SHALLOW-WIDE FINE-TUNING FOR CONTINUAL LEARNING WITH PRE-TRAINED MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Full fine-tuning of Pre-Trained Models (PTMs) on continually arriving downstream tasks often leads to catastrophic forgetting. Existing PTM-based Continual Learning (CL) methods typically freeze the backbone and attach auxiliary modules to preserve previously learned knowledge. However, frozen representations limit plasticity under large distribution shifts, and auxiliary modules incur additional inference overhead. To address these limitations, we propose Shallow-Wide Fine-Tuning (**SW-Tune**), which enables stable backbone updates without any auxiliary components. Inspired by the empirical insight that shallow-and-wide architectures exhibit inherently greater resilience to forgetting than deep-and-narrow ones, SW-Tune introduces a principled optimization framework that modulates the gradient subspace of a deep PTM to emulate shallow-and-wide learning dynamics without physical architectural changes. Specifically, *periodic block freezing* restricts trainable parameters to an equidistant, sparse subset of blocks, functionally reducing the effective trainable depth. Simultaneously, *gradient spectral flattening* regularizes the singular-value spectrum of layer-wise gradients, compelling parameter updates to span a broader orthogonal subspace, thereby expanding the effective capacity width. Extensive experiments demonstrate that SW-Tune outperforms existing PTM-based CL methods while introducing no additional inference overhead. Code is available at `https://anonymous.4open.science/r/SW-Tune`.

1 INTRODUCTION

The paradigm of pre-training on large-scale data followed by fine-tuning on downstream tasks has revolutionized modern machine learning (Alaparthi & Mishra, 2020; Dosovitskiy, 2020; Achiam et al., 2023). However, its success is built upon the assumption of static data distributions and fixed label spaces. In real-world scenarios, data arrive sequentially and new classes may emerge over time. When a Pre-Trained Model (PTM) is directly updated on each incoming task, it suffers from catastrophic forgetting, i.e., the severe and often irreversible degradation of previously learned knowledge (Zhang et al., 2023; Zhou et al., 2024a). This necessitates a careful balance between acquiring new knowledge (*plasticity*) and retaining what has already been learned (*stability*), giving rise to the study of PTM-based Continual Learning (CL) (Zhou et al., 2024b).

A central finding in this domain is that fine-tuning the entire backbone risks corrupting the general-purpose representations acquired during pre-training, which in turn undermines the model's capacity to adapt to new downstream tasks (Zhou et al., 2024a). To preserve these transferable representations, existing PTM-based CL methods typically freeze the backbone and introduce lightweight trainable modules to absorb task-specific information (Zhou et al., 2024b). For instance, L2P (Wang et al., 2022b) and DualPrompt (Wang et al., 2022a) prepend learnable prompts to the input, whereas SSIAT (Tan et al., 2024) and TUNA (Wang et al., 2025) attach adapter modules to intermediate layers. Although such freezing-based strategies effectively preserve stability, their plasticity is inherently bounded by the frozen backbone, particularly when downstream distributions deviate substantially from the pre-training domain (Lu et al., 2026). Moreover, the auxiliary modules are typically retained at inference time, incurring non-trivial computational overhead.

To address these limitations, we investigate whether the backbone itself can be updated stably during CL, thereby achieving high plasticity without introducing extra inference overhead. We examine this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

question through the lens of network architecture, a factor that has been shown to fundamentally influence forgetting behavior. Specifically, shallow-and-wide architectures have been shown, both theoretically and empirically, to exhibit greater resilience to forgetting than their deep-and-narrow counterparts (Mirzadeh et al., 2022a; Lu et al., 2024). This insight suggests that a deep PTM could be fine-tuned stably if its optimization process were modulated to mimic the effective behavior of a shallow-and-wide architecture. Since the physical architecture of a PTM cannot be altered, we propose to replicate this favorable configuration directly within the optimization subspace.

To this end, we propose Shallow-Wide Fine-Tuning (**SW-Tune**), a framework comprising two complementary components that target the respective geometric dimensions. To reduce the effective depth, *Periodic Block Freezing* (PBF) keeps only a periodically distributed subset of blocks trainable while freezing the rest. By treating the frozen blocks as fixed transformations, PBF restricts parameter updates to discrete layers, functionally compressing the optimization path and reducing the effective trainable depth. To increase the effective width, *Gradient Spectral Flattening* (GSF) flattens the singular-value spectrum of each linear layer's gradient. Standard fine-tuning often suffers from gradient collapse into a few dominant singular directions (i.e., low-rank updates), which restricts the model's capacity to a narrow subspace. By preventing this collapse, GSF forces updates to span the layer's full orthogonal subspace, expanding the effective capacity width. By combining these two components, SW-Tune enables the PTM to function as an effectively shallower and wider network, achieving a superior stability–plasticity trade-off without relying on auxiliary modules.

Our contributions are as follows:

- We introduce a novel perspective on PTM-based CL, showing that a shallow-and-wide optimization subspace inherently mitigates catastrophic forgetting without auxiliary modules.
- We propose SW-Tune, a fine-tuning framework that reshapes the PTMs' optimization subspace by equivalently reducing depth and expanding width, yielding a superior stability–plasticity trade-off.
- Extensive experiments demonstrate that SW-Tune outperforms existing state-of-the-art CL methods on challenging benchmarks while maintaining the PTM's original inference efficiency.

2 Related Work

Continual Learning (CL). CL aims to enable neural networks to learn from sequentially arriving tasks without catastrophic forgetting (Masana et al., 2022; Van de Ven et al., 2022; Wang et al., 2024). Prior CL research has largely followed three main paradigms. Replay-based methods retain a subset of previous data in a memory buffer and replay it to approximate old data distributions (Aljundi et al., 2019; Liu et al., 2020; Iscen et al., 2020; Zhao et al., 2021). Regularization-based methods introduce penalty terms that constrain model updates when learning new tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Li & Hoiem, 2017; Rebuffi et al., 2017; Feng et al., 2022). Parameter-isolation methods allocate task-specific parameter subspaces within the network for each new task (Kang et al., 2022; Konishi et al., 2023; Yan et al., 2021; Zhou et al., 2023).

Pre-Trained Model-based CL. With the advent of powerful PTMs such as ViTs (Dosovitskiy, 2020) and CLIP (Thengane et al., 2022), research attention has increasingly shifted toward PTM-based CL (Zhou et al., 2024b). Despite a few efforts to fine-tune the entire PTM backbone (Zhang et al., 2023), the prevailing practice is to freeze the backbone and learn lightweight auxiliary modules for adaptation to new tasks. Prompt-based methods, for instance, maintain a dynamic pool of prompt tokens to guide the model across sequentially arriving tasks, as exemplified by L2P (Wang et al., 2022b), DualPrompt (Wang et al., 2022a), and CODA-Prompt (Smith et al., 2023). Adapter-based methods follow a similar philosophy but replace prompts with dedicated adapter layers, e.g., SSIAT (Tan et al., 2024), MOS (Sun et al., 2025), and TUNA (Wang et al., 2025). However, these auxiliary modules introduce extra inference overhead, and the frozen backbone severely limits representational flexibility under large domain shifts. In this work, we demonstrate that fine-tuning the backbone with tailored strategies can simultaneously overcome both limitations.

Network Architecture for CL. Beyond parameter-level strategies, the interplay between network architecture and CL has drawn growing attention (Pham et al., 2022; Mirzadeh et al., 2022b). In particular, prior studies (Mirzadeh et al., 2022a; Lu et al., 2024) have empirically demonstrated that wider, shallower networks consistently suffer less catastrophic forgetting than their deeper, narrower counterparts with equivalent parameter budgets. Inspired by this insight, our method algorithmically

emulates the characteristics of wide, shallow networks through tailored optimization. Crucially, this avoids explicit architectural modifications, making our approach directly applicable to PTM-based CL, where the backbone structure is typically fixed.

3 PRELIMINARIES

This paper focuses on the exemplar-free class-incremental learning setting (Van de Ven et al., 2022), where a model sequentially learns from a stream of T disjoint tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$. For each task $\mathcal{T}_t$, the model is trained exclusively on its corresponding dataset $\mathcal{D}_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{N_t}$ with label space $\mathcal{Y}_t$. Crucially, the label spaces are mutually disjoint ($\mathcal{Y}_t \cap \mathcal{Y}_{t'} = \emptyset$ for $t \neq t'$), and no task identifiers are provided during inference. After learning tasks up to stage t, the model must generalize to the accumulated label space $\mathcal{Y}_{1:t} = \bigcup_{j=1}^{t} \mathcal{Y}_j$ by evaluating on the union of all past test sets $\bigcup_{j=1}^{t} \mathcal{D}_j^{\text{test}}$. The objective is to learn a unified classifier $f : \mathcal{X} \to \mathcal{Y}_{1:t}$ that minimizes the expected classification error over all seen classes:

$$f^* = \arg\min_{f \in \mathcal{H}} \mathbb{E}_{(\mathbf{x},y)\sim\bigcup_{j=1}^{t} \mathcal{D}_j^{\text{test}}} \mathbb{I}(y \neq f(\mathbf{x})), \tag{1}$$

where $\mathcal{H}$ denotes the hypothesis space and $\mathbb{I}(\cdot)$ is the indicator function. Following recent PTM-based CL methods (Smith et al., 2023; Sun et al., 2025), we initialize f using a pre-trained ViT. The model is decomposed as $f(\mathbf{x}) = W^\top \phi(\mathbf{x})$, where $\phi : \mathbb{R}^D \to \mathbb{R}^d$ is the feature extractor consisting of L stacked Transformer blocks, and W is the classification head. The feature extractor ϕ maps the input to a d-dimensional representation using the [CLS] token from the final block. To accommodate new classes, the classification head dynamically expands at each stage t, maintaining a weight matrix $W \in \mathbb{R}^{d\times|\mathcal{Y}_{1:t}|}$ that spans the accumulated label space $\mathcal{Y}_{1:t}$ and ultimately reaches a final dimension of $d \times |\mathcal{Y}_{1:T}|$ after all T tasks.

4 SHALLOW-WIDE FINE-TUNING (SW-TUNE)

Existing PTM-based CL methods preserve stability by freezing the backbone and attaching auxiliary modules, but this design inherently caps plasticity and adds inference overhead. In this section, we present SW-Tune, a fine-tuning framework that directly adapts the backbone without auxiliary modules while remaining resilient to catastrophic forgetting.

The key insight behind SW-Tune is that shallow-and-wide architectures are more resistant to forgetting than deep-and-narrow ones (Mirzadeh et al., 2022a; Lu et al., 2024). Although the physical architecture of a PTM is fixed, we can modulate its optimization process so that the effective training dynamics resemble those of a shallower and wider network. SW-Tune achieves this through two complementary strategies, illustrated in Figure 1. Specifically, PBF reduces the *effective depth* (i.e., the number of layers whose parameters receive gradient updates) by training only a periodically selected subset of Transformer blocks while freezing the rest. GSF expands the *effective width* (i.e., the effective rank of the gradient spectrum) by flattening the singular-value distribution of each linear layer's gradient so that updates span a broader orthogonal subspace.

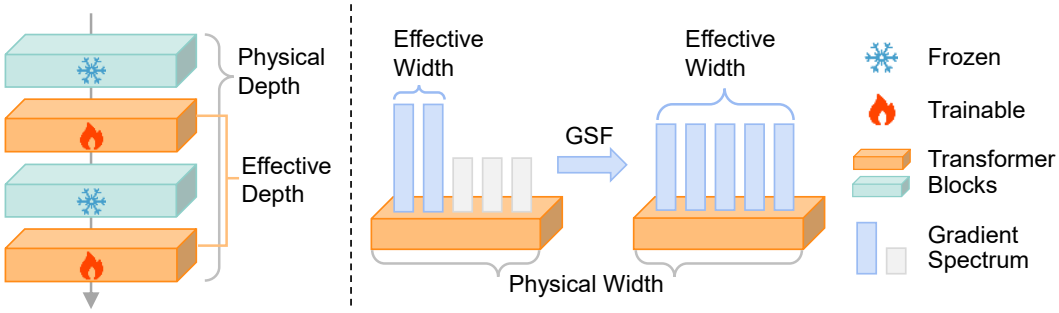


Figure 1: Illustration of the SW-Tune framework. **Left:** PBF freezes a subset of Transformer blocks, yielding a smaller *Effective Depth* than the *Physical Depth*. **Right:** GSF flattens the gradient spectrum so that updates engage a broader orthogonal subspace, thereby increasing the *Effective Width*.

4.1 Reducing Effective Depth via Periodic Block Freezing

In a standard Transformer with L blocks, gradient updates propagate through all L layers, creating a deep optimization path that is prone to catastrophic forgetting (Mirzadeh et al., 2022a). A natural remedy is to reduce the number of trainable layers. Naively freezing early or late blocks, however, sacrifices the ability to adapt features at certain levels of the representation hierarchy. PBF addresses this by distributing a small number of trainable blocks periodically across the full depth of the network, thereby reducing the effective depth while preserving multi-level feature adaptation.

Recall that the feature extractor ϕ consists of L Transformer blocks indexed by $\{0, 1, \ldots, L-1\}$. PBF retains only K trainable blocks ($K < L$). Assuming K divides L, we define the stride $s = L/K$, partition the blocks into K consecutive segments of length s, and keep only the last block of each segment trainable:

$$\mathcal{I}_{\text{train}} = \{(k+1)s - 1 \mid k = 0,\, 1,\, \ldots,\, K-1\}\,. \tag{2}$$

Let g_k denote the k-th trainable block, and let h_k denote the composition of the $s-1$ frozen blocks immediately preceding g_k. The backbone then factorizes as

$$\phi = g_{K-1} \circ h_{K-1} \circ \cdots \circ g_1 \circ h_1 \circ g_0 \circ h_0. \tag{3}$$

Because each h_k is a fixed, parameter-free transformation from the optimizer's perspective, gradient updates flow through only the K trainable blocks. This reduces the effective depth from L to K, conferring the stability benefits of a shallower architecture. Crucially, because the trainable blocks are distributed across shallow, middle, and deep portions of the network, PBF retains the ability to adapt features at multiple levels of abstraction.

Reducing the effective depth improves stability but may also reduce the plasticity available for learning new tasks. To balance these two objectives, we set $K = 3$ as the default. For a standard 12-block ViT, this yields a stride of $s = 4$ and trainable indices $\mathcal{I}_{\text{train}} = \{3,\, 7,\, 11\}$. An ablation study justifying this choice is provided in Section 5.4.

4.2 Expanding Effective Width via Gradient Spectral Flattening

In practice, the gradient of a linear layer often exhibits a highly skewed singular-value spectrum: a few dominant singular directions capture most of the update magnitude, while the remaining directions are underutilized. When multiple tasks share these dominant directions, their weight displacements overlap, increasing the risk of cross-task interference. GSF mitigates this by flattening the gradient spectrum, redistributing the update magnitude over a larger set of orthogonal directions and thereby expanding the effective width of each layer.

For a linear layer with weight matrix $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, let $\boldsymbol{G} = \nabla_{\boldsymbol{W}} \mathcal{L}$ denote the gradient of the loss with respect to $\boldsymbol{W}$. We compute the compact SVD $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_d)$ with $\sigma_1 \geq \cdots \geq \sigma_d \geq 0$ and $d = \min(m, n)$. Given a flattening exponent $p \in [0, 1]$, GSF applies a power transformation to the singular values:

$$\widetilde{\boldsymbol{G}} = \boldsymbol{U}\, \operatorname{diag}\big(\sigma_1^p,\, \ldots,\, \sigma_d^p\big)\, \boldsymbol{V}^\top. \tag{4}$$

When $p = 1$, the original gradient is recovered; as p decreases, the spectrum is progressively compressed; at $p = 0$, all non-zero singular values become equal, yielding maximum flattening.

To ensure that GSF modifies only the directional distribution of the gradient without altering its overall magnitude, we rescale $\widetilde{\boldsymbol{G}}$ to preserve the Frobenius norm:

$$\boldsymbol{G}^* = \widetilde{\boldsymbol{G}}\, \frac{\|\boldsymbol{G}\|_F}{\max(\|\widetilde{\boldsymbol{G}}\|_F,\, \epsilon_{\text{norm}})}, \tag{5}$$

where $\epsilon_{\text{norm}} > 0$ is a small constant for numerical stability. The resulting gradient $\boldsymbol{G}^*$ retains the original singular vectors but distributes the update energy more uniformly across them, effectively expanding the width of the optimization subspace without changing the layer's physical dimensions.

Computing an explicit SVD of $\boldsymbol{G}$ at every optimization step can be expensive for large layers. In practice, GSF can be implemented more efficiently via an eigendecomposition of the smaller Gram matrix $\boldsymbol{G}^\top\boldsymbol{G} \in \mathbb{R}^{n \times n}$ or $\boldsymbol{G}\boldsymbol{G}^\top \in \mathbb{R}^{m \times m}$. Further details are provided in Appendix A.3. To progressively strengthen flattening as tasks accumulate and subspace conflicts intensify, p is linearly annealed from 1 to 0 across the task sequence (i.e., $p_t = 1 - \frac{t-1}{T-1}$ for task t).

4.3 Training Pipeline

Within SW-Tune, the PTM is continually optimized across incremental tasks. Following recent representative PTM-based CL methods (Sun et al., 2025; Wang et al., 2025), we optimize the network with a cosine-similarity cross-entropy loss and adopt the off-the-shelf post-processing head alignment from TUNA (Wang et al., 2025). Both PBF and GSF are integrated into the training loop of each task. PBF restricts gradient updates to the K selected Transformer blocks and the classification head, with all other parameters remaining frozen. After backpropagation but before the optimizer step, GSF replaces the gradient $\boldsymbol{G}$ of every linear layer within the selected blocks (including attention projections and MLP layers) with its spectrally flattened counterpart $\boldsymbol{G}^*$ (Eq. equation 5). The classification head is updated with original gradients.

5 Experiments

5.1 Experimental Setup

Backbones and Benchmarks. We employ two representative ViTs as pre-trained backbones: ViT-B/16-IN1K and ViT-B/16-IN21K. Both are initially pre-trained on ImageNet-21K (Ridnik et al., 2021), while ViT-B/16-IN1K is further fine-tuned on ImageNet-1K (Deng et al., 2009). The need for CL is particularly pronounced when substantial distribution shifts exist between downstream tasks and the pre-training data. Accordingly, we select three datasets that exhibit notable domain shifts from ImageNet-21K and ImageNet-1K: ImageNet-R (Hendrycks et al., 2021a), ImageNet-A (Hendrycks et al., 2021b), and ObjectNet (Barbu et al., 2019). Following existing studies (Zhou et al., 2024b), we standardize each dataset to 200 classes and partition it into 10 tasks of 20 classes each, yielding three benchmarks: ImageNet-R Inc20, ImageNet-A Inc20, and ObjectNet Inc20.

Evaluation Metrics. We evaluate the overall CL performance using two metrics: (1) *Last Accuracy (LA)*: the classification accuracy over all seen classes after completing the final task, defined as $\mathrm{LA} = \frac{1}{T}\sum_{j=1}^{T} R_{T,j}$, where $R_{i,j}$ denotes the test accuracy on task j after learning task i. (2) *Average Incremental Accuracy (AIA)*: the mean accuracy across all incremental stages, defined as $\mathrm{AIA} = \frac{1}{T}\sum_{i=1}^{T}\left(\frac{1}{i}\sum_{j=1}^{i} R_{i,j}\right)$.

Baseline Methods. We compare SW-Tune against six representative PTM-based CL methods: L2P (Wang et al., 2022b), DualPrompt (Wang et al., 2022a), CODA-Prompt (Smith et al., 2023), SSIAT (Tan et al., 2024), MOS (Sun et al., 2025), and TUNA (Wang et al., 2025).

Implementation Details. For SW-Tune and its ablation variants, all models are trained for 10 epochs per task using SGD with momentum 0.9, weight decay 10^{-4}, and batch size 128. The initial learning rate is set to 0.01 and decayed via cosine annealing. For baselines, we follow their implementations and default hyperparameter configurations in the open-source library PILOT (Sun et al., 2023) to ensure faithful reproductions, while aligning the batch size (128) and epochs (10) with SW-Tune to ensure a fair evaluation. We report mean performance over five runs with different class orders where applicable. For analyses that are impractical to repeat across multiple orderings (e.g., performance curves and visualizations), we report results under a single class order and verify that the qualitative trends remain consistent across other orderings.

5.2 Main Results

Results with ViT-B/16-IN1K. Table 1 presents the comparison between SW-Tune and the baselines using ViT-B/16-IN1K. SW-Tune achieves the best performance across all three datasets under both evaluation metrics. On ImageNet-R, it achieves 81.25% LA and 85.25% AIA, outperforming the strongest baseline SSIAT by 2.49% and 3.61%, respectively. The advantage is even more pronounced on ImageNet-A, which exhibits the most severe domain shift, where SW-Tune reaches 65.24% LA and 72.00% AIA, surpassing the best competitor TUNA by 4.54% and 4.76%, respectively. On ObjectNet, SW-Tune retains a clear advantage, achieving 65.23% LA and 73.01% AIA and exceeding the strongest baseline SSIAT by 0.60% and 1.01%, respectively. These consistent gains highlight the effectiveness of SW-Tune under substantial domain shifts.

Table 1: Performance (%) of existing state-of-the-art CL methods and SW-Tune using ViT-B/16-IN1K. **Bold** indicates the best result; underline denotes the second best. "Improvement" is the gain of SW-Tune over the strongest baseline. Mean ± standard deviation over five runs is reported.

Method	ImageNet-R Inc20		ImageNet-A Inc20		ObjectNet Inc20	
	LA	AIA	LA	AIA	LA	AIA
L2P	71.91±0.27	76.76±0.45	42.58±0.39	50.42±1.12	55.91±0.33	64.17±1.32
DualPrompt	69.43±0.51	74.85±0.18	45.35±1.04	54.72±1.64	53.99±0.30	62.90±1.48
CODA-Prompt	73.24±0.40	77.13±0.49	45.36±0.78	57.04±0.94	56.24±0.39	65.95±0.35
SSIAT	78.76±0.24	81.64±0.34	59.57±0.32	66.54±1.36	64.63±0.28	72.00±1.29
MOS	74.07±0.36	78.84±0.43	57.71±0.55	65.84±1.00	62.75±0.30	70.62±1.49
TUNA	78.25±0.15	81.33±0.24	60.70±0.66	67.24±1.85	62.36±0.39	70.07±1.48
SW-Tune (Ours)	**81.25**±0.25	**85.25**±0.36	**65.24**±0.15	**72.00**±1.51	**65.23**±0.41	**73.01**±1.11
Improvement	+2.49	+3.61	+4.54	+4.76	+0.60	+1.01

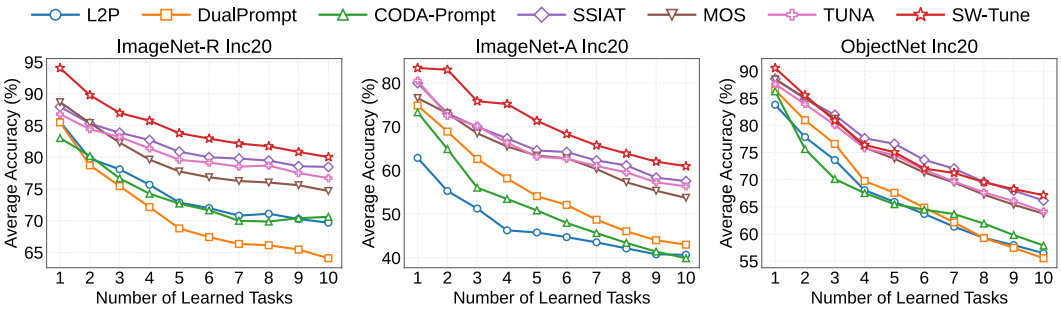


Figure 2: Performance curves of existing CL methods and SW-Tune using ViT-B/16-IN21K.

Results with ViT-B/16-IN21K. Figure 2 presents the performance curves of SW-Tune and the baselines using ViT-B/16-IN21K. As the number of learned tasks grows from 1 to 10, all methods exhibit varying degrees of performance degradation. Nevertheless, SW-Tune maintains the highest accuracy at the final stage on all three datasets. In particular, it consistently outperforms all baselines at every incremental stage on ImageNet-R and ImageNet-A. These results suggest that the advantages of SW-Tune generalize across different PTM initializations.

Analysis of Inference Overhead. Since SW-Tune introduces no auxiliary module, its inference requires only a single forward pass through the ViT backbone. To verify the resulting efficiency advantage, we compare the inference latency and throughput of SW-Tune against existing state-of-the-art CL methods, as reported in Table 2. All competing methods incur additional inference overhead due to their auxiliary components. By contrast, SW-Tune, free from any auxiliary-module overhead, achieves the lowest latency (6.99 ms) and the highest throughput (420.2 FPS), making it well suited for latency-sensitive deployment.

Table 2: Comparison of inference latency and throughput across different methods. Latency is measured with batch size 1, and throughput with batch size 64. All inputs are of size $3 \times 224 \times 224$.

Metric	L2P	DualP	CODA-P	SSIAT	MOS	TUNA	SW-Tune
Latency (ms) ↓	11.82	12.09	13.76	12.08	363.34	132.53	**6.99**
Throughput (FPS) ↑	197.8	210.8	213.2	398.3	35.8	36.4	**420.2**

5.3 Analysis of the Stability–Plasticity Trade-off

We further dissect the stability–plasticity trade-off through two complementary metrics. (1) *Average Optimal Accuracy (AOA)*: the mean of the best accuracy attained on each task over the entire CL process. Formally, AOA $= \frac{1}{T}\sum_{j=1}^{T} \max_{i \geq j} R_{i,j}$, where $R_{i,j}$ denotes the test accuracy on task j after learning task i. A higher AOA reflects greater plasticity. (2) *Average Forgetting (AF)*: the mean accuracy drop across all preceding tasks once the final task has been learned. Formally, AF $=$

$\frac{1}{T-1}\sum_{j=1}^{T-1}\left(\max_{j\le l<T} R_{l,j} - R_{T,j}\right)$. A lower AF indicates stronger stability. Notably, the overall performance metric LA admits the decomposition LA $=$ AOA $-\ \frac{T-1}{T}$AF; given that $T = 10$ in our experimental setup, this reduces to LA $=$ AOA $- 0.9 \times$ AF. For conciseness, this and all subsequent analyses are conducted exclusively with ViT-B/16-IN1K.

Figure 3 provides a quantitative breakdown of the stability–plasticity trade-off on three benchmarks. SW-Tune exhibits a slightly higher AF yet a substantially higher AOA, indicating that it sacrifices a marginal degree of stability in exchange for a considerably larger gain in plasticity. This results in a more favorable stability–plasticity balance and, consequently, superior overall CL performance, as evidenced by its highest LA among all compared methods.

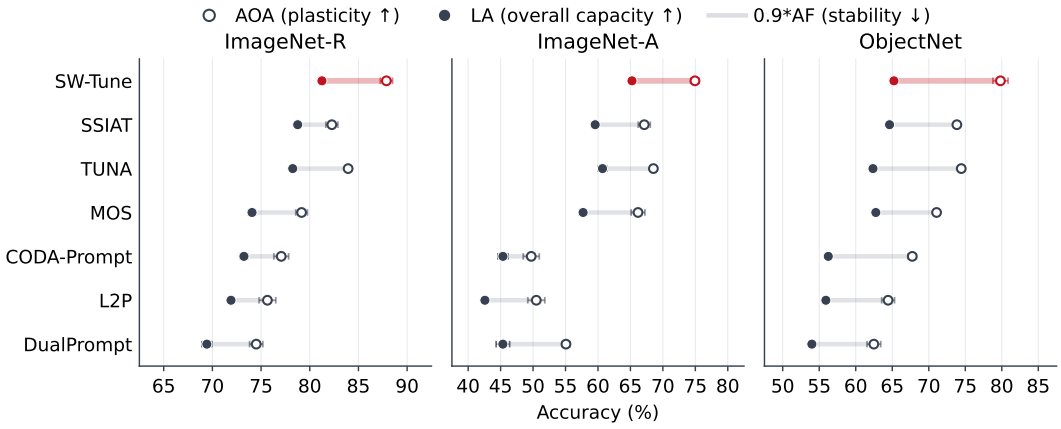


Figure 3: AOA, AF, and LA of existing CL methods and SW-Tune, averaged over five runs.

5.4 Ablation Study

Effectiveness of Each Component. To assess the contribution of each component, we compare SW-Tune against the following variants: *Full Fine-tuning* (with neither PBF nor GSF), *PBF Only*, *GSF Only*, and two PBF layout variants, namely *Bottom-Layer Tuning* and *Top-Layer Tuning*. The former keeps only a contiguous bottom region (blocks 0–2) trainable, while the latter keeps only a contiguous top region (blocks 9–11) trainable. Figure 4 reports the accuracy curves on three benchmarks as the number of learned classes increases. All methods exhibit accuracy decay as the task sequence lengthens, but the degree of degradation varies considerably. This comparison reveals two key findings. First, full fine-tuning suffers pronounced accuracy loss due to catastrophic forgetting. Applying either PBF or GSF alone alleviates forgetting to a moderate extent, yet their combination in SW-Tune achieves the most stable performance throughout the sequence. This confirms that the two components target different sources of forgetting and that their benefits are largely complementary. Second, SW-Tune with its periodic freezing layout $\{3, 7, 11\}$ outperforms both *Bottom-Layer Tuning* and *Top-Layer Tuning*, indicating that distributing trainable blocks across multiple depths, rather than concentrating them in a single contiguous region, yields a better stability–plasticity trade-off.

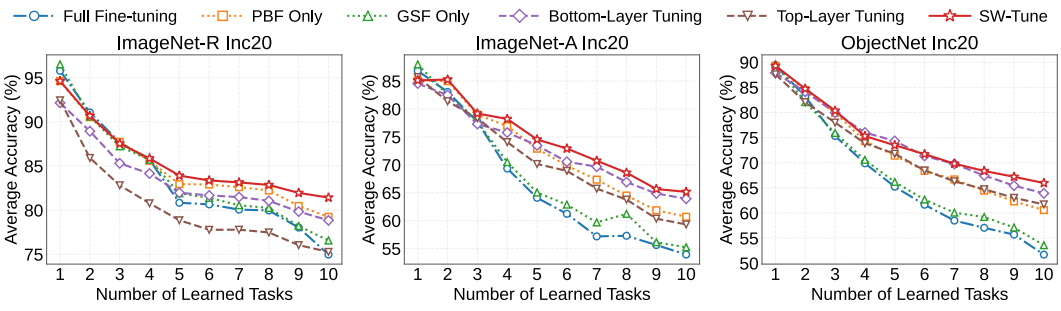


Figure 4: Accuracy curves of SW-Tune and its ablation variants on three CL benchmarks.

Impact of the Number of Trainable Blocks. We further investigate how the number of trainable blocks, K, affects the stability-plasticity trade-off. We evaluate PBF with $K \in \{2, 3, 4, 6, 12\}$ on ImageNet-A Inc20, with GSF disabled. As shown in Figure 5, increasing K monotonically improves plasticity (reflected by higher AOA), but stability degrades accordingly (reflected by higher AF). Consequently, the overall metric LA follows an inverted-U trend and reaches its peak at $K = 3$, indicating that this setting best balances the ability to learn new classes against the need to retain previously acquired knowledge. This justifies our default choice of $K = 3$.

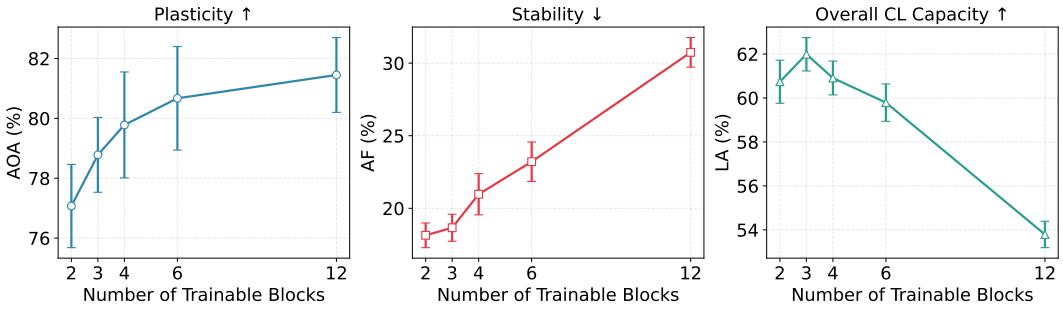


Figure 5: Effect of the number of trainable blocks K on ImageNet-A Inc20.

5.5 Mechanistic Analysis

To understand *why* SW-Tune facilitates CL, we conduct a mechanistic analysis at both the parameter and representation levels, comparing SW-Tune against full fine-tuning. We focus on three representative Transformer blocks (*Block 3*, *Block 7*, and *Block 11*) in the ImageNet-A Inc20 setting. Let M_i denote the model after completing tasks $1, \ldots, i$, with M_0 the initial pre-trained model and W_i its corresponding weight parameters.

Parameter-Level Analysis. We first examine the spectral properties of weight updates through two complementary metrics. (1) *Effective rank of the cumulative update.* We compute the Shannon entropy of the singular-value spectrum of $\Delta W = W_{10} - W_0$. A higher effective rank indicates that update energy is distributed more evenly across singular directions, rather than being concentrated along a few dominant modes. (2) *Similarity between updates.* We measure the absolute cosine similarity between the single-step updates of the first task ($W_1 - W_0$) and the last task ($W_{10} - W_9$). A lower similarity implies that the updates for different tasks occupy nearly orthogonal subspaces, suggesting reduced cross-task interference. As shown in Figure 6 (Left), SW-Tune consistently yields a higher effective rank and significantly lower update similarity than full fine-tuning across all three blocks. This indicates that SW-Tune distributes parameter changes more uniformly across the weight space while enforcing near-orthogonal update directions for successive tasks, thereby mitigating cross-task interference.

Representation-Level Analysis. We further assess representation stability using Centered Kernel Alignment (CKA) (Kornblith et al., 2019). For every pair of checkpoints (M_i, M_j) with $i, j \in \{1, \ldots, 10\}$, we compute the CKA similarity of their intermediate features at each block, yielding a 10×10 heatmap (Figure 6, Right). Higher CKA values indicate less representational drift over the course of CL. As expected, deeper blocks exhibit greater drift under both methods. Nevertheless, SW-Tune consistently preserves substantially higher CKA similarity than full fine-tuning at every depth. This confirms that SW-Tune more effectively retains shared representations across incremental tasks, enabling adaptation to new tasks with minimal disruption to previously acquired features and thus achieving a better stability–plasticity balance.

5.6 Comparison with SLCA

While the methods compared above adopt the mainstream freezing-based paradigm, SLCA (Zhang et al., 2023) takes a different route by fine-tuning the backbone with selectively lowered learning rates to mitigate catastrophic forgetting. Since both SW-Tune and SLCA fall under the backbone fine-tuning paradigm yet differ fundamentally in mechanism, a direct comparison is warranted. We conduct this comparison on ImageNet-A (Figure 7). Both SLCA and its enhanced variant

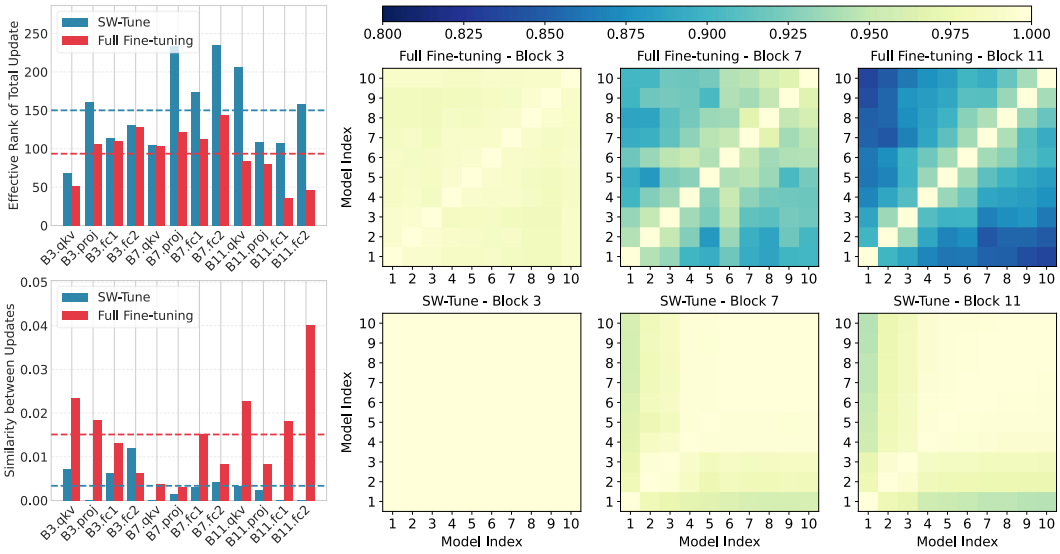


Figure 6: **Left:** Effective rank of the cumulative weight update (top) and absolute cosine similarity between the single-step updates of the first and last tasks (bottom). Dashed lines denote layer-wise means. **Right:** Pairwise CKA similarity heatmaps across the 10 model checkpoints.

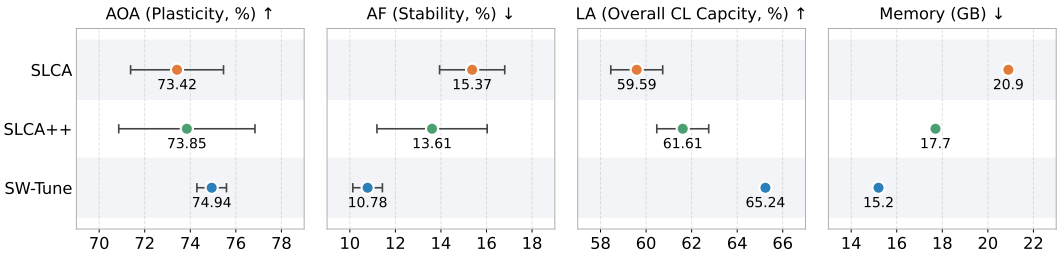


Figure 7: AOA, AF, LA, and training-time GPU memory consumption of SW-Tune and SLCA variants, averaged over five runs with different class orders.

SLCA++ (Zhang et al., 2024) exhibit limited plasticity (AOA of 73.42% and 73.85%) while still suffering considerable forgetting (AF of 15.37% and 13.61%), yielding an LA of only 59.59% and 61.61%, respectively. In contrast, SW-Tune improves both axes simultaneously: with an AOA of 74.94% and an AF of 10.78%, it surpasses SLCA++ by 1.09 and 2.83 percentage points on the two metrics, respectively, resulting in an LA of 65.24% that exceeds SLCA++ by 3.63 points. Notably, these gains arise from concurrent improvements in both plasticity and stability, indicating that SW-Tune expands the stability–plasticity Pareto frontier beyond what learning-rate adjustment alone can achieve. As an additional practical advantage, SW-Tune is more memory-efficient during training, consuming 15.2 GB of GPU memory compared to 17.7 GB for SLCA++ and 20.9 GB for SLCA.

6 Conclusion

In this work, we revisited the prevalent freezing-based paradigm for CL with PTMs, which mitigates forgetting at the expense of plasticity and often introduces additional inference overhead. To overcome these limitations, we proposed SW-Tune, which fine-tunes the backbone without auxiliary modules by reshaping the optimization subspace of PTMs toward an effectively shallower and wider configuration. Extensive experiments demonstrate that SW-Tune substantially outperforms existing state-of-the-art methods while adding no inference overhead. We hope this work encourages further research on principled optimization strategies for CL with foundation models.

Limitations. Although SW-Tune introduces no inference overhead, it incurs a slightly higher training cost compared to several existing methods (Appendix A.4). Moreover, extending SW-Tune beyond classification to broader downstream tasks remains an important direction for future work.

AI USE STATEMENT

In this work, generative AI tools were utilized exclusively for assisting with initial language proofreading and LaTeX template formatting. All core conceptual ideas, theoretical derivations, algorithm designs, code implementations, and experimental formulations were independently conceived, verified, and authored by the researchers. The authors take full responsibility for the entirety of the content, claims, and formulations presented in this paper.

ETHICS STATEMENT

This research focuses on foundational machine learning algorithms. All datasets used are standard, publicly available academic benchmarks. The proposed method does not generate harmful content, involve sensitive personal data, or introduce societal bias.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, comprehensive implementation details are provided in Sections 4 and 5. Code is available at `https://anonymous.4open.science/r/SW-Tune`.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Shivaji Alaparthi and Manit Mishra. Bidirectional encoder representations from transformers (bert): A sentiment analysis odyssey. *arXiv preprint arXiv:2007.01127*, 2020.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Tao Feng, Mang Wang, and Hangjie Yuan. Overcoming catastrophic forgetting in incremental object detection via elastic response distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9427–9436, 2022.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8340–8349, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15262–15271, 2021b.

Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pp. 699–715. Springer, 2020.

Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark Hasegawa-Johnson, Sung Ju Hwang, and Chang D Yoo. Forget-free continual learning with winning subnetworks. In *International Conference on Machine Learning*, pp. 10734–10750. PMLR, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Tatsuya Konishi, Mori Kurokawa, Chihiro Ono, Zixuan Ke, Gyuhak Kim, and Bing Liu. Parameter-level soft-masking for continual learning. In *International Conference on Machine Learning*, pp. 17492–17505. PMLR, 2023.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 12245–12254, 2020.

Aojun Lu, Tao Feng, Hangjie Yuan, Xiaotian Song, and Yanan Sun. Revisiting neural networks for continual learning: An architectural perspective. In *IJCAI*, pp. 4651–4659, 2024.

Aojun Lu, Tao Feng, Hangjie Yuan, Chunhui Ding, and Yanan Sun. Adapt before continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24061–24069, 2026.

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Wide neural networks forget less catastrophically. In *International conference on machine learning*, pp. 15699–15717. PMLR, 2022a.

Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning. *arXiv preprint arXiv:2202.00275*, 2022b.

Quang Pham, Chenghao Liu, and HOI Steven. Continual normalization: Rethinking batch normalization for online continual learning. In *International Conference on Learning Representations*, 2022.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *2023 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11909–11919. IEEE, 2023.

Hai-Long Sun, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Pilot: A pre-trained model-based continual learning toolbox. *arXiv preprint arXiv:2309.07117*, 2023.

Hai-Long Sun, Da-Wei Zhou, Hanbin Zhao, Le Gan, De-Chuan Zhan, and Han-Jia Ye. Mos: Model surgery for pre-trained model-based class-incremental learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 20699–20707, 2025.

Yuwen Tan, Qinhao Zhou, Xiang Xiang, Ke Wang, Yuchuan Wu, and Yongbin Li. Semantically-shifted incremental adapter-tuning is a continual vitransformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23252–23262, 2024.

Vishal Thengane, Salman Khan, Munawar Hayat, and Fahad Khan. Clip model is an efficient continual learner. *arXiv:2210.03114*, 2022.

Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

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

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

Yan Wang, Da-Wei Zhou, and Han-Jia Ye. Integrating task-specific and universal adapters for pre-trained model-based class-incremental learning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 806–816. IEEE, 2025.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pp. 631–648. Springer, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 139–149, 2022b.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3014–3023, 2021.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19091–19101. IEEE, 2023.

Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca++: Unleash the power of sequential fine-tuning for continual learning with pre-training. *arXiv preprint arXiv:2408.08295*, 2024.

Yuanhan Zhang, Zhenfei Yin, Jing Shao, and Ziwei Liu. Benchmarking omni-vision representation through the lens of visual realms. In *European Conference on Computer Vision*, pp. 594–611. Springer, 2022.

Hanbin Zhao, Hui Wang, Yongjian Fu, Fei Wu, and Xi Li. Memory-efficient class-incremental learning for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5966–5977, 2021.

Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *ICLR*, 2023.

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, pp. 1–21, 2024a.

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

A APPENDIX

A.1 EXPERIMENT DETAILS

We use seeds 1993, 1994, 1995, 1996, and 1997 to determine the class order for the five runs; for single-run experiments, we use seed 1993. Further details can be found in the code (`https://anonymous.4open.science/r/SW-Tune`). All experiments are conducted on a server equipped with eight NVIDIA RTX 3090 GPUs and two Intel Xeon Platinum 8160 CPUs, and each experiment is run on a single GPU.

A.2 JUSTIFICATION OF BENCHMARK SELECTION

To further motivate our choice of evaluation benchmarks, we examine the zero-shot performance of the pre-trained backbone across a diverse set of datasets. Specifically, we report the performance of ViT-B/16-IN21K on seven widely used benchmarks using SimpleCIL (Zhou et al., 2024a), which relies entirely on the frozen pre-trained features without any learning process. The results are summarized in Table 3.

Table 3: Zero-shot performance of ViT-B/16-IN21K via SimpleCIL across multiple datasets. All results are sourced from (Zhou et al., 2024a).

	CIFAR100	CUB	OmniBench	VTAB	ImageNet-R	ImageNet-A	ObjectNet
LA (%)	81.26	86.73	73.15	84.38	53.59	54.55	49.44

As shown, the PTM achieves strong zero-shot accuracy on datasets whose visual distributions closely align with ImageNet-21K, such as CIFAR100 (Krizhevsky et al., 2009) (81.26%), CUB (Wah et al., 2011) (86.73%), OmniBench (Zhang et al., 2022) (73.15%), and VTAB (Zhai et al., 2019) (84.38%). In these cases, the pre-trained representations already generalize well to the downstream tasks, leaving limited room for CL to provide additional gains. In contrast, the performance drops substantially on ImageNet-R (Hendrycks et al., 2021a) (53.59%), ImageNet-A (Hendrycks et al., 2021b) (54.55%), and ObjectNet (Barbu et al., 2019) (49.44%), all of which exhibit notable domain shifts from the standard ImageNet-21K pre-training data. These degraded results indicate that the frozen pre-trained features fail to capture the distributional characteristics of these datasets, creating a genuine need for the model to acquire new knowledge through CL.

A.3 EFFICIENT COMPUTATION OF GRADIENT SPECTRAL FLATTENING

We derive the GSF operator used in the main text and describe how to compute it efficiently from the eigendecomposition of a Gram matrix, avoiding an explicit singular-value decomposition of the gradient. The key observation is that the singular vectors of $\boldsymbol{G}$ are eigenvectors of the Gram matrices $\boldsymbol{G}^\top\boldsymbol{G}$ or $\boldsymbol{G}\boldsymbol{G}^\top$, and the corresponding eigenvalues are the squared singular values of $\boldsymbol{G}$.

For a gradient matrix $\boldsymbol{G} \in \mathbb{R}^{m\times n}$, let $d = \min(m, n)$, and write its reduced singular value decomposition as

$$\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \tag{6}$$

where $\boldsymbol{U} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times d}$ have orthonormal columns, and $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_d)$ with $\sigma_1 \geq \cdots \geq \sigma_d \geq 0$.

Tall case ($m \geq n$). Here $d = n$. We form the Gram matrix $\boldsymbol{A} = \boldsymbol{G}^\top\boldsymbol{G} \in \mathbb{R}^{n\times n}$, whose eigendecomposition is

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top, \tag{7}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ and $\lambda_i = \sigma_i^2$. For a given power $p \in [0, 1]$, we define the flattening operator as

$$\boldsymbol{M} = \boldsymbol{V}\boldsymbol{\Lambda}^{\frac{p-1}{2}}\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{\Sigma}^{p-1}\boldsymbol{V}^\top. \tag{8}$$

Right-multiplying $\boldsymbol{G}$ by $\boldsymbol{M}$ gives

$$\widetilde{\boldsymbol{G}} = \boldsymbol{G}\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}^{p-1}\boldsymbol{V}^\top = \boldsymbol{U}\boldsymbol{\Sigma}^p\boldsymbol{V}^\top. \tag{9}$$

This is exactly the singular-value power transformation in Eq. equation 4.

Wide case ($m < n$). Now $d = m$. We form the Gram matrix $\boldsymbol{A} = \boldsymbol{G}\boldsymbol{G}^\top \in \mathbb{R}^{m\times m}$, whose eigendecomposition is

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top, \tag{10}$$

where the eigenvectors are the left singular vectors $\boldsymbol{U}$, and the eigenvalues are again $\lambda_i = \sigma_i^2$. The flattening operator is

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{p-1}{2}}\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{\Sigma}^{p-1}\boldsymbol{U}^\top. \tag{11}$$

Left-multiplying $\boldsymbol{G}$ by $\boldsymbol{M}$ gives

$$\widetilde{\boldsymbol{G}} = \boldsymbol{M}\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}^{p-1}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{U}\boldsymbol{\Sigma}^{p}\boldsymbol{V}^\top. \tag{12}$$

Thus, this case also implements the same transformation $\sigma_i \mapsto \sigma_i^p$.

Therefore, both cases implement the singular-value power transformation without explicitly computing the full SVD of $\boldsymbol{G}$. In practice, we compute $\boldsymbol{M}$ from the eigendecomposition of the smaller Gram matrix, namely $\boldsymbol{G}^\top\boldsymbol{G}$ if $m \geq n$ and $\boldsymbol{G}\boldsymbol{G}^\top$ otherwise. Before raising the eigenvalues to the power $(p-1)/2$, we clamp them to a small positive constant to avoid numerical instability caused by zero or near-zero eigenvalues. The resulting matrix $\boldsymbol{M}$ is applied to $\boldsymbol{G}$ from the appropriate side, and the output is rescaled to preserve the original Frobenius norm as described in Eq. equation 5, with a small stability constant ϵ_{norm}.

A.4 Training Overhead

To evaluate the training overhead of SW-Tune, we compare the total training time and peak GPU memory consumption of different methods on ImageNet-A Inc20 under the same experimental setting. As shown in Table 4, SW-Tune incurs a higher training cost than several freezing-based baselines, as it updates the backbone under the proposed shallow-wide optimization scheme, whereas those methods only train lightweight auxiliary modules. Nevertheless, its time and memory costs remain comparable to other fine-tuning-based methods such as SLCA and SLCA++. Importantly, this additional cost is confined to training; SW-Tune introduces no extra computation at inference time, making it particularly suitable for scenarios where inference efficiency is critical.

Table 4: Training overhead on ImageNet-A Inc20. Lower is better for both metrics.

Metrics	L2P	DualP	CODA-P	SSIAT	MOS	TUNA	SLCA	SLCA++	SW-Tune
Time (s) ↓	765	719	785	1302	842	1131	1416	1445	1413
Memory (GB) ↓	16.0	13.2	19.5	15.2	12.0	15.2	20.9	17.7	15.2