MOS: Model Surgery for Pre-Trained Model-Based Class-Incremental Learning

Hai-Long Sun[1, 2], **Da-Wei Zhou**[1, 2*], **Hanbin Zhao**[3], **Le Gan**[1, 2], **De-Chuan Zhan**[1, 2], **Han-Jia Ye**[1, 2*]

[1]School of Artificial Intelligence, Nanjing University
[2]National Key Laboratory for Novel Software Technology, Nanjing University
[3]College of Computer Science and Technology, Zhejiang University
{sunhl, zhoudw, zhandc, yehj}@lamda.nju.edu.cn, ganle@nju.edu.cn, zhaohanbin@zju.edu.cn

Abstract

Class-Incremental Learning (CIL) requires models to continually acquire knowledge of new classes without forgetting old ones. Despite Pre-trained Models (PTMs) have shown excellent performance in CIL, catastrophic forgetting still occurs as the model learns new concepts. Existing work seeks to utilize lightweight components to adjust the PTM, while the forgetting phenomenon still comes from *parameter and retrieval* levels. Specifically, iterative updates of the model result in parameter drift, while mistakenly retrieving irrelevant modules leads to the mismatch during inference. To this end, we propose MOdel Surgery (MOS) to rescue the model from forgetting previous knowledge. By training task-specific adapters, we continually adjust the PTM to downstream tasks. To mitigate parameter-level forgetting, we present an adapter merging approach to learn task-specific adapters, which aims to bridge the gap between different components while reserve task-specific information. Besides, to address retrieval-level forgetting, we introduce a training-free self-refined adapter retrieval mechanism during inference, which leverages the model's inherent ability for better adapter retrieval. By jointly rectifying the model with those steps, MOS can robustly resist catastrophic forgetting in the learning process. Extensive experiments on seven benchmark datasets validate MOS's state-of-the-art performance.

Code — https://github.com/sun-hailong/AAAI25-MOS

Introduction

In recent years, deep learning has achieved significant results in many real-world applications (Deng et al. 2009; He et al. 2015; Cao et al. 2024b; Sun et al. 2024). While in the open world, data often appears in a streaming format (Golab and Özsu 2003), requiring a machine learning paradigm capable of incrementally acquiring new class knowledge, which is denoted as Class-Incremental Learning (CIL) (Rebuffi et al. 2017; Zhou et al. 2024b). One of the significant challenges in CIL is catastrophic forgetting, where the model, after learning new classes incrementally, gradually loses its ability to recognize the old ones (French 1999). In response to this challenge, the field of CIL is evolving with the emergence of pre-trained models (PTMs). Unlike the traditional approach of "training from scratch" (Li and Hoiem 2017; Zhou et al. 2023; Ye et al. 2019), contemporary CIL methods are increasingly leveraging PTMs, which are initially pre-trained on vast datasets using substantial resources (McDonnell et al. 2024; Jung et al. 2023). This pre-training process endows PTMs with robust generalization abilities. Consequently, designing an effective CIL method that leverages PTMs and resists catastrophic forgetting has garnered significant attention from researchers.

Due to the generalization of PTMs, existing works often freeze the pre-trained weights and adapt to incremental tasks using additional lightweight modules (Hu et al. 2022; Chao et al. 2020; Ye, Lu, and Zhan 2022). For example, visual prompt tuning (Jia et al. 2022) customizes prompts to modify model behavior, facilitating adaptation to downstream tasks. Specifically, L2P (Wang et al. 2022c) designs a key-query matching strategy to retrieve instance-specific prompts from a prompt pool. Based on L2P, DualPrompt (Wang et al. 2022b) introduces expert prompts to encode task-specific information and explores the impact of prompt depth. Furthermore, CODA-Prompt (Smith et al. 2023) proposes an attention-based weighting method for prompts to enhance the efficacy of prompt retrieval.

However, as the model learns new concepts, catastrophic forgetting still occurs. This forgetting phenomenon happens at both the *parameter and retrieval levels*. During the training stage, although many methods use lightweight components to adjust the PTM, iterative updates of these components will lead to parameter drift and trigger forgetting. Moreover, existing works devote to preventing conflicts between prompts or achieving orthogonal projection, which exacerbates parameter drift between new and old components. During inference, training multiple lightweight modules requires selecting the most relevant one, but the model may mistakenly retrieve the irrelevant modules, leading to the performance decay. This motivates us to question if it is possible to *jointly rectify the model to resist catastrophic forgetting at both the parameter and retrieval levels?*

Facing the challenges at both the parameter and retrieval levels, our model should be able to effectively design mechanisms to overcome these issues. To address forgetting at the parameter level, the model needs to develop effective update methods that ensure the updated parameters remain discriminative for old data. To overcome forgetting at the retrieval level, the model requires efficient self-correction

*Corresponding author.

strategies to help utilize relevant information, assisting in the instance-specific retrieval of lightweight modules.

To this end, we propose MOdel Surgery (MOS) for pre-trained model-based class-incremental learning to rescue the model from forgetting previous knowledge. This surgery is divided into the training and inference stages. To mitigate parameter-level forgetting, we present an *adapter merging* approach during training, which learns task-specific adapters while bridging gaps between components and retaining task-specific information. This strategy helps previously learned adapters aid in learning new tasks. To address retrieval-level forgetting, we introduce a training-free *self-refined adapter retrieval mechanism* during inference, which leverages the model's inherent ability for better adapter retrieval. This mechanism requires no additional training overhead, making the algorithm simple and efficient. Finally, to enable the model to balance the stability-plasticity dilemma, we present a model ensemble method that integrates the model's capabilities across multiple phases. It not only ensures strong generalization but also allows the model to quickly recognize and update information. Experiments on seven benchmark datasets validate the effectiveness of MOS. Additionally, the visualization of the self-refined adapter retrieval mechanism indicates that MOS effectively learns adapter retrieval for various downstream tasks.

Related Work

Class-Incremental Learning (CIL). It aims to enable models to acquire new classes knowledge while retaining previously learned information (Rebuffi et al. 2017). Existing works can be roughly categorized into several categories. Knowledge distillation-based methods (Li and Hoiem 2017; Rebuffi et al. 2017; Snell, Swersky, and Zemel 2017) establish a mapping between the former stage model and the current model, thereby aiding the latter in retaining characteristics from earlier updates during incremental learning (Hinton, Vinyals, and Dean 2015). Data rehearsal-based methods (Chaudhry et al. 2018; Liu et al. 2020; Zhao et al. 2021) select and replay crucial exemplars from old classes during training new ones to continuously revise former knowledge. Parameter regularization-based methods (Aljundi, Kelchtermans, and Tuytelaars 2019; Kirkpatrick et al. 2017) aim to predict and minimize the drift of key parameters by using regularization terms. Model rectification-based methods (Pham, Liu, and Steven 2022; Shi et al. 2022; Yu et al. 2020) focus on correcting the model's inductive bias to ensure unbiased estimations. Model expansion-based methods (Chen and Chang 2023; Hu et al. 2023; Wang et al. 2022a; Yan, Xie, and He 2021) construct non-interfering subnetworks for each task. During inference, they are combined to form a larger feature map and train a classifier to effectively calibrate across all classes.

Pre-Trained Model-Based CIL. PTM-based CIL has emerged as a hot topic in the current CIL research area. With advances in pre-training techniques, numerous parameter-efficient fine-tuning (PEFT) methods (Jia et al. 2022; Hu et al. 2022; Lian et al. 2022; Rebuffi, Bilen, and Vedaldi 2017; Cao et al. 2024a; Hu et al. 2024; Lu et al. 2024; Li et al. 2024; Wei et al. 2019) have been developed. These methods aim to improve model performance with minimal additional resources while freezing pre-trained weights. In this context, L2P (Wang et al. 2022c) introduces a prompt pool, selecting instance-specific prompts via a key-query matching selection mechanism to guide the PTM's response. DualPrompt (Wang et al. 2022b) extends L2P by designing G-Prompt and E-Prompt, which encode task-invariant and task-specific instructions, respectively. CODA-Prompt (Smith et al. 2023) innovates by developing decomposed prompts and combining them using an attention-based weighting method. DAP (Jung et al. 2023) extends prompt selection into prompt generation. SLCA (Zhang et al. 2023) reveals that fine-tuning a ViT backbone with a lower learning rate at the representation layer yields higher accuracy than prompt strategies. APER (Zhou et al. 2024a) explores various PEFT methods and shows that prototypical classifiers serve as a strong baseline, and RanPAC (McDonnell et al. 2024) further expands APER in random projection. EASE (Zhou et al. 2024c) concatenates the feature representations of multiple task-specific backbones.

Preliminaries

Class-Incremental Learning

Class-incremental learning aims to acquire knowledge from continuously evolving data streams that introduce new classes while retaining knowledge of previous ones to build a unified classifier (Rebuffi et al. 2017). Consider a series of B training stages, expressed as $\{\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^B\}$, where $\mathcal{D}^b = \{(\mathbf{x}_i^b, y_i^b)\}_{i=1}^{n_b}$ represents the b-th incremental stage containing n_b instances. Correspondingly, the testing set is denoted as $\{\mathcal{D}_t^1, \mathcal{D}_t^2, \cdots, \mathcal{D}_t^B\}$. Within this setting, each training instance $\mathbf{x}_i^b \in \mathbb{R}^D$ is associated with a class $y_i \in Y_b$. Here, Y_b defines the set of labels for task b, and it is ensured that $Y_b \cap Y_{b'} = \varnothing$ for any $b \neq b'$. During b-th training stage, the model is updated utilizing data exclusively from $\mathcal{D}^b$. In this paper, we follow the **exemplar-free setting** in (Wang et al. 2022c,b; Zhou et al. 2024a), which entails not using any historical exemplars from previous classes. Therefore, the model can only access data from $\mathcal{D}^b$ for training during the b-th stage. The effectiveness of the model is evaluated across all previously encountered classes, collectively represented as $\mathcal{Y}_b = Y_1 \cup \cdots \cup Y_b$, after each CIL task. Specifically, we aim to find a model $f(\mathbf{x}) : X \to \mathcal{Y}_b$ that minimizes empirical risk across all test datasets:

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \, \mathbb{E}_{(\mathbf{x},y)\sim \mathcal{D}_t^1 \cup \cdots \mathcal{D}_t^b} \mathbb{I}\left(y \neq f(\mathbf{x})\right), \qquad (1)$$

where $\mathcal{H}$ is the hypothesis space and $\mathbb{I}(\cdot)$ denotes the indicator function. $\mathcal{D}_t^b$ represents the testing set of task b. An effective CIL model satisfying Eq. 1 exhibits discriminative abilities across all classes. It achieves a balance between learning new classes and retaining information about old ones.

Following the typical PTM-based CIL works (Wang et al. 2022c,b), we assume that a PTM (*e.g.*, Vision Transformer (ViT) (Dosovitskiy et al. 2020)) is available as the initialization for $f(\mathbf{x})$. For clearer understanding, we decouple the PTM into two components: $f(\mathbf{x}) = W^\top \phi(\mathbf{x})$, where $\phi(\cdot) : \mathbb{R}^D \to \mathbb{R}^d$ is the feature extractor and $W \in \mathbb{R}^{d \times |\mathcal{Y}_b|}$

is the classifier. We denote the classifier for class k as $\boldsymbol{w}_k$: $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_{|\mathcal{Y}_b|}]$. For a standard ViT, the initial encoding layer converts the image into a sequence of output features, denoted as $\mathbf{x}_e \in \mathbb{R}^{L\times d}$, where L is the sequence length. We simplify this by treating the first token in $\mathbf{x}_e$ to be the `[CLS]` token. The sequence $\mathbf{x}_e$ is then processed through subsequent layers, including multi-head self-attention and MLP, to produce the final embeddings. Finally, the embedded `[CLS]` token is considered as $\phi(\mathbf{x})$.

Analysis of PTM-Based CIL

Learning with PTMs. A representative work in PTM-based CIL is L2P (Wang et al. 2022c). They introduce a strategy of freezing the pre-trained weights and constructing a learnable prompt pool that can be shared across all tasks. This prompt pool is denoted as $\mathbf{P} = \{P_1, P_2, \cdots, P_M\}$, where $P_j \in \mathbb{R}^{L_p\times d}$ is a single prompt with token length L_p and the same embedding size d as $\mathbf{x}_e$. M is the size of the prompt pool. Each prompt in this pool corresponds to a specific key $\{(\boldsymbol{k}_1, P_1), (\boldsymbol{k}_2, P_2), \cdots, (\boldsymbol{k}_M, P_M)\}$, where $\boldsymbol{k}_i \in \mathbb{R}^{d_k}$. First, they utilize a PTM without prompting (*i.e.*, $\phi(\cdot)$) to encode the features into the key's embedding space and retrieve prompts with similar keys. During inference, given an input $\mathbf{x}$, the model employs $\phi(\mathbf{x})$ to look up the top-N keys by solving the objective in Eq. 2. This process retrieves the most relevant keys and their corresponding prompts from the prompt pool.

$$\mathbf{K}_{\mathbf{x}} = \underset{\{s_i\}_{i=1}^{N}\subseteq[1,M]}{\operatorname{argmin}} \sum_{i=1}^{N} \gamma\left(\phi(\mathbf{x}), \boldsymbol{k}_{s_i}\right), \tag{2}$$

where $\mathbf{K}$ is the set of all keys and $\mathbf{K}_{\mathbf{x}}$ is the selected top-N keys. $\gamma(\cdot,\cdot)$ denotes the cosine distance. Finally, L2P minimize the end-to-end training loss function:

$$\min_{\mathbf{P},\mathbf{K},\phi} \ell(W^{\top}\phi(\mathbf{x};\mathbf{P}), y) + \lambda \sum_{\mathbf{K}_{\mathbf{x}}} \gamma\left(\phi(\mathbf{x}), \boldsymbol{k}_{s_i}\right), \tag{3}$$

where $\ell(\cdot,\cdot)$ is the cross-entropy loss that measures the discrepancy between prediction and ground truth. λ is a scalar to weight the loss. Optimizing Eq. 3 enhances the PTM's ability to incorporate task-specific information, allowing it to adapt more effectively to evolving data instances.

Forgetting of parameter and retrieval levels. L2P continually updates prompts and retrieves instance-specific prompts to guide the PTM's response. However, although the model learns new concepts, catastrophic forgetting still occurs at the *parameter and retrieval* levels. Specifically, Eq. 3 shows how L2P uses lightweight modules to adjust the PTM to downstream tasks. As the prompts are iteratively updated, they gradually adapt to the subsequent tasks, leading to parameter drift. On the other hand, training multiple lightweight modules requires selecting the most relevant one during inference, while the model may mistakenly retrieve the irrelevant modules, leading to the performance decay. The mistaken retrieval comes from three aspects: First, modules learned in previous tasks might be re-selected for new tasks, causing confusion between the retrieval of old and new modules. Besides, since the keys for subsequent tasks do not exist during current training, a gap may arise between the keys and the feature embeddings, leading to mistaken retrieval during inference. Therefore, it is essential to design a method to jointly rectify the model to resist catastrophic forgetting at both the parameter and retrieval levels.

MOS: Model Surgery for PTM-based CIL

Facing the challenge of resisting catastrophic forgetting, we need a method to jointly rectify the model. The key idea of MOS is to design *model surgery* in two aspects, *i.e.*, training stage surgery that mitigates parameter drift and testing stage surgery that retrieves better lightweight modules. Training stage surgery aims to use previously learned knowledge to improve performance on current tasks, allowing the model to adapt to new tasks more quickly. Testing stage surgery seeks to find a mechanism for better adapter retrieval without additional overhead. As a result, the model can benefit from continual lightweight module updates and effective retrieval ability without forgetting existing knowledge.

We first introduce the process of progressively merged adapters for mitigating parameter drift and then discuss the self-refined adapter retrieval mechanism. We summarize the inference function with pseudo-code in the last part.

Progressively Merged Adapters

To handle the parameter drift caused by the iterative updates of the model, we need to bridge the gap between different lightweight modules. In other words, as the model continually receives new data and tasks, it is crucial to effectively retain and utilize previously learned knowledge. This approach allows the model to transfer prior knowledge to new tasks and mitigates the parameter drift problem. In Eq. 3, the embedding of a given input $\mathbf{x}$ is obtained using instance-specific prompts. During the incremental phase, a potential problem can emerge, *i.e.*, iterative updates to existing prompts might cause them to better match new tasks, possibly resulting in forgetting older tasks.

Due to the large prompt pool in the above methods, which exacerbates mistaken retrieval, we suggest mitigating this problem by using a smaller number of lightweight modules. In detail, by directly incorporating adapter tuning (Rebuffi, Bilen, and Vedaldi 2017) into the PTM to optimize a single adapter for encoding task-specific information, we achieve this goal through the application of this method. This enhanced integration allows facilitates a more effective assimilation of task-specific information. With this approach, we only need to optimize a collection of adapters to encode task-specific information. Denote that there are L transformer blocks in the pre-trained model, each with a self-attention module and an MLP layer. We integrate an adapter into each layer's MLP via residual connections. An adapter is a bottleneck module comprising a down-projection layer $W_{down} \in \mathbb{R}^{d\times r}$, a non-linear activation function ReLU, and an up-projection layer $W_{up} \in \mathbb{R}^{d\times r}$. The output formula of the MLP is formatted as follows:

$$\mathbf{x}_o = \mathrm{MLP}(\mathbf{x}_i) + \mathrm{ReLU}(\mathbf{x}_i W_{down}) W_{up}, \tag{4}$$

where $\mathbf{x}_i$ and $\mathbf{x}_o$ are the input and output of the MLP, respectively. Eq. 4 illustrates how to enhance the task information

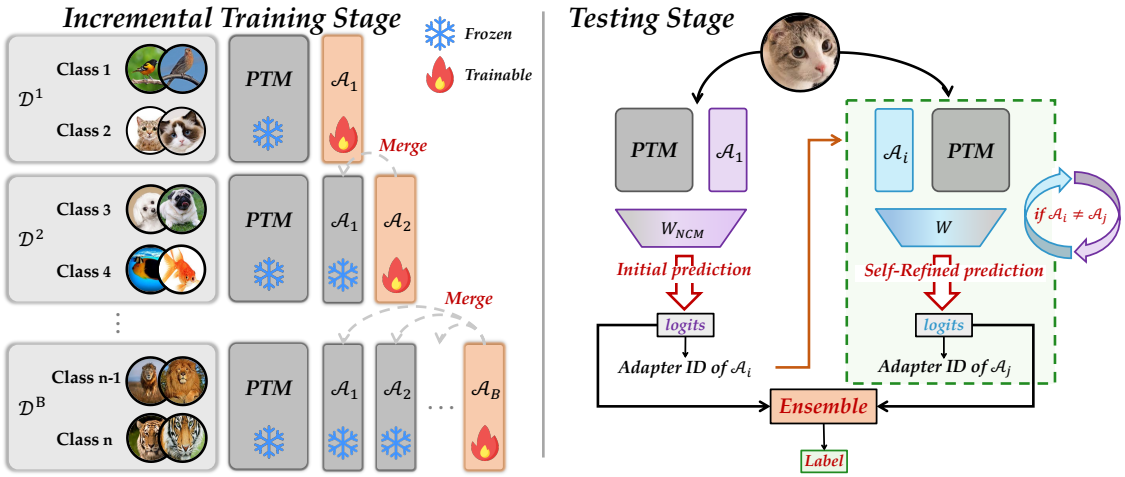


Figure 1: Illustration of MOS. **Left**: the training protocol of MOS. We use progressively merged adapters to incrementally adapt the PTM. **Right**: the self-refined adapter retrieval mechanism for the testing stage. We use the model's own capabilities to correct errors caused by the mistaken retrieval problem.

by adding residual connections of adapters to the original outputs. In the context of ViT and for a specific i-th task, we define the set of adapters across all L transformer blocks as $\mathcal{A}_i$, representing task-specific adapters. Furthermore, we denote the output embedding of a given $\mathcal{A}_i$, combined with the PTM, as $\phi(\mathbf{x};\mathcal{A}_i)$. Therefore, when a new task emerges, we freeze the weights of the PTM and focus solely on optimizing the adapters and the corresponding classifier W:

$$\min_{\mathcal{A}_i,W} \sum_{(\mathbf{x},y)\in\mathcal{D}^b} \ell\left(W^\top \phi\left(\mathbf{x};\mathcal{A}_i\right),y\right). \tag{5}$$

We enable the incorporation of task-specific information into embeddings through adapters by optimizing Eq. 5, facilitating the learning of new tasks. In an ideal scenario, if the task ID of each test sample is known, we can easily select the corresponding task-specific adapter using this ID to achieve optimal results.

However, in the CIL setting, obtaining such a task ID during the testing phase is forbidden. To address this challenge and mitigate parameter drift, we propose the training stage surgery which uses adapter merging strategy based on Exponential Moving Average (EMA) in Eq. 6. This approach allows subsequent adapters to retain some knowledge of their predecessors, ensuring satisfactory results even if an incorrect $\mathcal{A}$ is selected.

$$\mathcal{A}_b = (1-\alpha)\hat{\mathcal{A}}_b + \frac{\alpha}{b-1}\sum_{k=1}^{b-1}\mathcal{A}_k, \tag{6}$$

where $\hat{\mathcal{A}}_b$ represents the set of adapters for the b-th training stage and $\mathcal{A}_b$ is the final result after the EMA process. Specifically, given an adapter comprises W_{up} and W_{down}, we perform the merge process on both of them to facilitate the integration of adapters. When training a new $\mathcal{A}_b$, all previously trained $\mathcal{A}_k$ are frozen, and the adapter merging process is executed following each backpropagation step.

Effect of adapter merging strategy. Figure 1 (left) depicts this merging process. This strategy ensures that the training of the current adapter $\mathcal{A}_b$ does not interfere with the performance of already trained adapters, thereby preventing catastrophic forgetting. Moreover, it guarantees that each $\mathcal{A}$ retains task-specific information while maintaining remaining well-aligned in the feature space, even if an incorrect $\mathcal{A}$ is selected. In this way, we can mitigate parameter drift during iterative adapter updates. Moreover, because adapters are lightweight branches, they require significantly fewer parameters compared to fully fine-tuning the backbone. The parameter cost for saving these adapters is calculated as $(B \times L \times 2dr)$, where B denotes the number of tasks, L is the number of transformer blocks, and $2dr$ signifies the parameter count of each adapter (*i.e.*, linear projections).

Self-Refined Adapter Retrieval Mechanism

After obtaining these task-specific adapters, we utilize a prototype-based classifier (Snell, Swersky, and Zemel 2017) for prediction. Specifically, after the training process of each incremental stage, we extract the class prototype of the i-th class using adapter $\mathcal{A}_b$:

$$\boldsymbol{p}_{i,b} = \frac{1}{N}\sum_{j=1}^{|\mathcal{D}^b|}\mathbb{I}(y_j = i)\phi(\mathbf{x}_j;\mathcal{A}_b), \tag{7}$$

where N is the instance number of class i. Eq. 7 illustrates the construction of classifier. During inference, we directly adopt the class prototype as the classifier weight, *i.e.*, $\boldsymbol{w_i} = \boldsymbol{p_i}$, and utilize a cosine classifier for classification:

$$f(\mathbf{x}|\mathcal{A}_i) = \left(\frac{W}{\|W\|_2}\right)^\top \left(\frac{\phi(\mathbf{x};\mathcal{A}_i)}{\|\phi(\mathbf{x};\mathcal{A}_i)\|_2}\right), \tag{8}$$

where $\mathcal{A}_i$ denotes the selected adapter for the input $\mathbf{x}$.

Eq. 2 illustrates how prompts are selected from the prompt pool. Subsequently, L2P integrates the selected prompt into the original PTM (*i.e.*, $\phi(\mathbf{x};P)$) to guide the model's response. However, this approach heavily relies on the retrieval mechanism of key-query pairs. Mistakenly retrieving the irrelevant prompts often leads to performance decay. To address the retrieval-level issue, we design

Method	CIFAR B0 Inc5		CUB B0 Inc10		IN-R B0 Inc20		IN-A B0 Inc20		ObjNet B0 Inc10		OmniBench B0 Inc30		VTAB B0 Inc10	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	38.90	20.17	26.08	13.96	32.31	22.78	24.28	14.51	19.14	8.73	23.61	10.57	34.95	21.25
Finetune Adapter	60.51	49.32	66.84	52.99	58.17	52.39	45.41	41.10	50.22	35.95	62.32	50.53	48.91	45.12
LwF	46.29	41.07	48.97	32.03	45.72	34.17	37.75	26.84	33.01	20.65	47.14	33.95	40.48	27.54
L2P	85.94	79.93	67.05	56.25	75.46	69.77	49.39	41.71	63.78	52.19	73.36	64.69	77.11	77.10
DualPrompt	87.87	81.15	77.47	66.54	73.10	67.18	53.71	41.67	59.27	49.33	73.92	65.52	83.36	81.23
CODA-Prompt	89.11	81.96	84.00	73.37	77.97	72.27	53.54	42.73	66.07	53.29	77.03	68.09	83.90	83.02
SimpleCIL	87.57	81.26	92.20	86.73	61.26	54.55	59.77	48.91	65.45	53.59	79.34	73.15	85.99	84.38
APER+ Finetune	87.67	81.27	91.82	86.39	68.54	58.37	61.01	49.57	61.41	48.34	73.02	65.03	87.47	80.44
APER+ VPT-S	90.43	84.57	92.02	86.51	68.83	62.03	58.39	47.20	64.54	52.53	79.63	73.68	87.15	85.36
APER+ VPT-D	88.46	82.17	91.02	84.99	77.05	69.47	58.48	48.52	67.83	54.65	81.05	74.47	86.59	83.06
APER+ SSF	87.78	81.98	91.72	86.13	75.47	67.02	61.30	50.03	69.15	56.64	80.53	74.00	85.66	81.92
APER+ Adapter	90.65	85.15	92.21	86.73	75.82	67.95	60.47	49.37	67.18	55.24	80.75	74.37	85.95	84.35
SLCA	92.49	88.55	89.51	82.19	81.17	77.00	68.66	58.74	72.55	61.30	82.80	74.10	90.94	90.76
EASE	91.51	85.80	92.23	86.81	81.74	76.17	65.34	55.04	70.84	57.86	81.11	74.85	**93.61**	**93.55**
MOS	**93.30**	**89.25**	**93.49**	**90.12**	**82.96**	**77.93**	**69.13**	**59.12**	**74.69**	**63.62**	**85.91**	**80.05**	92.62	92.79

Table 1: Average and last performance comparison on seven datasets with **ViT-B/16-IN21K** as the backbone. 'IN-R/A' stands for 'ImageNet-R/A,' 'ObjNet' stands for 'ObjectNet,' and 'OmniBench' stands for 'OmniBenchmark.' We report all compared methods with their source code. The best performance is shown in bold. All methods are implemented without using exemplars.

the testing stage surgery which uses self-refined adapter retrieval mechanism. It is an efficient and training-free method that enables the model to autonomously correct this problem, thereby improving adapter retrieval. This mechanism does not require any additional training overhead and is only used during the inference process, making the algorithm both simple and efficient.

Since there is a gap between the PTM and downstream datasets, we first use an adapter to fine-tune the PTM on the first incremental task, denoting the model as $f(\mathbf{x}; \mathcal{A}_1)$. This process effectively bridges this gap and makes the model suitable as the initial selector. During inference, we utilize $f(\mathbf{x}; \mathcal{A}_1)$ to obtain the embedding of each testing example and perform the initial retrieval of task-specific adapters. Specifically, given an input $\mathbf{x}$, we first obtain the prediction result $f(\mathbf{x}|\mathcal{A}_1)$ of the model through Eq. 8. Afterwards, we can easily infer its corresponding task ID i:

$$i = \text{argmax}\ (f(\mathbf{x}|\mathcal{A}_1))\ \text{mod}\ |Y_b|, \tag{9}$$

where Y_b is the number of classes for each task. Building on this result, we introduce an iterative self-refined process. As defined in Eq. 8, this process primarily uses $f(\mathbf{x}; \mathcal{A}_i)$ to obtain prediction and identify the task ID j. *Since each adapter is task-specific, we can determine whether to end the iteration by checking if* $i = j$. Specifically, through $f(\mathbf{x}|\mathcal{A}_i)$, we can infer its corresponding task ID j:

$$j = \text{argmax}\ (f(\mathbf{x}|\mathcal{A}_i))\ \text{mod}\ |Y_b|. \tag{10}$$

For example, in a scenario where each task comprises 10 classes, classes 0 through 9 are in the task 0, while classes 10 through 19 are in the task 1. Subsequently, if $i \neq j$, we replace i with j and repeat the process of Eq. 10 until $i = j$, ensuring the self-consistency.

Effect of self-refined adapter retrieval mechanism. Figure 1 (right) illustrates the self-refined process. Firstly, $\phi(\mathbf{x}; \mathcal{A}_1)$ with the prototype-based classifier bridges the gap between upstream and downstream datasets, enhancing the model's ability to generalize to new classes. Hence, we use it as the initial selector to start the self-refined iteration. Moreover, this approach is training-free and does not incur any additional training costs, ensuring the algorithm's efficiency. Due to the self-refined adapter retrieval mechanism allowing the model to verify the correctness of its initial predictions, we can easily check model consistency, thereby alleviating the aforementioned mistaken retrieval problem. By using this mechanism, MOS successfully corrects some images that were originally incorrectly predicted. The detailed visualization examples will be provided in experiments.

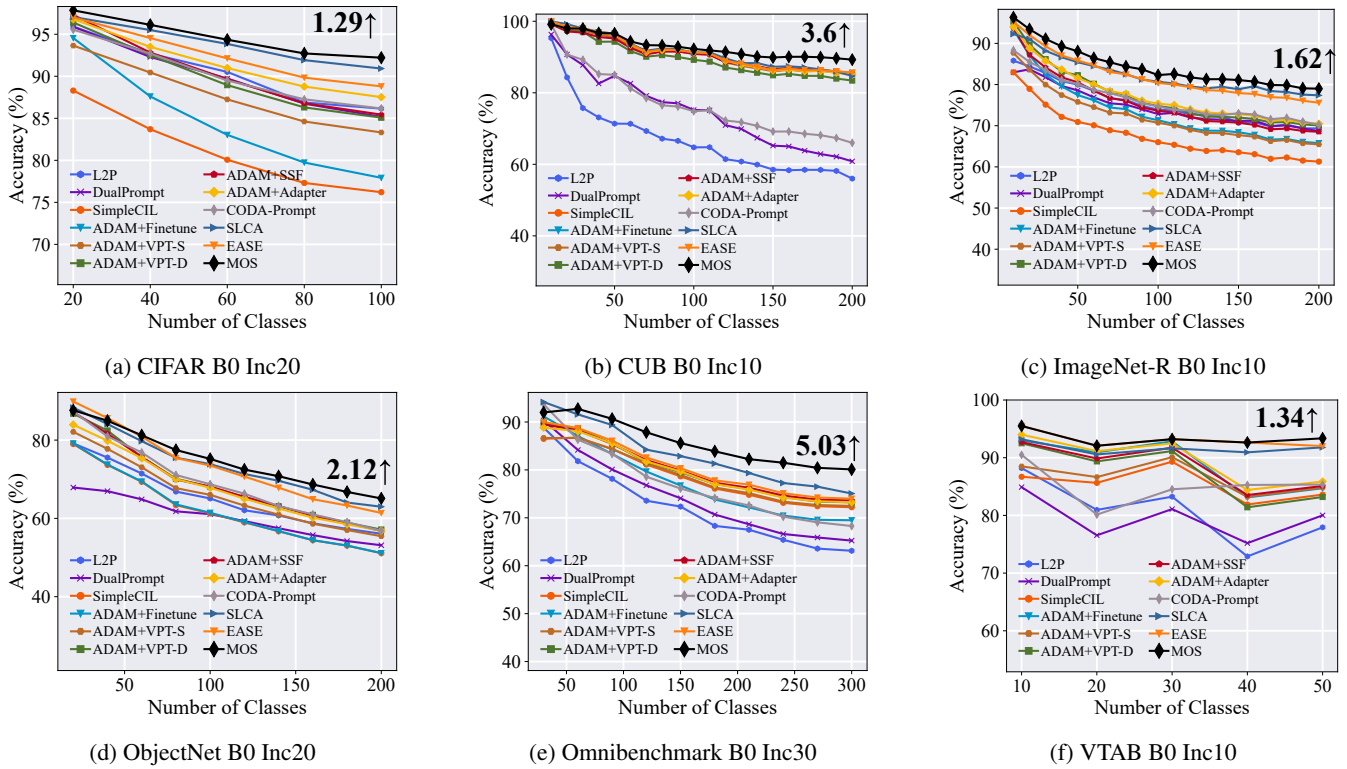
Multi-Stage Model Ensemble

Inspired by the *Complementary Learning System* of the human brain (McClelland, McNaughton, and O'Reilly 1995; Kumaran, Hassabis, and McClelland 2016), which suggests that the anterior cingulate circuit is responsible for rapid pattern recognition and unconscious memory, and the hippocampal circuit for deep processing and conscious memory. Therefore, we implement a two-stage model ensemble:

$$y^* = \underset{y}{\text{argmax}}\ (\underbrace{f(\mathbf{x}|\mathcal{A}_1)}_{\text{Part 1}} + \underbrace{f(\mathbf{x}|\mathcal{A}_j)}_{\text{Part 2}}). \tag{11}$$

In Eq. 11, Part 1, trained solely on the first incremental task, acts as a crucial bridge between the upstream and downstream datasets. It not only demonstrates strong generalization but also has the ability to quickly recognize and update information. In contrast, Part 2 employs progressively merged adapters and a self-refined adapter retrieval mechanism for deep processing and conscious memory.

Summary of MOS**.** We initialize and train an adapter for each incremental task to encode the task-specific information, and employ the adapter merging strategy to mitigate parameter drift phenomenon. Subsequently, we extract prototypes from the current dataset for the current adapter to complete the classifier head. During inference, we utilize the training-free self-refined adapter retrieval mechanism to correct the mistaken retrieval of irrelevant adapters. Finally, we implement a two-stage model ensemble to select the maximum logit.



(a) CIFAR B0 Inc20 (b) CUB B0 Inc10 (c) ImageNet-R B0 Inc10

(d) ObjectNet B0 Inc20 (e) Omnibenchmark B0 Inc30 (f) VTAB B0 Inc10

Figure 2: Performance curve of different methods under different settings. All methods are initialized with **ViT-B/16-IN1K**. We annotate the relative improvement of MOS above the runner-up method with numerical numbers at the last incremental stage.

Experiments

In this section, we evaluate MOS on seven benchmark datasets and compare it to other SOTA methods to demonstrate its superiority. Moreover, we provide an ablation study and a visualized analysis to validate the robustness of MOS.

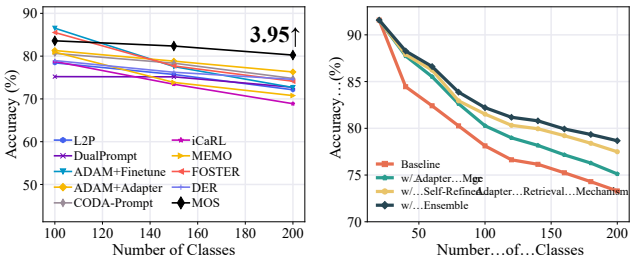
Implementation Details

Dataset: Since PTMs possess extensive knowledge regarding upstream tasks, we follow (Zhou et al. 2024a) to evaluate the performance on CIFAR100 (Krizhevsky, Hinton et al. 2009), CUB200 (Wah et al. 2011), ImageNet-R (Hendrycks et al. 2021a), ImageNet-A (Hendrycks et al. 2021b), objectNet (Barbu et al. 2019), Omnibenchmark (Zhang et al. 2022), and VTAB (Zhai et al. 2019). These datasets represent typical CIL benchmarks and include out-of-distribution datasets that exhibit a *significant domain gap* with ImageNet (*i.e.*, the pre-trained dataset). Specifically, There are 50 classes in VTAB, 100 classes in CIFAR100, 200 classes in CUB, ImageNet-R, ImageNet-A, ObjectNet, and 300 classes in OmniBenchmark. More details are reported in the supplementary.

Dataset split: Following the benchmark setting (Rebuffi et al. 2017; Wang et al. 2022c), we utilize the notation 'B-m Inc-n' to represent class splits, where m indicates the number of classes in the initial task, and n denotes the number of classes in each subsequent incremental task. $m = 0$ means the total classes are equally divided into each task. For a consistent and fair comparison, we randomly shuffle class orders using a random seed of 1993 before splitting the data. We ensure consistency in the training and testing sets across all methods, following (Zhou et al. 2024a).

Training details: We use PyTorch (Paszke et al. 2019) and PILOT (Sun et al. 2023) to implement all models on NVIDIA RTX 4090 with the *same* network backbone. Since the wide range of PTMs are publicly accessible (Wightman 2019), we choose two representative models following (Wang et al. 2022b; Zhou et al. 2024a), denoted as **ViT-B/16-IN1K** and **ViT-B/16-IN21K**. They are both initially pre-trained on ImageNet21K, while the former is further finetuned on ImageNet1K. In MOS, we set the batch size to 48 and train for 20 epochs using the SGD optimizer with momentum. The learning rate is initially set to 0.01 and follows a cosine annealing decay pattern. The projection dimension r in the adapter is set to 16, and the EMA factor parameter α is set to 0.1.

Comparison methods: We choose state-of-the-art PTM-based CIL methods for comparison, such as Finetune Adapter (Chen et al. 2022), L2P (Wang et al. 2022c), DualPrompt (Wang et al. 2022b), CODA-Prompt (Smith et al. 2023), SimpleCIL (Zhou et al. 2024a), APER (Zhou et al. 2024a), SLCA (Zhang et al. 2023), EASE (Zhou et al. 2024c). In addition, we compare MOS with traditional CIL methods modified by PTM, including LwF (Li and Hoiem 2017), FOSTER (Wang et al. 2022a), MEMO (Zhou et al. 2023), iCaRL (Rebuffi et al. 2017), DER (Yan, Xie, and He 2021). We report the baseline method, which sequentially finetunes the PTM, denoted as Finetune. All methods are implemented with the **same** PTM for a *fair* comparison.



(a) ImageNet-R B100 Inc50 (b) Ablation study

Figure 3: **Left:** Experimental results with large base classes. All methods are based on the same PTM (**ViT-B/16-IN1K**). **Right:** Ablation study of different components in MOS. We find each component within MOS enhances the performance.

Evaluation protocol: Following the benchmark established by (Rebuffi et al. 2017), we denote the Top-1 accuracy after the b-th stage as $\mathcal{A}_b$. Moreover, we use $\mathcal{A}_B$ (the performance after the last stage) and $\bar{\mathcal{A}} = \frac{1}{B}\sum_{b=1}^{B}\mathcal{A}_b$ (average performance along incremental stages) as measurements.

Benchmark Comparison

In this section, we compare MOS with other SOTA methods across seven datasets and various backbone weights. As detailed in Table 1, MOS shows the best performance across all seven benchmarks, significantly surpassing the SOTA methods, such as SLCA, EASE, and APER. Furthermore, we present an analysis of the incremental performance trend of different methods in Figure 2 with ViT-B/16-IN1K. Notably, MOS outperforms the runner-up method by 2%∼5% on CUB, ObjectNet, and OmniBenchmark, as highlighted in the annotations at the end of each image.

Beyond the B0 setting presented in Table 1 and Figure 2, we extend our experiments to a larger base setting. In Figure 3a, we compare MOS with several SOTA methods and traditional methods using the **same** PTM. Although traditional methods require storing exemplars to recover previous knowledge, MOS achieves SOTA performance in this setting as well.

Ablation Study

In this section, we conduct an ablation study by *incrementally adding each component* to evaluate their effectiveness within MOS. Specifically, we present this ablation study on ImageNet-R B0 Inc20 setting. As depicted in Figure 3b, **'Baseline'** refers to the PTM integrated with $\mathcal{A}_1$ (*i.e.*, $\phi(\mathbf{x}|\mathcal{A}_1)$). Since we aim to mitigate parameter drift and build task-specific adapters, we report **'w/ Adapter Merge'** by only using Eq. 6. Due to the mistaken retrieval issue, we propose using the model's inherent capabilities to correct errors. We report the performance of **'w/ Self-Refined Adapter Retrieval Mechanism'** by using this technique along with the adapter merging strategy. As shown in the figure, both the adapter merging strategy and self-refined adapter retrieval mechanism significantly improve the performance, which indicates MOS has the ability to correct itself and alleviate the catastrophic forgetting. Finally, we adjust the logits using Eq. 11 to trade off stability and plasticity, denoted as **'w/ Ensemble'**. Ablations verify that every component in MOS contributes to improving performance.

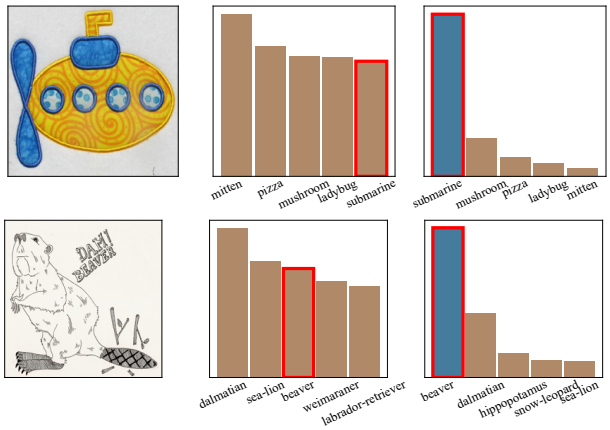


Figure 4: Visualizations of self-refined adapter retrieval mechanism on ImageNet-R. The original images are depicted in the first row, followed by the top-5 prediction probability before the self-refined process, and the probabilities after refinement in the last row. The ground-truth class is highlighted with red boxes.

Visualizations

In this section, we discuss how the self-refined adapter retrieval mechanism works. To illustrate this, we present the visualization of prediction results before and after the self-refined process and analyze their differences. We choose images from ImageNet-R and utilize the model trained under the B0 Inc20 setting. The results are shown in Figure 4. As shown in these figures, MOS is capable of rectifying incorrect predictions. This is evident even in the example below, where the initial top-5 class predictions do not include the ground truth, yet MOS accurately corrects this error. It demonstrates that the model, using its inherent capabilities, can select the most suitable adapter for the current sample. Hence, MOS can use this adapter to extract more suitable features, which aids in enhancing prediction accuracy. These visualizations reveal that the self-refined adapter retrieval mechanism can help to correct the outputs, thereby enhancing the attention of the ground-truth class.

Conclusion

Incremental learning is an increasingly prominent paradigm in real-world systems. This paper proposes a novel model surgery (MOS) for PTM-based CIL to rescue the model from forgetting previous knowledge. Specifically, we introduce an adapter merging method to mitigate parameter drift and design a training-free self-refined adapter retrieval mechanism for better adapter retrieval during inference. Our approach balances the stability-plasticity dilemma by leveraging the model's inherent capabilities, enhancing generalization and adaptability. Extensive experiments on seven benchmark datasets validate the effectiveness of MOS. In future work, we aim to explore further application scenarios, such as few-shot class-incremental learning.

Acknowledgments

This work is partially supported by Fundamental Research Funds for the Central Universities (2024300373, 14380021), Key Program of Jiangsu Science Foundation (BK20243012), CCF-Tencent Rhino-Bird Open Research Fund RAGR20240101, NSFC (62476123, 62376118, 62006112, 62250069, 61921006, 62402430), the AI & AI for Science Project of Nanjing University, Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

Aljundi, R.; Kelchtermans, K.; and Tuytelaars, T. 2019. Task-free continual learning. In *CVPR*, 11254–11263.

Barbu, A.; Mayo, D.; Alverio, J.; Luo, W.; Wang, C.; Gutfreund, D.; Tenenbaum, J.; and Katz, B. 2019. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *NeurIPS*, 32.

Cao, Q.; Xu, Z.; Chen, Y.; Ma, C.; and Yang, X. 2024a. Domain-controlled prompt learning. In *AAAI*, volume 38, 936–944.

Cao, Q.; Xu, Z.; Chen, Y.; Ma, C.; and Yang, X. 2024b. Domain prompt learning with quaternion networks. In *CVPR*, 26637–26646.

Chao, W.-L.; Ye, H.-J.; Zhan, D.-C.; Campbell, M.; and Weinberger, K. Q. 2020. Revisiting meta-learning as supervised learning. *arXiv preprint arXiv:2002.00573*.

Chaudhry, A.; Ranzato, M.; Rohrbach, M.; and Elhoseiny, M. 2018. Efficient Lifelong Learning with A-GEM. In *ICLR*.

Chen, S.; Chongjian, G.; Tong, Z.; Wang, J.; Song, Y.; Wang, J.; and Luo, P. 2022. AdaptFormer: Adapting Vision Transformers for Scalable Visual Recognition. In *NeurIPS*.

Chen, X.; and Chang, X. 2023. Dynamic Residual Classifier for Class Incremental Learning. In *ICCV*, 18743–18752.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

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

Golab, L.; and Özsu, M. T. 2003. Issues in data stream management. *ACM Sigmod Record*, 32(2): 5–14.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

Hendrycks, D.; Basart, S.; Mu, N.; Kadavath, S.; Wang, F.; Dorundo, E.; Desai, R.; Zhu, T.; Parajuli, S.; Guo, M.; et al. 2021a. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 8340–8349.

Hendrycks, D.; Zhao, K.; Basart, S.; Steinhardt, J.; and Song, D. 2021b. Natural adversarial examples. In *CVPR*, 15262–15271.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*.

Hu, Y.; Cheng, D.; Zhang, D.; Wang, N.; Liu, T.; and Gao, X. 2024. Task-aware Orthogonal Sparse Network for Exploring Shared Knowledge in Continual Learning. In *ICML*.

Hu, Z.; Li, Y.; Lyu, J.; Gao, D.; and Vasconcelos, N. 2023. Dense network expansion for class incremental learning. In *CVPR*, 11858–11867.

Jia, M.; Tang, L.; Chen, B.; Cardie, C.; Belongie, S. J.; Hariharan, B.; and Lim, S. 2022. Visual Prompt Tuning. In *ECCV*, 709–727. Springer.

Jung, D.; Han, D.; Bang, J.; and Song, H. 2023. Generating instance-level prompts for rehearsal-free continual learning. In *ICCV*, 11847–11857.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13): 3521–3526.

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

Kumaran, D.; Hassabis, D.; and McClelland, J. L. 2016. What learning systems do intelligent agents need? Complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7): 512–534.

Li, L.; Peng, J.; Chen, H.; Gao, C.; and Yang, X. 2024. How to configure good in-context sequence for visual question answering. In *CVPR*, 26710–26720.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *TPAMI*, 40(12): 2935–2947.

Lian, D.; Zhou, D.; Feng, J.; and Wang, X. 2022. Scaling & shifting your features: A new baseline for efficient model tuning. *NeurIPS*, 35: 109–123.

Liu, Y.; Su, Y.; Liu, A.-A.; Schiele, B.; and Sun, Q. 2020. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, 12245–12254.

Lu, Y.; Zhang, S.; Cheng, D.; Xing, Y.; Wang, N.; Wang, P.; and Zhang, Y. 2024. Visual Prompt Tuning in Null Space for Continual Learning. In *NeurIPS*.

McClelland, J. L.; McNaughton, B. L.; and O'Reilly, R. C. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3): 419.

McDonnell, M. D.; Gong, D.; Parvaneh, A.; Abbasnejad, E.; and van den Hengel, A. 2024. Ranpac: Random projections and pre-trained models for continual learning. *NeurIPS*, 36.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 8026–8037.

Pham, Q.; Liu, C.; and Steven, H. 2022. Continual Normalization: Rethinking Batch Normalization for Online Continual Learning. In *ICLR*.

Rebuffi, S.-A.; Bilen, H.; and Vedaldi, A. 2017. Learning multiple visual domains with residual adapters. *NeurIPS*, 30.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *CVPR*, 2001–2010.

Shi, Y.; Zhou, K.; Liang, J.; Jiang, Z.; Feng, J.; Torr, P. H.; Bai, S.; and Tan, V. Y. 2022. Mimicking the Oracle: An Initial Phase Decorrelation Approach for Class Incremental Learning. In *CVPR*, 16722–16731.

Smith, J. S.; Karlinsky, L.; Gutta, V.; Cascante-Bonilla, P.; Kim, D.; Arbelle, A.; Panda, R.; Feris, R.; and Kira, Z. 2023. CODA-Prompt: COntinual Decomposed Attention-based Prompting for Rehearsal-Free Continual Learning. In *CVPR*, 11909–11919.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *NeurIPS*, 4080–4090.

Sun, H.-L.; Zhou, D.-W.; Li, Y.; Lu, S.; Yi, C.; Chen, Q.-G.; Xu, Z.; Luo, W.; Zhang, K.; Zhan, D.-C.; et al. 2024. Parrot: Multilingual Visual Instruction Tuning. *arXiv preprint arXiv:2406.02539*.

Sun, H.-L.; Zhou, D.-W.; Ye, H.-J.; and Zhan, D.-C. 2023. PILOT: A Pre-Trained Model-Based Continual Learning Toolbox. *arXiv preprint arXiv:2309.07117*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wang, F.-Y.; Zhou, D.-W.; Ye, H.-J.; and Zhan, D.-C. 2022a. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, 398–414.

Wang, Z.; Zhang, Z.; Ebrahimi, S.; Sun, R.; Zhang, H.; Lee, C.-Y.; Ren, X.; Su, G.; Perot, V.; Dy, J.; et al. 2022b. DualPrompt: Complementary Prompting for Rehearsal-free Continual Learning. *ECCV*.

Wang, Z.; Zhang, Z.; Lee, C.-Y.; Zhang, H.; Sun, R.; Ren, X.; Su, G.; Perot, V.; Dy, J.; and Pfister, T. 2022c. Learning to prompt for continual learning. In *CVPR*, 139–149.

Wei, X.-S.; Ye, H.-J.; Mu, X.; Wu, J.; Shen, C.; and Zhou, Z.-H. 2019. Multi-instance learning with emerging novel class. *IEEE Transactions on Knowledge and Data Engineering*, 33(5): 2109–2120.

Wightman, R. 2019. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models.

Yan, S.; Xie, J.; and He, X. 2021. DER: Dynamically Expandable Representation for Class Incremental Learning. In *CVPR*, 3014–3023.

Ye, H.-J.; Lu, S.; and Zhan, D.-C. 2022. Generalized knowledge distillation via relationship matching. *TPAMI*, 45(2): 1817–1834.

Ye, H.-J.; Zhan, D.-C.; Li, N.; and Jiang, Y. 2019. Learning multiple local metrics: Global consideration helps. *TPAMI*, 42(7): 1698–1712.

Yu, L.; Twardowski, B.; Liu, X.; Herranz, L.; Wang, K.; Cheng, Y.; Jui, S.; and Weijer, J. v. d. 2020. Semantic drift compensation for class-incremental learning. In *CVPR*, 6982–6991.

Zhai, X.; Puigcerver, J.; Kolesnikov, A.; Ruyssen, P.; Riquelme, C.; Lucic, M.; Djolonga, J.; Pinto, A. S.; Neumann, M.; Dosovitskiy, A.; et al. 2019. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*.

Zhang, G.; Wang, L.; Kang, G.; Chen, L.; and Wei, Y. 2023. SLCA: Slow Learner with Classifier Alignment for Continual Learning on a Pre-trained Model. In *ICCV*.

Zhang, Y.; Yin, Z.; Shao, J.; and Liu, Z. 2022. Benchmarking omni-vision representation through the lens of visual realms. In *ECCV*, 594–611. Springer.

Zhao, H.; Wang, H.; Fu, Y.; Wu, F.; and Li, X. 2021. Memory efficient class-incremental learning for image classification. *IEEE Transactions on Neural Networks and Learning Systems*.

Zhou, D.-W.; Cai, Z.-W.; Ye, H.-J.; Zhan, D.-C.; and Liu, Z. 2024a. Revisiting Class-Incremental Learning with Pre-Trained Models: Generalizability and Adaptivity are All You Need. *International Journal of Computer Vision*.

Zhou, D.-W.; Sun, H.-L.; Ning, J.; Ye, H.-J.; and Zhan, D.-C. 2024b. Continual learning with pre-trained models: A survey. In *IJCAI*, 8363–8371.

Zhou, D.-W.; Sun, H.-L.; Ye, H.-J.; and Zhan, D.-C. 2024c. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *CVPR*, 23554–23564.

Zhou, D.-W.; Wang, Q.-W.; Ye, H.-J.; and Zhan, D.-C. 2023. A Model or 603 Exemplars: Towards Memory-Efficient Class-Incremental Learning. In *ICLR*.