International Journal of Computer Vision (2025) 133:1012–1032
https://doi.org/10.1007/s11263-024-02218-0

Revisiting Class-Incremental Learning with Pre-Trained Models: Generalizability and Adaptivity are All You Need

Da-Wei Zhou[1,2] · **Zi-Wen Cai**[1,2] · **Han-Jia Ye**[1,2] · **De-Chuan Zhan**[1,2] · **Ziwei Liu**[3]

Received: 4 December 2023 / Accepted: 5 August 2024 / Published online: 31 August 2024

Abstract

Class-incremental learning (CIL) aims to adapt to emerging new classes without forgetting old ones. Traditional CIL models are trained from scratch to continually acquire knowledge as data evolves. Recently, pre-training has achieved substantial progress, making vast pre-trained models (PTMs) accessible for CIL. Contrary to traditional methods, PTMs possess generalizable embeddings, which can be easily transferred for CIL. In this work, we revisit CIL with PTMs and argue that the core factors in CIL are adaptivity for model updating and generalizability for knowledge transferring. (1) We first reveal that frozen PTM can already provide generalizable embeddings for CIL. Surprisingly, a simple baseline (SimpleCIL) which continually sets the classifiers of PTM to prototype features can beat state-of-the-art even without training on the downstream task. (2) Due to the distribution gap between pre-trained and downstream datasets, PTM can be further cultivated with adaptivity via model adaptation. We propose AdaPt and mERge (Aper), which aggregates the embeddings of PTM and adapted models for classifier construction. Aper is a general framework that can be orthogonally combined with any parameter-efficient tuning method, which holds the advantages of PTM's generalizability and adapted model's adaptivity. (3) Additionally, considering previous ImageNet-based benchmarks are unsuitable in the era of PTM due to data overlapping, we propose four new benchmarks for assessment, namely ImageNet-A, ObjectNet, OmniBenchmark, and VTAB. Extensive experiments validate the effectiveness of Aper with a unified and concise framework. Code is available at https://github.com/zhoudw-zdw/RevisitingCIL.

Keywords Class-incremental learning · Pre-trained models · Continual learning · Catastrophic forgetting

1 Introduction

With the advancement of deep learning, deep models have achieved impressive feats in many fields (Zhong et al., 2021; Liu et al., 2017; Hassan et al., 2023; Wang et al., 2019; Jaimes & Sebe, 2007; Yang, 2002). However, most research focuses on recognizing a limited number of classes in static environments. In the real world, applications often deal with streaming data with incoming new classes (Gomes et al., 2017). To address this issue, Class-Incremental Learning (CIL) has been proposed, which allows the model to learn from the evolving data and *continuously* build a unified classification model. Nevertheless, when new classes are added sequentially, the notorious *catastrophic forgetting* occurs (French, 1999), which erases the previously learned knowledge. While typical CIL methods assume that the model is "*trained from scratch*," recent advancements in pre-training (Han et al., 2021) have made Pre-Trained Models (PTMs) more accessible for designing models in downstream tasks. These PTMs are often trained on massive

Communicated by ZHUN ZHONG.

✉ Han-Jia Ye
yehj@lamda.nju.edu.cn

✉ Ziwei Liu
ziwei.liu@ntu.edu.sg

Da-Wei Zhou
zhoudw@lamda.nju.edu.cn

Zi-Wen Cai
caizw@lamda.nju.edu.cn

De-Chuan Zhan
zhandc@nju.edu.cn

[1] School of Artificial Intelligence, Nanjing University, Nanjing 210023, China

[2] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

[3] S-Lab, Nanyang Technological University, 639798 Singapore, Singapore

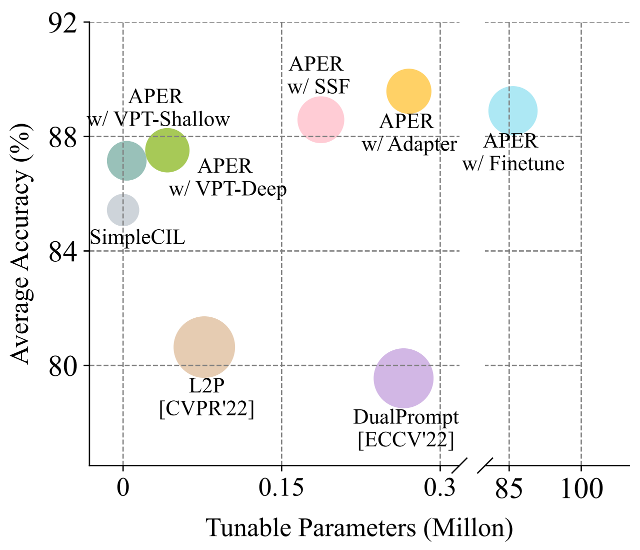


Fig. 1 Comparison of different PTM-based CIL methods on VTAB dataset. The X-axis stands for the number of tunable parameters, and the Y-axis represents the average accuracy. The radius stands for the training time. Although consuming more tuning parameters and training time, current state-of-the-art (*i.e.*, L2P and DualPrompt) still show inferior performance than the baseline method SimpleCIL. By contrast, our APER consistently improves the baseline with tiny costs. For a fair comparison, all methods are based on pre-trained ViT-B/16-IN1K. SimpleCIL utilizes the training set to calculate the average embeddings

corpus (Radford et al., 2021) or abundant images (Deng et al., 2009; Ridnik et al., 2021) with handcrafted tricks (Steiner et al., 2022), resulting in strong *generalizability*. Consequently, several methods (Wang et al., 2022b, a, d; Villa et al., 2023) propose to leverage PTM for better incremental learning.

Powerful PTMs alleviate the burden of CIL (Zhou et al., 2023a). However, upon revisiting the objective of CIL, we find essential differences between these protocols. Without PTMs, CIL models are trained from *random initialization* to *continually acquire* the knowledge of new classes and build a unified embedding space, which requires the **adaptivity** for sequential updating. In contrast, PTMs are trained with massive datasets, which makes it easier to achieve a powerful embedding space with strong generalizability. To use a human learning analogy, non-PTM methods aim to teach an *infant* to grow up and continually acquire knowledge through college, while PTM-based methods teach an experienced *adult* to do the same thing, which is much easier.

To evaluate the generalizability of PTMs, we formulate a CIL task using the VTAB (Zhai et al., 2019) dataset and test the performance of state-of-the-art PTM-based methods (Wang et al., 2022a, b) with a pre-trained ViT-B/16-IN1K in Fig. 1. As a comparison, we present a simple baseline SimpleCIL to evaluate the quality of the pre-trained features. With the pre-trained embedding function frozen, SimpleCIL sets the classifier weights to the average embeddings (Snell et al., 2017) of each new class for classification. The average embeddings are calculated with training images. In this way, SimpleCIL serves as a direct indicator of the quality of pre-trained features. If PTMs possess generalizable features, directly matching the average pattern to each query instance could also achieve competitive results. Surprisingly, Fig. 1 shows that SimpleCIL outperforms the current SOTA by 5% even without any tuning on these downstream tasks, verifying its strong *generalizability* in knowledge transfer.

Although PTMs are generalizable for CIL, a *domain gap* may still exist between pre-trained and incremental datasets (Zhou et al., 2022b; You et al., 2020). For instance, the ImageNet pre-trained model may not generalize well to out-of-distribution (Hendrycks et al., 2021b) or specialized tasks (Alfassy et al., 2022). Under such circumstances, freezing the embedding for knowledge transferring is not a "*panacea*." Accordingly, *adaptivity* becomes essential to enable the model to grasp *task-specific features*. Nevertheless, sequentially tuning the PTM will harm the structural information and weaken the generalizability (Kumar & Raghunathan, 2022), leading to the irreversible forgetting of previous knowledge. Is there a way to *unify* the generalizability of PTM with the adaptivity of the adapted model?

In this paper, we present AdaPt and mERge (APER) for CIL, which employs PTM to enhance generalizability and adaptivity in a unified framework. To improve adaptivity, we adapt the PTM in the first incremental stage via parameter-efficient tuning. Adapting the model helps to obtain task-specific features and fills the domain gap between PTM and incremental data. We then concatenate the adapted model with the PTM to extract average embeddings as the classifier, thereby maintaining generalizability. APER restricts model tuning in the first stage, striking a balance between adaptivity and generalizability. Moreover, typical ImageNet-based CIL benchmarks are unsuitable for evaluation due to overlapping between pre-trained and downstream tasks. Therefore, we benchmark PTM-based CIL with four new datasets that have *large* domain gaps with the pre-trained data. Extensive experiments under various settings demonstrate the effectiveness of APER. Our main contributions can be summarized as follows:

- With extensive empirical evaluations, we reveal a simple baseline (*i.e.*, SimpleCIL) can directly transfer the strong generalizability of PTMs in CIL, which even outperforms current state-of-the-art without training on the downstream task.
- We propose APER that employs PTM to enhance generalizability and adaptivity in a unified framework. It enjoys the generalizability of PTMs using a prototype-based classifier and the strong adaptivity of downstream tasks by model adaptation. Due to its uniformity, APER

can be applied to different network structures and different tuning techniques;
- Due to the overlapping between pre-trained data and traditional CIL benchmarks, we benchmark pre-trained model-based CIL with several new datasets with large domain gaps with ImageNet. Extensive experiments on these benchmark datasets verify APER's state-of-the-art performance.

2 Related Work

2.1 Class-Incremental Learning (CIL)

Class-incremental learning enables a learning system to continually incorporate new concepts without forgetting old ones (Zhou et al., 2023a; Wang et al., 2023a; Masana et al., 2022; Zhang et al., 2023b; Li et al., 2023; Cai et al., 2024; Yang et al., 2024; Li et al., 2023; Ding et al., 2023). Typical CIL methods can be divided into several categories. Exemplar-based methods save and replay exemplars from old classes to recover former knowledge (Aljundi et al., 2019; Chaudhry et al., 2018; Iscen et al., 2020; Liu et al., 2020). Apart from direct saving exemplars, other methods work on saving features (Zhao et al., 2021; Iscen et al., 2020; Zhu et al., 2021) or using generative models (Shin et al., 2017; Jiang et al., 2021; Smith et al., 2021; Gao & Liu, 2023) to construct the memory. Knowledge distillation-based methods aim to align the outputs of old and new models during updating, thereby maintaining knowledge of old concepts (Li & Hoiem, 2017; Rebuffi et al., 2017b; Douillard et al., 2020; Zhang et al., 2020; Xinting et al., 2021). The alignment can be built in several aspects, resulting in different optimization targets. iCaRL (Rebuffi et al., 2017b) and LwF (Li & Hoiem, 2017) utilize logit distillation, which requires the output logits on old classes to be the same. LUCIR (Hou et al., 2019) utilizes feature distillation and forces the output features to be the same across models. Some following works distill other feature products to resist forgetting, *e.g.*, attention map (Dhar et al., 2019), weighted feature map (Kang et al., 2022), pooled features (Douillard et al., 2020), casual effect (Xinting et al., 2021), subspace feature (Simon et al., 2021), and spatial/temporal features (Zhao et al., 2021). Additionally, other works also consider distilling the relational information among a group of instances (Dong et al., 2021; Gao et al., 2020; Tao et al., 2020; Dong et al., 2023). The third group finds the inductive bias in the incremental model and designs rectification algorithms for an unbiased prediction. BiC (Yue et al., 2019) and IL2M (Belouadah & Popescu, 2019) calibrate the logit scales between old and new classes to resist forgetting former classes. WA (Zhao et al., 2020) directly normalizes the fully connected layer to alleviate its influence on the final prediction. SDC (Lu et al., 2020) estimates the prototype drift of former classes via new class instances. TEEN (Wang et al., 2023b) designs a prototype calibration process to adjust the few-shot prototypes of new classes for better classification. The following works also consider rectifying the biased BN statistics (Pham & Liu, 2022) and feature representations (Shi et al., 2022). Recently, network expansion-based methods have shown competitive performance, which can be further divided into neuron-wise, backbone-wise, and token-wise. Neuron-wise (Yoon et al., 2018; Ju & Zhu, 2018) expansion aims to expand the network's width to enhance its representation ability. Backbone-wise (Yan et al., 2021; Wang et al., 2022; Zhou et al., 2023b; Wang et al., 2023c) expansion methods aim to build a holistic embedding by training a separate backbone for each new task and aggregate them as the final representation. Finally, token-wise (Douillard et al., 2022; Wang et al., 2022a, b, d) expansion are designed to add lightweight tokens to adapt the model while preserving its knowledge. CIL algorithms are also widely adopted in other real-world applications, *e.g.*, federated learning (Dong et al., 2022), semantic segmentation (Cermelli et al., 2022), text-to-image diffusion (Sun et al., 2024), and object detection (Dong et al., 2021; Perez-Rua et al., 2020).

2.2 CIL with Pre-Trained Models

Pre-trained model-based CIL (Zhou et al., 2024) is becoming popular with the increasing prevalence of pre-trained models (Dosovitskiy et al., 2020; Radford et al., 2021). The aim is to sequentially adjust the PTM to stream data with new classes without forgetting. L2P (Wang et al., 2022b) applies visual prompt tuning (Jia et al., 2022) to CIL based on the pre-trained Vision Transformer (Dosovitskiy et al., 2020) and learns a prompt pool to select the instance-specific prompt. During training, L2P retrieves the nearest prompts to the query instance and appends them to get the instance-specific embedding. DualPrompt (Wang et al., 2022a) extends L2P with general and expert prompts. Specifically, general prompts are equally assigned to all tasks, while expert prompts are selected for the specific task via the prompt retrieval process. Unlike the key-value search in L2P, CODA-Prompt (Smith et al., 2023) improves the prompt selection process with an attention mechanism so that the reweighted prompt can reflect the task-specific information of all seen classes. Furthermore, DAP (Jung et al., 2023) learns a prompt generator that can generate instance-specific prompts instead of the complex prompt retrieval process. SLCA (Zhang et al., 2023a) explores the classifier rectification process (Zhu et al., 2021) during model updating. ESN (Wang et al., 2023c) adopts the anchor-based energy self-normalization strategy to aggregate multiple pre-trained classifiers. CPP (Li et al., 2024) designs task-specific prompt-tuning with a contrastive learning objective. LAE (Gao et al.,

2023) utilizes exponential moving average (EMA) among online and offline models to resist forgetting. Although it also considers learning multiple models for CIL, our work differs from it in the inference format and the updating policy. Apart from the single modality for visual recognition, recent research also involves incremental learning of pre-trained vision-language models (Jiahui et al., 2022; Yuan et al., 2021; Tschannen et al., 2022). When changing ViT into CLIP (Radford et al., 2021), S-Prompts (Wang et al., 2022d) and Pivot (Villa et al., 2023) extend L2P by learning prompts for both text and image modalities (Zhou et al., 2022d). A contemporary work (Liu et al., 2023) explores the application of PTMs in class-incremental novel class discovery (Roy et al., 2022), which shows a frozen PTM can detect and learn new classes with high performance. However, apart from the frozen PTM, this manuscript also explores the effect of downstream data in adapting the model, aiming to unify the generalizability of PTM and adaptivity of the downstream data.

2.3 Parameter-Efficient Tuning for Pre-Trained Models

Parameter-efficient tuning aims to adapt the pre-trained model to downstream tasks by tuning only a small number of (extra) parameters. Compared to fully finetuning, parameter-efficient tuning obtains competitive or even better performance at a much lower cost. Visual prompt tuning (VPT) (Jia et al., 2022) prepends tunable prefix tokens (Li & Liang, 2021) to the input or hidden layers. LoRA (Hu et al., 2022) learns low-rank matrices to approximate parameter updates. AdaptFormer (Chen et al., 2022) learns extra adapter (Rebuffi et al., 2017a) modules with downsize and upsize projection. AdapterFusion (Pfeiffer et al., 2021) merges the learned adapters with a fusion module. SSF (Lian et al., 2022) addresses the scaling and shifting operation for model tuning. BitFit (Zaken et al., 2022) only tunes the bias term in the pre-trained model, and FacT (Jie & Deng, 2023) tensorizes the weights of each ViT into a single 3D tensor and updates it during finetuning. Apart from additional modules in the network, Visual prompting (Bahng et al., 2022) proposes learning tunable parameters in the input space. MAM-Adapter (He et al., 2022a) formulates these works in a unified framework, and NOAH (Zhang et al., 2022b) searches for the optimal design of prompt modules for downstream tasks. Apart from tuning a pre-trained Vision Transformer, CoOp (Zhou et al., 2022d) and CoCoOp (Zhou et al., 2022c) explore the application of prompt tuning for CLIP (Radford et al., 2021) via learning textual prompts. Maple (Khattak et al., 2023) further promotes strong coupling between the vision-language prompts to ensure mutual synergy.

3 From Old Classes to New Classes

3.1 Class-Incremental Learning

CIL aims to learn from an evolving data stream with new classes to build a unified classifier (Rebuffi et al., 2017b). There is a sequence of B training tasks $\left\{\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^B\right\}$, where $\mathcal{D}^b = \left\{\left(\mathbf{x}_i^b, y_i^b\right)\right\}_{i=1}^{n_b}$ is the b-th incremental step with n_b instances. Here, the training instance $\mathbf{x}_i^b \in \mathbb{R}^D$ belongs to class $y_i \in Y_b$, where Y_b is the label space of task b. $Y_b \cap Y_{b'} = \varnothing$ for $b \neq b'$. During the b-th training stage, we can only access data from $\mathcal{D}^b$ for model updating. This paper focuses on the exemplar-free CIL setting (Zhu et al., 2021; Wang et al., 2022b), where *no historical data* can be fetched for rehearsal. The goal of CIL is to build a unified model for all seen classes incrementally, *i.e.*, acquiring knowledge from new classes and meanwhile preserving knowledge from former ones. The model's capability is evaluated over all seen classes $\mathcal{Y}_b = Y_1 \cup \cdots Y_b$ after each incremental task. Formally, the target is to fit a model $f(\mathbf{x}) : X \rightarrow \mathcal{Y}_b$ that minimizes the empirical risk across all testing datasets:

$$\frac{1}{N} \sum_{(\mathbf{x}_j, y_j) \in \mathcal{D}_t^1 \cup \cdots \mathcal{D}_t^b} \ell\left(f\left(\mathbf{x}_j\right), y_j\right), \tag{1}$$

where $\ell(\cdot, \cdot)$ measures the discrepancy between prediction and ground-truth label. $\mathcal{D}_t^b$ denotes the testing set of task b, and N is the number of instances. A good CIL model satisfying Eq. 1 has discriminability among all classes, which strikes a balance between learning new classes and remembering old ones.

Following (Wang et al., 2022b, a; Zhou et al., 2024), we assume the availability of a pre-trained model (*e.g.*, a ViT (Dosovitskiy et al., 2020) or ResNet (He et al., 2016c)) on ImageNet (Deng et al., 2009), which we use as the initialization of $f(\mathbf{x})$. For clarity, we decouple the deep model into two parts: $f(\mathbf{x}) = W^\top \phi(\mathbf{x})$, where $\phi(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^d$ is the embedding function and $W \in \mathbb{R}^{d \times |\mathcal{Y}_b|}$ is the classification head. We denote the classifier for class k as $\mathbf{w}_k$: $W = [\mathbf{w}_1, \cdots, \mathbf{w}_{|\mathcal{Y}_b|}]$. We refer to the features after pooling as $\phi(\mathbf{x})$ for convolutional networks. In a plain ViT, the input encoding layer transforms the image into a sequence of output features $\mathbf{x}_e \in \mathbb{R}^{L \times d}$, where L is the sequence length. We assume the first token in $\mathbf{x}_e$ to be the `[CLS]` token to simplify notation. $\mathbf{x}_e$ is then fed into the subsequent layers (*i.e.*, multi-head self-attention and MLP) to produce the final embeddings. We treat the embedded `[CLS]` token as $\phi(\mathbf{x})$ for ViT.

3.2 Adaptivity and Generalizability in Class-Incremental Learning

CIL with adaptivity: Before introducing PTMs into CIL, models are trained from scratch to gradually acquire knowledge of new classes. The common solution is to update the incremental model with cross-entropy loss, which equips the model with *adaptivity* to adapt to new tasks:

$$\mathcal{L} = \sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}^b} \ell\left(f\left(\mathbf{x}_i\right), y_i\right) + \mathcal{L}_{reg} \,, \tag{2}$$

where $\mathcal{L}_{reg}$ stands for the regularization terms to resist forgetting, *e.g.*, knowledge distillation (Hinton et al., 2015; Li & Hoiem, 2017) or parameter regularization (Kirkpatrick et al., 2017).
CIL with generalizability: With the introduction of PTM to CIL (Wang et al., 2022b), continual learners are born with *generalizability*, which can be directly transferred to downstream tasks without learning. Correspondingly, we define a simple baseline, SimpleCIL, to transfer PTM for incremental tasks. With the embedding function $\phi(\cdot)$ *frozen* throughout the learning process, we extract average embedding (*i.e.*, prototype (Snell et al., 2017)) of each class:

$$\mathbf{p}_i = \frac{1}{K}\sum_{j=1}^{|\mathcal{D}^b|} \mathbb{I}(y_j = i)\phi(\mathbf{x}_j) \,, \tag{3}$$

where $K = \sum_{j=1}^{|\mathcal{D}^b|} \mathbb{I}(y_j = i)$, and $\mathbb{I}(\cdot)$ is the indicator function. The averaged embedding represents the most common pattern of the corresponding class. We set the prototype as the classifier, *i.e.*, $\mathbf{w}_i = \mathbf{p}_i$, to directly adjust the PTM for CIL. SimpleCIL demonstrates competitive performance in Fig. 1, confirming the strong generalizability of pre-trained models.
Generalizability vs. adaptivity: Equations 2 and 3 address different aspects of CIL models. The former aims to enhance the adaptivity by enabling the model to be gradually tuned. By contrast, the latter highlights the model's generalizability by freezing it throughout the learning process. To understand their roles in CIL, we conduct an experiment on CIFAR100 with 20 incremental tasks and compare the performance of finetuning versus SimpleCIL. These methods are based on pre-trained ViT-B/16-IN21K, and we separately report the performance of new (Y_b) and old ($\mathcal{Y}_{b-1}$) classes in Fig. 2. Specifically, SimpleCIL relies on the generalizability of PTM, which works competitively even without training on the target dataset. However, it can be further improved to grasp the task-specific features, and finetuning shows better performance in new classes with the help of adaptivity. However, finetuning suffers catastrophic forgetting of old classes since features are continually changing.

To summarize, these characteristics are two core aspects of CIL — adaptivity enables the model to bridge the domain gap between pre-training and incremental learning. At the same time, generalizability encourages knowledge transfer from pre-training to incremental learning. Therefore, both of them should be cultivated to facilitate CIL.

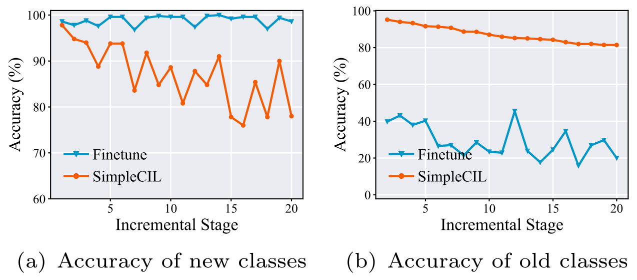


Fig. 2 Performance of new and old classes in CIL with PTM. Sequentially finetuning the model fills the domain gap and performs better on new classes, while freezing the model has better generalizability and performs better on old classes

4 Aper: AdaPt and mERge PTMs for CIL

Motivated by the potential for enhancing generalizability and adaptivity, can we achieve these characteristics in a unified framework? Specifically, we aim to achieve this goal from two aspects. On the one hand, to bridge the domain gap between the PTM and downstream datasets, *model adaptation* is essential to move the PTM towards incremental data. On the other hand, since the adapted model may lose the generalizability of high-level features, we attempt to *merge* the adapted model and PTM into a unified network for future tasks. The merged embedding function is kept frozen throughout the incremental learning process, transferring the generalizable embedding of model sets to incoming new classes. In this way, generalizability and adaptivity are achieved in the unified framework. We first introduce the framework of Aper and then discuss the specific techniques for model adaptation.

4.1 Training Procedure of Aper

Although PTMs have discriminative features, there may exist a significant domain gap between the pre-trained dataset and incremental data. For example, the PTM is optimized to capture the characteristics of classes in ImageNet, while the incremental data stream may correspond to specialized data requiring domain knowledge or has extensive concept drift from ImageNet. To bridge this gap, an adapting process can be developed with the incremental data:

$$f^*(\mathbf{x}) = \mathcal{F}(f(\mathbf{x}), \mathcal{D}, \Theta) \,, \tag{4}$$

where the adapting algorithm $\mathcal{F}$ takes the current model $f(\mathbf{x})$ and the dataset $\mathcal{D}$ as input. It optimizes the parameter set Θ and produces the adapted model $f^*(\mathbf{x})$ that gains the domain-specific knowledge in the corresponding dataset. We introduce the variations of $\mathcal{F}$ in Sect. 4.2. If we could obtain all the incremental training sets at once, adapting the model via $\mathcal{F}(f(\mathbf{x}), \mathcal{D}^1 \cup \mathcal{D}^2 \cdots \cup \mathcal{D}^B, \Theta)$ can transfer the knowledge from the PTM to the incremental dataset and grasp the task-specific features for better performance.

However, since data in CIL arrives sequentially, we cannot hold all the training sets at once. Continuously adapting the model would consequently result in catastrophic forgetting (as shown in Fig. 2(b)). Hence, an alternative choice is to adapt the model only *in the first incremental stage*:

$$f^*(\mathbf{x}) = \mathcal{F}(f(\mathbf{x}), \mathcal{D}^1, \Theta)\,. \tag{5}$$

Since $\mathcal{D}^1$ is a subset of the incremental data stream, it also possesses *domain-specific* knowledge that could facilitate model adaptation. The tuning process enhances the adaptivity of the CIL model, and the next question is to ensure *generalizability*. Since Eq. 5 forces the original generalizable feature to become more specialized to the downstream task, high-level features irrelevant to $\mathcal{D}^1$ shall be *overwritten and forgotten*. Therefore, a better solution is to concatenate the features extracted by the PTM and the adapted model, *i.e.*, $[\phi^*(\mathbf{x}), \phi(\mathbf{x})]$, where $\phi^*(\mathbf{x})$ and $\phi(\mathbf{x})$ stand for the adapted and pre-trained embedding functions, respectively.

To maintain generalizability, we *freeze* the concatenated embedding functions $[\phi^*(\cdot), \phi(\cdot)]$ after adaptation and extract prototypes for the following classes:

$$\mathbf{p}_i = \tfrac{1}{K}\textstyle\sum_{j=1}^{|\mathcal{D}^b|} \mathbb{I}(y_j = i)[\phi^*(\mathbf{x}_j), \phi(\mathbf{x}_j)]\,, \tag{6}$$

where $K = \sum_{j=1}^{|\mathcal{D}^b|} \mathbb{I}(y_j = i)$. Compared to Eqs. 3 and 6 contains additional information from the adapted model, which incorporates domain-specific features for better recognition. These prototypes reveal the most common patterns from the adapted and pre-trained models, ensuring both generalizability and adaptivity. We directly adopt the class prototype as the classifier weight, *i.e.*, $\mathbf{w}_i = \mathbf{p}_i$, and utilize a cosine classifier for classification: $f(\mathbf{x}) = (\frac{W}{\|W\|_2})^\top(\frac{[\phi^*(\mathbf{x}),\phi(\mathbf{x})]}{\|[\phi^*(\mathbf{x}),\phi(\mathbf{x})]\|_2})$. Based on the similarity between instance embedding and class prototype, it assigns a higher probability to the class with a more similar prototype.

Effect of adapt and merge: We give the visualizations of APER in Fig. 3 (left). Although $\mathcal{D}^1$ is a subset of the entire training set, adapting with it still helps transfer the PTM from the upstream dataset to the downstream task. The adapting process can be viewed as a further pre-training procedure, which adapts the PTM to the incremental dataset and bridges the domain gap. By merging the embedding functions of the PTM and the adapted model, the extracted features are more representative than any one of them alone. Additionally, since the model is only trainable in the first incremental task, the efficiency of APER is comparable to SimpleCIL, which does not require sequential tuning. On the other hand, since the model is frozen in the subsequent tasks, it does not suffer catastrophic forgetting of former concepts. We give the pseudo-code of APER in Algorithm 1. Given the pre-trained model, we first adapt it with the first training dataset via Eq. 5 to get the adapted model. Afterward, we freeze the pre-trained model and adapted model and merge the embeddings. For the subsequent tasks, we get a new dataset and replace the classifier weights with prototypical features (*i.e.*, class centers). In the extreme case where the adaptation process in Eq. 5 does nothing to the PTM, APER will degrade to SimpleCIL, which guarantees the performance lower bound.

Algorithm 1 AdaPt and mERge (APER) for CIL

```
Input: Incremental datasets: ${\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^B}$, Pre-trained Model: $f(\mathbf{x})$;
Output: Updated model;
1: Adapt the model to $\mathcal{D}^1$ via Eq. 5;                      ▷ Model adapt
2: Freeze the embedding functions $\phi^*(\cdot)$ and $\phi(\cdot)$;
3: Merge the embeddings, i.e., $[\phi^*(\mathbf{x}), \phi(\mathbf{x})]$;   ▷ Model merge
4: for $b = 1, 2 \cdots, B$ do                                 ▷ Incremental learning
5:     Get the incremental training set $\mathcal{D}^b$;
6:     Extract the prototypes via Eq. 6;
7:     Replace the classifier with prototype;
8: end for
       return the updated model;
```

4.2 Adapting the PTM

To bridge the distribution gap between the pre-trained and incremental datasets, APER's performance depends on the effective adapting algorithm $\mathcal{F}$. In this section, we discuss six specializations of $\mathcal{F}$ in APER that can handle different types of PTMs, such as ViTs and CNNs.

Fully finetune: is a common solution when transferring the model to downstream tasks. It involves tuning all parameters in the adapting process, *i.e.*, $\Theta = \theta_\phi \cup \theta_W$, and minimizing the discrepancy between the model's output and the ground truth:

$$\min_{\theta_\phi \cup \theta_W} \sum_{(\mathbf{x}_j, y_j) \in \mathcal{D}^1} \ell\left(f\left(\mathbf{x}_j\right), y_j\right)\,. \tag{7}$$

However, the tuning cost could be relatively *high* for large-scale PTMs, *e.g.*, ViTs. Therefore, some parameter-efficient tuning techniques can alleviate the tuning cost and be better solutions.

Visual prompt tuning (VPT) (Jia et al., 2022): is a lightweight tuning technique for adapting ViTs, which only prepends some learnable prompts $\mathbf{P} \in \mathbb{R}^{p\times d}$ to form the

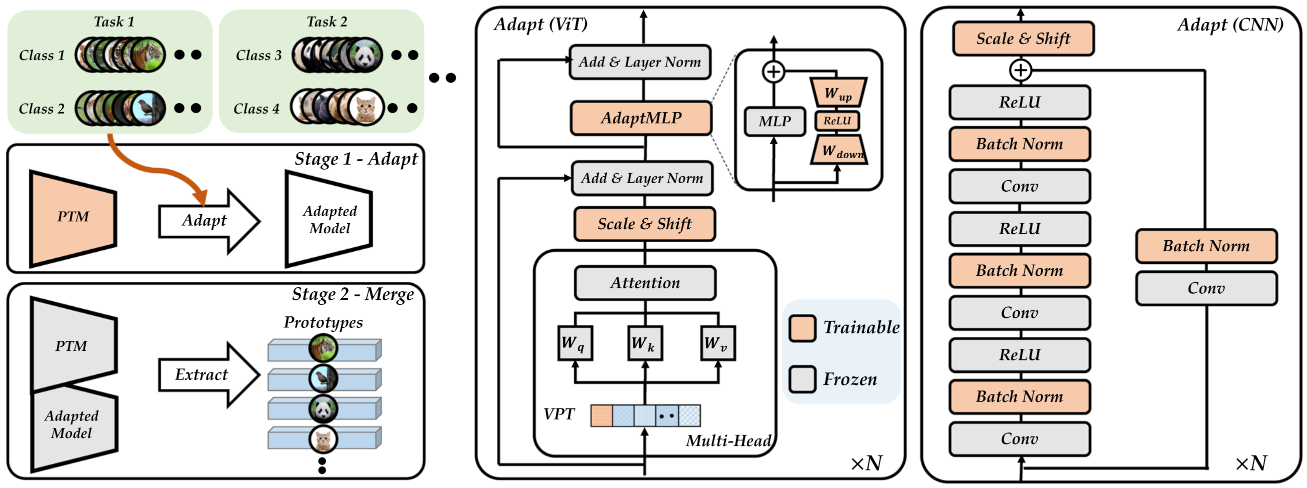


Fig. 3 Illustration of APER. Left: the training protocol of APER. We adapt the PTM using the first stage training set $\mathcal{D}^1$ and then concatenate the embedding functions of PTM and the adapted model to maintain *generalizability* and *adaptivity*. The aggregated embedding function $[\phi^*(\cdot), \phi(\cdot)]$ is frozen throughout the following stages, and we extract the prototypes via Eq. 6 to set the classifier. Middle: adapting pre-trained ViT for CIL. We provide VPT Deep/Shallow, Scale & Shift, and Adapter for model adaptation. Right: adapting pre-trained CNN for CIL. We provide BN tuning and Scale & Shift for model adaptation. APER is a general framework that can be orthogonally combined with these adapting techniques. Red modules in the figure are trainable, while gray ones are frozen (Color figure online)

extended features $[\mathbf{P}, \mathbf{x}_e]$, where $\mathbf{x}_e$ is the encoded features of the input image. The extended features are then fed into the subsequent layers of ViT to calculate the final embeddings. There are two variations of VPT: VPT-Deep, which prepends the prompts at every attention layer, and VPT-Shallow, which only prepends the prompts at the first layer. During optimization, it freezes the pre-trained weights in the embedding function and optimizes these prompts and classification head, *i.e.*, $\Theta = \theta_{\mathbf{P}} \cup \theta_W$.

Scale & Shift (SSF) (Lian et al., 2022): aims to adjust the feature activation by scaling and shifting. It appends an extra SSF layer after each operation layer (*i.e.*, MSA and MLP) and adjusts the output of these operations. Given the input $\mathbf{x}_i \in \mathbb{R}^{L\times d}$, the output $\mathbf{x}_o \in \mathbb{R}^{L\times d}$ is formulated as:

$$\mathbf{x}_o = \gamma \otimes \mathbf{x}_i + \beta \,, \tag{8}$$

where $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ are the scale and shift factors, respectively. $\otimes$ is Hadamard product (element-wise multiplication). The model optimizes the SSF layers and classifier, *i.e.*, $\Theta = \theta_{SSF} \cup \theta_W$, to trace the features of new tasks.

Adapter (Houlsby et al., 2019; Chen et al., 2022): is a bottleneck module which contains a down-projection $W_{\text{down}} \in \mathbb{R}^{d\times r}$ to reduce the feature dimension, a non-linear activation function, and an up-projection $W_{\text{up}} \in \mathbb{R}^{r\times d}$ to project back to the original dimension. We follow (Chen et al., 2022) to equip the original MLP structure in ViT with the adapter. We denote the input of the MLP layer as $\mathbf{x}_\ell$, and the output of AdaptMLP is formatted as:

$$\text{MLP}(\mathbf{x}_\ell) + \text{ReLU}(\mathbf{x}_\ell W_{\text{down}}) W_{\text{up}} \,. \tag{9}$$

With pre-trained weights frozen, it optimizes the adapter and classification head, *i.e.*, $\Theta = \theta_{W_{\text{down}}} \cup \theta_{W_{\text{up}}} \cup \theta_W$.

Batch normalization tuning: If the PTM is a convolutional network, *e.g.*, CNNs, we can adjust the BN (Ioffe & Szegedy, 2015) parameters. Since the running mean and variance in BN are compatible with the upstream data distribution, they could be *unstable* for downstream tasks. Correspondingly, we can reset the running statistics in BN and adapt to the current data via forward passing. No backpropagation is required, making it quick and simple for the pre-trained model.

Discussions: We visualize the adapting process of APER in Fig. 3. Compared to fully finetuning, parameter-efficient tuning adjusts the PTM towards the downstream task with lightweight modules. Besides, recent parameter-efficient tuning methods show stronger performance than fully finetune, indicating stronger adaptivity in the adapting process. The adapted model can capture the specialized features in the incremental data, leading to better adaptivity. Since L2P and DualPrompt are based on pre-trained ViT, they cannot be deployed with CNN. In contrast, APER is a general framework that efficiently handles diverse structures. Specifically, APER can be combined with VPT/SSF/Adapter for ViT and SSF/BN Tuning for CNN. Since APER adopts the prototype-based classifier, the linear classifier W will be dropped after adaptation.

4.3 Discussions on Related Concepts

There is a famous concept in CIL, namely "stability-plasticity dilemma" (Grossberg, 2012; Mermillod et al., 2013; Mirzadeh et al., 2020). Specifically, "stability" denotes the ability of a continual learner to remember old knowledge, while "plasticity" refers to the ability to learn new concepts. These concepts are similar to the concept of "generalizability and adaptivity" in this paper. However, there are some main differences that need to be highlighted.

Firstly, "stability-plasticity dilemma" mainly refers to the problem of *training from scratch* (*i.e.*, randomly initialized weights), where the model needs to balance learning new concepts and remembering the old. These concepts are two ultimate goals of continual learning, which do not conflict with "generalizability and adaptivity" raised in this paper. Secondly, "Generalizability and adaptivity" in this paper is the new characteristic in the era of PTMs. Specifically, a randomly initialized model does not have such "generalizability," which cannot be directly applied to the downstream tasks. However, continual learners are born with "generalizability" if starting with a PTM, and we observe a simple baseline shows strong performance. Furthermore, we find that "generalizability" is insufficient for all downstream tasks, especially when downstream tasks come from a different distribution. In this case, we need to enhance the PTM's "adaptivity" by further tuning it with the downstream task. Finally, by aggregating the features extracted by the pre-trained and adapted models, we unify these characteristics in a single model.

In summary, the "generalizability and adaptivity" in this paper is a new characteristic in class-incremental learning with pre-trained models. We aim to unify these characteristics in CIL and propose our Aper by aggregating the adapted and pre-trained models.

5 Experiments

This section compares Aper with state-of-the-art methods on benchmark datasets to show its superiority. Due to the overlap between pre-trained datasets and traditional class-incremental learning benchmarks, we also advocate four new benchmarks for evaluating PTM-based methods. Ablations and visualizations verify the effectiveness of Aper with new classes. We also explore the performance of different pre-trained models in class-incremental learning.

5.1 Implementation Details

Dataset: Following (Wang et al., 2022a; Lu et al., 2020), we evaluate the performance on CIFAR100 (Krizhevsky et al., 2009), CUB200 (Wah et al., 2011), and ImageNet-R (Hendrycks et al., 2021a). Since PTMs are often trained with ImageNet21K (Deng et al., 2009), evaluating PTM-based methods with ImageNet is meaningless. Hence, we advocate four new datasets that have *large domain gap* with ImageNet, namely ImageNet-A (Hendrycks et al., 2021b), ObjectNet (Barbu et al., 2019), Omnibenchmark (Zhang et al., 2022a) and VTAB (Zhai et al., 2019). Among them, ImageNet-A and ObjectNet contain *challenging samples* that ImageNet pre-trained models cannot handle, while Omnibenchmark and VTAB contain diverse classes from multiple *complex realms*. To construct the CIL task, we sample 200 classes from ObjectNet and ImageNet-A, and 300 from Omnibenchmark. We sample 5 datasets from VTAB, each containing 10 classes, to construct the cross-domain CIL setting. The aim of using the subset is to ensure a simple split of training classes. Following (Rebuffi et al., 2017b), we shuffle the classes with the same random seed and split them into 'B/Base-m, Inc-n.' It means the first dataset contains m classes, and each following dataset contains n classes. $m = 0$ means the total classes are equally divided into each task.

Comparison methods: We first compare to SOTA PTM-based CIL methods L2P (Wang et al., 2022b), Dual-Prompt (Wang et al., 2022a), CODA-Prompt (Smith et al., 2023), CPP (Li et al., 2024), and LAE (Gao et al., 2023). Additionally, we also modify classical CIL methods LwF (Li & Hoiem, 2017), SDC (Lu et al., 2020), iCaRL (Rebuffi et al., 2017b), LUCIR (Hou et al., 2019), DER (Yan et al., 2021), FOSTER (Wang et al., 2022), MEMO (Zhou et al., 2023b), FACT (Zhou et al., 2022a) to utilize the same PTM as the initialization. Apart from SimpleCIL, we also report the baseline, sequentially tuning the model, denoted as Finetune.

Training details: We use PyTorch (Paszke et al., 2019) and Pilot (Sun et al., 2023) to *deploy all models on Tesla V100 with the same network backbone*. As there are various PTMs publicly available (Wightman, 2019), we follow (Wang et al., 2022b, a) to choose the most representative ones, denoted as ViT-B/16-IN1K and ViT-B/16-IN21K. Both are pre-trained on ImageNet21K, while the former is additionally finetuned on ImageNet1K. During adaptation, we train the model with a batch size of 48 for 20 epochs and use SGD with momentum for optimization. The learning rate starts from 0.01 and decays with cosine annealing. The prompt length p is 5 for VPT, and the projection dim r is 16 for Adapter.

Evaluation protocol: Following (Rebuffi et al., 2017b), we denote accuracy after the b-th stage as $\mathcal{A}_b$. 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.

5.2 Benchmark Comparison

We report the performance against SOTA methods in Table 1, where all methods are based on the pre-trained ViT-B/16-IN21K. We also train these models with pre-trained ViT-

Table 1 Average and last performance comparison on seven datasets with ViT-B/16-IN21K as the backbone

Method	CIFAR B0 Inc5		CUB B0 Inc10		IN-R B0 Inc5		IN-A B0 Inc10		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	21.61	10.79	21.60	10.96	19.14	8.73	23.61	10.57	34.95	21.25
Finetune adapter (Chen et al., 2022)	60.51	49.32	66.84	52.99	47.59	40.28	43.05	37.66	50.22	35.95	62.32	50.53	48.91	45.12
LwF (Li & Hoiem, 2017)	46.29	41.07	48.97	32.03	39.93	26.47	35.39	23.83	33.01	20.65	47.14	33.95	40.48	27.54
SDC (Lu et al., 2020)	68.21	63.05	70.62	66.37	52.17	49.20	26.65	23.57	39.04	29.06	60.94	50.28	45.06	22.50
L2P (Wang et al., 2022b)	85.94	79.93	67.05	56.25	66.53	59.22	47.16	38.48	63.78	52.19	73.36	64.69	77.11	77.10
DualPrompt (Wang et al., 2022a)	87.87	81.15	77.47	66.54	63.31	55.22	52.56	42.68	59.27	49.33	73.92	65.52	83.36	81.23
CODA-prompt (Smith et al., 2023)	89.11	81.96	84.00	73.37	64.42	55.08	48.51	36.47	66.07	53.29	77.03	68.09	83.90	83.02
CPP (Li et al., 2024)	85.21	78.64	86.60	85.27	64.33	60.74	53.70	40.70	60.44	49.92	71.52	73.26	85.92	84.30
LAE (Gao et al., 2023)	**92.47**	**87.62**	83.13	77.78	69.05	63.17	57.19	46.41	62.28	50.57	73.80	70.63	86.14	84.39
SimpleCIL	87.57	81.26	92.20	**86.73**	62.58	54.55	60.50	49.44	65.45	53.59	79.34	73.15	85.99	84.38
APER w/ Finetune	87.67	81.27	91.82	86.39	70.51	62.42	61.57	50.76	61.41	48.34	73.02	65.03	**87.47**	80.44
APER w/ VPT-Shallow	90.43	84.57	92.02	86.51	66.63	58.32	57.72	46.15	64.54	52.53	79.63	73.68	87.15	**85.36**
APER w/ VPT-Deep	88.46	82.17	91.02	84.99	68.79	60.48	60.59	48.72	67.83	54.65	**81.05**	**74.47**	86.59	83.06
APER w/ SSF	87.78	81.98	91.72	86.13	68.94	60.60	**62.81**	**51.48**	**69.15**	**56.64**	80.53	74.00	85.66	81.92
APER w/ adapter	90.65	85.15	**92.21**	**86.73**	**72.35**	**64.33**	60.53	49.57	67.18	55.24	80.75	74.37	85.95	84.35

’IN-R/A’ stands for ‘ImageNet-R/A,’ ’ObjNet’ stands for ’ObjectNet,’ and ‘OmniBench’ stands for ‘OmniBenchmark.’ The best performance is shown in bold

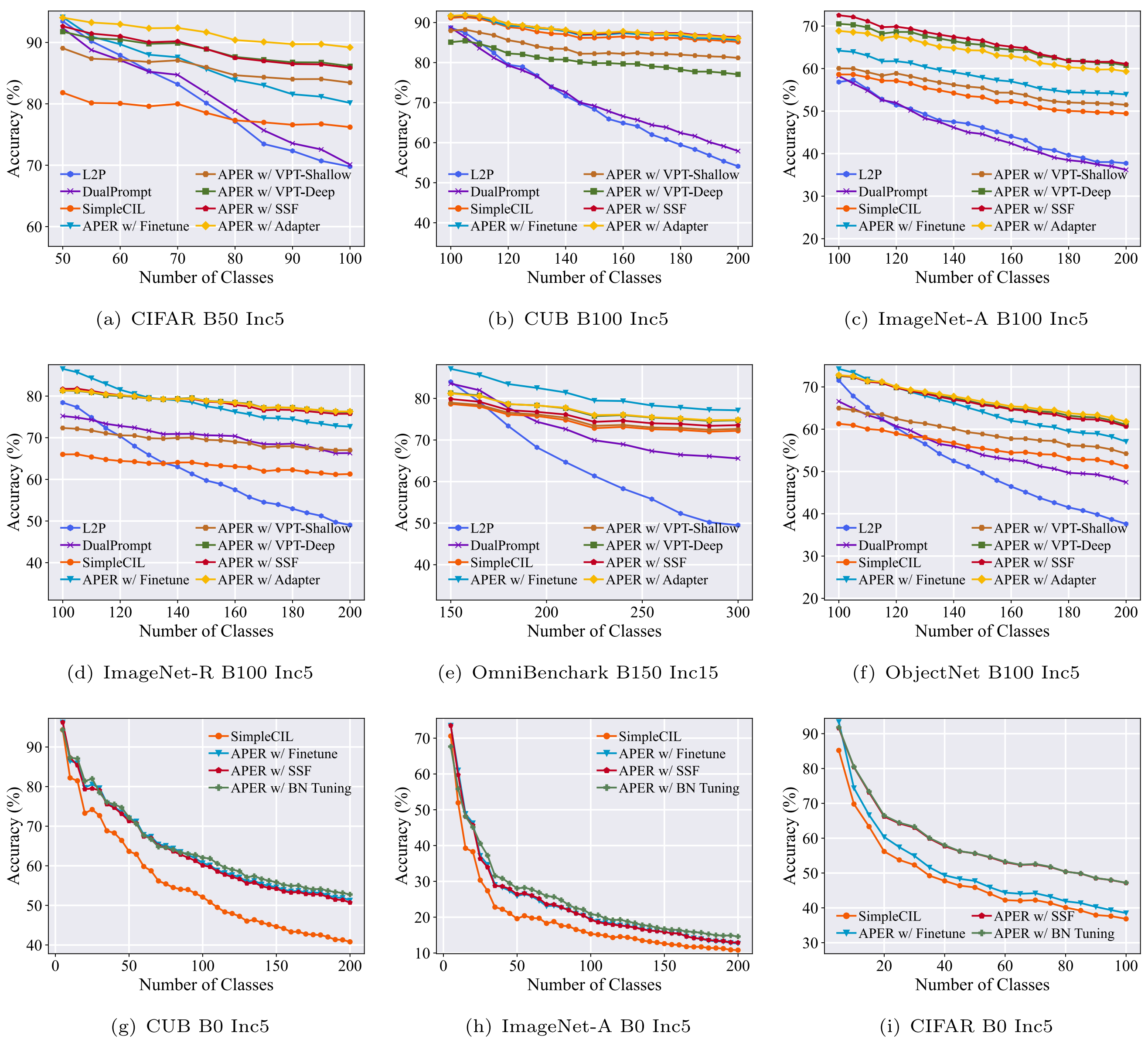


Fig. 4 **a–f**: Incremental performance with ViT-B/16-IN1K as the backbone when half of the total classes are base classes. **g–i:** Incremental performance when using ResNet18 as backbone. Since L2P and DualPrompt cannot be deployed with ResNet, we do not report their performance in (**g**)–(**i**). Aper consistently improves the performance of different backbones, *i.e.*, ViT and CNN

B/16-IN1K and show the incremental trend in Fig. 4(a)–(f). These data splits include settings with large and small base classes for a holistic evaluation.

Firstly, we can infer that the embeddings of PTMs are generalizable and can be directly applied for CIL to beat the SOTA. Specifically, the baseline SimpleCIL outperforms DualPrompt by **20%** on CUB and **8%** on ImageNet-A in terms of $\mathcal{A}_B$. However, strong PTMs can be further improved if they are adapted by Aper, as downstream tasks have a large domain gap with the pre-trained dataset. Specifically, we find Aper *consistently* outperforms SimpleCIL in seven benchmark datasets. In contrast, sequentially finetuning the model suffers severe forgetting, which verifies the effectiveness of the adapt and merge protocol. Specifically, L2P and DualPrompt suffer from forgetting due to prompts being overwritten in the latter stages and the linear layer's imbalanced weight norms. Since Aper only requires tuning the PTM in the first stage, it requires less training time and extra parameters than L2P and DualPrompt, as shown in Fig. 1. Among the variations of adapting techniques, we find *SSF and Adapter are more efficient than VPT*. We also compare to state-of-the-art traditional CIL methods and modify their

Table 2 Comparison to SOTA classical CIL methods with ViT-B/16-IN1K

Method	ObjNet B0 Inc20		IN-A B0 Inc20	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
iCaRL	33.43	19.18	29.22	16.16
LUCIR	41.17	25.89	31.09	18.59
DER	35.47	23.19	33.85	22.27
FOSTER	37.83	25.07	34.82	23.01
MEMO	38.52	25.41	36.37	24.46
FACT	60.59	50.96	60.13	49.82
SimpleCIL	62.11	51.13	59.67	49.44
Aper w/ SSF	**68.75**	**56.79**	**63.59**	**52.67**

The best performance is shown in bold
All methods are deployed without exemplars

backbones into pre-trained ViT for a fair comparison. However, we can infer from Table 2 that these methods work poorly without exemplars. Hence, Aper achieves the best performance compared to various CIL algorithms under the fair comparison.

Apart from ViTs, Aper also works well with pre-trained CNNs. We adopt the ImageNet1K pre-trained ResNet18 (He et al., 2016c) for evaluation and plot the incremental performance in Fig. 4(g), (h), (i). Results show that Aper consistently boosts the performance of pre-trained ViTs and CNNs. Specifically, we find a simple BN tuning technique achieves better performance than fully or partially finetuning the ResNet.

Lastly, as shown in Table 1, the performance on typical benchmarks is approaching saturation as they have a small domain gap with ImageNet. By contrast, due to the large domain gap between our newly established benchmarks and ImageNet, there is still space for improvement, indicating the effectiveness and necessity of these new benchmarks.

5.3 Ablation Study

In this section, we conduct an ablation study to investigate the influence of each part in Aper, *e.g.*, using sampled features, part of its components, and different tuning stages. We also analyze the parameter number of different methods.

5.3.1 Downscale Features

Since the feature of Aper is aggregated with PTM and adapted model, it has a larger feature dimension than a vanilla PTM (*e.g.*, 1536 versus 768). We conduct an ablation with Aper w/ SSF on ObjectNet B100 Inc5 to show whether these features are essential for CIL. Specifically, we train a PCA (Pearson, 1901) model in the first stage to reduce embedding dimension for the following stages. Denote the target dimension as k, we train the PCA model $\text{PCA}([\phi^*(\mathbf{x}), \phi(\mathbf{x})]) : \mathbb{R}^d \rightarrow \mathbb{R}^k$, and append it to the feature extractor. Hence, the features and prototypes are projected to k dimensions. We plot the performance with the change of k in Fig. 5(a). Specifically, we find Aper obtains competitive performance to DualPrompt (with 768 dims) even if the features are projected to 30 dims.

Apart from the PCA projection, we also experiment by randomly sampling k features from the original feature space and report the results in Fig. 5(b). The conclusions are consistent with the former ones, showing that randomly sampling 100 dimensions in the concatenated space achieves the same performance scale as DualPrompt. We show the accuracy-dimension curves in Fig. 5(c).

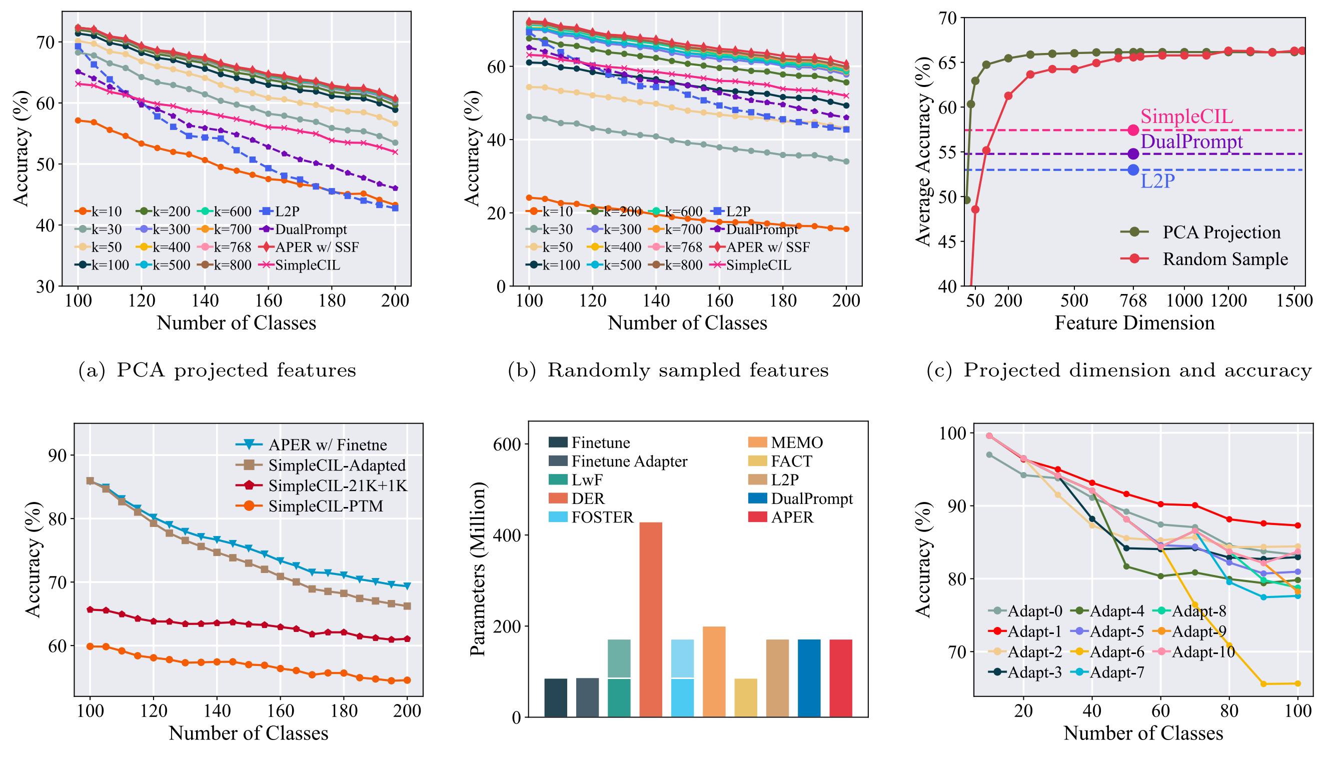
5.3.2 Sub-Modules

Since Aper is concatenated with PTM and adapted model, we conduct ablations on ImageNet-A Base100 Inc5 with ViT-B/16-IN21K to compare Aper w/ Finetune and its sub-modules. Specifically, we build SimpleCIL with $\phi(\cdot)$ and $\phi^*(\cdot)$, respectively, denoted as **SimpleCIL-PTM** and **SimpleCIL-Adapted**. The former represents the capability of PTM, while the latter stands for the power of the adapted model. Both are compositional modules in Aper. Besides, we build SimpleCIL based on concatenated pre-trained ViT-B/16-IN21K and ViT-B/16-IN1K, denoted as SimpleCIL-21K+1K. It utilizes the aggregated features of two embedding functions, which has the same dimension as Aper.

As shown in Fig. 5(d), SimpleCIL-PTM performs the worst among all variations, indicating that although pre-trained features are effective and generalizable, it still requires extracting features of downstream tasks for better representation. In comparison, SimpleCIL-Adapted outperforms SimpleCIL-PTM, indicating the importance of model adaptation and adaptivity. However, adapting the model also overwrites the high-level features, which reduces the model's generalizability. The adapted model suffers more extensive performance degradation than vanilla SimpleCIL, indicating the effect of generalizability in resisting forgetting. Finally, Aper w/ Finetune outperforms any of these sub-modules with the help of unified adaptivity and generalizability.

5.3.3 Parameter Scale

The parameter scale is another core factor influencing CIL algorithms in real-world applications, *e.g.*, edge devices or mobile phones. In this section, we compare the parameter scale of different methods to investigate the possibility of real-world applications. In the comparison, all methods are based on the pre-trained ViT. For methods requiring back-



(a) PCA projected features (b) Randomly sampled features (c) Projected dimension and accuracy

(d) Compositional components (e) Number of parameters (f) Adapting stages

Fig. 5 Ablation study. **a–c**: We use PCA or random sample to downscale the dimension of aggregated embeddings. **d**: We compare APER to its sub-modules for ablation. **e**: Number of total parameters of different compared methods. The bars with shadow denote the parameters used during training but dropped during inference. **f**: The accuracy trend with the change of adapting stages. Adapt-T denotes the model is adapted for the first T incremental tasks. $T = 0$ denotes SimpleCIL

bone expansion (*e.g.*, DER, MEMO, and FOSTER), we also use pre-trained ViT as the initialization of new backbones.

We list the total number of parameters of all compared methods in Fig. 5(e), which indicates that APER obtains better performance than the compared methods with the same scale or fewer parameters. Since L2P and DualPrompt have prompt pools, they rely on another pre-trained ViT as the 'retriever' to search for the instance-specific prompt. Hence, APER shares the same scale of total parameters as these methods. Additionally, since APER utilizes parameter-efficient tuning techniques to obtain the adapted model, most of the parameters in the adapted model are the same as the pre-trained weight. Hence, the memory budget of APER can be further alleviated, which we will explore in future works.

5.3.4 Influence of Adapting Stages

In APER, we only adapt the pre-trained model in the first incremental stage with $\mathcal{D}^1$. There are two reasons: 1) Sequentially tuning the model will suffer catastrophic forgetting. 2) Since we utilize a prototype-based classifier, tuning the model with multiple stages will result in *incompatible* features between former and new prototypes.

In this section, we conduct an ablation to determine the influence of adapting stages and report the results in Fig. 5(f). We conduct the experiment on CIFAR100 Base0 Inc10 setting with pre-trained ViT-B/16-IN21K. There are 10 incremental stages in total. We denote the tuning stages as T and train APER w/ Adapter for ablation. Specifically, we change the tuning stages among $\{0, 1, 2, \cdots, 10\}$ to determine the influence on the final performance. In the first T stages, we adapt the PTM incrementally with adapter and replace the classifier with prototypes. Afterward, in the T-th stage, we freeze the encoding functions and only extract prototypes for the following stages. $T = 0$ denotes vanilla SimpleCIL. To prevent forgetting, we freeze the classifier weight of former classes when learning new classes.

As shown in the figure, tuning the model with the first stage achieves the best performance among all settings. Specifically, multi-stage tuning harms generalizability and results in the incompatible features of former and new classes.

5.3.5 Different PTMs

Observing the performance gap between ViT-B/16-IN21K and ViT-B/16-IN1K, we seek to explore different kinds

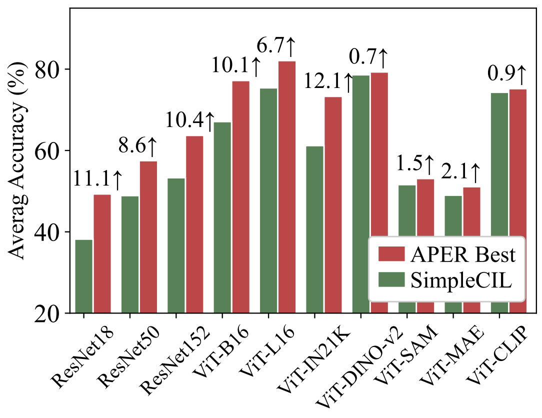


Fig. 6 CIL with different kinds of PTMs on ImageNet-R Base0 Inc20. APER consistently improves the performance of different PTMs. We report the best variation of APER in the figure and its relative improvement above SimpleCIL in the figure

of PTMs on ImageNet-R Base0 Inc20. We choose publicly available PTMs, *i.e.*, ResNet18/50/152 (He et al., 2016c), ViT-B/16-IN1K/21K, ViT-L/16-IN1K, ViT-B/16-DINO-v2 (Oquab et al., 2023), ViT-B/16-SAM (Chen et al., 2022), ViT-B/16-MAE (He et al., 2022b), ViT-B/16-CLIP (Radford et al., 2021) (image encoder) for a holistic evaluation, and report the results in Fig. 6. We can draw three main conclusions. Firstly, pre-trained ViTs show better generalizability than ResNets by achieving better performance with SimpleCIL. The main reason comes from the more extensive training data and parameters. Secondly, larger ViTs generalize better than small ones, and ViTs trained with supervised loss perform better than unsupervised ones. However, DINO-v2 shows the best performance due to its 1.2B training instances. Thirdly, owing to the massive training corpus and the cross-modal information, CLIP performs better than ImageNet21K pre-trained ViTs. Finally, we find the best APER variation consistently improves the performance of SimpleCIL for any PTM, thus validating its effectiveness.

5.4 Visualization of Incremental Sessions

In this section, we visualize the learned decision boundaries with t-SNE (Van der Maaten & Hinton, 2008) on CIFAR100 dataset between two incremental stages, as shown in Fig. 7(a), (b). We visualize the classes from the first and second incremental tasks with colorful dots and triangles. Correspondingly, the class prototypes are represented by squares. As we can infer from these figures, PTM works competitively, which well separates the instances into their corresponding classes. The class prototypes are situated at the center of each class, verifying their representativeness in recognition. When extending the model from the first to the second stage, we find APER performs well on both old and new classes. Visualizations verify the generalizability and adaptivity of APER.

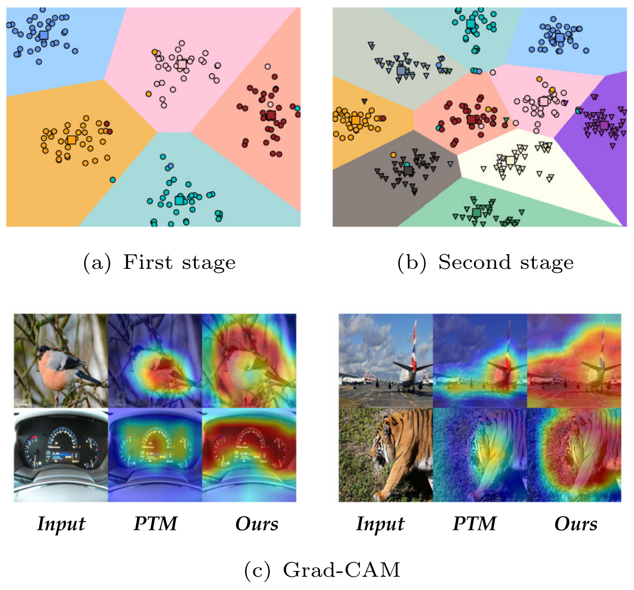


Fig. 7 **Top**: visualization of the decision boundary on CIFAR100 between two incremental tasks. Dots represent old classes, and triangles stand for new classes. Decision boundaries are shown with the shadow region. **Bottom**: Grad-CAM visualizations of PTM and APER. Important regions are highlighted with warm colors (Color figure online)

We also visualize the Grad-CAM (Selvaraju et al., 2017) results on OmniBenchmark dataset based on pre-trained ResNet18. Grad-CAM is utilized to highlight the critical regions in the image to predict the corresponding concept. The results are shown in Fig. 7(c), indicating APER concentrates more on the task-specific features than vanilla PTMs. Hence, visualizations verify the importance of adaptivity in PTM-based class-incremental learning.

5.5 Experiments with Multiple Domains

This paper mainly considers the CIL setting, where all the data are from the same domain. However, we also consider two challenging tasks to investigate the performance with significant domain gaps. (1): Domain-incremental learning with Office-Home dataset (Venkateswara et al., 2017). Office-Home is a benchmark dataset for domain adaptation, containing four domains, each consisting of 65 categories. The four domains are: art, clip-art, product, and real-world, with an average of around 70 images per class and a maximum of 99 images in a class. We follow (Wang et al., 2022b) to organize the domain-incremental learning scenario where each task contains the classes of a new domain. (2): Class-incremental learning with cross-domain data. We follow (Smith et al., 2023) and split DomainNet (Peng et al., 2019) into five tasks. Specifically, DomainNet is a dataset of common objects in six different domains. All domains

Table 3 Experiments on domain-incremental learning and cross-domain CIL

Method	Office-Home		DomainNet	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
L2P	71.61	59.48	57.90	57.59
DualPrompt	70.85	58.26	62.41	54.81
SimpleCIL	72.18	75.85	54.80	58.14
APER w/ Finetune	75.44	77.47	64.17	60.78
APER w/ VPT-Shallow	71.44	63.26	58.75	60.84
APER w/ VPT-Deep	71.06	61.27	60.38	58.31
APER w/ SSF	**80.81**	**83.14**	65.79	65.02
APER w/ Adapter	72.46	63.73	**67.50**	**66.90**

The best performance is shown in bold
All methods are implemented with pre-trained ViT-B/16-IN1K

include 345 classes of objects. The domains include clip-art, real, sketch, infograph, painting, and quickdraw. To construct a class-incremental learning setting, we split the dataset into five tasks, each containing 69 classes of individual domains.

The above settings contain datasets from multiple domains, and we report the experimental results in Table 3. We adopt the same pre-trained ViT-B/16-IN1K for all compared methods. Although APER is not specially designed with multiple stages, it still shows competitive performance against L2P and DualPrompt in these cross-domain tasks.

6 Conclusion

Learning with incremental classes is of great importance in real-world applications, which requires *adaptivity* for updating and *generalizability* for knowledge transfer. In this paper, we systematically revisit CIL with PTMs and draw three conclusions. Firstly, a frozen PTM can provide *generalizable* embeddings for CIL, enabling a prototype-based classifier to outperform the current state-of-the-art. Secondly, due to the distribution gap between pre-trained and downstream datasets, PTMs can be further harnessed to enhance their *adaptivity*. To this end, we propose APER, which can be orthogonally combined with any parameter-efficient tuning method to unify generalizability and adaptivity for CIL. Lastly, due to data overlapping, traditional ImageNet-based benchmarks are unsuitable for evaluation in the era of PTM. Hence, we propose four new benchmarks to evaluate PTM-based CIL methods. Extensive experiments verify APER's state-of-the-art performance. Future work includes exploring task-specific tuning methods and structures.

Limitations: The limitations are two-fold. Firstly, the model cannot make full use of exemplars since the adapted model should fully reflect the downstream features. It turns into exemplar-based CIL if sufficient old class instances are available, where adaptivity can be further addressed through data rehearsal. Secondly, since APER is only adapted with the first incremental stage, it shall face challenges when extensive domain gaps exist in the continual learning process, *i.e.*, domain-incremental learning. In summary, extending APER to these more challenging scenarios are interesting future works.

Appendix A More Results

A.1 Experiments on ImageNet

Traditional class-incremental learning mainly considers three benchmark datasets, *i.e.*, CIFAR100 (Krizhevsky et al., 2009), ImageNet100 and ImageNet1000 (Deng et al., 2009). Among them, ImageNet100 is a subset of ImageNet1000 containing 100 classes. However, current Vision Transformers are widely pre-trained with the ImageNet21K (Steiner et al., 2022) dataset (a superset of ImageNet1000). Consequently, incremental learning with its subset becomes less reasonable since the pre-trained model has already seen these classes.

To verify this claim, we experiment with ImageNet100 B0 Inc10 setting and report the results in Table 4. Specifically, we have two major conclusions from these results. First, all methods can achieve a competitive performance of around 90%, indicating that ImageNet100 is quite easy for the pre-trained model to tackle. Besides, SimpleCIL achieves an accuracy of 92%, indicating that the pre-trained model can handle the ImageNet100 dataset even without tuning. These experiments verify that ImageNet and its subsets are unsuitable for the incremental task due to the overlapping of pre-training data and downstream data.

A.2 Trainable Parameters

In this section, we further report the number of learnable parameters in various parameter-efficient tuning methods in Table 5. As we can infer from the table, APER and its

Table 4 Experiments on ImageNet100

Method	ImageNet100 B0 Inc10	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
L2P	92.45	89.54
DualPrompt	92.85	89.66
SimpleCIL	92.83	90.52
APER w/ Adapter	**94.79**	**92.78**

The best performance is shown in bold
All methods are implemented with pre-trained ViT-B/16-IN1K

Table 5 Number of trainable parameters in different methods

Method	Trainable parameters
L2P	84530
DualPrompt	291890
SimpleCIL	0
APER w/ VPT-Shallow	3840
APER w/ VPT-Deep	46080
APER w/ SSF	205825
APER w/ adapter	297408

variations only cost a tiny portion of trainable parameters. Besides, the cost is required only once since the model is finetuned with the first incremental stage using parameter-efficient tuning modules. By contrast, L2P and DualPrompt shall require more parameters when facing longer learning sequences since the small prompt pool may not be diverse enough to capture multiple tasks.

A.3 Module Trade-Off

In the main paper, we mainly consider the exemplar-free scenario where the model cannot save any exemplars. In this section, we can also treat the pre-trained model and adapted model as two separate modules for ensemble with the help of exemplars. We denote the output of pre-trained model as $f(\mathbf{x}) = W^\top \phi(\mathbf{x})$, and the output of the adapted model as $f^*(\mathbf{x}) = W^{*\top} \phi^*(\mathbf{x})$. The linear classifiers W and W^* are obtained via the prototype extraction. In this way, $f(\mathbf{x})$ corresponds to the generalizability while $f^*(\mathbf{x})$ stands for the adaptivity. Instead of concatenating the features, we consider the weighted ensemble between these modules:

$$\alpha f(\mathbf{x}) + f^*(\mathbf{x})\,, \tag{10}$$

where α is the trade-off parameter (scalar) to adjust the importance of different modules. To obtain α in a data-driven way, we consider using *auxiliary information from the exemplar set*, which contains a small portion of previous instances. Afterward, we freeze $f(\mathbf{x})$ and $f^*(\mathbf{x})$ and optimize α with the exemplar set. This enables learning a suitable trade-off parameter to capture the pre-trained and adapted models' capabilities. In the implementation, we find saving a small portion of exemplars is enough for the learning process, *e.g.*, 5 per class. We denote the inference with Eq. 10 as APER†, and report the results in Table 6. As we can infer from the table, utilizing the weighted ensemble strikes a balance between adaptivity and generalizability, which further improves the performance of APER by 3∼5%. However, it must be noted that APER†requires extra exemplars for the parameter optimization. Without such extra exemplars, the average ensemble (*i.e.*, $\alpha = 1$) even shows inferior performance than the vanilla APER on VTAB.

A.4 Large Base Classes

In this section, we report the detailed performance of different methods with large base classes in Table 7. As we can infer from the table, APER consistently achieves the best performance in the current setting.

A.5 Influence of Hyper-Parameters

In this section, we explore the influence of hyper-parameters in APER. Specifically, since APER is optimized with parameter-efficient tuning, the only hyper-parameters come from these tuning methods, *i.e.*, the prompt length p in VPT and the projection dim r in Adapter. We train the model on CIFAR100 B50 Inc5 with pre-trained ViT-B/16-IN1K.

Firstly, we investigate the influence of the prompt length in Fig. 8. The figure shows that APER's performance is robust with the change of the prompt length. Therefore, we use $p = 5$ as a default parameter for all settings. Similarly, we observe similar phenomena in Fig. 9, where the performance is robust with the change of the projection dimension. As a result, we set $r = 16$ as a default parameter for all settings.

A.6 Combining with CLIP

Apart from the typical backbone Vision Transformer adopted in the main paper, in this section, we also combine APER with CLIP (Radford et al., 2021) to show its compatibility to enhance CLIP's incremental learning ability.

We treat CoOp (Zhou et al., 2022d) as the baseline and combine APER w/ CoOp on CIFAR100 B0 Inc10 and

Table 6 Experiments on the generalizability-adaptivity trade-off

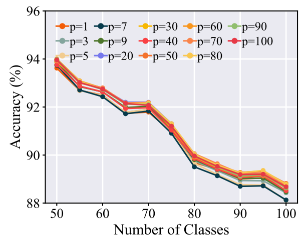
Method	Exemplar	OmniBenchmark B0 Inc30		VTAB B0 Inc10	
		$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
APER w/ Finetune	0	73.02	65.03	87.47	80.44
APER† w/ Finetune, $\alpha = 1$	0	73.42	65.16	86.29	80.03
APER† w/ Finetune	5/class	**76.03**	**67.39**	**91.74**	**85.66**

The best performance is shown in bold
APER† indicates further tuning APER via weighted ensemble

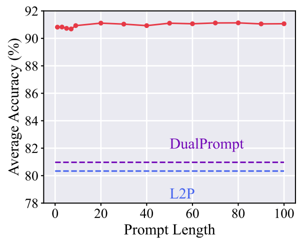
Table 7 Experiments with vast base classes on seven datasets with ViT-B/16-IN21K as the backbone

Method	CIFAR B50 Inc5		CUB B100 Inc5		IN-R B100 Inc5		IN-A B100 Inc5		ObjNet B100 Inc10		OmniBench B150 Inc15	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	48.91	35.10	52.88	34.61	40.09	34.10	13.12	6.19	35.43	30.78	50.90	43.83
Finetune Adapter (Chen et al., 2022)	88.58	84.54	73.12	66.07	69.29	65.53	49.89	42.86	62.32	53.64	72.65	67.34
LwF (Li & Hoiem, 2017)	81.69	66.49	53.80	30.28	57.37	49.42	24.62	19.63	33.74	24.88	43.37	39.80
SDC (Lu et al., 2020)	79.12	69.54	63.36	49.70	65.00	38.80	34.56	26.32	54.91	42.96	65.79	54.62
L2P (Wang et al., 2022b)	81.27	70.96	70.53	55.85	60.55	46.94	44.56	35.31	54.26	41.63	63.96	50.58
DualPrompt (Wang et al., 2022a)	82.12	72.04	74.57	62.32	64.96	57.57	43.26	32.63	61.61	53.18	72.62	63.29
CODA-Prompt (Smith et al., 2023)	80.26	70.49	66.12	51.82	67.79	61.63	54.52	46.68	55.77	45.49	64.73	53.98
SimpleCIL	83.78	81.26	88.93	86.73	56.88	54.55	53.26	48.91	58.61	53.59	75.25	73.15
Aper w/ Finetune	83.78	81.26	86.96	84.10	75.79	69.35	52.78	48.19	61.63	53.35	**78.71**	74.50
Aper w/ VPT-Shallow	86.25	84.13	88.66	86.64	64.89	62.58	52.81	48.26	62.32	57.24	75.26	73.33
Aper w/ VPT-Deep	91.04	88.71	89.50	87.53	74.15	71.15	63.23	58.39	67.14	62.16	77.55	75.46
Aper w/ SSF	89.91	87.17	89.15	87.23	**76.42**	**72.87**	**63.73**	**58.66**	67.81	62.49	76.20	74.20
Aper w/ Adapter	**91.89**	**89.67**	**89.84**	**87.70**	75.47	72.63	61.73	56.75	**68.65**	**63.55**	77.60	**75.54**

‘IN-R/A’ stands for ‘ImageNet-R/A,’ ‘ObjNet’ stands for ‘ObjectNet,’ and ‘OmniBench’ stands for ‘OmniBenchmark.’ The best performance is shown in bold

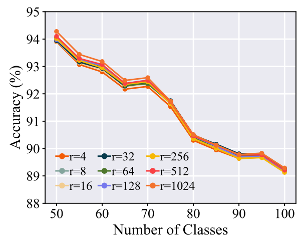


(a) Performance curve

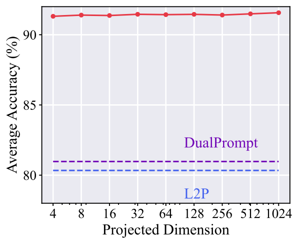


(b) Accuracy-Prompt length curve

Fig. 8 Influence of prompt length. The performance of APER is robust with the change of the prompt length, and we set $p = 5$ as the default setting



(a) Performance curve



(b) Accuracy-Projected dimension curve

Fig. 9 Influence of projected dimension in adapter. The performance of APER is robust with the change of the projected dimension, and we set $r = 16$ as the default setting

ImageNet-R B0 Inc20 settings. Specifically, the CLIP model possesses two branches of encoders, *i.e.*, the visual encoder and textual encoder, and utilizes the matching target with max similarity for prediction. CoOp, the most popular tuning technique in CILP, finetunes a set of prompts and freezes the other parameters to adapt it to downstream tasks. We only learn the prompts of CoOp in the first stage and conduct inference for the following incremental tasks. We implement these methods using CLIP with OpenAI weights (Radford et al., 2021) as initialization. For APER w/ CoOp, we utilize CoOp to finetune the model with the first incremental stage and adopt Eq. 10 for inference.

Table 10 Experiments on ImageNet-R Base0 Inc20

Method	ImageNet-R Base0 Inc20	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
CPP	68.25	64.45
APER w/ SSF	**75.62**	**67.12**

The best performance is shown in bold
All methods are implemented with pre-trained ViT-B/16-IN21K

As shown in Tables 8 and 9, CoOp shows poor performance in the incremental learning process when the prompts are only tuned for the first stage. However, when combining APER w/ CoOp, the incremental learning performance can be further improved, indicating APER can tackle various backbones and improve the incremental learning ability.

A.7 ImageNet-R B0 Inc20

In this section, we compare APER to CPP with ImageNet-R Base0 Inc20 setting. As shown in Table 10, APER still shows substantial improvement to CPP (3% by last accuracy and 7% by average accuracy), indicating its strong performance against the state-of-the-art.

A.8 Ablations on ImageNet-A

In the main paper, we show the improvement of best APER variation to SimpleCIL with different backbones on ImageNet-R. In this section, we report the results with ImageNet-A in Fig. 10. As shown in the figure, APER still consistently improves the performance of SimpleCIL in this new benchmark. We also find that the improvement for large models like ViT-L is also 6%, indicating that APER also shows stable improvements to larger models.

Table 8 Experiments on pre-trained CLIP under CIFAR100 Base0 Inc10 setting. APER can also be combined with CoOp to improve CLIP's incremental learning ability

Method	Accuracy in each task (%) ↑									
	1	2	3	4	5	6	7	8	9	10
CoOp	97.50	55.75	48.17	39.45	37.18	32.45	28.76	25.76	23.82	22.41
APER w/ CoOp	97.70	91.45	88.40	84.00	80.16	78.38	76.99	74.58	71.72	70.34

Table 9 Experiments on pre-trained CLIP under ImageNet-R Base0 Inc20 setting. APER can also be combined with CoOp to improve CLIP's incremental learning ability

Method	Accuracy in each task (%) ↑									
	1	2	3	4	5	6	7	8	9	10
CoOp	94.63	73.81	62.48	55.40	50.82	49.08	46.32	43.46	40.82	39.13
APER w/ CoOp	94.94	88.27	84.54	81.06	78.93	77.17	75.01	73.19	72.03	71.42

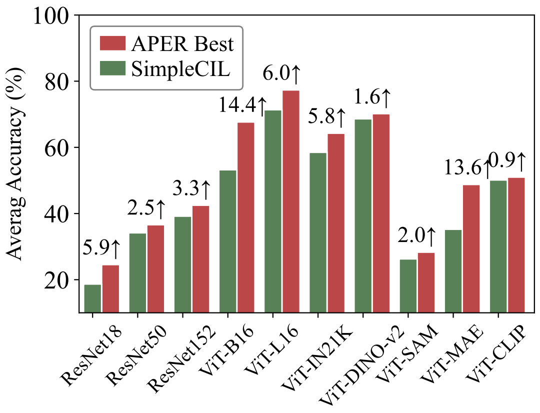


Fig. 10 CIL with different kinds of PTMs on ImageNet-A Base0 Inc10. APER consistently improves the performance of different PTMs. We report the best variation of APER in the figure and its relative improvement above SimpleCIL in the figure

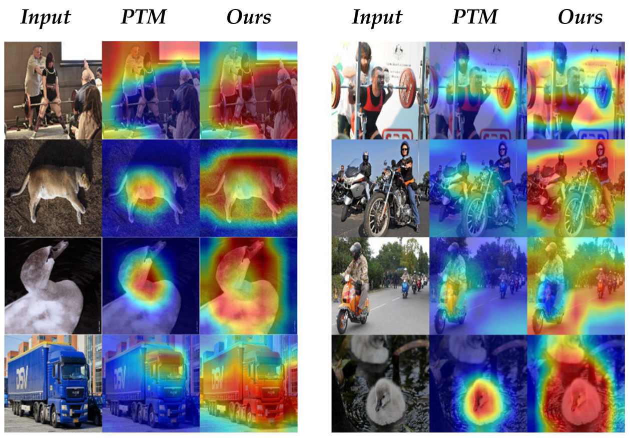


Fig. 11 Supplied Grad-CAM results. Important regions are highlighted with warm colors. Compared to PTM, APER concentrates more on task-specific features (Color figure online)

Table 11 Average and last performance comparison on ObjectNet300 with ViT-B/16-IN21K as the backbone

Method	ObjectNet300 B0 Inc30	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	28.00	10.46
Finetune Adapter (Chen et al., 2022)	60.03	41.53
LwF (Li & Hoiem, 2017)	29.30	10.44
L2P (Wang et al., 2022b)	59.36	47.62
DualPrompt (Wang et al., 2022a)	58.40	49.30
CODA-Prompt (Smith et al., 2023)	61.72	52.41
CPP (Li et al., 2024)	53.33	45.75
LAE (Gao et al., 2023)	62.85	53.20
SimpleCIL	57.30	46.88
APER w/ Finetune	58.88	48.42
APER w/ VPT-Shallow	60.53	50.01
APER w/ VPT-Deep	63.94	53.11
APER w/ SSF	**65.09**	53.80
APER w/ Adapter	64.48	**53.83**

The best performance is shown in bold

A.9 ObjectNet300

In the main paper, we sample 200 out of the 313 classes in ObjectNet for class-incremental learning. In this section, we also sample 300 classes from it, denoted as ObjectNet300. We conduct experiments against other state-of-the-art methods and report the results in Table 11. Results indicate that APER still performs better than competitors in this more "natural" setting.

A.10 Grad-CAM

In this section, we have show more Grad-CAM results. As shown in Fig. 11, APER concentrates more on task-specific features than vanilla PTM, confirming the effectiveness of model adaptation in capturing task-specific features.

Funding This work is partially supported by National Science and Technology Major Project (2022ZD0114805), Fundamental Research Funds for the Central Universities (2024300373), NSFC (62376118, 62006112, 62250069, 61921006), Collaborative Innovation Center of Novel Software Technology and Industrialization, China Scholarship Council, Ministry of Education, Singapore, under its MOE AcRF Tier 2 (MOET2EP20221- 0012), NTU NAP, and under the RIE2020 Industry Alignment Fund - Industry Collaboration Projects (IAF-ICP) Funding Initiative.

Code Availability Code is available at https://github.com/zhoudw-zdw/RevisitingCIL.

References

Akbari, H., Kondratyuk, D., Cui, Y., Hornung, R., Wang, H., & Adam, H. (2023) Alternating gradient descent and mixture-of-experts for integrated multimodal perception. *arXiv preprint* arXiv:2305.06324.
Alfassy, A., Arbelle, A., Halimi, O., Harary, S., Herzig, R., Schwartz, E., Panda, R., Dolfi, M., Auer, C., Staar, P., et al. (2022). Feta: Towards specializing foundational models for expert task applications. *NeurIPS, 35*, 29873–29888.
Aljundi, R., Lin, M., Goujaud, B., & Bengio, Y. (2019). Gradient based sample selection for online continual learning. *In NeurIPS, 32*, 11816–11825.
Bahng, H., Jahanian, A., Sankaranarayanan, S. & Isola, P. (2022). Visual prompting: Modifying pixel space to adapt pre-trained models. *arXiv preprint* arXiv:2203.17274.
Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J. & Katz, B. (2019). Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *NeurIPS, 32*.
Belouadah E & Popescu A. (2019). Il2m: Class incremental learning with dual memory. *In ICCV,* pp. 583–592.
Cai X, Lou J, Bu J, Dong J, Wang H, Yu H. (2024). Single depth image 3d face reconstruction via domain adaptive learning. *Frontiers of Computer Science, 18*(1).

Cermelli F, Fontanel D, Tavera A, Ciccone M, Caputo B. (2022). Incremental learning in semantic segmentation from image labels. *In CVPR,* pp. 4371–4381.

Chaudhry, A., Dokania, P. K., Ajanthan, T., & Torr, P. H. (2018). Understanding forgetting and intransigence. Riemannian walk for incremental learning. *In ECCV,* pp.532–547.

Chen, S., Ge, C., Tong, Z., Wang, J., Song, Y., Wang, J., & Luo, P. (2022). Adaptformer: Adapting vision transformers for scalable visual recognition. *NeurIPS,* pp. 16664–16678.

Chen, X., Hsieh C.-J., Gong, B. (2022). When vision transformers outperform resnets without pre-training or strong data augmentations. In *ICLR*

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

Dhar, P., Singh, R. V., Peng, K.-C., Ziyan, W., & Chellappa, R. (2019). Learning without memorizing. *In CVPR,* pp. 5138–5146.

Ding, Z., Xie, H., Li, P., & Xin, X. (2023). A structural developmental neural network with information saturation for continual unsupervised learning. *CAAI Transactions on Intelligence Technology, 8*(3), 780–795.

Dong, J., Cong, Y., Sun, G., Ma, B., & Wang, L. (2021). I3dol: Incremental 3d object learning without catastrophic forgetting. *In AAAI,* pp. 6066–6074.

Dong, S., Hong, X., Tao, X., Chang, X., Wei, X., & Gong, Y. (2021). Few-shot class-incremental learning via relation knowledge distillation. *In AAAI,* pp. 1255–1263.

Dong, J., Liang, W., Cong, Y., & Sun, G. (2023). Heterogeneous forgetting compensation for class-incremental learning. *In ICCV,* pp. 11742–11751.

Dong, J., Wang, L., Fang, Z., Sun, G., Shichao, X., Wang, X., & Zhu, Q. (2022). *Federated class-incremental learning. In CVPR,* pp. 10164–10173.

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*.

Douillard, A., Cord, M., Ollion, C., Robert, T., & Valle, E. (2020). Podnet: Pooled outputs distillation for small-tasks incremental learning. *In ECCV,* pp. 86–102.

Douillard, A., Ramé, A., Couairon, G., & Cord, M. (2022). Dytox: Transformers for continual learning with dynamic token expansion. *In CVPR,* pp. 9285–9295.

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

Gao, R., & Liu, W. (2023). Ddgr: Continual learning with deep diffusion-based generative replay. *In ICML,* pp. 10744–10763.

Gao, Q., Zhao, C., Ghanem, B., & Zhang, J. (2022). R-DFCIL: relation-guided representation learning for data-free class incremental learning. *In ECCV,* pp. 423–439.

Gao, Q., Zhao, C., Sun, Y., Xi, T., Zhang, G., Ghanem, B., & Zhang, J. (2023). A unified continual learning framework with general parameter-efficient tuning. *In ICCV,* pp. 11483–11493.

Gomes, H. M., Barddal, J. P., Enembreck, F., & Bifet, A. (2017). A survey on ensemble learning for data stream classification. *CSUR, 50*(2), 1–36.

Grossberg, S. .T. . (2012). *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control* (Vol. 70). Springer Science & Business Media.

Han, X., Zhang, Z., Ding, N., Yuxian, G., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L., et al. (2021). Pre-trained models: Past, present and future. *AI Open, 2*, 225–250.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022). Masked autoencoders are scalable vision learners. *In CVPR,* pp. 16000–16009.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In CVPR,* pp. 770–778.

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., & Neubig, G. (2022). Towards a unified view of parameter-efficient transfer learning. In *ICLR*.

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

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., & Song, D. (2021). *Natural adversarial examples. In CVPR,* pp. 15262–15271.

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

Hou, S., Pan, X, L. C. Change, Wang, Z., & Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. *In CVPR,* pp. 831–839.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., & Gelly, Sylvain. (2019). Parameter-efficient transfer learning for nlp. *In ICML,* pp. 2790–2799.

Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, Wang L, Chen W. Lora et al. (2022). Lora: Low-rank adaptation of large language models. In *ICLR*.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In ICML,* pp. 448–456.

Iscen, A., Zhang, J., Lazebnik, S., & Schmid, C. (2020). Memory-efficient incremental learning through feature adaptation. *In ECCV,* pp. 699–715.

Jaimes, A., & Sebe, N. (2007). Multimodal human-computer interaction: A survey. *Computer Vision and Image Understanding, 108*(1–2), 116–134.

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S. J., Hariharan, B., & Lim, S.-N. (2022). *Visual prompt tuning. In ECCV,56*, pp. 709–727. Springer.

Jiahui, Y., Wang, Z., Vasudevan, V., Yeung, L., & Seyedhosseini, M. (2022). *and Yonghui Wu*. Coca: Contrastive captioners are image-text foundation models. Transactions on Machine Learning Research.

Jiang, J., Cetin, E., & Celiktutan, O. (2021). Ib-drr-incremental learning with information-back discrete representation replay. *In CVPRW,* pp. 3533–3542.

Jie, S., & Deng, Z.-H. (2023). Fact: Factor-tuning for lightweight adaptation on vision transformer. *In AAAI,* pp. 1060–1068.

Ju, X., & Zhu, Z. (2018). *Reinforced continual learning. In NeurIPS,* pp. 899–908.

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

Kang, M., Park, J., & Han, B. (2022). Class-incremental learning by knowledge distillation with adaptive feature consolidation. *In CVPR,* pp. 16071–16080.

Khattak, M. U., Rasheed, H., Maaz, M., Khan, S., & Khan, F. S. (2023). Multi-modal prompt learning. *Maple. In CVPR,* 19113–19122.

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.

Kumar, A., & Raghunathan, A. (2022). *Robbie Matthew Jones, Tengyu Ma, and Percy Liang*. In ICLR: Fine-tuning can distort pretrained features and underperform out-of-distribution.

Li, X.L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *In ACL,* 4582–4597.

Li, L., Peng, J., Chen, H., Gao, C., & Yang, X. (2023). How to configure good in-context sequence for visual question answering. *arXiv preprint* arXiv:2312.01571.

Li, Z., Zhao, L., Zhang, Z., Zhang, H., Liu, D., Liu, T., & Metaxas, D.N. (2024). Steering prototypes with prompt-tuning for rehearsal-free continual learning. *In WACV,* 2523–2533.

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

Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE Transactions on pattern analysis and machine intelligence, 40*(12), 2935–2947.

Li, L., Huang, J., Chang, L., Weng, J., Chen, J., & Li, J. (2023). Dpps: A novel dual privacy-preserving scheme for enhancing query privacy in continuous location-based services. *Frontiers of Computer Science, 17*(5), 175814.

Liu, H., Ji, R., Yongjian, W., Huang, F., & Zhang, B. (2017). Cross-modality binary code learning via fusion similarity hashing. *In CVPR,* pp. 7380–7388.

Liu, M., Roy, S., Zhong, Z., Sebe, N., & Ricci, E. (2023). Large-scale pre-trained models are surprisingly strong in incremental novel class discovery. arXiv:2303.15975

Liu, Y., Yuting, S., Liu, A.-A., Schiele, B., & Sun, Q. (2020). Mnemonics training: Multi-class incremental learning without forgetting. *In CVPR,* pp. 12245–12254.

Lu Y., Twardowski, B., Liu, X., Herranz, L., Wang, K., Cheng, Y., Jui, S., & van de Weijer, J. (2020). Semantic drift compensation for class-incremental learning. *In CVPR,* pp. 6982–6991.

Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., & Van De Weijer, J. (2022). Class-incremental learning: Survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 45*(5), 5513–5533.

Mermillod, M., Bugaiska, A., & Bonin, P. (2013). The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology, 4*, 54654.

Mirzadeh, S. I., Farajtabar, M., Pascanu, R., & Ghasemzadeh, H. (2020). Understanding the role of training regimes in continual learning. *NeurIPS, 33*, 7308–7320.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. (2023). Dinov2: Learning robust visual features without supervision. *arXiv preprint* arXiv:2304.07193.

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,* pp.8026–8037.

Pearson, K. (1901). Liii. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin PPhilosophical Magazine and Journal of Science, 2*(11), 559–572.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., & Wang, B. (2019). Moment matching for multi-source domain adaptation. *In ICCV,* pp. 1406–1415.

Perez-Rua, J.-M., & Zhu, X. (2020). Timothy M Hospedales, and Tao Xiang. *Incremental few-shot object detection. In CVPR,* pp. 13846–13855.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., & Gurevych, I. (2021). Adapterfusion: Non-destructive task composition for transfer learning. *In ACL,* pp. 487–503.

Pham, Q., & Liu, C. (2022). *and HOI Steven*. Continual normalization: Rethinking batch normalization for online continual learning. In ICLR.

Radford, A., Kim, Jong W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *In ICML,* pp. 8748–8763. PMLR.

Rebuffi, S.-A., Bilen, H., & Vedaldi, A. (2017). Learning multiple visual domains with residual adapters. *NIPS,* pp. 506–516.

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

Ridnik, T., Ben-Baruch, E., Noy, A., & Zelnik-Manor, L. (2021). Imagenet-21k pretraining for the masses. *arXiv preprint* arXiv:2104.10972

Roy, S., Liu, M., Zhong, Z., Sebe, N., & Ricci, E. (2022). *Class-incremental novel class discovery. In ECCV,* pp. 317–333.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. *In ICCV,* pp.618–626.

Shin, H., Lee, J. K., Kim, J., & Kim, J. (2017). Continual learning with deep generative replay. *In NIPS,* pp. 2990–2999.

Shi Y, Zhou K, Liang J, Jiang Z, Feng J, Torr PH, Bai S, & Tan VY. (2022). Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. *In CVPR,* pp. 16722–16731.

Simon, C., Koniusz, P., & Harandi, M. (2021). On learning the geodesic path for incremental learning. *In CVPR,* pp. 1591–1600.

Smith, J., Hsu, Y.-C., Balloch, J., Shen, Y., Jin, H., & Kira, Z. (2021). Always be dreaming: A new approach for data-free class-incremental learning. *In ICCV,* pp. 9374–9384.

Smith, J.S., Karlinsky, L., Gutta, V., Cascante-Bonilla, P., Kim, D., Arbelle, A., Panda, R., Feris, R. & Kira, Z. (2023). Continual decomposed attention-based prompting for rehearsal-free continual learning. *Coda-prompt. In CVPR,* pp. 11909–11919.

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *In NIPS, 56*, 4080–4090.

Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J. & Beyer, L. (2022). How to train your vit? data, augmentation, and regularization in vision transformers. *Transactions on Machine Learning Research*.

Sun, G., Liang, W., Dong, J., Li, J., & Ding, Z. (2024). *and Yang Cong*. Create your world: Lifelong text-to-image diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Sun, H.-L., Zhou, D.-W., Ye, H.-J., & Zhan, D.-C. (2023). Pilot: A pre-trained model-based continual learning toolbox. *arXiv preprint* arXiv:2309.07117.

Tao, X., Hong, X., Chang, X., Dong, S., Wei, X., & Gong, Y. (2020). *Few-shot class-incremental learning. In CVPR,* pp. 12183–12192.

Tschannen, M., Mustafa, B., & Houlsby, N. (2022). Image-and-language understanding from pixels only. *arXiv preprint* arXiv:2212.08045.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research,9*(11).

Venkateswara, H., Eusebio, J., Chakraborty, S., & Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. *In CVPR,* pp. 5018–5027.

Villa, A., Alcázar, J.L., Alfarra, M., Alhamoud, K., Hurtado, J., Heilbron, F.C., Soto, A. and Ghanem, B. (2023) Prompting for video continual learning. Pivot. *In CVPR,* pp. 24214–24223.

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

Wang, Y., Ma, Z., Huang, Z., Wang, Y., Su, Z., & Hong, X. (2023). Isolation and impartial aggregation: A paradigm of incremental learning without interference. *In AAAI,* pp. 10209–10217.

Wang, Z., Wang, Z., Zheng, Y., Chuang, Y.-Y., & Satoh, S. (2019). Learning to reduce dual-level discrepancy for infrared-visible person re-identification. *In CVPR,* pp. 618–626.

Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.-Y., Ren, X., Guolong, S., Perot, V., Dy, J., et al. (2022). Dualprompt:

Complementary prompting for rehearsal-free continual learning. *In ECCV,* pp. 631–648.

Wang, Z., Zhang, Z., Lee, C.-Y., Zhang, H., Sun, R., Ren, X., Guolong, S., Perot, V., Dy, J., & Pfister, T. (2022). Learning to prompt for continual learning. *In CVPR,* pp. 139–149.

Wang, L., Zhang, X., Su, H., & Zhu, J. (2023). A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint* arXiv:2302.00487.

Wang, F.-Y., Zhou, D.-W., Liu, L., Ye, H.-J., Bian, Y., & Zhan, D.-C. (2023). *and Peilin Zhao*. BEEF: Bi-compatible class-incremental learning via energy-based expansion and fusion. In ICLR.

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

Wang, Q.-W., Zhou, D.-W., Zhang, Y.-K., Zhan, D.-C., & Ye, H.-J. (2023). Few-shot class-incremental learning via training-free prototype calibration. *NeurIPS,36*.

Wang, Y., Huang, Z., & Hong, X. (2022). S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *NeurIPS, 35*, 5682–5695.

Wightman, R. (2019). Pytorch image models. https://github.com/rwightman/pytorch-image-models.

Xinting, H., Tang, K., Miao, C., Hua, X.-S., & Zhang, H. (2021). Distilling causal effect of data in class-incremental learning. *In CVPR,* pp. 3957–3966.

Yan, S., Xie, J., & He, X. (2021). Der: Dynamically expandable representation for class incremental learning. *In CVPR,* pp. 3014–3023.

Yang, X., Yongliang, W., Yang, M., Chen, H., & Geng, X. (2024). Exploring diverse in-context configurations for image captioning. *NeurIPS,36*.

Yang, M.-H. (2002). Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(1), 34–58.

Yoon, J., Yang, E., Lee, J., & Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. In *ICLR*.

You, K., Kou, Z., Long, M., & Wang, J. (2020). *Co-tuning for transfer learning. NeurIPS,* pp. 17236–17246.

Yuan, L., Chen, D., Chen, Y.-L., Codella, N., Dai, X., Gao, J., Hu, H., Huang, X., Li, B., Li, C., et al. (2021). Florence: A new foundation model for computer vision. *arXiv preprint* arXiv:2111.11432.

Yue, W., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., & Yun, F. (2019). *Large scale incremental learning. In CVPR,* pp. 374–382.

Zaken, E. B., Ravfogel, S., & Goldberg, Y. (2022). Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *In ACL,* pp. 1–9.

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A., 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., & Wei, Y. (2023). Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. *In ICCV,* pp. 19148–19158.

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

Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., & Kuo J. C.-C. (2020). Class-incremental learning via deep model consolidation. *In WACV,* pp. 1131–1140.

Zhang, Y., Zhou, K., & Liu, Z. (2022). Neural prompt search. *arXiv preprint* arXiv:2206.04673.

Zhang, T., Zhang, H., & Li, X. (2023). Vision-audio fusion slam in dynamic environments. *CAAI Transactions on Intelligence Technology, 8*(4), 1364–1373.

Zhao, H., Qin, X., Shihao, S., Yongjian, F., Lin, Z., & Li, X. (2021). When video classification meets incremental classes. *In ACM MM,* pp. 880–889.

Zhao, B., Xiao, X., Gan, G., Zhang, B., & Xia, S.-T. (2020). Maintaining discrimination and fairness in class incremental learning. *In CVPR,* pp. 13208–13217.

Zhao, H., Wang, H., Yongjian, F., Fei, W., & Li, X. (2021). Memory-efficient class-incremental learning for image classification. *IEEE Transactions on Neural Networks and Learning Systems, 33*(10), 5966–5977.

Zhong, Z., Fini, E., Roy, S., Luo, Z., Ricci, E., & Sebe, N. (2021). Neighborhood contrastive learning for novel class discovery. *In CVPR,* pp. 10867–10875.

Zhou, D.-W., Wang, Q.-W., Qi, Z.-H., Ye, H.-J., Zhan, D.-C., & Liu, Z. (2023). Class-incremental learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Zhou, D.-W., Sun, H.-L., Ning, J., & Ye, H.-J. (2024). *and De-Chuan Zhan*. Continual learning with pre-trained models: A survey. In IJCAI.

Zhou, D. W., Wang, Q. W., Ye, H. J., & Zhan, D. C. (2023). A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *ICLR*.

Zhou, D.-W., Wang, F.-Y., Ye, H.-J., Ma, L., Shiliang, P., & Zhan, D.-C. (2022). Forward compatible few-shot class-incremental learning. *In CVPR,* pp. 9046–9056.

Zhou, K., Yang, J., Loy, C. C., & Liu, Z. (2022). Conditional prompt learning for vision-language models. *In CVPR,* pp. 16816–16825.

Zhou, K., Liu, Z., Qiao, Y., Xiang, T., & Loy, C. C. (2022). Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 45*(4), 4396–4415.

Zhou, K., Yang, J., Loy, C. C., & Liu, Z. (2022). Learning to prompt for vision-language models. *International Journal of Computer Vision, 130*(9), 2337–2348.

Zhu, F., Zhang, X.-Y., Wang, C., Yin, F., & Liu, C.-L. (2021). Prototype augmentation and self-supervision for incremental learning. *In CVPR,* pp. 5871–5880.