From Generic to Dedicated: A Novel Optimizer for Online Continual Learning

Yongyi Wu[*]
University of Houston
Houston, USA
yongyiwu.cn@gmail.com

Zheng Wang
University of Houston
Houston, USA
zwang214@cougarnet.uh.edu

Sen Lin[†]
University of Houston
Houston, USA
slin50@central.uh.edu

Abstract

Online Continual Learning (OCL) requires models to learn from non-stationary data streams under a strict single-pass constraint, making them highly susceptible to catastrophic forgetting. While existing studies have explored various strategies from data, architecture, and optimization perspectives, the optimizer often directly uses generic approaches such as SGD and AdamW. In this work, we reveal a new link between optimization dynamics and OCL needs by recasting Polyak-Ruppert averaging as the engine of "plasticity" and Primal averaging as the anchor of "stability". This inspires our novel optimizer, SPIN (**S**tability-**P**lasticity **IN**terpolation), which explicitly decouples plasticity and stability by interpolating between a fast-moving "plasticity" sequence and an adaptive "stability" sequence. We theoretically demonstrate that SPIN implicitly employs an inverse Hessian approximation, providing crucial Tikhonov regularization to damp noisy, single-sample Hessian estimates and mitigate forgetting. More importantly, SPIN can be seamlessly integrated into existing OCL methods by taking resulted gradients as input and replacing standard optimizers. Extensive experiments show that using SPIN instead of standard optimizers yields consistent and significant performance gains across multiple OCL approaches, including the challenging rehearsal-free setting and under strict memory constraints.

1 Introduction

Continual Learning (CL) aims to enable AI systems to continuously acquire new knowledge while retaining previously learned skills, a capability essential for many real-world applications where data arrives sequentially [62, 76, 59, 9]. However, the common assumption in offline CL, that models are able to revisit task data multiple times during training, rarely holds. This has motivated the rise of Online Continual Learning (OCL), a more realistic paradigm where each sample is observed only once [21, 58, 41]. Nevertheless, OCL inevitably exacerbates key challenges like catastrophic forgetting [16, 27] and the plasticity-stability dilemma [43] under strict single-pass constraints.

To address this, existing OCL research has largely followed the classical offline CL paradigm, focusing on data, loss functions, and architectural designs, such as replay, regularization, and distillation [50, 7, 24, 74, 71]. In most of these approaches, the lower-level optimizer is treated as a fixed, secondary component, directly using generic strategies such as vanilla SGD [51] or AdamW [26, 39]. In both SGD and AdamW, the gradient is evaluated at the current parameter given the newly arrived sample. Consequently, the optimization direction is dominated by the instantaneous data distribution, even when momentum is used. This behavior can lead to accelerated forgetting under non-stationary data streams. From the perspective of online optimization theory, however, the optimizer itself

[*]Work done during an internship at University of Houston
[†]Corresponding author: slin50@central.uh.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

plays a decisive role in determining convergence, stability, and generalization [23]. Many works have explored the role of optimization in OCL, proposing methods that modify gradient updates via projection, reweighting, proximal constraints, or curvature-aware preconditioning [38, 8, 69, 73]. These approaches operate primarily at the level of gradient manipulation, adapting the update direction while still relying on standard optimizers. Therefore, a critical question arises: "*Instead of relying on generic optimizers, can we design a novel optimizer specifically tailored to OCL that effectively mitigates the forgetting problem and further pushes the limits of existing OCL approaches?*"

To answer this question, we revisit the fundamentals of stochastic optimization and establish a direct connection between its averaging mechanisms and the stability–plasticity dilemma in OCL. Specifically, we reinterpret *Polyak–Ruppert averaging* [48, 52] as the engine of **plasticity**, since it evaluates gradients at the current, fast-evolving iterate and thus quickly adapts to new data. In contrast, *Primal averaging* [44, 10, 13] serves as the anchor of **stability**, as it computes gradients at the averaged, slowly updating point, thereby retaining previously acquired knowledge. We empirically examine these two averaging schemes, as shown in Figure 1. Polyak–Ruppert averaging achieves higher learning accuracy but suffers from greater forgetting, reflecting its strong plasticity yet limited stability. Primal averaging, on the other hand, exhibits the opposite behavior, showing lower plasticity while maintaining superior stability.

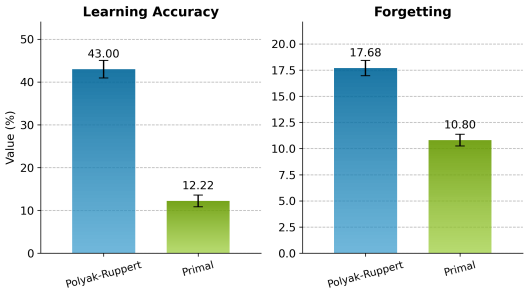


Figure 1: The comparison of plasticity (Learning Accuracy) and stability (Average Forgetting) of Polyak-Ruppert averaging and Primal averaging on split CIFAR-100. Learning Accuracy for the j-th task is the accuracy evaluated on the test set of task j immediately after the network has been trained sequentially from task 1 up to task j.

Driven by these observations, we propose a novel optimizer, SPIN (**S**tability-**P**lasticity **IN**terpolation), which explicitly manages this trade-off by operating on two coupled iterate sequences. SPIN maintains a plasticity model sequence for rapid adaptation and a stability model sequence for consolidated knowledge. Its core mechanism links these sequences in two ways. First, gradients are evaluated at an interpolated point between the two models, enabling a unified view of multiple averaging strategies. Second, the stability sequence is updated by smoothly integrating information from the plastic sequence, preventing corruption under the high-variance, non-stationary OCL stream. We find that this formulation not only blends the benefits of different averaging approaches, but also connects conceptually to several regularization-based and meta-continual learning methods. Our main contributions can be summarized as follows:

(1) We propose a novel optimizer dedicated for OCL, namely SPIN, which explicitly decouples plasticity and stability at the model level. Crucially, SPIN can be directly integrated into existing OCL methods by simply replacing their lower-level optimizers. In other words, *SPIN operates similarly to SGD by taking only the gradients as input, regardless of how those gradients are modified by the upstream OCL methods.*

(2) We establish a unified optimization framework for OCL, providing new theoretical insights into the plasticity-stability dilemma. Within this framework, we theoretically justify the design of SPIN and its effectiveness in mitigating catastrophic forgetting.

(3) Extensive experiments validate the effectiveness and generality of SPIN, showing consistent, significant performance gains across diverse OCL approaches, including the challenging rehearsal-free setting and under strict memory constraints.

2 Related Work

Continual Learning. CL aims to enable models to learn from a sequence of tasks without forgetting previous knowledge [47, 76, 35]. Conventionally, CL strategies primarily fall into three categories: Memory-based methods revisit past knowledge, including replay-based methods that retrain on stored old data [50, 6, 65, 14] and orthogonal projection-based methods that project gradients to prevent interference [69, 34, 8]; Regularization-based methods, which add penalties to the loss function to

prevent significant changes to weights deemed important for old tasks [60, 27, 77]; and Parameter Isolation methods, which use dedicated network parameters for different tasks, either by expanding the architecture or by isolating pathways within a fixed network [70, 61, 33, 72, 32]. CLS-ER combines short- and long-term semantic memories with experience replay to support adaptation and knowledge retention [2]. Parameter averaging offers another approach: CoMA combines model parameters from previous and current tasks to balance stability and plasticity, while CoFiMA further weights parameters using Fisher information [42]. While optimization-oriented methods exist in offline CL (e.g., [66, 34, 54]), their fundamental reliance on accessing past data makes them incompatible with the single-pass constraints of OCL.

Online Continual Learning. OCL requiress models to learn from a strict single-pass data stream [23]. Replay-based strategies dominate this space [57], ranging from simple buffer combinations (ER [50], DER [6]) to advanced objective modifications addressing representation drift (ER-ACE, OCM, GSA, OnPro [7, 17, 67, 18]). Recent methods further boost adaptability via multi-level supervision (MOSE [71]) or advanced state-space and collaborative modeling (CCL-DC [63], S6MOD [37]). CBA addresses task-recency bias under distribution shifts through a continual bias adaptor [64]. A-GEM constrains the current gradient to avoid increasing the loss on past data by projecting it onto the feasible region defined by reference gradients from memory [8]. LPR applies a layer-wise proximal preconditioning that stabilizes optimization by limiting abrupt changes to past representations while learning from both new and replay data [73]. *In contrast, our work focuses on the lower-level optimizer design that takes upper-level gradients as input, and provides a complementary and orthogonal perspective that can be applied on top of diverse OCL methods.*

Averaging-Based Optimization. Classical optimization methods use coupled iterate sequences and averaging to accelerate convergence or reduce variance, as exemplified by Nesterov's accelerated methods and Polyak–Ruppert averaging [45, 46, 48, 52]. Further analyses establish connections between iterate averaging, momentum, and convergence guarantees [56, 11]. In deep learning, stochastic weight averaging (SWA) improves generalization by averaging weights along the training trajectory [25], while Lookahead periodically interpolates and synchronizes fast and slow weights [75]. Schedule-Free optimization combines interpolated gradient evaluation with cumulative iterate averaging to avoid a prespecified training horizon [12], with nonconvex convergence guarantees established for its SGD variant [1]. Generalized Primal Averaging (GPA) uses independent coefficients for gradient evaluation and iterate averaging to smooth DiLoCo updates and improve LLM training efficiency [13]. SPIN shares this coupled interpolation-and-averaging structure. Our focus is on how the gradient evaluation point and averaging dynamics jointly influence the stability–plasticity trade-off in non-stationary OCL, where each update must accommodate incoming data while preserving knowledge of previously encountered tasks.

3 Preliminaries

Online Continual Learning Setting. Consider the standard OCL setting, where a model f_θ with parameters θ learns from a non-stationary data stream of T tasks [21, 58, 41]. The dataset for the t-th task is denoted as $\mathcal{D}_t = \{(v_i, y_i)\}$ with input-label pairs, where $\mathcal{C}_t$ is the set of classes for the t-th task. Let $\mathcal{D} = \bigcup_{t=1}^{T} \mathcal{D}_t$. Following the common protocol that $\mathcal{C}_t \cap \mathcal{C}_{t'} = \emptyset$ for $t \neq t'$, the model must infer the correct class from the entire set of seen classes $\mathcal{Y} = \bigcup_{i=1}^{T} \mathcal{C}_i$ at test time [57]. The model processes this data stream sequentially online, where data from each task is provided as a stream of mini-batches. We denote a single mini-batch from the current stream as $\mathcal{B}_{\text{stream}} = \{(v_i, y_i)\}_{i=1}^{B}$, with batch size B. Crucially, OCL imposes a strict **single-pass constraint**: the model observes each training sample (v, y) *exactly once* and cannot re-access past task data (unless stored in a buffer). This constraint, combined with data distribution shift, leads to catastrophic forgetting [16], where the model's performance on previous tasks degrades significantly as it learns new ones. To mitigate forgetting, many OCL methods employ experience replay [50]. A small memory buffer $\mathcal{M}$ with a fixed capacity M is maintained to store a subset of past samples. At each training step, the model trains on the union of the current stream batch and a replay batch drawn from $\mathcal{M}$. The goal is to learn parameters θ_T at the end of the training stream that minimizes the empirical risk across all encountered tasks: $\min_{\theta_T} \sum_{t=1}^{T} \mathbb{E}_{(v,y)\sim\mathcal{D}_t} [\mathcal{L}(f(v; \theta_T), y)]$, where $\mathcal{L}$ is the loss function.

Averaging-Based Methods in Optimization. Nesterov's accelerated method is an optimization algorithm designed to achieve faster convergence rates for L-smooth convex functions [45, 46]. One

common form, known as AC-SA [29], utilizes an interpolation between two sequences of iterates:

$$x_t = (1 - c_{t+1})\theta_t + c_{t+1}\phi_t, \tag{1}$$

$$\phi_{t+1} = \phi_t - \frac{t+1}{2L}\nabla\mathcal{L}(x_t), \tag{2}$$

$$\theta_{t+1} = (1 - c_{t+1})\theta_t + c_{t+1}\phi_{t+1}. \tag{3}$$

Here, ϕ_t is the base sequence, θ_t is the averaged sequence, and x_t is the point at which the gradient is evaluated. The coefficient c_{t+1} is typically set as $c_{t+1} = 2/(t+2)$ for non-strongly convex functions.

Gradient-Space Interpolation in OCL. Many replay-based OCL methods can be written as a gradient-space interpolation between the current task and past tasks [69]. Let g_t^{cur} denote the gradient on the current sample at step t, and $\{g_t^{(i)}\}_{i<t}$ the gradients estimated from the replay buffer for previous tasks. A generic form is

$$u_t \;=\; g_t^{\text{cur}} \;+\; \textstyle\sum_{i<t} \lambda_i \, g_t^{(i)}, \tag{4}$$

where u_t is the update direction fed to lower-level optimizers and λ_i are task-dependent weights.

Different choices of λ_i recover many existing OCL algorithms: experience replay methods such as ER and DER use uniform averaging over buffered gradients [50, 6, 69]; gradient-alignment methods such as GEM, A-GEM, and MER implicitly realize Eq. 4 via projection-based or bi-level updates [38, 8, 49]; Pareto-style methods (e.g. , POCL) optimize λ to balance previous tasks within a multi-objective formulation [69]. In all these cases, the gradients are evaluated at the current parameter θ_t, and only their linear combination in gradient space is modified.

Which Averaging Mechanism Suits OCL. Under the viewpoint of Polyak–Ruppert averaging and Primal averaging, the two possible ways to form an averaged direction are: (i) averaging the gradients at ϕ_t and θ_t, or (ii) averaging the iterates and then taking a single gradient at the interpolated point. Gradient-space averaging's formulation can be generalized as a weighted average of gradients:

$$g_t^{\text{grad}} = (1 - \beta)\nabla\mathcal{L}(\phi_t) + \beta\nabla\mathcal{L}(\theta_t). \tag{5}$$

Compared with directly averaging gradients from different iterates, parameter-space averaging evaluates the gradient at a single interpolated point between two coupled sequences:

$$g_t^{\text{param}} = \nabla\mathcal{L}((1 - \beta)\phi_t + \beta\theta_t)\,. \tag{6}$$

This formulation induces a coherent optimization direction along the shared parameter manifold while implicitly incorporating local curvature information through the interpolated evaluation point. In contrast, gradient-space averaging mixes gradients computed at different locations in parameter space, which may lead to inconsistent update directions under non-stationary data streams. Motivated by these observations, we adopt parameter-space interpolation as the core design principle of SPIN. In Section 4.2, we further show that this formulation admits a unified optimization interpretation and naturally connects to curvature-aware continual learning methods. Empirical comparisons between gradient-space and parameter-space averaging are presented in Section 5.3.

4 Method

In this section, we first present the general form of our SPIN optimizer in Section 4.1, and then provide a theoretical perspective that connects our optimizer with existing methods in Section 4.2. Finally, we establish the theoretical guarantees and empirical validation of our optimizer in Section 4.3.

4.1 The SPIN Optimizer

We propose our SPIN optimizer as follows:

$$x_t = (1 - \beta)\phi_t + \beta\theta_t \tag{7}$$

$$\phi_{t+1} = \mathit{Opt}(\phi_t, \nabla\mathcal{L}(x_t), \gamma) \tag{8}$$

$$\theta_{t+1} = \theta_t + \alpha(\phi_{t+1} - \theta_t) \tag{9}$$

where $\mathit{Opt}(\cdot)$ denotes the update function of a standard base optimizer (e.g. , SGD or AdamW) with learning rate γ. Here, ϕ_t is the fast-moving plasticity sequence, x_t is the interpolated point where

the stochastic gradient $\nabla\mathcal{L}(x_t)$ is evaluated, and θ_t is a slow-moving stability sequence that evolves jointly. The coefficients $\alpha, \beta \in [0, 1]$ control the synchronization strength and the interpolation ratio between the two sequences, respectively. Note that $\mathcal{L}$ is non-stationary due to the changing data distributions [57, 41] and depends on the upstream algorithm design. Detailed pseudo-code of the SPIN algorithm is provided in Appendix I.

SPIN adopts a synchronous dual-sequence coupling mechanism: the plasticity sequence ϕ_t rapidly adapts to incoming data, while the stability sequence θ_t incrementally consolidates these updates into a robust representation. This design ensures that both sequences evolve jointly at every iteration. The plasticity sequence ϕ_t captures fast adaptation to new data, whereas the stability sequence θ_t gradually consolidates these changes into a consistent and robust representation. In the OCL setting, where the data distribution drifts over time, such tightly coupled dynamics allow θ_t to promptly absorb relevant new information while preventing abrupt forgetting of past knowledge. Moreover, SPIN evaluates the gradient at the interpolated point x_t. This intermediate evaluation allows the optimizer to blend the short-term plastic response of ϕ_t with the long-term stability encoded in θ_t, effectively guiding the update toward a compromise direction between adaptation and retention.

4.2 A Unified View

To understand the internal dynamics of the SPIN optimizer, we first analyze its update rules from first principles. The optimizer maintains two sequences, ϕ_t and θ_t, updated as per Eq. 9. Gradients $g_t = \nabla\mathcal{L}(x_t)$ are computed at $x_t = \theta_t + (1 - \beta)d_t$, where $d_t = \phi_t - \theta_t$ is the gap vector. First, assuming the loss function $\mathcal{L}_t(\theta)$ is sufficiently smooth, we employ the first-order Taylor expansion of the gradient $g_t = \nabla\mathcal{L}(x_t)$ around the iterate θ_t, which gives:

$$g_t \approx \nabla\mathcal{L}_t(\theta_t) + \nabla^2\mathcal{L}_t(\theta_t)(x_t - \theta_t) = \nabla\mathcal{L}_t(\theta_t) + (1 - \beta)H_t d_t \tag{10}$$

where $H_t := \nabla^2\mathcal{L}_t(\theta_t)$ is the Hessian at θ_t. Second, invoking a steady-state assumption for the gap vector d_t (i.e. , $d_{t+1} \approx d_t$), allows us to express d_t as a function of the gradient $\nabla\mathcal{L}_t(\theta_t)$ and Hessian H_t [3]. This analysis leads to the following result:

Proposition 1 *Suppose the gradient $g_t = \nabla\mathcal{L}(x_t)$ can be approximated by its first-order Taylor expansion around θ_t as in Eq. 10, and assume the gap vector $d_t = \phi_t - \theta_t$ reaches a quasi-steady state ($d_{t+1} \approx d_t$). Then the effective update for the iterate θ_t simplifies to:*

$$\theta_{t+1} - \theta_t \approx -\gamma[I + \rho_t H_t]^{-1}\nabla\mathcal{L}_t(\theta_t), \tag{11}$$

where $\rho_t = \gamma(1 - \alpha)(1 - \beta)/\alpha \geq 0$. (Full derivation in Appendix A.1.)

Recent work [68] unifies regularization-based methods and Meta-Continual Learning by showing that they both approximate a unified iterative update of the form:

$$\theta \leftarrow \theta - \alpha\tilde{H}^{-1}\nabla\mathcal{L}(\theta), \tag{12}$$

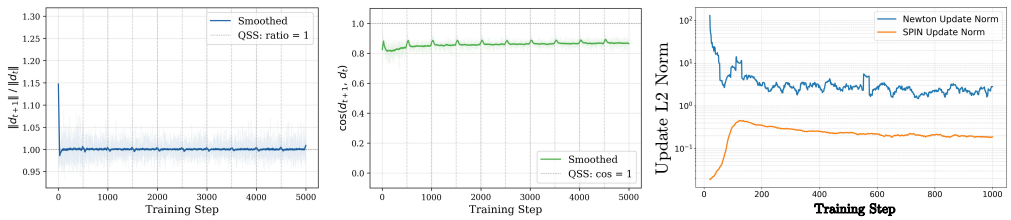
where $\tilde{H}^{-1}$ represents a specific (explicit or implicit) inverse Hessian approximation (e.g. , H_{reg}^{-1} from past-task penalties or H_M^{-1} from bi-level meta-objectives). Upon comparing our derived effective update for SPIN (Eq. 11) with this unified form (Eq. 12), a clear structural equivalence emerges. The SPIN optimizer, despite its distinct formulation, implicitly arrives at the same general structure by effectively using $\tilde{H}^{-1} \approx \gamma(I + \rho_t H_t)^{-1}$ as its inverse Hessian approximation. This finding allows us to surprisingly place SPIN optimization within the same conceptual framework as regularization and meta-CL methods, as summarized in Table 1. This derived update rule (Eq. 11) provides a new lens for understanding SPIN's underlying mechanism, revealing a deep and previously unnoted connection to a unified theoretical framework that links seemingly disparate CL approaches.

Table 1: A Unified View of CL methods as Hessian Approximation.

Method Family	Hessian Approximation
Regularization (EWC) [27]	$\approx (\sum_{i=1}^{j-1} H_i)^{-1}$
Meta-CL (La-MAML) [19]	$\approx (H_M^j)^{-1}$
SPIN	$\approx \gamma(I + \rho_t H_t)^{-1}$

4.3 Theoretical Analysis and Empirical Validation

Theoretical Mechanisms. Our analysis in Section 4.2 reveals that SPIN implicitly utilizes an effective inverse Hessian approximation of the form $\tilde{H}^{-1} \approx \gamma(I + \rho_t H_t)^{-1}$. A primary difficulty



(a) (b)

Figure 2: **Empirical validation of SPIN.** (a) Verification of the quasi-steady state assumption. (b) Stability of the update L2 norm compared to a baseline optimizer.

in OCL arises from high-variance data streams [68], where the local Hessian H_t estimated from individual samples is extremely noisy and ill-conditioned. Conventional CL methods relying on purely data-dependent Hessian approximations risk catastrophic forgetting because they over-trust unreliable low-curvature estimates. In contrast, SPIN's formulation adds a "floor" (akin to Tikhonov regularization [5, 4]). This effectively damps extreme curvature ratios while preserving relative geometry. It acts as a curvature-aware method in high-curvature directions but gracefully reverts to a stable update in uncertain, low-curvature directions.

Empirical Validation. We empirically validate these core mechanisms. (For detailed experimental setup, see the appendix.) First, our derivation relies on the quasi-steady state (QSS) assumption ($d_{t+1} \approx d_t$). Figure 2a validates this assumption, showing that $\frac{\|d_{t+1}\|}{\|d_t\|} \approx 1$ throughout training, indicating stable magnitude. Meanwhile, $\cos(d_{t+1}, d_t)$ remains close to 1, implying strong directional consistency. Notably, both properties remain stable even around task boundaries, indicating that d_t evolves smoothly despite distribution shifts, which justifies the QSS assumption. Second, we validate our theory by comparing the L2 norm of the optimization updates, as this behavior is the direct, measurable symptom of the underlying scaling factor stability our theory addresses. As Figure 2b illustrates, the magnitude of SPIN's updates (orange line) is stable and "not violently fluctuating," in stark contrast to the naive optimizer's volatile updates (blue line).

Besides, we have the following theorem to further characterize the performance of SPIN.

Theorem 1 (Regret Bound) *Assume the loss $\mathcal{L}_t$ are convex, G-Lipschitz and η-smooth over a closed convex domain $\mathcal{X} \subset \mathbb{R}^d$ with diameter D, and that the gradients are unbiased with bounded variance $\mathbb{E}[\|g_t - \nabla\mathcal{L}_t(x_t)\|^2] \leq \sigma^2$ and bounded magnitude $\|g_t\| \leq G$. Let $\theta^\star = \arg\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \mathcal{L}_t(\theta)$ and define the static regret $R_T = \sum_{t=1}^{T} \big(\mathcal{L}_t(\theta_t) - \mathcal{L}_t(\theta^\star)\big)$. Let $\alpha \in (0,1)$ and step size $\gamma^\star = \frac{D}{G\sqrt{T}}\sqrt{\frac{\alpha}{2-\alpha}}$. Then with probability at least $1-\delta$ for any $\delta \in (0,1)$, we have*

$$R_T \leq DG\Big(\sqrt{\tfrac{(2-\alpha)T}{\alpha}} + \tfrac{2}{3}\ln\tfrac{1}{\delta}\Big) + D\sigma\sqrt{2T\ln\tfrac{1}{\delta}}. \tag{13}$$

Theorem 1 shows that, under mild assumptions, our proposed optimizer achieves a nearly-optimal $\tilde{\mathcal{O}}(\sqrt{T})$ regret bound with high probability in the online optimization setting. This result theoretically substantiates the efficiency and stability of the proposed optimization framework. Detailed proof is provided in Appendix A.2.

5 Experiments

5.1 Experimental Setups

Datasets. We evaluate our method on four datasets, i.e. , CIFAR-100 (100 classes) [28], Tiny-ImageNet (200 classes) [31], CLEAR100 (100 classes) [36], and Imagenet-1K (1000 classes) [53]. More details are provided in Appendix D.1.

Baselines. To show the effectiveness of our proposed optimizer, we integrate it into 11 representative OCL methods, which span various categories, include ER [50], DER++ [6], COPE [30], SCR [40],

Table 2: Comparison of final **Average Accuracy** (Acc. %) using different optimizers on **Split CIFAR-100 (M=2k)** and **Split Tiny-ImageNet (M=5k)**. Our **SPIN** is benchmarked against 4 standard optimizers across 11 SOTA OCL methods. The best results are highlighted using **bold** font.

	Split CIFAR-100 (10 tasks, M=2k)						Split Tiny-ImageNet (20 tasks, M=5k)					
Baseline	SGD	**SPIN-SGD**	RMSprop	Adam	AdamW	**SPIN-AdamW**	SGD	**SPIN-SGD**	RMSprop	Adam	AdamW	**SPIN-AdamW**
ER (2019)	27.94	**33.31**	26.38	31.02	33.07	**35.92**	21.68	**25.24**	19.25	23.38	23.90	**25.94**
DER++ (2020)	26.10	**35.90**	27.17	30.84	31.11	**36.06**	18.95	**27.27**	23.37	26.34	26.69	**29.79**
COPE (2021)	30.81	**34.18**	31.12	32.90	33.55	**35.63**	18.31	**20.43**	18.01	19.80	20.77	**21.66**
SCR (2021)	29.82	**31.53**	28.07	30.52	31.32	**32.86**	17.26	**18.77**	17.11	18.30	18.74	**20.39**
ER-ACE (2022)	28.54	**36.14**	27.65	32.43	34.77	**37.02**	24.95	**28.11**	23.75	28.10	28.52	**30.42**
OCM (2022)	33.21	**36.06**	33.69	36.78	37.45	**40.15**	22.19	**24.91**	23.47	24.83	25.68	**27.29**
OnPro (2023)	32.44	**34.35**	29.56	33.56	34.25	**36.61**	21.38	**23.42**	20.21	21.99	22.26	**24.94**
GSA (2023)	33.25	**36.79**	30.78	35.90	36.53	**37.73**	20.80	**22.44**	20.84	22.16	25.67	**26.63**
MOSE (2024)	44.18	**47.18**	43.46	46.21	46.95	**48.33**	28.21	**29.91**	28.02	29.57	29.87	**30.63**
CCL-DC (2024)	38.80	**41.74**	42.17	42.97	43.99	**45.68**	27.01	**28.91**	27.84	28.10	28.35	**29.32**
S6MOD (2025)	34.12	**34.81**	33.08	35.19	35.55	**37.37**	22.02	**24.00**	20.94	23.43	24.01	**25.17**
Average	32.66	**36.54**	32.10	35.30	36.23	**38.49**	21.25	**23.95**	21.01	24.09	24.95	**26.56**

Table 3: Comparison of **Averaged Anytime Accuracy** (AAA %) using different optimizers on **Split CIFAR-100 (M=2k)** and **Split Tiny-ImageNet (M=5k)**. Our **SPIN** is benchmarked against 4 standard optimizers across 11 SOTA OCL methods. The best results are highlighted using **bold** font.

	Split CIFAR-100 (10 tasks, M=2k)						Split Tiny-ImageNet (20 tasks, M=5k)					
Baseline	SGD	**SPIN-SGD**	RMSprop	Adam	AdamW	**SPIN-AdamW**	SGD	**SPIN-SGD**	RMSprop	Adam	AdamW	**SPIN-AdamW**
ER (2019)	36.86	**46.18**	34.54	40.68	41.50	**48.08**	28.36	**34.38**	28.14	32.47	32.81	**37.33**
DER++ (2020)	35.16	**45.49**	33.68	37.99	38.86	**46.26**	21.20	**35.39**	30.50	36.49	36.87	**40.08**
COPE (2021)	37.76	**44.62**	34.09	39.48	42.01	**47.38**	26.49	**26.80**	24.44	28.54	27.99	**31.01**
SCR (2021)	42.55	**45.70**	37.74	41.24	42.44	**48.20**	24.03	**26.01**	23.15	26.19	26.24	**28.30**
ER-ACE (2022)	38.38	**45.91**	38.48	41.26	42.05	**46.61**	32.20	**37.25**	31.56	36.55	37.54	**40.03**
OCM (2022)	40.17	**41.83**	39.98	42.26	43.39	**47.06**	27.28	**30.73**	26.44	30.08	31.50	**34.21**
OnPro (2023)	38.09	**40.29**	37.20	41.21	41.44	**43.97**	22.44	**29.86**	24.87	29.41	28.24	**32.48**
GSA (2023)	44.91	**50.43**	39.44	44.40	45.40	**51.37**	27.45	**31.16**	27.08	31.87	33.08	**35.93**
MOSE (2024)	56.53	**58.55**	53.49	58.94	58.76	**61.79**	33.65	**36.89**	33.06	36.48	37.01	**40.74**
CCL-DC (2024)	47.80	**51.29**	49.53	55.30	55.61	**58.58**	34.21	**36.52**	33.79	35.13	35.84	**37.48**
S6MOD (2025)	42.21	**45.02**	41.35	44.17	44.67	**47.96**	28.08	**29.93**	29.24	30.21	31.00	**32.85**
Average	41.86	**46.85**	39.96	44.27	45.10	**49.75**	27.76	**32.27**	28.54	32.13	32.55	**34.59**

ER-ACE [7], OCM [17], OnPro [67], GSA [18], MOSE [71], CCL-DC [63], S6MOD [37]. To rigorously isolate and evaluate the contribution of our optimization strategy, we compare the performance of these OCL methods under our **SPIN** optimizer and that under four widely-used generic optimizers: SGD [51], Adam [26], AdamW [39] and RMSProp [51].

Implementation details. For all experiments, we employ a ResNet-18 [22] backbone, by following the common architecture in the broad OCL literature [63, 37, 67]. The streaming input batch size is set to 10 and the memory buffer batch size to 64, consistent with other methods [63, 37]. All main results use fixed hyperparameters $\alpha = \beta = 0.05$. For each OCL method, we use the same learning rate and weight decay for AdamW and SPIN-AdamW, and likewise for SGD and SPIN-SGD. All other settings follow the corresponding baseline implementations. Results are averaged over 5 runs except for ImageNet-1K, which is evaluated with a single run due to computational constraints. For evaluation, we measure the model performance using Final Average Accuracy (Acc) and Averaged Anytime Accuracy (AAA). Details are provided in Appendix D.3.

5.2 Main Results

Performance on Standard Benchmarks. As shown in Table 2 and Table 3, SPIN consistently outperforms generic optimizers across 11 OCL baselines on Split CIFAR-100 and Tiny-ImageNet. Notably, SPIN-AdamW raises MOSE's final accuracy (ACC) from 46.95% to 48.33% and GSA's anytime accuracy (AAA) from 45.40% to 51.37% on CIFAR-100. This demonstrates its strong ability to improve peak performance and effectively mitigate forgetting.

Scalability to Million-Scale Data and Robustness to Natural Distribution Shifts. On the large-scale ImageNet-1K dataset, replacing SGD with SPIN yields substantial gains (Table 4). For instance, it improves DER++'s ACC from 14.19% to 17.51% and AAA from 18.50% to 22.02%, confirming its effectiveness and scalability on complex data streams. On CLEAR100, which exhibits real world natural shifts devoid of clear task boundaries, SPIN consistently surpasses both standard SGD and

Table 4: Comparison of final Average Accuracy (Acc.) and Averaged Anytime Accuracy (AAA) on Imagenet-1K.

	Acc.(%)		AAA(%)	
Baseline	SGD	SPIN	SGD	SPIN
ER	13.21	**15.37**	19.76	**21.58**
DER++	14.19	**17.51**	18.50	**22.02**

Table 5: Comparison on CLEAR100 dataset (Acc %).

	M=200			M=5000		
Baseline	SGD	SGD-M	SPIN	SGD	SGD-M	SPIN
ER	32.63	36.01	**37.99**	41.38	51.87	**53.02**
DER++	33.54	36.87	**37.96**	41.20	50.49	**52.21**
GSA	37.76	40.62	**42.38**	43.86	46.20	**47.88**

SGD-M, i.e., SGD based on Exponential Moving Average (EMA), across varying memory sizes (Table 5), proving its stability-plasticity coupling is superior to naive parameter averaging.

5.3 Ablation Studies and Analysis

To further understand the behavior of SPIN, we conduct a series of ablation studies and analyses to answer the following key questions:

Q1: Does SPIN effectively balance stability and plasticity?

To better understand the learning dynamics, Figure 3 compares the task-wise Learning Accuracy (reflecting plasticity) and Relative Forgetting (reflecting stability) between SPIN and AdamW. SPIN consistently achieves higher learning accuracy while maintaining lower relative forgetting throughout the task stream. These results suggest that SPIN enables effective adaptation to new tasks without excessively overwriting previously acquired knowledge. This behavior is consistent with the core design of SPIN, where the interpolation between plastic and stable iterate sequences promotes a better balance between adaptation and retention during continual optimization. We further compare Relative Forgetting(RF) across 11 OCL methods on Split CIFAR-100 (Table 6). SPIN lowers average RF from 31.94% to 29.99% with SGD and from 30.83% to 28.63% with AdamW. Combined with the higher final accuracies in Table 2, these results indicate a better overall accuracy–forgetting trade-off.

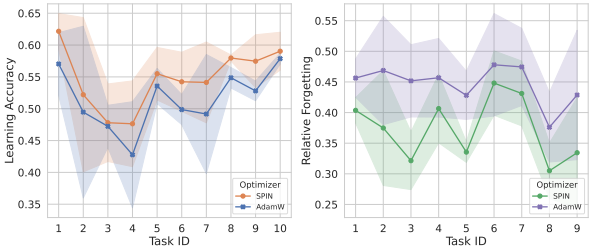


Figure 3: Task-wise Learning Accuracy and Relative Forgetting of the ER method ($M = 2000$), comparing the performance of AdamW and SPIN.

Table 6: Relative Forgetting(RF) (% ↓) on Split CIFAR-100 ($M = 2$k). Best in each optimizer pair is bolded.

Method	SGD	SPIN-SGD	AdamW	SPIN-AdamW
ER	39.14	**36.70**	36.15	**33.09**
DER++	53.30	**42.41**	45.43	**36.49**
COPE	13.64	**12.50**	12.07	**10.91**
SCR	**50.47**	51.33	**46.98**	50.19
ER-ACE	**17.24**	19.27	17.20	**16.72**
OCM	14.01	**13.78**	13.51	**12.98**
OnPro	16.40	**15.24**	21.88	**17.46**
GSA	**36.53**	39.45	35.76	**34.76**
MOSE	33.47	**31.35**	37.20	**36.11**
CCL-DC	37.56	**33.78**	37.62	**32.05**
S6MOD	39.62	**34.11**	35.35	**34.20**
Average	31.94	**29.99**	30.83	**28.63**

Table 7: Component ablation of SPIN-AdamW with GSA on Split CIFAR-100 ($M = 1$k), comparing boundary settings of α and β. ACC and AAA (%) are reported. Best results are bolded.

Method	Acc.(%) ↑	AAA(%) ↑
AdamW	27.84	39.53
SPIN(α=0)	1.04	2.93
SPIN(α=1)	27.52	37.70
SPIN(β=1)	22.30	30.63
SPIN(β=0)	31.34	40.28
SPIN	**32.20**	**42.39**

Q2: What are the key components behind SPIN's effectiveness? We first analyze several boundary cases of SPIN in Table 7. When $\alpha = 0$, the optimization collapses since the model parameters remain nearly unchanged throughout training. When $\alpha = 1$, SPIN degenerates into standard AdamW. Similarly, when $\beta = 1$, gradients are evaluated entirely on the stability sequence, resulting in overly conservative updates that hinder adaptation to new tasks. When $\beta = 0$, SPIN reduces to an EMA-based optimizer (i.e. , AdamW+EMA), similar to prior temporal ensemble approaches [58]. Although EMA improves optimization stability, SPIN consistently achieves superior performance across all

Table 8: Comparison of **Averaged Anytime Accuracy** (AAA %) and **Average Accuracy** (ACC %) using different Rehearsal-Free methods on **Split CIFAR-10** and **Split CIFAR-100**. The best and second best results are highlighted using **bold** and underline fonts.

	Split CIFAR-10		Split CIFAR-100	
Baseline	AAA	Acc	AAA	Acc
on-EWC	40.27	18.46	13.35	4.99
SI	39.84	17.49	12.32	5.20
SPIN-SGD	40.52	19.09	14.63	**5.91**
SPIN-AdamW	**41.16**	**19.18**	**15.60**	5.58

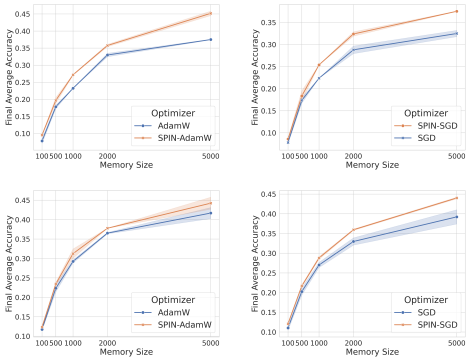


Figure 4: Impact of Replay Buffer Size

Table 9: Runtime per training step (ms/step) and peak memory (MB) of different optimizers with GSA on split CIFAR-100.

Method	**Runtime**	**Peak Memory**
SGD	1.04(ms/step)	155.86(MB)
SGD+EMA	1.08(ms/step)	240.75(MB)
AdamW	1.68(ms/step)	267.58(MB)
SPIN-SGD	**1.13**(ms/step)	**240.75**(MB)

Table 10: Accuracy under approximately matched total memory with GSA on CIFAR-100. SGD uses a 30k replay buffer to offset SPIN's additional optimizer states with a 15k buffer. Best results are bolded.

	SGD		**SPIN**
Buffer Size	**15k**	**30k**	**15k**
Acc (%)	41.61	46.93	**48.46**
AAA (%)	47.41	51.97	**55.67**

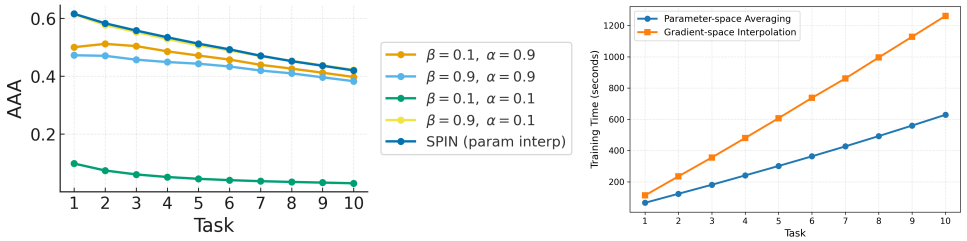
settings. Additional comparisons with SGD+EMA baselines are provided in Appendix F. These results suggest that the gains of SPIN cannot be solely attributed to temporal averaging, and instead highlight the importance of gradient-location interpolation between the plastic and stable sequences.

Q3: Is SPIN robust under different memory settings? An ideal CL method should be robust across different memory sizes in real world application. To investigate whether SPIN is effective with different memory constraints, we conducted experiments on CIFAR-100 using various memory sizes. As seen from Figure 4, SPIN consistently improves performance across all replay buffer sizes, including strict memory-constrained settings, suggesting that SPIN is effective even with different buffer sizes.

Q4: Can SPIN maintain stability without replay memory? To further investigate whether SPIN can induce intrinsic stability beyond replay mechanisms, we evaluate SPIN under the rehearsal-free setting. We compare SPIN against representative rehearsal-free continual learning methods, including On-EWC [24] and SI [74], which explicitly regularize parameter updates using importance-aware constraints. As shown in Table 8, both SPIN-SGD and SPIN-AdamW consistently outperform these rehearsal-free baselines across all metrics. These results suggest that SPIN is able to preserve past knowledge through its optimization dynamics alone, which is consistent with our theoretical analysis that SPIN implicitly induces Hessian-aware stabilization during continual optimization.

Q5: Are SPIN's gains simply due to increased optimizer memory? As shown in Table 9, SPIN-SGD introduces only modest runtime and memory overhead compared to vanilla SGD, while remaining substantially more efficient than AdamW. To rule out the possibility that SPIN's gains arise merely from additional optimizer states, we evaluate SPIN under a strict equal-memory budget. On CIFAR-100, the extra optimizer state of SPIN-SGD is equivalent to approximately 15k replay samples. We therefore compare SPIN with a 15k replay buffer against SGD with a 30k replay buffer, ensuring comparable total memory usage. As shown in Table 10, SPIN (15k) still achieves higher ACC and AAA than the memory-matched SGD (30k). These results suggest that SPIN's improvements primarily arise from its optimization dynamics and stability–plasticity balancing mechanism.

Q6: Which averaging mechanism is more suitable for OCL? We compare parameter-space interpolation with gradient-space averaging under the same optimization framework. As shown in



(a) Task-wise AAA under different averaging mechanisms. (b) Training time comparison.

Figure 5: Comparison between parameter-space interpolation and gradient-space averaging in OCL.

Figure 6a, parameter-space interpolation consistently achieves higher task-wise AAA, suggesting that evaluating gradients at a shared interpolated point yields a more coherent optimization trajectory under non-stationary data streams. In contrast, directly averaging gradients computed at different parameter locations introduces conflicting update directions. We further compare their computational efficiency in Figure 6b. Since gradient-space averaging requires two forward-backward passes, its training time is nearly twice that of parameter-space interpolation. In contrast, SPIN only requires a single gradient evaluation while achieving better continual learning performance.

6 Conclusion and Limitations

In this paper, we investigate the optimizer design for OCL and introduce a novel and dedicated optimizer SPIN, which explicitly balances the stability and plasticity tradeoff. Through its interpolation-based update mechanism, SPIN achieves smooth and robust optimization under non-stationary data streams, effectively mitigating catastrophic forgetting. Extensive experiments across diverse OCL frameworks demonstrate that SPIN consistently improves both accuracy and stability compared to widely used generic optimizers, regardless of the presence or size of replay buffers and even under strict memory constraints. These findings highlight that optimizer design plays a critical role in achieving a stable learning–forgetting balance.

Limitations: Our theoretical analysis relies on smoothness assumptions and the quasi-steady-state approximation, which may not fully characterize the optimization dynamics of highly over-parameterized large-scale neural networks. In addition, although maintaining dual iterate sequences brings substantial continual learning improvements, it also introduces additional optimizer states compared to standard optimizers. Exploring more parameter-efficient interpolation mechanisms that maintain the trade-off between stability and plasticity, thereby making them more effective on edge devices, remains an important direction for future research. Another promising direction is to integrate SPIN with continual learning methods built on large-scale pretrained models, such as CLIP and pretrained ViTs, and investigate its role in balancing adaptation and retention during prompt tuning and low-rank adaptation.

Acknowledgments and Disclosure of Funding

Acknowledgments. We thank the anonymous reviewers and the area chair for their constructive feedback, which helped improve this work.

Funding. The authors received no external funding or third-party support for this work.

Competing interests. The authors declare no competing interests.

References

[1] Kwangjun Ahn, Gagik Magakyan, and Ashok Cutkosky. General framework for online-to-nonconvex conversion: Schedule-free SGD is also effective for nonconvex optimization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 772–795. PMLR, 2025. URL `https://proceedings.mlr.press/v267/ahn25a.html`.

[2] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uxxFrDwrE7Y.

[3] Michal Arbel and Julien Mairal. Amortized implicit differentiation for stochastic bilevel optimization. *ArXiv*, abs/2111.14580, 2021. URL https://api.semanticscholar.org/CorpusID:244714983.

[4] Hedy Attouch, Aïcha Balhag, Zaki Chbani, and Hassan Riahi. Accelerated gradient methods combining tikhonov regularization with geometric damping driven by the hessian. *Applied Mathematics & Optimization*, 88(2):29, 2023.

[5] Radu Ioan Boţ, Ernö Robert Csetnek, and Szilárd Csaba László. Tikhonov regularization of a second order dynamical system with hessian driven damping. *Mathematical Programming*, 189 (1):151–186, 2021.

[6] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *ArXiv*, abs/2004.07211, 2020. URL https://api.semanticscholar.org/CorpusID:215768806.

[7] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. *ArXiv*, abs/2203.03798, 2021. URL https://api.semanticscholar.org/CorpusID:247315339.

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *ArXiv*, abs/1812.00420, 2018. URL https://api.semanticscholar.org/CorpusID:54443381.

[9] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7): 3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446.

[10] Aaron Defazio. Momentum via primal averaging: Theoretical insights and learning rate schedules for non-convex optimization, 2020. URL https://api.semanticscholar.org/CorpusID:235266276.

[11] Aaron Defazio and Robert Mansel Gower. The power of factorial powers: New parameter settings for (stochastic) optimization. In *Asian Conference on Machine Learning*, 2020. URL https://api.semanticscholar.org/CorpusID:224815911.

[12] Aaron Defazio, Xingyu Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled, 2024.

[13] Aaron Defazio, Konstantin Mishchenko, Parameswaran Raman, Hao-Jun Michael Shi, and Lin Xiao. Smoothing diloco with primal averaging for faster training of llms. *arXiv preprint arXiv:2512.17131*, 2025.

[14] Junze Deng, Qinhang Wu, Peizhong Ju, Sen Lin, Yingbin Liang, and Ness B. Shroff. Unlocking the power of rehearsal in continual learning: A theoretical perspective. *ArXiv*, abs/2506.00205, 2025. URL https://api.semanticscholar.org/CorpusID:279074801.

[15] David A Freedman. On tail probabilities for martingales. *the Annals of Probability*, pages 100–118, 1975.

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

[17] Yiduo Guo, Bing Liu, and Dongyan Zhao. Online continual learning through mutual information maximization. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8109–8126, 17–23 Jul 2022.

[18] Yiduo Guo, Bing Liu, and Dongyan Zhao. Dealing with cross-task class discrimination in online continual learning. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11878–11887, 2023. URL https://api.semanticscholar.org/CorpusID:258865951.

[19] Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-maml: Look-ahead meta learning for continual learning. *arXiv preprint arXiv:2007.13904*, 2020.

[20] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

[21] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13926–13935, 2020.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.

[24] Ferenc Huszár. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 115:E2496 – E2497, 2017. URL https://api.semanticscholar.org/CorpusID:3415694.

[25] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1803.05407.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

[27] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114: 3521 – 3526, 2016. URL https://api.semanticscholar.org/CorpusID:4704285.

[28] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

[29] Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1):365–397, 2012.

[30] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8230–8239, 2020. URL https://api.semanticscholar.org/CorpusID:221447946.

[31] Yann Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[32] Hongbo Li, Sen Lin, Lingjie Duan, Yingbin Liang, and Ness B Shroff. Theory on mixture-of-experts in continual learning. *arXiv preprint arXiv:2406.16437*, 2024.

[33] Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647, 2024.

[34] Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Trgp: Trust region gradient projection for continual learning. In *The Tenth International Conference on Learning Representations*, 2022.

[35] Sen Lin, Peizhong Ju, Yingbin Liang, and Ness Shroff. Theory on forgetting and generalization of continual learning. In *International Conference on Machine Learning*, pages 21078–21100. PMLR, 2023.

[36] Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[37] Sihao Liu, Yibo Yang, Xiaojie Li, David A Clifton, and Bernard Ghanem. Enhancing online continual learning with plug-and-play state space model and class-conditional mixture of discretization. *arXiv preprint arXiv:2412.18177*, 2024.

[38] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:37308416.

[39] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017. URL https://api.semanticscholar.org/CorpusID:3312944.

[40] Zheda Mai, Ruiwen Li, Hyunwoo J. Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3584–3594, 2021. URL https://api.semanticscholar.org/CorpusID:232352581.

[41] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.10.021. URL https://www.sciencedirect.com/science/article/pii/S0925231221014995.

[42] Imad Eddine Marouf, Subhankar Roy, Enzo Tartaglione, and Stéphane Lathuilière. Weighted ensemble models are strong continual learners. In *European Conference on Computer Vision (ECCV)*, 2024. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/9002_ECCV_2024_paper.php.

[43] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

[44] Yu Nesterov and Vladimir Shikhman. Quasi-monotone subgradient methods for nonsmooth convex minimization. *Journal of Optimization Theory and Applications*, 165(3):917–940, 2015.

[45] Yurii Nesterov. A method for solving the convex programming problem with convergence rate o (1/k2). In *Dokl akad nauk Sssr*, volume 269, page 543, 1983.

[46] Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

[47] Oleksiy Ostapenko, Mihai Marian Puscas, Tassilo Klein, Patrick Jähnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11313–11321, 2019. URL https://api.semanticscholar.org/CorpusID:102353035.

[48] Boris T Polyak. New stochastic approximation type procedures. *Automat. i Telemekh*, 7(98-107): 2, 1990.

[49] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *In International Conference on Learning Representations (ICLR)*, 2019.

[50] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Neural Information Processing Systems*, 2018. URL https://api.semanticscholar.org/CorpusID:53860287.

[51] Sebastian Ruder. An overview of gradient descent optimization algorithms. *ArXiv*, abs/1609.04747, 2016. URL https://api.semanticscholar.org/CorpusID:17485266.

[52] David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. 1988. URL https://api.semanticscholar.org/CorpusID:108279905.

[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL http://arxiv.org/abs/1409.0575.

[54] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021.

[55] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2025.

[56] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *ArXiv*, abs/1212.1824, 2012. URL https://api.semanticscholar.org/CorpusID:1217704.

[57] Albin Soutif-Cormerais, Antonio Carta, Andrea Cossu, Julio Hurtado, Vincenzo Lomonaco, Joost Van de Weijer, and Hamed Hemati. A comprehensive empirical evaluation on online continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3518–3528, 2023.

[58] Albin Soutif-Cormerais, Antonio Carta, and Joost Van de Weijer. Improving online continual learning performance and stability with temporal ensembles. In *Conference on Lifelong Learning Agents*, pages 828–845. PMLR, 2023.

[59] Gido M. van de Ven and Andreas Savas Tolias. Three scenarios for continual learning. *ArXiv*, abs/1904.07734, 2019. URL https://api.semanticscholar.org/CorpusID:119309522.

[60] Liyuan Wang, Mingtian Zhang, Zhongfan Jia, Qian Li, Chenglong Bao, Kaisheng Ma, Jun Zhu, and Yi Zhong. Afec: Active forgetting of negative transfer in continual learning. *arXiv preprint arXiv:2110.12187*, 2021.

[61] Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. *Advances in Neural Information Processing Systems*, 2023.

[62] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE TPAMI*, 46(8):5362–5383, 2024. doi: 10.1109/TPAMI.2024.3367329.

[63] Maorong Wang, Nicolas Michel, Ling Xiao, and Toshihiko Yamasaki. Improving plasticity in online continual learning via collaborative learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23460–23469, June 2024.

[64] Quanziang Wang, Renzhen Wang, Yichen Wu, Xixi Jia, and Deyu Meng. Cba: Improving online continual learning via continual bias adaptor. In *ICCV*, 2023.

[65] Quanziang Wang, Renzhen Wang, Yuexiang Li, Dong Wei, Hong Wang, Kai Ma, Yefeng Zheng, and Deyu Meng. Relational experience replay: Continual learning by adaptively tuning task-wise relationship. *IEEE TMM*, 26:9683–9698, 2024. doi: 10.1109/TMM.2024.3397048.

[66] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *CVPR*, pages 184–193, 2021.

[67] Yujie Wei, Jiaxin Ye, Zhizhong Huang, Junping Zhang, and Hongming Shan. Online prototype learning for online continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18764–18774, 2023.

[68] Yichen Wu, Long-Kai Huang, Renzhen Wang, Deyu Meng, and Ying Wei. Meta continual learning revisited: Implicitly enhancing online hessian approximation via variance reduction. In *ICLR*, 2024.

[69] Yichen Wu, Hong Wang, Peilin Zhao, Yefeng Zheng, Ying Wei, and Long-Kai Huang. Mitigating catastrophic forgetting in online continual learning by modeling previous task interrelations via pareto optimization. In *ICML*, 2024.

[70] Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning. *arXiv preprint arXiv:2501.13198*, 2025.

[71] Hongwei Yan, Liyuan Wang, Kaisheng Ma, and Yi Zhong. Orchestrate latent expertise: Advancing online continual learning with multi-level supervision and reverse self-distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23670–23680, 2024.

[72] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3013–3022, 2021. URL `https://api.semanticscholar.org/CorpusID:232427923`.

[73] Jason Yoo, Yunpeng Liu, Frank Wood, and Geoff Pleiss. Layerwise proximal replay: A proximal point method for online continual learning. *arXiv preprint arXiv:2402.09542*, 2024.

[74] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. Pmlr, 2017.

[75] Michael R Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. *arXiv preprint arXiv:1907.08610*, 2019.

[76] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE TPAMI*, 46(12):9851–9873, 2024. doi: 10.1109/TPAMI.2024.3429383.

[77] Minghao Zhou, Quanziang Wang, Jun Shu, Qian Zhao, and Deyu Meng. Diagnosing batch normalization in class incremental learning. *arXiv preprint arXiv:2202.08025*, 2022.

Appendix

Appendix contents

A Proofs **17**

A.1 Proof of Proposition 1 (Implicit Hessian) 17

A.2 Proof of Theorem 1 (Regret Bound) 18

B An Intuitive Understanding of Two Averaging **19**

C pipeline **19**

D Implementation Details **20**

D.1 Dataset Details 20

D.2 Evaluation Metrics 20

D.3 Training Details 21

E Performance on Different Architecture **22**

F Analysis of Hyperparameter Robustness And Comparisons with SGD+EMA Baselines. **23**

G Relation to Multi-Sequence Optimizers **23**

H Detailed Experimental Setup for Sec. 4.3 **24**

H.1 Setup for verifying the quasi-steady-state assumption 24

H.2 Setup for comparing update norms with Newton's method 25

I Algorithm Pseudocode **26**

A Proofs

A.1 Proof of Proposition 1 (Implicit Hessian)

We analyze the local dynamics of the SPIN algorithm to verify the effective update rule presented in the main text (Proposition 1). Let $d_t := \phi_t - \theta_t$ denote the displacement between the fast (plastic) and slow (stable) iterates. Using the interpolation $x_t = \theta_t + (1-\beta)d_t$, we rewrite the system dynamics in terms of the variables (θ_t, d_t):

$$\begin{aligned} \theta_{t+1} - \theta_t &= \alpha(d_t - \gamma g_t), \\ d_{t+1} &= (1-\alpha)(d_t - \gamma g_t), \end{aligned} \tag{14}$$

where $g_t = \nabla \mathcal{L}_t(x_t)$.

Local Linearization. To expose the curvature dependence, we perform a first-order Taylor expansion of the gradient around the slow iterate θ_t. Let $H_t = \nabla^2 \mathcal{L}_t(\theta_t)$ denote the local Hessian. Neglecting higher-order terms, we approximate the gradient at the interpolated point x_t as:

$$g_t = \nabla \mathcal{L}_t(\theta_t + (1-\beta)d_t) \approx \nabla \mathcal{L}_t(\theta_t) + (1-\beta)H_t d_t. \tag{15}$$

Substituting this linear approximation into the evolution equation for d_t in Eq. 14 yields:

$$d_{t+1} \approx (1-\alpha)\left[d_t - \gamma\left(\nabla \mathcal{L}_t(\theta_t) + (1-\beta)H_t d_t\right)\right]. \tag{16}$$

Quasi-Steady State Analysis. Since SPIN is designed as a two-time-scale system where the fast variate ϕ_t (and consequently d_t) evolves more rapidly than θ_t, we apply a quasi-steady-state approximation [3]. We solve for the equilibrium displacement d_t^* by imposing the stationarity condition $d_{t+1} \approx d_t$:

$$d_t^* = (1-\alpha)d_t^* - (1-\alpha)\gamma \nabla \mathcal{L}_t(\theta_t) - (1-\alpha)\gamma(1-\beta)H_t d_t^*. \tag{17}$$

Rearranging terms to isolate d_t^*, we obtain:

$$\left[\alpha I + (1-\alpha)\gamma(1-\beta)H_t\right] d_t^* = -(1-\alpha)\gamma \nabla \mathcal{L}_t(\theta_t). \tag{18}$$

Solving for d_t^* gives the explicit form:

$$d_t^* = -(1-\alpha)\gamma\left[\alpha I + (1-\alpha)\gamma(1-\beta)H_t\right]^{-1} \nabla \mathcal{L}_t(\theta_t). \tag{19}$$

Effective Update Rule. We now substitute the equilibrium displacement d_t^* back into the update rule for the slow iterate θ_t in Eq. 14. Note that the update term is $\alpha(d_t - \gamma g_t)$. From the stationarity condition $d_t \approx (1-\alpha)(d_t - \gamma g_t)$, we have the identity $d_t - \gamma g_t \approx \frac{1}{1-\alpha} d_t$. Therefore, the update becomes:

$$\theta_{t+1} - \theta_t \approx \frac{\alpha}{1-\alpha} d_t^* = -\alpha\gamma\left[\alpha I + (1-\alpha)\gamma(1-\beta)H_t\right]^{-1} \nabla \mathcal{L}_t(\theta_t). \tag{20}$$

To align with the form presented in the main text, we factorize α from the inverse term. We define the effective regularization coefficient ρ_t as:

$$\rho_t = \frac{\gamma(1-\alpha)(1-\beta)}{\alpha}. \tag{21}$$

Using this definition, we can rewrite the inverse term as:

$$[\alpha I + (1-\alpha)\gamma(1-\beta)H_t]^{-1} = \frac{1}{\alpha}\left[I + \frac{\gamma(1-\alpha)(1-\beta)}{\alpha} H_t\right]^{-1} = \frac{1}{\alpha}[I + \rho_t H_t]^{-1}. \tag{22}$$

Substituting this back into the update equation, the α terms cancel out, yielding the final effective update:

$$\theta_{t+1} \approx \theta_t - \gamma\left[I + \rho_t H_t\right]^{-1} \nabla \mathcal{L}_t(\theta_t). \tag{23}$$

Connection to Preconditioning. For sufficiently small step sizes, applying the Neumann series approximation $(I + A)^{-1} \approx I - A$ yields:

$$\theta_{t+1} \approx \theta_t - \gamma\left(I - \rho_t H_t\right) \nabla \mathcal{L}_t(\theta_t). \tag{24}$$

This confirms that SPIN performs a gradient descent step preconditioned by the implicit inverse Hessian approximation $\tilde{H}^{-1} \approx \gamma(I + \rho_t H_t)^{-1}$.

A.2 Proof of Theorem 1 (Regret Bound)

Theorem 1. *Let $\mathcal{X} \subset \mathbb{R}^d$ be a closed convex set with diameter D. Under Assumptions (i)–(iv), if we choose $\gamma = \frac{D}{G\sqrt{T}}\sqrt{\frac{\alpha}{2-\alpha}}$ and initialize $\theta_1 = \phi_1$, the regret R_T satisfies the following bound with probability at least $1-\delta$:*

$$R_T \leq \tilde{\mathcal{O}}(\sqrt{T}). \tag{25}$$

We consider online convex optimization [55, 20] with the update rule defined as:

$$x_t = (1-\beta)\phi_t + \beta\theta_t, \tag{26}$$

$$\phi_{t+1} = \Pi_{\mathcal{X}}\big(\phi_t - \gamma g_t\big), \qquad g_t := \nabla\mathcal{L}_t(x_t), \tag{27}$$

$$\theta_{t+1} = \theta_t + \alpha(\phi_{t+1} - \theta_t), \tag{28}$$

where $\alpha \in (0,1)$ is a constant. We let $\mathcal{F}_t$ denote the filtration generated by the random variables up to round t.

For the analysis, we adopt the following standard assumptions:

(i) **Domain:** $\mathcal{X}$ is a closed convex set with diameter bounded by D, i.e., $\sup_{x,y\in\mathcal{X}} \|x-y\| \leq D$.

(ii) **Loss Function:** Each loss function $\mathcal{L}_t$ is convex and G-Lipschitz continuous on $\mathcal{X}$.

(iii) **Stochastic Gradient:** The stochastic gradients g_t are unbiased, $\mathbb{E}[g_t \mid \mathcal{F}_t] = \nabla\mathcal{L}_t(x_t)$, and bounded such that $\|g_t\| \leq G$ almost surely.

(iv) **Variance:** The conditional variance of the stochastic gradients is bounded by σ^2, i.e., $\mathbb{E}[\|g_t - \nabla\mathcal{L}_t(x_t)\|^2 \mid \mathcal{F}_t] \leq \sigma^2$.

We aim to bound the static regret $R_T := \sum_{t=1}^{T}(\mathcal{L}_t(\theta_t) - \mathcal{L}_t(\theta^\star))$.

1. Regret Decomposition. By the convexity of $\mathcal{L}_t$, we have $\mathcal{L}_t(x_t) - \mathcal{L}_t(\theta^\star) \leq \langle\nabla\mathcal{L}_t(x_t), x_t - \theta^\star\rangle$ [20]. We decompose the instantaneous regret as follows:

$$\begin{aligned}\mathcal{L}_t(\theta_t) - \mathcal{L}_t(\theta^\star) &= \big(\mathcal{L}_t(\theta_t) - \mathcal{L}_t(x_t)\big) + \big(\mathcal{L}_t(x_t) - \mathcal{L}_t(\theta^\star)\big) \\ &\leq G\|\theta_t - x_t\| + \langle\nabla\mathcal{L}_t(x_t), x_t - \theta^\star\rangle.\end{aligned} \tag{29}$$

Using the definition of x_t in Eq. 26, we have $\|\theta_t - x_t\| = (1-\beta)\|\theta_t - \phi_t\|$. Next, we introduce the martingale difference sequence $\xi_t := \langle\nabla\mathcal{L}_t(x_t) - g_t, x_t - \theta^\star\rangle$. We rewrite the inner product term using the stochastic gradient g_t:

$$\langle\nabla\mathcal{L}_t(x_t), x_t - \theta^\star\rangle = \langle g_t, x_t - \theta^\star\rangle + \xi_t = \langle g_t, \phi_t - \theta^\star\rangle + \langle g_t, x_t - \phi_t\rangle + \xi_t.$$

Noting that $x_t - \phi_t = \beta(\theta_t - \phi_t)$ and $\|g_t\| \leq G$ (Assumption iii), we bound $\langle g_t, x_t - \phi_t\rangle \leq G\beta\|\theta_t - \phi_t\|$. Combining these terms yields:

$$\mathcal{L}_t(\theta_t) - \mathcal{L}_t(\theta^\star) \leq \langle g_t, \phi_t - \theta^\star\rangle + G\|\theta_t - \phi_t\| + \xi_t. \tag{30}$$

2. Bounding the Gradient Descent Term. The update for ϕ_t corresponds to Online Gradient Descent (OGD) [20]. By the standard analysis of OGD and the non-expansiveness of the projection operator $\Pi_{\mathcal{X}}$, we have:

$$\langle g_t, \phi_t - \theta^\star\rangle \leq \frac{\|\phi_t - \theta^\star\|^2 - \|\phi_{t+1} - \theta^\star\|^2}{2\gamma} + \frac{\gamma}{2}\|g_t\|^2. \tag{31}$$

3. Bounding the Drift Term. Let $\Delta_t = \theta_t - \phi_t$. From the update rule $\theta_{t+1} = (1-\alpha)\theta_t + \alpha\phi_{t+1}$, the recurrence for the difference is:

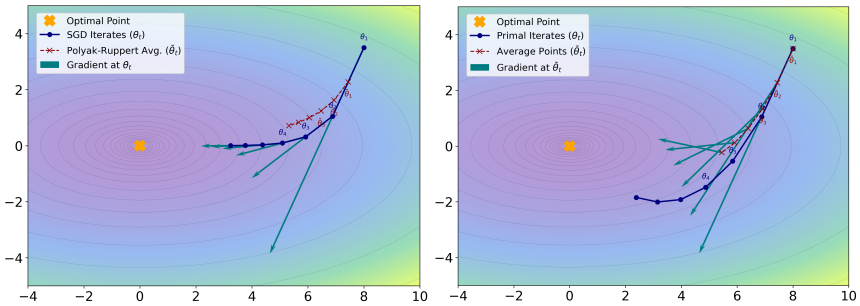
$$\Delta_{t+1} = (1-\alpha)\theta_t + \alpha\phi_{t+1} - \phi_{t+1} = (1-\alpha)(\theta_t - \phi_{t+1}). \tag{32}$$

Taking norms and applying the triangle inequality with $\|\phi_t - \phi_{t+1}\| \leq \gamma G$:

$$\|\Delta_{t+1}\| \leq (1-\alpha)\|\Delta_t\| + (1-\alpha)\gamma G. \tag{33}$$

Summing this geometric series over $t = 1, \ldots, T$ yields:

$$\sum_{t=1}^{T}\|\theta_t - \phi_t\| \leq \frac{\|\theta_1 - \phi_1\|}{\alpha} + T\frac{1-\alpha}{\alpha}\gamma G. \tag{34}$$



(a) Polyak-Ruppert averaging. (b) Primal averaging.

Figure 6: Trajectories of Polyak-Ruppert Averaging and Primal Averaging. **Polyak-Ruppert Averaging** computes the gradient at the current, fast-moving model iterate θ_t: $g_t = \nabla\mathcal{L}(\theta_t)$. The final model is the average of this noisy sequence: $\bar{\theta}_T = \frac{1}{T}\sum_{t=1}^{T}\theta_t$. **Primal Averaging** computes the gradient at the averaged iterate over the current trajectory: $g_t = \nabla\mathcal{L}(\bar{\theta}_t)$, where $\bar{\theta}_t = \frac{1}{t}\sum_{i=1}^{t}\theta_i$. This stable gradient is then used to guide the update.

4. Concentration Analysis. Since $\|x_t - \theta^\star\| \le D$ (Assumption i) and gradients are bounded by G, the martingale difference is bounded as $|\xi_t| \le 2GD$. The conditional variance is bounded by $\mathbb{E}[\xi_t^2 \mid \mathcal{F}_t] \le D^2\sigma^2$ (Assumption iv). Applying Freedman's inequality [15], with probability at least $1-\delta$:

$$\sum_{t=1}^{T}\xi_t \le D\sigma\sqrt{2T\ln\tfrac{1}{\delta}} + \frac{2GD}{3}\ln\tfrac{1}{\delta}. \tag{35}$$

5. Final Bound. Summing Eq. 30 over $t = 1$ to T and substituting the bounds derived above:

$$R_T \le \frac{G\|\theta_1 - \phi_1\|}{\alpha} + \left[\frac{D^2}{2\gamma} + \frac{\gamma TG^2}{2} + \gamma TG^2\frac{1-\alpha}{\alpha}\right] + \sum_{t=1}^{T}\xi_t. \tag{36}$$

Substituting the optimal step size $\gamma^\star = \frac{D}{G\sqrt{T}}\sqrt{\frac{\alpha}{2-\alpha}}$, the term in brackets simplifies to $DG\sqrt{T}\sqrt{\frac{2-\alpha}{\alpha}}$. Thus, with probability $\ge 1-\delta$:

$$R_T \le \frac{G\|\theta_1 - \phi_1\|}{\alpha} + DG\sqrt{T}\sqrt{\frac{2-\alpha}{\alpha}} + D\sigma\sqrt{2T\ln\tfrac{1}{\delta}} + \frac{2GD}{3}\ln\tfrac{1}{\delta}. \tag{37}$$

Assuming the initialization $\theta_1 = \phi_1$, the first term vanishes, and we obtain the final asymptotic bound:

$$R_T \le \tilde{\mathcal{O}}(\sqrt{T}). \tag{38}$$

Conclusion. Theorem 1 establishes that the proposed algorithm achieves sublinear regret matching the optimal $\tilde{\mathcal{O}}(\sqrt{T})$ lower bound for stochastic OCO. The constant $\sqrt{(2-\alpha)/\alpha}$ highlights the trade-off between the memory retention of the slow weights (parameterized by α) and the convergence speed.

B An Intuitive Understanding of Two Averaging

We present these two averaging schemes, as shown in Figure 6.

C pipeline

We present the pipeline of SPIN in Figure 7.

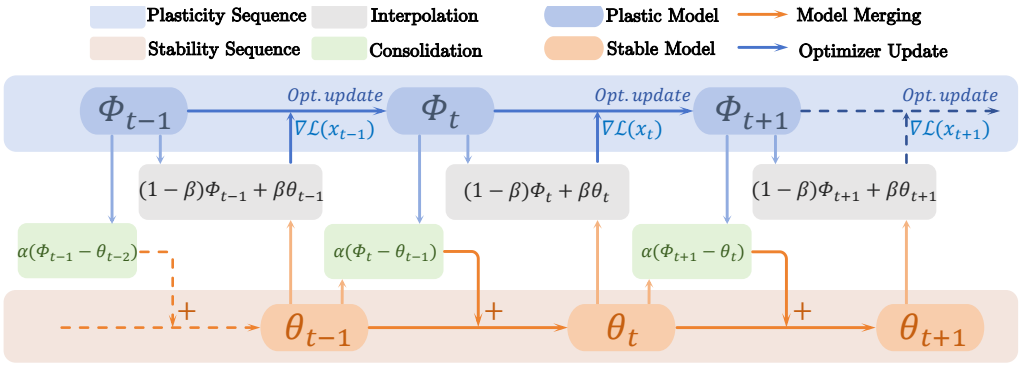


Figure 7: The pipeline of SPIN. The gradient $\nabla\mathcal{L}$ at the model interpolation is obtained from the upstream design of OCL algorithms.

D Implementation Details

D.1 Dataset Details

In this work, we experiment with five benchmark datasets in online continual learning setting: CIFAR10 [28], CIFAR100 [28], Tiny-Imagenet [31], CLEAR100 [36], and Imagenet1K [53], which are detailed as follows:

Seq-CIFAR10 [28]. Seq-CIFAR10 is a widely used benchmark in Online Continual Learning (OCL), constructed by splitting the CIFAR-10 dataset into sequential tasks. It contains 10 classes and 60,000 RGB images of size 32×32, and is commonly used to evaluate catastrophic forgetting under class-incremental learning settings.

Seq-CIFAR100 [28]. Seq-CIFAR100 is a continual learning benchmark derived from CIFAR-100, consisting of 100 fine-grained categories and 60,000 images. Due to its larger number of classes and more complex task splits, it provides a more challenging setting for evaluating the stability and generalization ability of OCL methods.

Tiny-ImageNet [31]. Tiny-ImageNet is a reduced version of ImageNet containing 200 classes with 64×64 resolution images. Its higher visual diversity and semantic complexity make it a popular benchmark for testing representation learning and forgetting resistance in online continual learning scenarios.

CLEAR100 [36]. CLEAR100 is specifically designed for online continual learning and emphasizes realistic streaming data settings. It includes distribution shifts, sequential data streams, and limited replay constraints, making it suitable for evaluating adaptability and robustness in dynamic learning environments.

ImageNet-1K [53]. ImageNet-1K is one of the most influential large-scale image classification datasets, containing 1,000 categories and over one million high-resolution images. In OCL research, it is commonly used to build large-scale class-incremental or streaming benchmarks for evaluating long-term memory retention and continual adaptation.

D.2 Evaluation Metrics

We adopt several metrics commonly used in online continual learning. Let $a_{i,j}$ denote the accuracy of the model after learning task i on the test data of task j.

Average Accuracy (Acc). Acc measures the final performance after completing all tasks:

$$\text{Acc} = \frac{1}{T}\sum_{j=1}^{T} a_{T,j}. \tag{39}$$

Averaged Anytime Accuracy (AAA). AAA evaluates the performance throughout the learning process:

$$\mathrm{AAA} = \frac{1}{T}\sum_{i=1}^{T}\left(\frac{1}{i}\sum_{j=1}^{i} a_{i,j}\right). \tag{40}$$

Learning Accuracy (LA). LA quantifies plasticity as the accuracy on each task immediately after it is learned:

$$l_j = a_{j,j}, \tag{41}$$

and the overall learning accuracy is

$$\mathrm{LA} = \frac{1}{T}\sum_{j=1}^{T} l_j. \tag{42}$$

Average Forgetting (AF). AF measures the degradation on old tasks:

$$\mathrm{AF} = \frac{1}{T-1}\sum_{j=1}^{T-1}\left(\max_{k\le T} a_{k,j} - a_{T,j}\right). \tag{43}$$

Relative Forgetting (RF). Relative Forgetting measures the proportion of performance degradation on previous tasks. For the j-th task after learning task k, it is defined as

$$f_j^k = \max_{i\in\{1,\ldots,k\}}\left(1 - \frac{a_j^k}{a_j^i}\right), \quad j \le k. \tag{44}$$

The overall relative forgetting averaged across all tasks is

$$\mathrm{RF} = \frac{1}{T}\sum_{j=1}^{T} f_j^T. \tag{45}$$

D.3 Training Details

For all baselines except MOSE [71] and SCR [40], we follow the standard experimental configuration used in CCL-DC [63]. All methods adopt a randomly initialized ResNet-18 backbone without pretraining. The model operates in a strict one-pass online continual learning setting, where each training sample is observed exactly once. The streaming batch size is fixed to 10, and at every iteration a mini-batch of 64 samples is drawn from the replay buffer. The memory buffer is maintained using reservoir sampling, and identical data augmentation is applied across all baselines to ensure fairness.

MOSE and SCR follow the experimental setup of MOSE. Both methods use a nearest-class-mean (NCM) classifier for final prediction. The same online protocol is applied, including a streaming batch size of 10, a replay batch size of 64, and reservoir-based memory management. All experiments were conducted using one NVIDIA RTX 4090 GPU.

Learning rates for the 11 OCL methods are summarized in Table 11. Note that in most of our experiments, SPIN can replace the existing optimizer without any additional adjustments.

Data Augmentation. Data augmentation has been shown to effectively improve the performance of online continual learning (OCL) methods [6, 63]. Since different methods may respond differently to augmentation strength, we consider two commonly used augmentation strategies to ensure a fair and comprehensive comparison.

1. **Partial Augmentation Strategy.** This strategy adopts relatively weak data augmentation, including random cropping with probability $p = 0.5$ and random horizontal flipping with probability $p = 0.5$.
2. **Full Augmentation Strategy.** The full augmentation strategy extends the partial augmentation setting by additionally incorporating color jitter and random grayscale transformations. Specifically, the color jitter parameters are set to $(0.4, 0.4, 0.4, 0.1)$ with probability $p = 0.8$, while random grayscale is applied with probability $p = 0.2$.

Table 11: Learning rates for each OCL method and optimizer.

Method	SGD	SPIN-SGD	AdamW	SPIN-AdamW
ER	0.01	0.01	0.0005	0.0005
DER++	0.03	0.03	0.0005	0.0005
COPE	0.03	0.03	0.0005	0.0005
SCR	0.05	0.05	0.005	0.005
ER-ACE	0.01	0.01	0.0005	0.0005
OCM	0.05	0.05	0.001	0.001
OnPro	0.03	0.03	0.0005	0.0005
GSA	0.03	0.03	0.0005	0.0005
MOSE	0.05	0.05	0.001	0.001
CCL-DC	0.03	0.03	0.0005	0.0005
S6MOD	0.03	0.03	0.0005	0.0005

Table 12: Hyperparameter for part of baseline methods.

Method	HP	Values
ER	optimizer	[SGD, AdamW, SPIN, RMSProp, Adam]
	lr	[0.1, 0.3, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]
	weight decay	[1e-4]
	momentum (for SGD)	[0.9]
	aug. strat.	[partial, full]
DER++	optimizer	[SGD, AdamW, SPIN, RMSProp, Adam]
	lr	[0.1, 0.3, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]
	weight decay	[0, 1e-4]
	momentum (for SGD)	[0, 0.9]
	aug. strat.	[partial, full]
	alpha	[0.2]
	beta	[1.0]
GSA	optimizer	[SGD, AdamW, SPIN, RMSProp, Adam]
	lr	[0.1, 0.3, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]
	weight decay	[1e-4]
	momentum (for SGD)	[0.9]
	aug. strat.	[partial, full]

For fair comparison, the models trained with SPIN also adopt the same augmentation strategy in the baseline loss branch. Note that the overall augmentation setting is kept consistent with CCL-DC [63]. We present hyperparameter searches for some of the methods in Table 12.

E Performance on Different Architecture

To further evaluate the robustness of our method across different model architectures, we additionally conduct experiments on a Transformer-based backbone. Specifically, we compare the optimization performance of AdamW and our SPIN under the Vision Transformer (ViT) architecture using the GSA framework. In these experiments, the memory buffer size is set to 2000 and the learning rate is fixed at 0.0001. The results demonstrate that our method consistently improves performance across different optimization landscapes and network structures, highlighting its generalization capability in online continual learning settings. Note that ResNet-18 is the standard backbone in OCL, while ViT remains largely unexplored.

Table 13: Optimization performance on the ViT architecture.

Metric	AdamW	SPIN
Acc (%)	25.89	**26.72**
AAA (%)	33.65	**34.98**

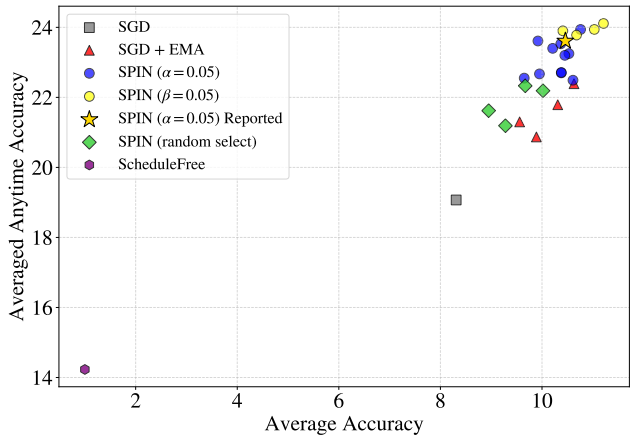


Figure 8: Analysis of hyperparameter robustness and comparisons with different baselines.

F Analysis of Hyperparameter Robustness And Comparisons with SGD+EMA Baselines.

We evaluate SPIN's sensitivity to α and β in Figure 8. Even randomly selected (α, β) pairs consistently outperform standard SGD. Specifically, fixing one parameter to an empirical default (e.g. , $\alpha = 0.05$ or $\beta = 0.05$) while tuning the other within $[0.01, 0.99]$ yields peak performance, allowing SPIN to consistently surpass the tuned EMA baseline. This superiority arises because EMA effectively operates as a subset of SPIN. Finally, ScheduleFree performs poorly in this context, as its equal-weighted scheduling implicitly assumes a stationary distribution, which mismatches the strictly non-stationary nature of OCL.

G Relation to Multi-Sequence Optimizers

We discuss how multi-sequence optimizers choose gradient evaluation points and average iterates. Throughout this section, z_t denotes the base iterate, x_t the averaged iterate, and y_t the gradient evaluation point. The formulas below use gradient-descent updates without weight decay to isolate the averaging mechanisms.

Lookahead. Lookahead [75] performs k base-optimizer steps on fast weights, interpolates the slow weights toward the resulting fast weights, and resets the fast weights to the updated slow weights. With SGD as the base optimizer and $z_{t,0} = x_t$, these updates are

$$z_{t,i+1} = z_{t,i} - \gamma_{t,i} g_{t,i}(z_{t,i}), \quad i = 0, \dots, k-1, \tag{46}$$

$$x_{t+1} = (1-\lambda)x_t + \lambda z_{t,k}, \tag{47}$$

$$z_{t+1,0} = x_{t+1}. \tag{48}$$

Here, t indexes synchronization rounds, i indexes inner steps, $g_{t,i}$ is the stochastic gradient at that step, and $\lambda \in (0, 1]$ is the slow-weight interpolation coefficient. Its averaging is periodic; the methods below instead couple their sequences at every step.

AC-SA form. For an L-smooth convex objective f, the accelerated formulation discussed in the main text can be written as [45, 29, 12]

$$y_t = (1 - c_{t+1})x_t + c_{t+1}z_t, \tag{49}$$

$$z_{t+1} = z_t - \frac{t+1}{2L}\nabla f(y_t), \tag{50}$$

$$x_{t+1} = (1 - c_{t+1})x_t + c_{t+1}z_{t+1}, \tag{51}$$

where $c_{t+1} = 2/(t+2)$. The same time-dependent coefficient controls both interpolations, together with a growing base step size. This coupling is motivated by accelerated convergence for a fixed objective.

Schedule-Free optimization. The basic uniform-averaging version of Schedule-Free SGD takes the form [12]

$$y_t = (1 - \beta)z_t + \beta x_t, \tag{52}$$

$$z_{t+1} = z_t - \gamma_t g_t(y_t), \tag{53}$$

$$x_{t+1} = \frac{t}{t+1}x_t + \frac{1}{t+1}z_{t+1}, \tag{54}$$

where $g_t(y_t)$ is a stochastic gradient. Unlike the AC-SA form above, the gradient interpolation coefficient is fixed while the averaging coefficient decreases. The framework also supports weighted averages, and nonconvex guarantees have been established for Schedule-Free SGD [1]. In OCL, a decreasing averaging coefficient may reduce responsiveness to newly arriving data; its effect on retention and adaptation requires empirical assessment.

Generalized Primal Averaging (GPA). GPA [13] uses two independent interpolation coefficients:

$$y_t = \mu_y x_t + (1 - \mu_y)z_t, \tag{55}$$

$$z_{t+1} = z_t - \gamma_t g_t(y_t), \tag{56}$$

$$x_{t+1} = \mu_x x_t + (1 - \mu_x)z_{t+1}. \tag{57}$$

Here, μ_y controls the gradient evaluation point, whereas μ_x controls the persistence of the averaged sequence. Expanding the last recursion gives

$$x_{t+1} = \mu_x^{t+1}x_0 + (1 - \mu_x)\sum_{j=0}^{t}\mu_x^{t-j}z_{j+1}. \tag{58}$$

Thus, GPA assigns exponentially decaying weights to past base iterates, with a constant weight on each new iterate. GPA was developed to smooth the periodic updates of single-worker DiLoCo and improve training efficiency. Its fixed averaging coefficient does not automatically inherit the schedule-free property of cumulative averaging.

Perspective for OCL. Our focus is the behavior of these dynamics under non-stationary, single-pass learning: how gradient interpolation and averaging balance acquisition of new knowledge with retention of previous tasks. Motivated by the stability–plasticity dilemma, SPIN interprets the fast sequence as adaptation and the slow sequence as progressive consolidation. The curvature interpretation in Appendix 4.2 and the accuracy–forgetting analysis in Appendix 5.3 examine these questions in the OCL setting.

H Detailed Experimental Setup for Sec. 4.3

H.1 Setup for verifying the quasi-steady-state assumption

To validate the quasi-steady-state (QSS) assumption used in our derivation, we measure

$$r_t = \frac{\|d_{t+1}\|}{\|d_t\|}, \tag{59}$$

where $d_t = \phi_t - \theta_t$ is the gap between the plasticity sequence and the stability sequence.

We use the ER method as the base OCL framework and keep the setup fully consistent with the main experiments. The experiment is conducted on Split CIFAR-100, with 10 tasks and 10 classes per task. The backbone is ResNet-18. The stream batch size is 10, the replay batch size is 64, and the replay buffer size is 2000. We use SPIN-AdamW with $\alpha = 0.05$ and $\beta = 0.05$. All remaining hyperparameters are the same as those used in the corresponding main experiments.

During training, we record r_t at every optimization step. The resulting curve is reported in Figure 2a of the main paper.

H.2 Setup for comparing update norms with Newton's method

To further validate the curvature-related interpretation of SPIN, we compare the ℓ_2 norm of its update with that of a Newton-type method.

Because computing Newton updates on CIFAR-100 with a ResNet-18 backbone is computationally too expensive due to the high dimensionality of the model parameters and the cost of Hessian computation, we instead perform this experiment on MNIST. This choice allows efficient computation of second-order information while preserving the purpose of the experiment, namely, comparing update norm behavior.

We train a small multilayer perceptron (MLP) classifier on MNIST using the standard cross-entropy loss. The network consists of two fully connected hidden layers with 256 units each and ReLU activations. The batch size is set to 128, and the model is trained for 20 epochs.

We compare two optimization methods:

- **SPIN**: our proposed optimizer built on top of SGD.
- **Newton**: a Newton-type method that computes updates using the gradient and the Hessian of the current mini-batch loss.

For SPIN, we use a base learning rate of **0.01**, with hyperparameters $\alpha = 0.05$ and $\beta = 0.05$, which are consistent with the settings used in the main experiments.

At each training step t, we compute the parameter update vector Δ_t produced by each method and record its ℓ_2 norm $\|\Delta_t\|_2$. For SPIN, Δ_t denotes the actual update applied by the optimizer to the model parameters. For the Newton method, Δ_t corresponds to the second-order update computed from the gradient and Hessian.

Both methods use the same data order and training objective to ensure a fair comparison. Since the purpose of this experiment is theoretical validation rather than performance benchmarking, we focus on the evolution of the update norm during training instead of the final classification accuracy. The resulting comparison is shown in fig. 2b of the main paper.

I Algorithm Pseudocode

Algorithm 1: SPIN-SGD for OCL

Input: Task stream $\{\mathcal{D}_t\}_{t=1}^{T}$; replay buffer capacity $\mathcal{M}$; interpolation coefficient β; stability merging coefficient α; learning rate γ; weight decay λ.
Output: Final stability parameters θ_S.
Initialize network parameters θ_1; set $\phi_1 \leftarrow \theta_1$.
Initialize replay buffer $\mathcal{M} \leftarrow \emptyset$.
Initialize global step counter $s \leftarrow 1$.
for $t = 1$ **to** T **do**
  for *each incoming mini-batch* $\mathcal{B}^{\textit{stream}} \subset \mathcal{D}_t$ **do**
    Sample replay batch $\mathcal{B}^{\mathcal{M}} \sim \mathcal{M}$ if $\mathcal{M}$ is not empty.
    Construct training batch $\mathcal{B}_s \leftarrow \mathcal{B}^{\text{stream}} \cup \mathcal{B}^{\mathcal{M}}$.
    Compute interpolated parameters $x_s \leftarrow (1-\beta)\phi_s + \beta\theta_s$.
    Compute loss $\mathcal{L}_s$ on $\mathcal{B}_s$ using the upstream OCL method with parameters x_s.
    Compute stochastic gradient $g_s \leftarrow \nabla\mathcal{L}_s(x_s)$.
    `// Plasticity update`
    $\phi_{s+1} \leftarrow \phi_s - \gamma(g_s + \lambda x_s)$.
    `// Stability update`
    $\theta_{s+1} \leftarrow \theta_s + \alpha(\phi_{s+1} - \theta_s)$.
    Update replay buffer $\mathcal{M}$ with $\mathcal{B}^{\text{stream}}$.
    $s \leftarrow s + 1$.
  end
end
return θ_S

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe the main contributions of the paper, including the proposed method and experimental validation. These claims are supported by the theoretical analysis and experimental results presented in Section 4 and 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of this work are discussed in the final section ("Conclusion and Limitations"). Specifically, we discuss the reliance of the theoretical analysis on smoothness and quasi-steady-state assumptions, as well as the additional optimizer-state overhead introduced by maintaining dual iterate sequences. We also outline future directions toward more parameter-efficient interpolation mechanisms for edge-device deployment.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All theoretical assumptions are explicitly stated in Sections 4.2–4.3, and complete proofs of Proposition 1 and Theorem 1 are provided in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experimental settings, datasets, baselines, evaluation metrics, and implementation details are described in Section 5 and Appendix B. We report the mean results over 5 runs and provide all relevant hyperparameter settings to facilitate reproducibility. The code will be released upon publication.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We use publicly available benchmark datasets, and the implementation code will be released upon publication to support reproducibility.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 and Appendix B provide detailed descriptions of datasets, baselines, hyperparameters, evaluation protocols, and implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Most experimental results are averaged over 5 independent runs to improve reliability, while the large-scale ImageNet-1K experiments are conducted with a single run due to computational cost. Although formal statistical significance tests or confidence intervals are not included, additional robustness analyses under different optimization configurations are provided in the Appendix, showing consistent performance trends.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computational resources used for the experiments are described in the mainbody and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and complied with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: To the best of our knowledge, this paper constitutes foundational research in the field of online continuous learning and does not directly address its social implications.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This work proposes an optimization method for online continual learning and does not involve the release of high-risk models, generative systems, scraped datasets, or other assets that pose substantial misuse concerns. The proposed optimizer is designed to improve learning stability and mitigate catastrophic forgetting in continual learning settings. Therefore, we believe no additional safeguards are required beyond standard responsible research practices.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets, models, and existing methods used in this work are publicly available research assets and are properly cited in the paper. We use standard academic benchmarks and publicly available implementations in accordance with their respective licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: No new datasets or models are released in this paper.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve human subjects or crowdsourcing experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This study does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs have been used for language polishing or writing assistance only, without affecting the scientific contributions or methodology of the paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.