A.1.

i) **the empirical loss term:** $\boldsymbol{g} = \nabla \mathcal{L}(\boldsymbol{\theta})$ as the original gradients derived from the vanilla optimizer.

ii) **the zeroth-order sharpness term:** $\boldsymbol{g}_s = \nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_0^*)$ as the perturbed SAM gradients, where $\boldsymbol{g}_s - \boldsymbol{g} = \nabla \mathcal{R}_\rho^0(\boldsymbol{\theta})$ measures the sharpness of the loss landscape.[1]

iii) **the first-order flatness term:** $\boldsymbol{g}_f = \nabla \mathcal{R}_\rho^1(\boldsymbol{\theta}) = \boldsymbol{g}_1 - \boldsymbol{g}_0$ as the regularization gradients, which quantifies the flatness of the loss landscape. Here, the intermediate gradients are given by $\boldsymbol{g}_0 = \nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*)$ and $\boldsymbol{g}_1 = \nabla \mathcal{L}\left(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^* + \rho \cdot \frac{\nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*)}{\left\| \nabla \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*) \right\|} \right)$.

As shown above, optimizing the gradients of the C-Flat objective can be decomposed into three components: ***the empirical loss term $\boldsymbol{g}$, the sharpness term $\boldsymbol{g}_s - \boldsymbol{g}$, and the flatness term $\boldsymbol{g}_f$***. Among them, $\boldsymbol{g}$ is required at every update step to reduce the empirical risk, whereas computing $\boldsymbol{g}_s$ and $\boldsymbol{g}_f$ incurs one and two additional backward passes, respectively, using the perturbed models $\boldsymbol{\theta} + \boldsymbol{\epsilon}_0^*$ and $\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*$. Both regularization terms help identify flatter regions of the loss landscape. Despite their performance benefits, searching for such regions is computationally expensive and imposes a substantial burden on CL systems. Therefore, developing time-efficient solutions becomes imperative.

3.2. C-Flat Turbo

In this section, we introduce C-Flat Turbo, an enhanced variant of C-Flat designed to accelerate training in continual learning scenarios. C-Flat Turbo exploits the observation that the orthogonal component of the first-order flatness gradient varies much more slowly than other gradient terms, allowing the optimizer to skip redundant computations associated with proxy and proxy-perturbed gradients. In addition, we incorporate an adaptive mechanism that monitors sharpness online and selectively applies C-Flat only when beneficial, falling back to vanilla optimizers otherwise.

[1] We use $\boldsymbol{g}_s - \boldsymbol{g}$ rather than $\boldsymbol{g}_{vs}$ to measure sharpness, for better alignment with the flatness term $\boldsymbol{g}_f = \boldsymbol{g}_1 - \boldsymbol{g}_0$ used later. Though it is not fully orthogonal to the gradient direction $\boldsymbol{g}$, it still captures the direction that promotes zeroth-order sharpness, and has been widely used in [61, 62] when combined with the vanilla gradient $\boldsymbol{g}$ to form a sharpness-aware update direction.

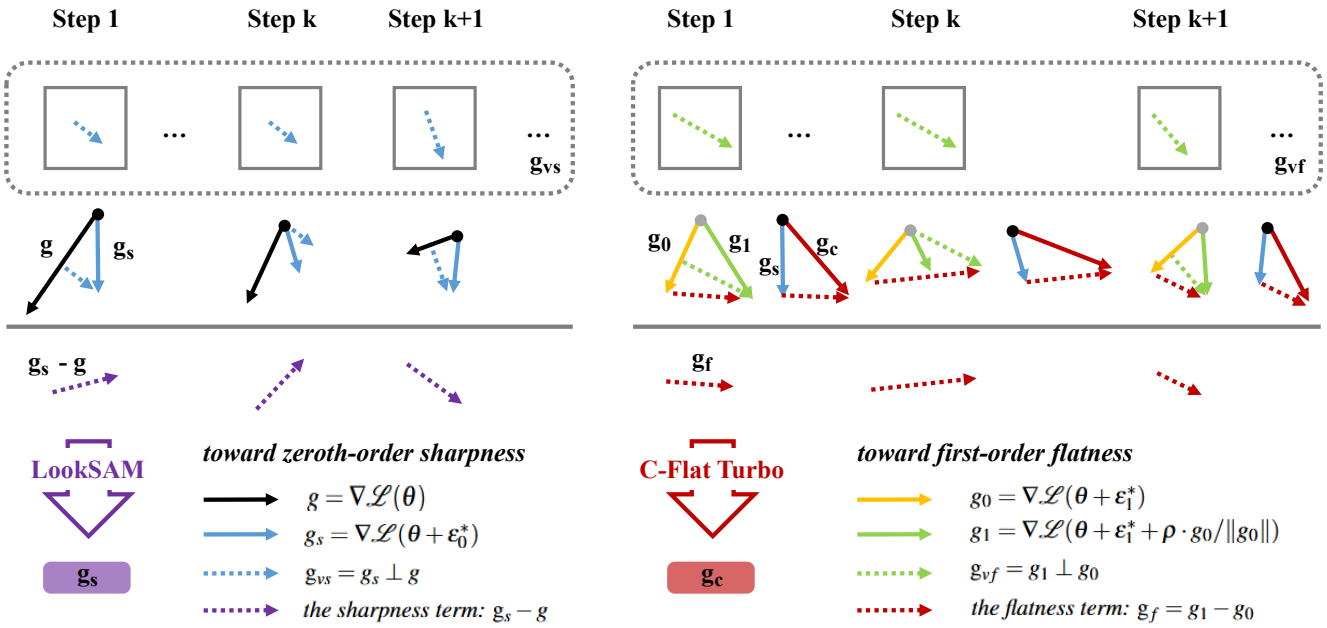


Figure 2. Schematic illustration of C-Flat Turbo. **Left:** LookSAM [36] decomposes the SAM gradients into two components: one parallel to $\boldsymbol{g}$ that reduces the empirical loss, and an orthogonal component $\boldsymbol{g}_{\text{vs}}$ that guides convergence toward a common low-loss region. Empirical results show that $\boldsymbol{g}_{\text{vs}}$ varies significantly more slowly than $\boldsymbol{g}$. **Right:** C-Flat Turbo investigates the latent invariance of $\boldsymbol{g}_{\text{vf}}$, the flatness component orthogonal to the gradients at the perturbed model $\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*$, which exhibits even slower variation than $\boldsymbol{g}_{\text{vs}}$ in LookSAM.

3.2.1. Taking Shortcuts Toward Flatness

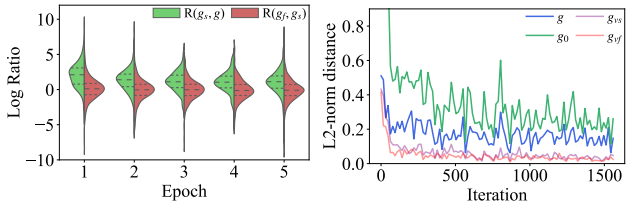
Base optimizers like SGD and Adam reduce the loss function along gradient directions. However, the resulting model parameters are often sensitive to small perturbations or noise, which can lead to degradation of previously learned parameters when adapting to new tasks. Seeking a unified low-loss landscape within a local region around the global minima has proven effective for continual learning. C-Flat builds upon this idea by promoting flatter regions via first-order flatness, in conjunction with zeroth-order sharpness.

In the context of zeroth-order sharpness, the increment $\boldsymbol{g}_s - \boldsymbol{g}$ captures the sharpness-aware correction introduced by SAM. Following the algorithm in Algorithm 1, we further define a direction-invariant sharpness component

$$\boldsymbol{g}_{vs} := \boldsymbol{g}_s - \frac{\langle \boldsymbol{g}_s, \boldsymbol{g} \rangle}{\|\boldsymbol{g}\|^2} \boldsymbol{g}, \tag{6}$$

which is orthogonal to the primary gradient direction $\boldsymbol{g}$. It has been observed that this component behaves as a more slowly varying, curvature-sensitive part of the SAM update, while facilitating the exploration of flatter regions in LookSAM [36]. This phenomenon further motivates us to investigate whether similar direction-invariant components exist in the optimization of the first-order flatness term.

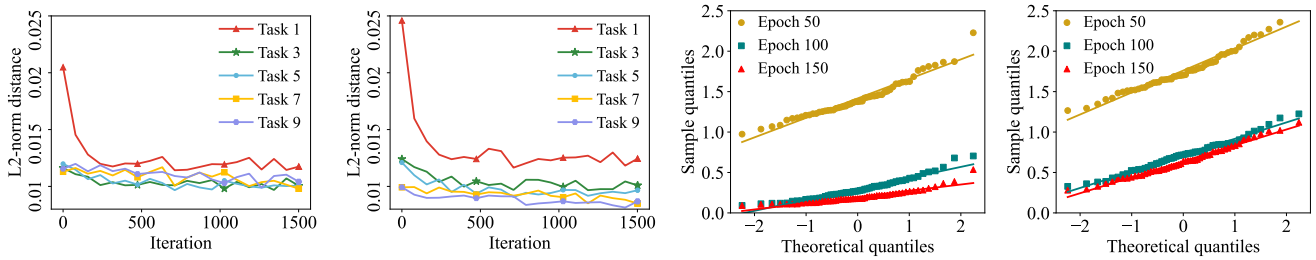
To visualize the optimization dynamics of $\boldsymbol{g}_s - \boldsymbol{g}$ and $\boldsymbol{g}_f$ relative to their reference directions, $\boldsymbol{g}$ for SAM and $\boldsymbol{g}_s$ for C-Flat, we define the gradient correction ratio as



(a) Gradient correction ratio. (b) Training with PTM.

Figure 3. **Left:** Distributions of gradient correction ratios for $\boldsymbol{g}_s - \boldsymbol{g}$ and $\boldsymbol{g}_f$ across training epochs. A larger portion of data near the distribution tails indicates increasingly pronounced differences. **Right:** L2-norm distances between gradients and their counterparts from five steps earlier. Changes along the sharpness-related direction ($\boldsymbol{g}_{vs}$) and flatness-related direction ($\boldsymbol{g}_{vf}$) evolve more slowly than those along the empirical gradient directions ($\boldsymbol{g}$ and $\boldsymbol{g}_0$).

$\text{ratio}(m,n) = \log\left(\left|\frac{m}{n+\epsilon}\right|\right)$, $\epsilon = 10^{-12}$, and then conduct experiments in EASE over 5 epochs. Figure 3a illustrates the distributions of $(\boldsymbol{g}_s - \boldsymbol{g}, \boldsymbol{g})$ and $(\boldsymbol{g}_f, \boldsymbol{g}_s)$ in Task 1. It can be observed that $\boldsymbol{g}_s - \boldsymbol{g}$ exhibits heavier tails than $\boldsymbol{g}_f$ in the early stages, indicating stronger adjustment introduced by zeroth-order sharpness. As the optimization converges to a local minimum, sharpness and flatness jointly contribute to exploring flatter regions, with sharpness still playing a dominant role. This phenomenon suggests that $\boldsymbol{g}_f$ acts as a relatively small rectification upon SAM, even subtler than the correction $\boldsymbol{g}_s - \boldsymbol{g}$ induced by SAM itself.



(a) sharpness (b) flatness (c) $\|\boldsymbol{g}\|^2$ (d) $\|\boldsymbol{g}_0\|^2$

Figure 4. **Left:** Gradient dynamics across tasks in CL. Both sharpness- and flatness-related gradients fluctuate substantially in early stages but progressively stabilize as training proceeds. **Right:** Q–Q plots of $\|\boldsymbol{g}\|^2$ and $\|\boldsymbol{g}_0\|^2$, showing that both statistics gradually approach a normal distribution over the course of learning.

In approximating $\mathcal{R}_\rho^1(\boldsymbol{\theta})$ in Eq. 5, the perturbation $\boldsymbol{\epsilon}_1^*$ naturally emerges as a small step aligned with the sharpness term. Consequently, the point $\boldsymbol{\theta}_p = \boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*$ serves as a proxy model for computing the flatness term. As illustrated in Figure 1, $\boldsymbol{g}_f$ plays a role analogous to SAM, but is defined at the proxy model $\boldsymbol{\theta}_p$ located in a region of high curvature. In this sense, we extract a direction-invariant flatness component $\boldsymbol{g}_{vf}$ from $\boldsymbol{g}_f$ with respect to the proxy gradient $\boldsymbol{g}_0$:

$$\boldsymbol{g}_{vf} := \boldsymbol{g}_f - \frac{\langle \boldsymbol{g}_f, \boldsymbol{g}_0 \rangle}{\|\boldsymbol{g}_0\|^2} \boldsymbol{g}_0, \tag{7}$$

which is orthogonal to $\boldsymbol{g}_0$ and captures the flatness-promoting update direction that is invariant to the base gradient direction of the proxy model.

Figure 3b illustrates the gradient differences across iterations, measured by the L2-norm distance to the gradients from five steps earlier. We observe that $\boldsymbol{g}_{vs}$ fluctuates more slowly than $\boldsymbol{g}$, which is consistent with the findings in [36]. Moreover, the proxy gradient $\boldsymbol{g}_0$ is noticeably more unstable than $\boldsymbol{g}$, reflecting its high-curvature nature and strong sensitivity to parameter changes. Interestingly, despite the considerable variation in $\boldsymbol{g}_0$, the flatness-oriented gradient $\boldsymbol{g}_f$ remains substantially more stable, and the direction-invariant component $\boldsymbol{g}_{vf}$ is even more stable than $\boldsymbol{g}_{vs}$, in line with the observations in Figure 3a. These insights support extracting the orthogonal component $\boldsymbol{g}_{vf}$ from $\boldsymbol{g}_f$ with respect to $\boldsymbol{g}_0$ as a flatness-promoting direction, as illustrated in Figure 2. Consequently, for the subsequent $k-1$ surrogate steps, the cached $\boldsymbol{g}_{vf}$ can be reused to efficiently guide the search toward low-curvature directions based on the proxy model at $\boldsymbol{\theta} + \boldsymbol{\epsilon}_1^*$. Concretely, we approximate the flatness update by adding a scaled direction-invariant term to the proxy gradient, $\boldsymbol{g}_f \approx \boldsymbol{g}_0 + \beta \frac{\|\boldsymbol{g}_0\|}{\|\boldsymbol{g}_{vf}\|} \boldsymbol{g}_{vf}$, thus avoiding recomputation of $\boldsymbol{g}_1$ when evaluating flatness at the proxy model and substantially reducing computational cost.

3.2.2. Dynamic Control for C-Flat

Stage-wise turbo step scheduling. As continual learning progresses, the learned parameter space becomes increasingly flatter, with earlier classes becoming more distinctly separated. Figures 4a and 4b depict the evolution of the sharpness and flatness terms over time. Both gradients fluctuate considerably during the initial incremental stage but stabilize in later stages, with this effect being particularly pronounced for flatness. This observation motivates assigning smaller turbo steps to earlier tasks and larger intervals to later ones. To this end, we introduce a linear scheduler that adaptively adjusts the step size for computing sharpness- and flatness-aware gradients throughout training. It is simply formulated as Turbo-k_0 with $k_t = k_0 + 10 \cdot t/N$, where k_0 and k_t denote the initial and task-n step sizes, respectively, with a total of N tasks. Empirically, C-Flat Turbo equipped with this scheduler achieves substantial speedup while maintaining competitive performance. Notably, an imprecise or even unknown value of N does not pose a serious concern, since the scheduler mainly serves to gradually increase k_t, and small distortions of the schedule in later tasks have little practical impact.

Adaptive triggering of regularization. Existing research on zeroth-order sharpness has proposed reducing computational overhead by combining SAM updates with standard ERM. For example, SS-SAM [63] employs a Bernoulli trial to decide whether to apply SAM, while AE-SAM [24] applies SAM only when a sharpness measure falls below a dynamically updated threshold. However, first-order flatness has received relatively little attention. Figures 4c and 4d visualize the distributional properties of $\|\boldsymbol{g}\|^2$ and $\|\boldsymbol{g}_0\|^2$ using quantile–quantile (Q–Q) plots. Points that lie closer to the reference line indicate that the corresponding variable more closely follows a normal distribution. Motivated by this observation, we use an exponential moving average (EMA) to estimate the mean and dispersion of $\|\boldsymbol{g}_0\|^2$, following [24]:

$$\begin{aligned} \mu_{f,j} &= \delta \mu_{f,j-1} + (1-\delta)\|\boldsymbol{g}_{0j}\|^2, \\ \sigma_{f,j} &= \delta \sigma_{f,j-1} + (1-\delta)\big(\|\boldsymbol{g}_{0j}\|^2 - \mu_{f,j}\big)^2, \end{aligned} \tag{8}$$

where j denotes the current iteration and $\delta = 0.9$ is a decay factor that discounts outdated gradient values. The flatness regularization is triggered only when

$$\|\boldsymbol{g}_{0j}\|^2 > \mu_{f,j} + \sigma_{f,j}. \tag{9}$$

A detailed description of the overall optimization procedure is provided in Algorithm 1 in the Appendix.

3.3. Convergence Analysis

We analyze the convergence of Turbo in the $k-1$ steps where the optimizer uses surrogate gradients instead of recomputing the full C-Flat gradients. Since C-Flat itself has been proven to converge in [6, 61], we only need to control the additional approximation error introduced by these surrogate steps. A detailed convergence proof of Turbo under this surrogate-gradient view is provided in Appendix A.9.

4. Experiments

4.1. Experiment Setup

Datasets. Following [70], we perform the evaluation on CIFAR100 [31], CUB200 [53], ImageNet-R [20] (IN-R), and ObjectNet [4] (ObjNet). These datasets contain 100 classes in CIFAR100, 200 classes in CUB200, and cover ImageNet-R and ObjectNet, which exhibit large domain gaps relative to the pre-trained datasets (ImageNet [9]). Following [66, 70], we denote the data split as 'B-m Inc-n', meaning that the initial task contains m classes, and each subsequent task contains n classes. The random seed for class-order shuffling is fixed at 1993 [65, 70].

Baselines. Typical CL and pre-trained model (PTM)-based CL methods [70] are used to assess C-Flat Turbo. For the former, we cover the classical iCaRL [42] and MEMO [67] methods. For the latter, we compare against L2P [57], Ranpac [38], and EASE [71], spanning common CL categories.

Implementation details. All experiments and compared methods are implemented and reproduced in PyTorch and PILOT [6, 65, 70] on a single RTX 3090 GPU. Unless otherwise specified, all hyperparameters and configurations remain unchanged from the open-source repository [70]. To ensure a fair comparison, we evaluate all methods with the same model and vanilla SGD optimizer (Adam for L2P) and adopt ViT-B/16-IN1K as the representative pre-trained models [57, 69]. Implementation details can be found in Sec. A.3. For evaluation, we primarily present the results in terms of average accuracy (Avg), final task accuracy (Last), and the average running speed (Img/s).

4.2. Faster and Stronger Performance

We thoroughly evaluate the performance of C-Flat Turbo. As shown in Table 1, although pre-trained models exhibit strong generalization capabilities, their feature spaces remain susceptible to contamination during continual adaptation to evolving data distributions, thereby exacerbating catastrophic forgetting. While C-Flat mitigates this degradation through flat region search mechanisms, it incurs significant computational overhead. On the one hand, C-Flat Turbo addresses this limitation by reusing flatness-related shortcut directions extracted from earlier steps, without repeated computation. On the other hand, it flexibly combines C-Flat with base optimizers through a stage-wise step schedule and an adaptive trigger for regularization, further accelerating the training process. A more detailed per-task accuracy progression and ablation studies are provided in the Appendix A.7. Experimental results demonstrate that C-Flat Turbo achieves better accuracy than C-Flat, while significantly reducing training time. Notably, we find that C-Flat Turbo remains stable even in PTM scenarios with larger generalization gaps (e.g., CUB200, ImageNet-R, and ObjectNet). C-Flat Turbo trains CL models at about 2× the speed of C-Flat, and 0.6× that of SGD. Overall, whether applied to typical CL benchmarks or to scenarios with large domain gaps in PTM, C-Flat Turbo maintains strong performance while offering superior training efficiency compared to the baseline C-Flat optimizer.

4.3. Training From Scratch

Table 2 presents the accuracy results of iCaRL and MEMO with ResNet-18 and ResNet-34 trained from scratch. Notably, iCaRL benefits significantly from C-Flat Turbo, achieving an increase of 1.61% in last accuracy on ResNet-18 and 1.08% on ResNet-34. Similarly, MEMO exhibits substantial gains, particularly in final stage accuracy, where C-Flat Turbo improves performance by 3.05% on ResNet-18 and 2.28% on ResNet-34. Moreover, C-Flat Turbo forgets less than C-Flat across training, likely due to its softer sharpness constraint around local minima. The consistent improvements across different backbone architectures highlight the robustness of C-Flat Turbo in mitigating catastrophic forgetting while maintaining computational efficiency. Furthermore, the larger performance gains observed in MEMO suggest that C-Flat Turbo is particularly beneficial for expansion-based approaches, which involve multiple module updates and often struggle with stability and adaptability. These findings validate C-Flat Turbo as a highly effective strategy for continual learning, offering substantial accuracy improvements while enabling flexible training strategies.

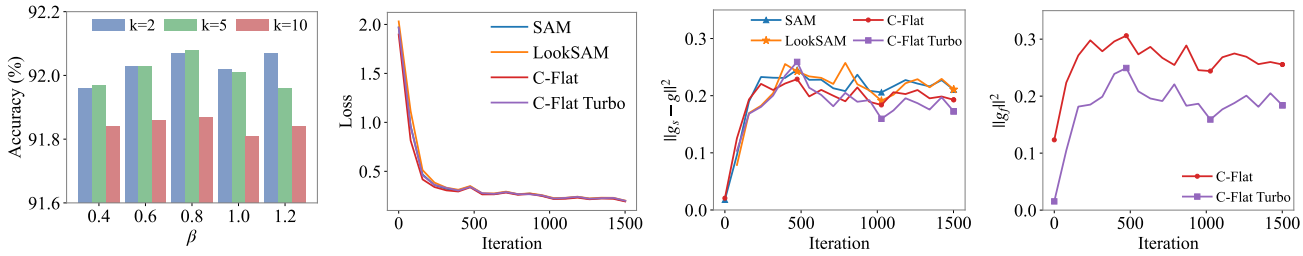
4.4. Comparison with Other Optimizers

In Table 3, we report the average accuracy, last accuracy, and training speeds on the CIFAR100 B0_Inc10 setting compared to various zeroth-order optimizers. In particular, the update interval for the sharpness term in LookSAM and the flatness term in C-Flat Turbo is fixed to 5.

For the performance, as shown in Table 3, we first observe that SAM and LookSAM do not offer obvious benefits over the vanilla optimizer, but C-Flat series shows significant improvement. The reason is that the backbone parameters loaded from the pre-trained model already possess strong generalization ability, resulting in uniformly low

Table 1. Accuracy and training speeds using five state-of-the-art methods, with a pre-trained ViT-B/16-IN1K backbone. Red and green denote the baseline and the efficient optimizer, respectively. **Bolded** indicates the best result.

Model	Method	CIFAR100 B0_Inc10		CUB B0_Inc10		IN-R B0_Inc20		ObjNet B0_Inc10		Img/s↑
		Avg↑	Last↑	Avg↑	Last↑	Avg↑	Last↑	Avg↑	Last↑	
Typical	iCaRL [42]	77.83	66.64	82.91	74.00	72.13	61.62	48.06	28.20	73.35 (100%)
	+*C-Flat [6]*	79.72	67.15	83.47	74.81	72.92	62.35	49.59	29.03	19.72 (26.9%)
	+*C-Flat Turbo*	**79.82**	**68.54**	**84.00**	**75.12**	**73.11**	**62.38**	**50.49**	**29.30**	45.89 (62.6%)
	MEMO [67]	82.26	73.89	86.66	79.90	70.96	61.05	56.22	38.32	135.14 (100%)
	+*C-Flat [6]*	82.61	75.49	87.03	80.73	71.69	62.93	56.50	39.45	46.11 (34.1%)
	+*C-Flat Turbo*	**83.02**	**75.76**	**87.15**	**80.92**	**72.38**	**63.55**	**57.52**	**40.62**	91.91 (68.0%)
PTM-based	L2P [57]	89.36	83.94	73.04	59.14	78.05	72.60	64.18	52.10	110.29 (100%)
	+*C-Flat [6]*	89.56	84.35	**74.36**	**62.11**	78.67	73.78	64.53	52.47	28.63 (30.0%)
	+*C-Flat Turbo*	**89.78**	**84.69**	74.12	61.97	**78.86**	**73.82**	**64.64**	**52.55**	65.50 (59.4%)
	Ranpac [38]	94.32	90.72	92.61	88.68	82.07	76.80	71.66	60.17	154.64 (100%)
	+*C-Flat [6]*	94.41	90.70	92.67	88.76	82.66	77.25	72.15	60.33	42.98 (27.8%)
	+*C-Flat Turbo*	**94.45**	**90.74**	**93.12**	**89.02**	**83.13**	**77.83**	**72.16**	**60.33**	94.34 (61.0%)
	EASE [71]	91.91	87.30	89.16	83.96	80.49	75.05	64.38	52.02	166.67 (100%)
	+*C-Flat [6]*	92.05	87.91	89.37	84.05	80.97	75.64	64.89	52.47	44.25 (26.5%)
	+*C-Flat Turbo*	**92.36**	**87.96**	**89.56**	**84.18**	**81.18**	**75.76**	**64.96**	**52.61**	102.74 (61.6%)



(a) Sensitivity analysis (b) Variation on loss (c) Variation on sharpness (d) Variation on flatness

Figure 5. (a) Sensitivity analysis of k. (b) - (d) Evolution of loss, sharpness and flatness on EASE w/ and w/o C-Flat Turbo.

losses around local minima under various parameter perturbations. Nevertheless, C-Flat Turbo further refines the strong generalization of the pre-trained model by the horizontal and vertical components of the oracle gradient, thus boost CL.

For the training speeds, as concluded in Table 3, although LookSAM significantly accelerates training compared to SAM by reusing historical gradients, it degrades performance on EASE due to its single zeroth-order regularization and the simplistic use of past iteration gradients. C-Flat Turbo differs from LookSAM in that it progressively updates sharpness gradients to vanilla components and imposes more strict constraints to encourage convergence to a flatter region. Compared with C-Flat, our efficient method improves throughput from 30.0% to 59.4% on L2P and from 26.5% to 61.6% on EASE, making it even faster than SAM.

4.5. Ablation Studies

Parameter Sensitivity. In Figure 5a, we empirically investigate the sensitivity of the scale factor β and the sampling step k in EASE, trained under the CIFAR100 B0_Inc10 setting. Here, k denotes the frequency of leveraging the sharpness and flatness properties in C-Flat Turbo. For fairness, the sharpness and flatness steps are kept equal, meaning that C-Flat Turbo-k indicates the cached sharpness and flatness gradients are updated every k iterations. The hyperparameters ρ and λ are fixed at 0.05 and 0.2, respectively. Figure 5a demonstrates that $\beta = 0.8$ is the optimal choice in most cases. Moreover, when $k = 2$ and $k = 5$, the performance exhibits similar accuracy fluctuations as long as β lies within the range [0.4, 1.0].

Evolution of Sharpness and Flatness. We approximate the sharpness and flatness gradients using cached branches and current SGD directions, following the same optimization procedure as C-Flat. This efficient approach does not slow convergence. As shown in Figure 5b, C-Flat Turbo converges as fast as other optimizers. Figure 5c illustrates that all optimizers start with low sharpness initially, owing to the pre-trained backbone's generalization, and that the sharpness then declines alongside the training loss. Notably, the sharpness and flatness gradients in C-Flat Turbo converge to lower values than in C-Flat, due to the intermediate gradient descent steps being free of regularization constraints.

Table 2. Accuracy results with ResNet-18 and ResNet-34 trained from scratch. **Bolded** indicates the best result.

Method	ResNet-18			ResNet-34		
	Avg↑	Last↑	Img/s↑	Avg↑	Last↑	Img/s↑
iCaRL [42]	$59.13_{\pm 0.30}$	$41.23_{\pm 0.91}$	2333.3 (100%)	$58.80_{\pm 1.04}$	$41.26_{\pm 0.99}$	1250.4 (100%)
+*C-Flat [6]*	$59.45_{\pm 0.18}$	$42.47_{\pm 0.06}$	686.3 (29.4%)	$59.55_{\pm 0.93}$	$42.09_{\pm 0.61}$	359.8 (28.8%)
+*C-Flat Turbo*	$\mathbf{59.84_{\pm 0.05}}$	$\mathbf{42.84_{\pm 0.18}}$	1750.1 (75.0%)	$\mathbf{59.75_{\pm 0.55}}$	$\mathbf{42.34_{\pm 0.54}}$	960.6 (76.8%)
MEMO [67]	$48.63_{\pm 0.78}$	$29.19_{\pm 0.89}$	2413.8 (100%)	$68.49_{\pm 1.74}$	$57.05_{\pm 1.46}$	1873.2 (100%)
+*C-Flat [6]*	$49.98_{\pm 0.61}$	$30.76_{\pm 0.57}$	886.1 (36.7%)	$69.00_{\pm 1.39}$	$59.29_{\pm 0.73}$	569.1 (30.4%)
+*C-Flat Turbo*	$\mathbf{50.51_{\pm 0.55}}$	$\mathbf{32.24_{\pm 0.27}}$	1891.9 (78.4%)	$\mathbf{69.48_{\pm 1.25}}$	$\mathbf{59.33_{\pm 0.65}}$	1372.5 (73.3%)

Table 3. Accuracy and training speeds of L2P and EASE across various optimizers. Task orders are shuffled for evaluation on dynamic sequences. Red and green denote the baseline and the efficient optimizer. **Bolded** indicates the best result.

Method	L2P [57]			EASE [71]		
	Avg↑	Last↑	Img/s↑	Avg↑	Last↑	Img/s↑
Vanilla	$87.92_{\pm 1.30}$	$83.66_{\pm 1.49}$	110.29 (100%)	$91.16_{\pm 0.71}$	$87.49_{\pm 0.32}$	166.67 (100%)
+*SAM [14]*	$88.18_{\pm 1.39}$	$83.84_{\pm 1.39}$	56.60 (51.3%)	$91.36_{\pm 0.82}$	$87.61_{\pm 0.27}$	86.71 (52.0%)
+*LookSAM [36]*	$88.35_{\pm 1.39}$	$83.80_{\pm 1.20}$	88.76 (80.5%)	$90.89_{\pm 0.86}$	$86.99_{\pm 0.28}$	132.74 (79.6%)
+*C-Flat [6]*	$88.62_{\pm 0.88}$	$84.04_{\pm 0.60}$	28.63 (30.0%)	$91.60_{\pm 1.18}$	$87.69_{\pm 0.50}$	44.25 (26.5%)
+*C-Flat Turbo*	$\mathbf{89.57_{\pm 0.36}}$	$\mathbf{84.48_{\pm 0.09}}$	65.50 (59.4%)	$\mathbf{91.75_{\pm 0.80}}$	$\mathbf{87.74_{\pm 0.20}}$	102.74 (61.6%)

4.6. Discussion on Scheduler Choices

Figure 6 compares the accuracy and training speed of MEMO and EASE. For the linear scheduler, we increase the step size k with the task number n, following $k = 5{+}10{\cdot}n/N$, whereas in the variant without a scheduler, k is fixed at 5. MEMO expands its architecture for each new task, which increases both the number of parameters and the computational cost, thereby prolonging training. The figure shows that C-Flat Turbo, with or without the scheduler consistently outperforms vanilla C-Flat in accuracy. In terms of speed, both variants are faster, and the scheduler yields an additional speedup of approximately 15%. In contrast, EASE reuses frozen adapters, which keeps the training cost stable. As k increases, the scheduler provides roughly a 30% faster over C-Flat while maintaining comparable accuracy.

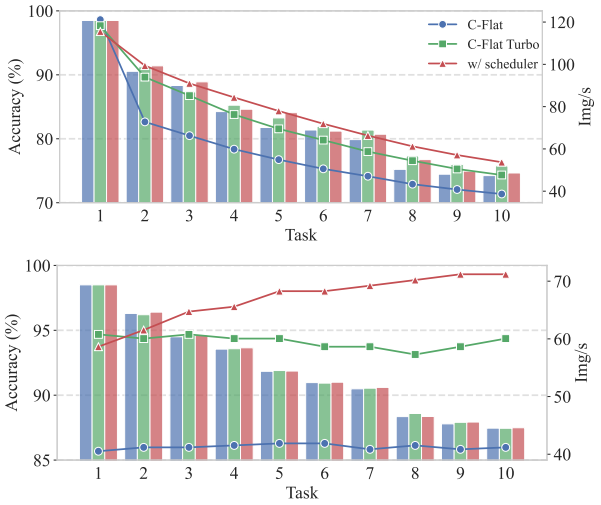


Figure 6. Accuracy and training speed comparison of different Turbo scheduling strategies. The upper part shows results on MEMO, while the lower part shows results on EASE.

5. Conclusion

This paper proposes C-Flat Turbo, an efficient optimizer that selectively takes shortcuts along stable directions toward flatness. By reusing historical sharpness and flatness signals instead of recomputing them at every step, C-Flat Turbo accelerates C-Flat while preserving its regularization effect. Building on the empirical observation that flatness gradients diminish across tasks, we further introduce (i) a linear scheduler that adaptively adjusts the turbo interval and (ii) an adaptive trigger that selectively activates C-Flat regularization only when it is most beneficial. Overall, our results reveal that adaptively modifying the oracle gradient can yield tangible efficiency gains in continual learning, and that C-Flat Turbo can be seamlessly plugged into a wide range of CL methods to provide consistent training speedups.

Limitations. Although C-Flat Turbo is more efficient than C-Flat, it remains costlier than standard optimizers due to the overhead of estimating the flatness component. Further reducing this cost, such as through cheaper finite-difference schemes or lower-rank approximations, is important future work. Moreover, our evaluation is limited to typical and PTM-based CIL benchmarks. Extending C-Flat Turbo to other CL scenarios, like vision-language models or reinforcement learning agents, may further reveal its generalization and robustness.

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *CVPR*, 2020. 2

[2] Afra Feyza Akyürek, Ekin Akyürek, Derry Tanti Wijaya, and Jacob Andreas. Subspace regularizers for few-shot class incremental learning. *ICLR*, 2022. 2

[3] Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. Shaping the learning landscape in neural networks around wide flat minima. *Proceedings of the National Academy of Sciences*, 117(1):161–170, 2020. 2

[4] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 2019. 6

[5] Prashant Bhat, Bahram Zonooz, and Elahe Arani. Task-aware information routing from common representation space in lifelong learning. *ICLR*, 2023. 2

[6] Ang Bian, Wei Li, Hangjie Yuan, Mang Wang, Zixiang Zhao, Aojun Lu, Pengliang Ji, Tao Feng, et al. Make continual learning stronger via c-flat. *Advances in Neural Information Processing Systems*, 37:7608–7630, 2024. 1, 2, 3, 6, 7, 8, 5

[7] Sungmin Cha, Hsiang Hsu, Taebaek Hwang, Flavio P Calmon, and Taesup Moon. Cpr: classifier-projection regularization for continual learning. *ICLR*, 2021. 2

[8] Danruo Deng, Guangyong Chen, Jianye Hao, Qiong Wang, and Pheng-Ann Heng. Flattening sharpness for dynamic gradient projection memory benefits continual learning. *NeurIPS*, 34, 2021. 1, 2

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 2009. 6

[10] Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent YF Tan. Efficient sharpness-aware minimization for improved training of neural networks. *arXiv preprint arXiv:2110.03141*, 2021. 2

[11] Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. *Advances in Neural Information Processing Systems*, 35:23439–23451, 2022. 2

[12] Tao Feng, Mang Wang, and Hangjie Yuan. Overcoming catastrophic forgetting in incremental object detection via elastic response distillation. In *CVPR*, 2022. 2

[13] Tao Feng, Wei Li, Didi Zhu, Hangjie Yuan, Wendi Zheng, Dan Zhang, and Jie Tang. Zeroflow: Overcoming catastrophic forgetting is easier than you think. In *Forty-second International Conference on Machine Learning*, 2025. 1

[14] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020. 1, 2, 3, 8

[15] Yulu Gan, Yan Bai, Yihang Lou, Xianzheng Ma, Renrui Zhang, Nian Shi, and Lin Luo. Decorate the newcomers: Visual domain prompt for continual test time adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7595–7603, 2023. 2

[16] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-dfcil: Relation-guided representation learning for data-free class incremental learning. In *ECCV*, 2022. 2

[17] Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11483–11493, 2023. 2

[18] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020. 1, 2

[19] Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. *NeurIPS*, 32, 2019. 1, 2

[20] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021. 6

[21] Zhiyuan Hu, Yunsheng Li, Jiancheng Lyu, Dashan Gao, and Nuno Vasconcelos. Dense network expansion for class incremental learning. In *CVPR*, 2023. 2

[22] Kishaan Jeeveswaran, Prashant Bhat, Bahram Zonooz, and Elahe Arani. Birt: Bio-inspired replay in vision transformers for continual learning. *ICML*, 2023. 2

[23] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022. 1, 2

[24] Weisen Jiang, Hansi Yang, Yu Zhang, and James Kwok. An adaptive policy to employ sharpness-aware minimization. *arXiv preprint arXiv:2304.14647*, 2023. 5, 2

[25] Dahuin Jung, Dongyoon Han, Jihwan Bang, and Hwanjun Song. Generating instance-level prompts for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11847–11857, 2023. 1, 2

[26] Muhammad Gul Zain Ali Khan, Muhammad Ferjad Naeem, Luc Van Gool, Didier Stricker, Federico Tombari, and Muhammad Zeshan Afzal. Introducing language guidance in prompt-based continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11463–11473, 2023. 1, 2

[27] Muhammad Uzair Khattak, Syed Talal Wasim, Muzammal Naseer, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Self-regulating prompts: Foundational model adaptation without forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15190–15200, 2023. 1, 2

[28] Do-Yeon Kim, Dong-Jun Han, Jun Seo, and Jaekyun Moon. Warping the space: Weight space rotation for class-incremental few-shot learning. In *ICLR*, 2022. 2

[29] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran

Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017. 2

[30] Yajing Kong, Liu Liu, Huanhuan Chen, Janusz Kacprzyk, and Dacheng Tao. Overcoming catastrophic forgetting in continual learning by exploring eigenvalues of hessian matrix. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11):16196–16210, 2024. 1

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report*, 2009. 6

[32] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, 2018. 2

[33] Huiwei Lin, Baoquan Zhang, Shanshan Feng, Xutao Li, and Yunming Ye. Pcr: Proxy-based contrastive replay for online class-incremental continual learning. In *CVPR*, 2023. 2

[34] Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Trgp: Trust region gradient projection for continual learning. *arXiv preprint arXiv:2202.02931*, 2022. 2

[35] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *CVPR*, 2021. 2

[36] Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12360–12370, 2022. 1, 2, 4, 5, 8

[37] Ziwei Liu, Borui Kang, Hangjie Yuan, Zixiang Zhao, Wei Li, Yifan Zhu, and Tao Feng. Continual gui agents. *arXiv preprint arXiv:2601.20732*, 2026. 1

[38] Mark D. McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *Advances in Neural Information Processing Systems*, pages 12022–12053. Curran Associates, Inc., 2023. 2, 6, 7

[39] Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *J. Mach. Learn. Res.*, 2023. 1

[40] Youngmin Oh, Donghyeon Baek, and Bumsub Ham. Alife: Adaptive logit regularizer and feature replay for incremental semantic segmentation. *NeurIPS*, 2022. 2

[41] Haomiao Qiu, Miao Zhang, Ziyue Qiao, and Liqiang Nie. Train with perturbation, infer after merging: A two-stage framework for continual learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 1

[42] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017. 2, 6, 7, 8

[43] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *NeurIPS*, 2019. 2

[44] Tim GJ Rudner, Freddie Bickford Smith, Qixuan Feng, Yee Whye Teh, and Yarin Gal. Continual learning via sequential function-space variational inference. In *International Conference on Machine Learning*, pages 18871–18887. PMLR, 2022. 2

[45] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2020. 2

[46] Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, pages 4555–4564, 2018. 2

[47] Guangyuan Shi, Jiaxin Chen, Wenlong Zhang, Li-Ming Zhan, and Xiao-Ming Wu. Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima. *NeurIPS*, 2021. 1, 2

[48] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33: 18098–18109, 2020. 3

[49] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11909–11919, 2023. 1, 2

[50] Wenju Sun, Qingyong Li, Jing Zhang, Wen Wang, and Yangli-ao Geng. Decoupling learning and remembering: A bilevel memory framework with knowledge projection for task-incremental learning. In *CVPR*, 2023. 2

[51] Zhicheng Sun, Yadong Mu, and Gang Hua. Regularizing second-order influences for continual learning. In *CVPR*, 2023. 2

[52] Yu-Ming Tang, Yi-Xing Peng, and Wei-Shi Zheng. When prompt-based incremental learning does not meet strong pretraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1706–1716, 2023. 1, 2

[53] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 6

[54] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024. 1, 2

[55] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *Advances in Neural Information Processing Systems*, 35:5682–5695, 2022. 2

[56] Yabin Wang, Zhiheng Ma, Zhiwu Huang, Yaowei Wang, Zhou Su, and Xiaopeng Hong. Isolation and impartial aggregation: A paradigm of incremental learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10209–10217, 2023. 2

[57] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 1, 2, 6, 7, 8

[58] Shipeng Yan, Jiangwei Xie, and Xuming He. DER: dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021. 2

[59] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *International Conference on Learning Representations*, 2020. 2

[60] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19148–19158, 2023. 2

[61] Xingxuan Zhang, Renzhe Xu, Han Yu, Hao Zou, and Peng Cui. Gradient norm aware minimization seeks first-order flatness and improves generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20247–20257, 2023. 1, 3, 6, 4

[62] Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning. In *International Conference on Machine Learning*, pages 26982–26992. PMLR, 2022. 3

[63] Yang Zhao, Hao Zhang, and Xiuyuan Hu. Ss-sam: Stochastic scheduled sharpness-aware minimization for efficiently training deep neural networks. *ArXiv*, abs/2203.09962, 2022. 5, 2

[64] Qihuang Zhong, Liang Ding, Li Shen, Peng Mi, Juhua Liu, Bo Du, and Dacheng Tao. Improving sharpness-aware minimization with fisher mask for better generalization on language models. *arXiv preprint arXiv:2210.05497*, 2022. 1

[65] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: A python toolbox for class-incremental learning, 2023. 6

[66] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Deep class-incremental learning: A survey. *arXiv preprint arXiv:2302.03648*, 2023. 1, 6

[67] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. *ICLR*, 2023. 2, 6, 7, 8

[68] Da-Wei Zhou, Yuanhan Zhang, Jingyi Ning, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Learning without forgetting for vision-language models. *arXiv preprint arXiv:2305.19270*, 2023. 2

[69] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 2024. 2, 6

[70] Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. *arXiv preprint arXiv:2401.16386*, 2024. 1, 2, 6

[71] Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. *arXiv preprint arXiv:2403.12030*, 2024. 2, 6, 7, 8

[72] Zhanpeng Zhou, Mingze Wang, Yuchen Mao, Bingrui Li, and Junchi Yan. Sharpness-aware minimization efficiently selects flatter minima late in training. *arXiv preprint arXiv:2410.10373*, 2024. 2

[73] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *CVPR*, pages 5871–5880, 2021. 2

[74] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *CVPR*, 2022. 2

[75] Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha Dvornek, Sekhar Tatikonda, James Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. *arXiv preprint arXiv:2203.08065*, 2022. 1