

Integrating Task-Specific and Universal Adapters for Pre-Trained Model-based Class-Incremental Learning

Yan Wang, Da-Wei Zhou^(✉), Han-Jia Ye

School of Artificial Intelligence, Nanjing University

National Key Laboratory for Novel Software Technology, Nanjing University

{wangy, zhoudw, yehj}@lamda.nju.edu.cn

Abstract

Class-Incremental Learning (CIL) requires a learning system to continually learn new classes without forgetting. Existing pre-trained model-based CIL methods often freeze the pre-trained network and adapt to incremental tasks using additional lightweight modules such as adapters. However, incorrect module selection during inference hurts performance, and task-specific modules often overlook shared general knowledge, leading to errors on distinguishing between similar classes across tasks. To address the aforementioned challenges, we propose integrating Task-Specific and Universal Adapters (TUNA) in this paper. Specifically, we train task-specific adapters to capture the most crucial features relevant to their respective tasks and introduce an entropy-based selection mechanism to choose the most suitable adapter. Furthermore, we leverage an adapter fusion strategy to construct a universal adapter, which encodes the most discriminative features shared across tasks. We combine task-specific and universal adapter predictions to harness both specialized and general knowledge during inference. Extensive experiments on various benchmark datasets demonstrate the state-of-the-art performance of our approach. Code is available at <https://github.com/LAMDA-CL/ICCV2025-TUNA>.

1. Introduction

The advent of deep learning leads to the remarkable performance of deep neural networks in practical applications [8, 9, 14, 55]. However, in real-world scenarios, data often arrive in a continuous stream, necessitating a learning system capable of progressively assimilating knowledge of emerging classes, a process known as class-incremental learning (CIL) [40]. CIL faces a formidable challenge: the process of acquiring new classes often results in the erosion of previously learned knowledge, precipitating a phe-

nomenon known as catastrophic forgetting of established features [13]. Correspondingly, recent breakthroughs in pre-training [17] have prompted the research community to leverage pre-trained models (PTMs) as a means to mitigate the issue of forgetting [45, 49, 50]. Leveraging vast datasets and considerable computing resources, PTMs naturally produce features with strong generalization capabilities. As a result, the development of a robust CIL methodology that harnesses the power of PTMs while mitigating catastrophic forgetting has attracted considerable attention from the research community [48, 64–66].

Owing to the remarkable generalization properties of PTMs, existing approaches frequently involve freezing the pre-trained weights and adapting to incremental tasks through the integration of lightweight modules [7, 25, 33]. Many of these methods rely on visual prompt tuning [43, 49]. During training, they learn task-specific prompt parameters and a set of keys. These keys are later used for task selection through query-key matching during inference. However, these methods suffer from two drawbacks. First, continual learning requires models to dynamically adapt to a sequence of tasks while maintaining stability on previous ones. Current prompt-based methods rely on accurate retrieval of task-specific prompts during inference. However, incorrect key matching, especially in scenarios with task ambiguity or distribution shifts, can lead to the selection of irrelevant prompts, degrading performance. Second, these methods predominantly concentrate on the acquisition of task-specific knowledge and ignore the general knowledge shared between different tasks. Thus, they tend to make mistakes when distinguishing between highly similar classes across different tasks.

To overcome the challenges outlined above, we introduce integrating Task-Specific and Universal Adapters (TUNA) in this paper, which explicitly disentangles continual learning into two complementary components: (1) specialized adapters that extract task-discriminative features, and (2) a universal adapter that consolidates cross-task shared knowledge through fusion. This decomposition

[†]Correspondence to: Da-Wei Zhou (zhoudw@lamda.nju.edu.cn)

not only mitigates task interference but also enables more robust generalization to semantically overlapping classes.

First, we use orthogonal loss to train task-specific adapters. To enhance the accuracy of module selection, we introduce an entropy-based adapter selection strategy that routes inputs to the most relevant task-specific adapter based on prediction uncertainty, eliminating reliance on brittle key-query matching. Second, for knowledge consolidation, we leverage an adapter fusion technique that merges task-specific adapters into a universal adapter, preserving shared features while minimizing redundancy. During inference, our framework leverages both task-specific and universal adapters in a coordinated manner. Our comprehensive experiments validate that the proposed method achieves state-of-the-art results across benchmark datasets, demonstrating notable improvements on challenging datasets such as ImageNet-A and ObjectNet.

2. Related Work

Class-Incremental Learning (CIL): necessitates a learning system capable of continuously assimilating new class information while preserving previously acquired knowledge without forgetting [16, 31, 34, 35, 61, 62]. This paradigm can be broadly categorized into several categories. Data rehearsal-based methods [3, 5, 6, 41, 60] involve carefully selecting and reintroducing exemplars from earlier classes during the acquisition of new classes. Knowledge distillation-based methods [11, 21, 27, 32, 42] try to establish a mapping between the model from previous stages and the current model through the process of knowledge distillation [20]. These mappings, represented as logits or feature representations, assist the incremental model in retaining essential characteristics from earlier phases during updating. Model rectification-based methods [1, 39, 52, 56, 59] seek to rectify the inductive bias inherent in incremental models, ensuring unbiased predictions during the updating process. Moreover, parameter regularization-based methods [2, 29, 30, 57] impose regularization penalties on the drift of crucial parameters throughout model adaptation, thereby safeguarding earlier knowledge. Expandable networks have recently shown strong performance in incremental learning [12, 23, 46, 53]. These methods preserve the original backbone and initialize a new one for each task, combining their outputs into a large feature map and training a classifier with exemplars for calibration.

Pre-Trained Model-Based CIL: is now a hot topic in today’s CIL field. Most pre-trained model-based CIL methods utilize the parameter-efficient fine-tuning mechanism to adapt the model efficiently while keeping the pre-trained model frozen. L2P [50] leverages a pre-trained model and dynamically learns a prompt pool to guide the model in addressing specific tasks. DualPrompt [49] introduces a novel approach by learning two mutually independent prompt

spaces: the general prompt and the expert prompt, which encode task-invariant and task-specific knowledge, respectively. CODA-Prompt [43] presents a decomposition-based, attention-driven continual learning prompting method, offering a significantly larger learning capacity compared to existing prompt-based techniques. Instead of directly optimizing prompt parameters, DAP [26] designs prompt generators to generate instance-specific information in prompts. SLCA [58] employs distinct learning rates for the backbone and classifier, it also models class-wise feature distributions [67] and replays them to calibrate the classifier. APER [64] proposes constructing the classifier by merging embeddings from both the pre-trained model and the adapted downstream model. EASE [65] innovatively concatenates feature representations from multiple task-specific backbones, further enhancing model capabilities. Furthermore, RanPAC [37] introduces a random projection approach that constructs robust high-dimensional randomized features, proving effective for continual learning tasks.

3. Preliminaries

In this section, we introduce the background of class-incremental learning and corresponding baselines.

3.1. Class-Incremental Learning

CIL focuses on continuously learning from evolving data streams that introduce new classes, while preserving the knowledge of previously encountered classes to construct a unified classifier [40]. Consider a series of T training stages, represented as $\{\mathcal{D}^1, \mathcal{D}^2, \dots, \mathcal{D}^T\}$, where $\mathcal{D}^t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_t}$ denotes the t -th incremental stage containing n_t instances. Correspondingly, the testing set is denoted as $\{\mathcal{D}_t^1, \mathcal{D}_t^2, \dots, \mathcal{D}_t^T\}$. Each training instance $\mathbf{x}_i^t \in \mathcal{D}^t$ is associated with a class label $y_i^t \in Y_t$. Here, Y_t defines the set of labels for training task t , and it is guaranteed that $Y_t \cap Y_{t'} = \emptyset$ for any $t \neq t'$. In this paper, we follow the **exemplar-free setting** in [65], which means that no historical exemplars from previous classes are used. Consequently, the model only has access to data from \mathcal{D}^t for training during the t -th stage. The model’s performance is evaluated across all previously encountered classes, denoted as $\mathcal{Y}_t = Y_1 \cup \dots \cup Y_t$, after each incremental learning task. Specifically, our objective is to learn a model $f(\mathbf{x}) : X \rightarrow \mathcal{Y}_t$ that minimizes empirical risk across all test classes:

$$f^* = \operatorname{argmin}_{f \in \mathcal{H}} \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}_t^1 \cup \dots \cup \mathcal{D}_t^t} \mathbb{I}(y \neq f(\mathbf{x})), \quad (1)$$

where \mathcal{H} is the hypothesis space and $\mathbb{I}(\cdot)$ denotes the indicator function, \mathcal{D}_t^b refers to the testing set for task b . An effective CIL model satisfying Eq. 1 demonstrates strong discriminative abilities across all classes. It strikes a bal-

ance between acquiring knowledge of new classes and preserving information from previously learned ones.

In line with typical PTM-based CIL works [49, 50, 65], we assume that a pre-trained Vision Transformer (ViT) [10] is available as the initialization for $f(\mathbf{x})$. To facilitate understanding, we decompose the PTM into two components: $f(\mathbf{x}) = W^\top \phi(\mathbf{x})$, where $\phi(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^d$ is the feature extractor and $W \in \mathbb{R}^{d \times |\mathcal{Y}_i|}$ is the classifier. We denote the classifier for class k as \mathbf{w}_k , so that $W = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{|\mathcal{Y}_i|}]$.

3.2. Baselines in PTM-Based CIL

In the era of PTMs, many methods seek to modify the PTM slightly to maintain the pre-trained knowledge [43, 47, 49, 50]. These methods usually involve freezing the pre-trained weights and training additional modules like prompt pool to incorporate task-specific information. A representative example is L2P [50], which proposes a key-query matching strategy. Specifically, every prompt $P_i \in \mathbb{R}^{L \times d}$, with L denoting the prompt length, is associated with a learnable key vector $\mathbf{k}_i \in \mathbb{R}^d$. The prompt pool is defined as $\mathbf{P} = \{(\mathbf{k}_1, P_1), (\mathbf{k}_2, P_2), \dots, (\mathbf{k}_Q, P_Q)\}$, where Q is the pool size. The optimization objective is formulated as:

$$\min_{W, \mathbf{P}} \sum_{(\mathbf{x}, y) \in D^t} \mathcal{L}_{ce}(W^\top \bar{\phi}(\mathbf{x}; \mathbf{P}), y) + \mathcal{L}_{reg}(\mathbf{P}), \quad (2)$$

where $\bar{\phi}(\cdot)$ represents the frozen pre-trained backbone parameters, \mathcal{L}_{ce} corresponds to the cross-entropy loss, and \mathcal{L}_{reg} serves as the prompt selection regularization term. During inference, the most appropriate prompts are selected by identifying the top- N keys:

$$\mathbf{K} = \underset{\{t_i\}_{i=1}^N \subseteq [1, Q]}{\operatorname{argmin}} \sum_{i=1}^N d(\phi(\mathbf{x}), \mathbf{k}_{t_i}), \quad (3)$$

where $\{t_i\}_{i=1}^N$ is the selected index set, and \mathbf{K} is the selected top- N keys, $d(\cdot, \cdot)$ denotes the cosine distance.

This approach has two main limitations. First, it demands precise selection of the most appropriate lightweight modules during inference, as guided by Eq. 3. However, the key-query matching process is fragile, making it prone to selecting unsuitable modules, which in turn leads to performance degradation. Second, it primarily focuses on task-specific knowledge while neglecting shared general knowledge between tasks. For instance, if the model learns to classify dogs and cats in different tasks, it may confuse similar-looking classes like a fluffy dog or a cat with a long snout due to its narrow focus on task-specific features. Consequently, it tends to make mistakes when distinguishing highly similar classes across tasks.

4. Methodology

To address the aforementioned challenges, we introduce TUNA in this paper. First, we train task-specific adapters and use an entropy-based mechanism to select the best adapter for each input. Second, we fuse these adapters into a universal adapter to retain shared knowledge across tasks. During inference, we employ a dual-adapter strategy that simultaneously leverages both the selected task-specific adapter and the universal adapter to boost the accuracy.

4.1. Learning Orthogonal Task-Specific Adapters

In this paper, we follow [64] to use adapter [7] to efficiently adapt the PTM to downstream tasks. An adapter is a bottleneck structure that can be incorporated into a pre-trained vision-transformer to facilitate transfer learning. Suppose we have L transformer blocks in the pre-trained vision-transformer, each with a self-attention module and an MLP layer. We can insert an adapter into each block’s MLP via residual connections. An adapter comprises a down-projection layer $W_{down} \in \mathbb{R}^{d \times r}$, a non-linear activation function ReLU and an up-projection layer $W_{up} \in \mathbb{R}^{r \times d}$. The output formula of the MLP layer is formulated as follows:

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

where \mathbf{x}_i and \mathbf{x}_o are the input and output of the MLP, respectively. Eq. 4 illustrates how to inject task-specific information by adding residual connections of adapters to the original outputs. For a specific task i , we define the set of adapters across all transformer blocks as \mathcal{A}_i , which represents task-specific adapters. Furthermore, we denote the output embedding of a given \mathcal{A}_i combined with the PTM as $\phi(\mathbf{x}; \mathcal{A}_i)$, the corresponding prediction as $f(\mathbf{x}; \mathcal{A}_i) = W^\top \phi(\mathbf{x}; \mathcal{A}_i)$. During the learning process of task t , we initialize a new adapter \mathcal{A}_t , which is composed of W_{down}^t and W_{up}^t , and then freeze the weights of the PTM, focus solely on optimizing the task-specific adapters and the corresponding classifier:

$$\mathcal{L}_{cls} = -\frac{1}{n_t} \sum_{(\mathbf{x}, y) \in D^t} \log \frac{\exp(\mathbf{w}_y^\top \phi(\mathbf{x}; \mathcal{A}_t))}{\sum_{i=1}^{|\mathcal{Y}_t|} \exp(\mathbf{w}_i^\top \phi(\mathbf{x}; \mathcal{A}_t))}, \quad (5)$$

where n_t denotes the number of instances in t -th incremental stage. After the first task, we utilize an orthogonal loss function to ensure that the trainable weights remain orthogonal to those learned from previous tasks:

$$\mathcal{L}_{orth} = \sum_{i=1}^{t-1} \left\| W_{up}^t \cdot W_{up}^{i \top} \right\|_1, \quad (6)$$

where $\|\cdot\|_1$ represents the L_1 norm. The up-projection weights in the adapter module plays a key role in projecting intermediate features into a higher-dimensional space,

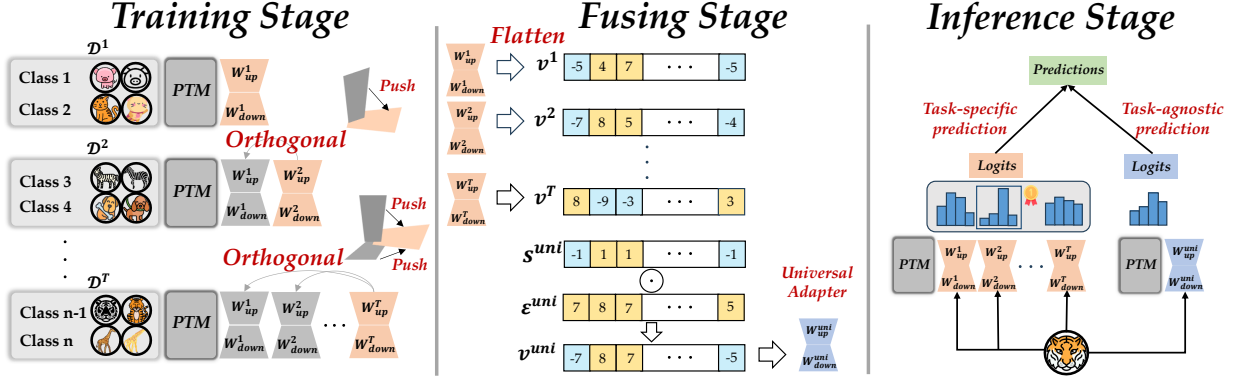


Figure 1. Illustration of TUNA. **Left:** The training protocol of TUNA. We use orthogonal loss to train task-specific adapters. **Middle:** The fusing process. We construct an aggregated sign vector and a magnitude vector, which are combined to form the universal task vector. **Right:** During the inference phase, we select the most appropriate task-specific adapter based on entropy, and then combine the outputs from both the task-specific and universal adapters.

which is essential for encoding task-specific information. By imposing orthogonality constraints on the up-projection weights, we ensure that the current adapter learns *unique* and non-redundant features, effectively differentiating it from previously learned adapters. The overall optimization target is formulated as:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{orth}, \quad (7)$$

where λ is a scalar to weight the loss. After training T tasks by optimizing Eq. 7, we get a list of T adapters: $\{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_T\}$. These adapters effectively capture the most salient features for their respective tasks.

Effect of task-specific adapters: Figure 1 (Left) shows the training protocol. We independently train and optimize adapter modules for each incremental task, ensuring each module extracts maximally discriminative task-specific features. This framework is general and can be seamlessly integrated with various parameter-efficient fine-tuning techniques like LoRA [22] and VPT [25]. Additionally, the lightweight architecture of adapters requires significantly fewer trainable parameters than full-model fine-tuning.

4.2. Multi-Stage Adapter Fusion

After training on t tasks, we obtain a set of task-specific adapters $\{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_t\}$. These adapters are derived by optimizing the same PTM via Eq. 7, ensuring that each adapter is discriminative for its respective task and functions as a ‘task expert.’ For example, if the first task involves classifying ‘tigers,’ the first adapter will focus on features like furpatterns and stripes. If the next task contains ‘birds,’ the adapter will emphasize characteristics such as beaks and feathers. Thus, each adapter is typically limited to task-specific knowledge and struggles to differentiate between similar classes across tasks. In a simplified scenario

where the task identity is known, we could directly use the corresponding expert adapter for prediction. However, in class-incremental learning, where task identity is not available, it is necessary to create a unified embedding space that accommodates all tasks. Drawing on insights from model merging techniques [24, 36, 54], we want to integrate these task-specific adapters into a *universal* adapter that can capture the high-level features shared across all tasks.

To achieve this, we begin by flattening the weights of the task-specific adapters into vectors: $\mathbf{v}^i = \text{Flatten}(\mathcal{A}_i)$, resulting in a collection of task-specific vectors, denoted as $\{\mathbf{v}^1, \mathbf{v}^2, \dots, \mathbf{v}^t\}$. Next, we construct the universal sign vector by determining the dominant sign for each parameter across all task-specific vectors. This is done by taking the sign of the sum of the corresponding parameters:

$$\mathbf{s}^{\text{uni}} = \text{sgn} \left(\sum_{i=1}^t \mathbf{v}^i \right), \quad (8)$$

where $\text{sgn}(\cdot)$ denotes the sign function. For each parameter, we then identify the maximum absolute value among all task vectors that maintain the consensus sign direction, forming the magnitude vector. Specifically, the j -th dimension of the magnitude vector ϵ_j^{uni} is calculated as:

$$\epsilon_j^{\text{uni}} = \begin{cases} \text{abs}(\max(v_j^1, \dots, v_j^t)) & \text{if } s_j^{\text{uni}} > 0 \\ \text{abs}(\min(v_j^1, \dots, v_j^t)) & \text{if } s_j^{\text{uni}} < 0 \end{cases}, \quad (9)$$

where s_j^{uni} denotes the j -th dimension of the sign vector. Then the universal task vector is generated through Hadamard (element-wise) multiplication:

$$\mathbf{v}^{\text{uni}} = \epsilon^{\text{uni}} \odot \mathbf{s}^{\text{uni}}. \quad (10)$$

Finally, we reshape \mathbf{v}^{uni} to match the original dimensions of the adapter, yielding the universal adapter \mathcal{A}_{uni} .

Effect of the universal adapter: Figure 1 (Middle) illustrates the fusion process, which employs two principled operations: sign summation and max-absolute-value selection. The sign summation operates as a voting system that maintains dominant feature orientations across tasks. Concurrently, the max-absolute-value selection with sign consistency suppresses noisy minor activations while preserving task-specific feature magnitudes without attenuation. This operation is theoretically grounded in max-out networks [15] and has been shown to preserve discriminative features.

Through the sign and max operation, the resulting universal adapter captures high-level features common to all tasks, which may not be fully represented by individual task-specific adapters. By using the universal adapter, we can effectively leverage shared knowledge and ensure the model is better equipped to handle all encountered tasks.

4.3. Adapter Selection via Prediction Uncertainty

Suppose the model has progressively learned t tasks and is now required to classify a test image, which may belong to any of the previously learned t tasks. The primary challenge lies in selecting the most suitable task-specific adapter for this prediction. For a sample \mathbf{x} , the predictions of PTM combined with different task-specific adapters are denoted as $f(\mathbf{x}; \mathcal{A}_1), f(\mathbf{x}; \mathcal{A}_2), \dots, f(\mathbf{x}; \mathcal{A}_t)$. Previous research have observed that minimizing entropy on test samples during optimization enables the pre-trained model to effectively adjust to previously unseen test data distributions. Nevertheless, it is still unclear whether entropy minimization can reliably function as a proxy objective for identifying the optimal task-specific adapter. To investigate this, we conduct a pilot study. Specifically, we choose Imagenet-A [19] and Imagenet-R [18] as the datasets and split them into 10 tasks. We assess the model’s performance when combined with each of the 10 different task-specific adapters: $f(\mathbf{x}; \mathcal{A}_1), f(\mathbf{x}; \mathcal{A}_2), \dots, f(\mathbf{x}; \mathcal{A}_{10})$ respectively. We compute the corresponding entropy and prediction accuracy. As illustrated in Figure 2a and Figure 2b, lower entropy is associated with higher prediction accuracy. In other words, the greater the model’s confidence in its predictions, the more accurate it tends to be. Consequently, we conclude that entropy minimization effectively acts as a robust proxy objective for identifying the optimal task-specific adapter. When we have t task-specific adapters, we can choose the most suitable adapter \mathcal{A}^* according to the following formula:

$$\mathcal{A}^* = \underset{\mathcal{A}_i \in \{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_t\}}{\arg \min} \left(- \sum_{c=1}^{\mathcal{Y}_t} f_c(\mathbf{x}; \mathcal{A}_i) \log f_c(\mathbf{x}; \mathcal{A}_i) \right), \quad (11)$$

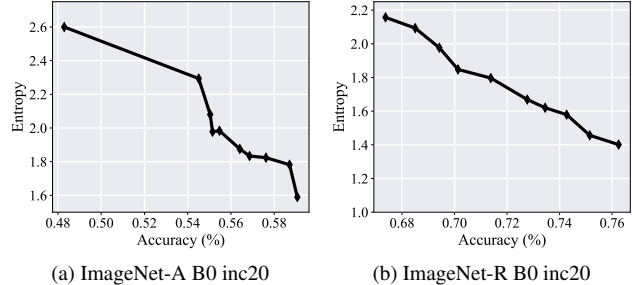


Figure 2. Relationship between accuracy and entropy.

where $f_c(\mathbf{x}; \mathcal{A}_i)$ denotes the predicted probability of class c for input \mathbf{x} using adapter \mathcal{A}_i .

Effect of entropy-based adapter selection: Entropy serves as a natural indicator of adapter-task alignment, when an adapter properly matches the input task, it generates confident, peaked predictions (low entropy), whereas mismatched adapters produce uncertain, flat distributions (high entropy). This intrinsic property makes entropy a reliable metric for selecting the most suitable adapter.

4.4. Task-Specific and Universal Model Ensemble

While task-specific adapters excel at extracting discriminative features for individual tasks, their narrow focus often fails to capture transferable patterns that could aid in distinguishing visually similar classes across different tasks. Our objective is to leverage both specialized and general features effectively, enabling better discrimination between visually similar classes from distinct tasks. Building on this insight, we propose a novel inference strategy: given a test image, we not only select the most suitable task-specific adapter \mathcal{A}^* according to Eq. 11 but also incorporate the predictions generated by the universal adapter to enhance classification robustness:

$$y^* = \underset{y}{\arg \max} (f_y(\mathbf{x}; \mathcal{A}^*) + f_y(\mathbf{x}; \mathcal{A}_{\text{uni}})). \quad (12)$$

Summary of TUNA: As illustrated in Figure 1, we initialize and train an adapter for each incremental task to encode the task-specific information, and then we compute the class-wise mean and variance upon completing the training of each task-specific adapter. These statistical features are subsequently replayed during future incremental learning tasks to alleviate catastrophic forgetting in the classification head. Finally, we fuse these task-specific adapters into a universal adapter, which amalgamates cross-task knowledge while preserving domain-invariant representations. During the inference phase, we employ an entropy-guided adapter selection mechanism that combines the most confident task-specific adapter with the universal adapter to generate more accurate predictions.

Table 1. Average and last performance comparison on four datasets with **ViT-B/16-IN21K** as the backbone. We report all compared methods with their source code. The best performance is highlighted in bold. None of the methods utilize exemplars in their implementation.

Method	CIFAR B0 Inc5		ImageNet-R B0 Inc20		ImageNet-A B0 Inc20		ObjectNet B0 Inc20	
	$\bar{\mathcal{A}}$	\mathcal{A}_B	$\bar{\mathcal{A}}$	\mathcal{A}_B	$\bar{\mathcal{A}}$	\mathcal{A}_B	$\bar{\mathcal{A}}$	\mathcal{A}_B
L2P [50]	85.94	79.93	75.46	69.77	49.39	41.71	63.78	52.19
DualPrompt [49]	87.87	81.15	73.10	67.18	53.71	41.67	59.27	49.33
CODA-Prompt [43]	89.11	81.96	77.97	72.27	53.54	42.73	66.07	53.29
SLCA [58]	92.49	88.55	81.17	77.00	68.66	58.74	72.55	61.30
SSIAT [45]	93.52	90.07	83.20	78.85	70.83	62.23	73.65	62.45
MOS [44]	93.30	89.25	82.96	77.93	67.08	56.22	74.69	63.62
SimpleCIL [64]	87.57	81.26	61.26	54.55	59.77	48.91	65.45	53.59
APER + Adapter [64]	90.65	85.15	75.82	67.95	60.47	49.37	67.18	55.24
RanPAC [37]	94.00	90.62	82.98	77.94	69.32	61.82	72.76	62.02
EASE [65]	91.51	85.80	81.74	76.17	65.34	55.04	70.84	57.86
TUNA (Ours)	94.44	90.74	84.22	79.42	73.78	64.78	76.46	66.32

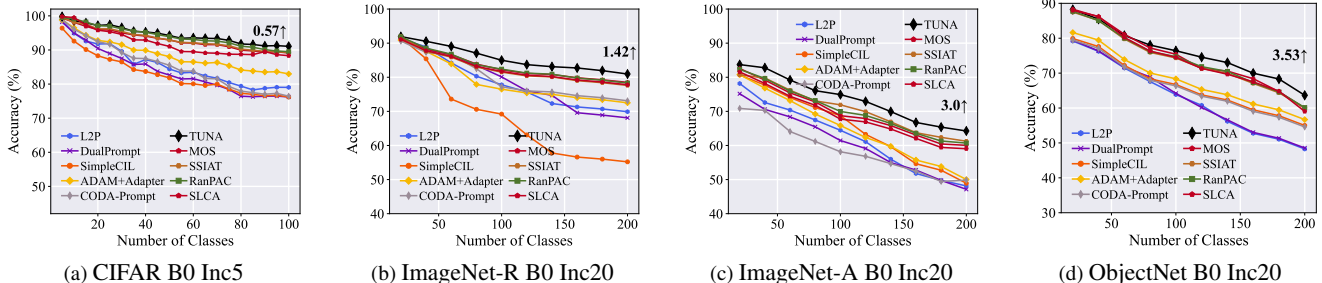


Figure 3. Performance curve of different methods under different settings. All methods are initialized with **ViT-B/16-IN1K**. The relative improvement over the second-best method is annotated with numerical values above the curves at the final incremental stage.

5. Experiments

In this section, we conduct a thorough evaluation of our proposed method using four benchmark datasets, comparing its performance against state-of-the-art methods to demonstrate its advantages. Additionally, we provide an ablation study and further analysis to validate the robustness and effectiveness of our approach.

5.1. Implementation Details

Dataset: Given that pre-trained models encapsulate extensive knowledge from upstream tasks, we adopt the evaluation framework proposed in [64] to assess the performance on various benchmark datasets, including CIFAR100 [28], ImageNet-R [18], ImageNet-A [19], and ObjectNet [4]. These datasets represent typical CIL benchmarks and include out-of-distribution datasets that exhibit a *significant domain gap* relative to ImageNet. Specifically, there are 100 classes in CIFAR100, 200 classes in ImageNet-R, ImageNet-A and ObjectNet.

Dataset split: In accordance with the benchmark protocols established in [40], we employ the notation ‘B- m Inc- n ’ to represent class splits, where m indicates the number of classes in the initial task, and n denotes the number of

classes in each subsequent incremental task. To ensure a fair and consistent comparison, we follow [40] and randomly shuffle class orders using a random seed of 1993 before splitting the data. We ensure consistency in the training and testing sets across all methods, following [45, 64, 65].

Comparison methods: We compare our approach with state-of-the-art PTM-based CIL methods, including prompt-based techniques (L2P [50], DualPrompt [49], and CODA-Prompt [43]), full-model fine-tuning approaches like SLCA [58], and adapter-based methods such as SSIAT [45], EASE [65], and MOS [44]. We also consider prototype-based SimpleCIL [64] and first-session adaptation approaches including RanPAC [37] and APER [64]. All comparative methods employ identical pre-trained models and experimental setups to guarantee fair comparison.

Training details: We use PyTorch [38] to implement all models on NVIDIA RTX 4090 with the *same* network backbone. Since the wide range of PTMs are publicly accessible [51], we choose two representative models following [64], denoted as **ViT-B/16-IN1K** and **ViT-B/16-IN21K**. They are both initially pre-trained on ImageNet21K, while the former is further finetuned on ImageNet1K. In our method, we set the batch size to 48 and train for 20 epochs

Table 2. In contrast to conventional exemplar-based continual learning approaches, TUNA operates without storing any exemplars. All compared methods utilize the identical pre-trained backbone architecture (ViT-B/16-IN21K) for fair evaluation.

Method	Exemplars	ImageNet-R B0 Inc20		CIFAR B0 Inc10	
		$\bar{\mathcal{A}}$	\mathcal{A}_B	$\bar{\mathcal{A}}$	\mathcal{A}_B
iCaRL [40]	20 / class	72.42	60.67	82.46	73.87
DER [53]	20 / class	80.48	74.32	86.04	77.93
FOSTER [46]	20 / class	81.34	74.48	89.87	84.91
MEMO [63]	20 / class	74.80	66.62	84.08	75.79
TagFex [62]	20 / class	83.23	78.45	92.17	89.26
TUNA	0	85.90	80.95	95.05	92.15

using the SGD optimizer with momentum. The learning rate is initially set to 0.01 and follows a cosine annealing decay pattern. The projection dimension r in the adapter is set to 16, the weight λ in Eq. 7 is initialized at 1e-3 and follows an exponential decay schedule.

Evaluation protocol: Following the benchmark established by [40], we denote the Top-1 accuracy after the b -th stage as \mathcal{A}_b . Moreover, we use \mathcal{A}_B (the performance after the last stage) and $\bar{\mathcal{A}} = \frac{1}{B} \sum_{b=1}^B \mathcal{A}_b$ (average performance along incremental stages) as measurements.

5.2. Benchmark Comparison

In this section, we conduct a comprehensive comparison of our proposed method, TUNA, against state-of-the-art approaches on four benchmark datasets and different backbone weights. Table 1 reports the comparison of different methods with ViT-B/16-IN21K. We can infer that our method achieves the best performance among all four benchmarks, substantially outperforming the current SOTA methods. We also report the incremental performance trend of different methods in Figure 3 with ViT-B/16-IN1K. As annotated at the end of each image, we find our method consistently outperforms the runner-up method, further underscoring its effectiveness.

To further validate the robustness of our approach, we extend our evaluation beyond the standard B0 benchmark (presented in Table 1 and Figure 3) to a large-base setting. In Figure 4, we compare our method with several SOTA methods with vast base classes. As shown in Figure 4, TUNA still outperforms other methods. Additionally, we also compare TUNA to traditional CIL methods such as iCaRL [40], DER [53], FOSTER [46], MEMO [63], TagFex [62] by implementing them with the same pre-trained weight in Table 2. Notably, TUNA maintains its leading performance, achieving a higher average accuracy than the closest competitor while remaining exemplar-free—a key advantage in memory-constrained scenarios.

It is important to highlight that traditional CIL methods rely on storing exemplars to retain previously learned knowledge, whereas our approach eliminates this requirement. We follow [40] to set the exemplar number to 20 per class for these methods. TUNA still works competitively in

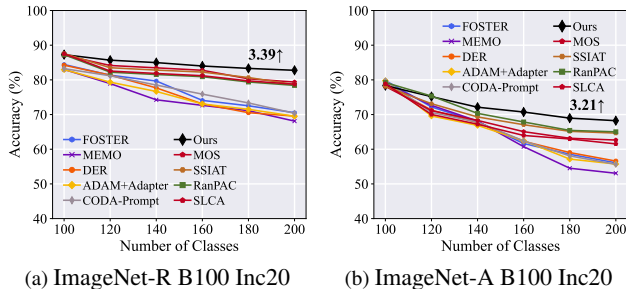


Figure 4. Experimental results on ImageNet-R and ImageNet-A with large base classes. All methods are based on the same PTM.

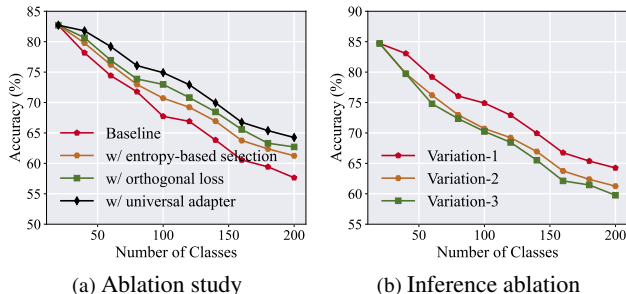


Figure 5. **Left:** Ablation study of different components in TUNA. We find each component contributes to enhancing the performance. **Right:** Experimental results on ImageNet-A B0 inc20 with different inference strategies.

comparison to these exemplar-based methods.

5.3. Ablation Study

In this section, we perform an ablation study to evaluate the contribution of each component in TUNA. Specifically, we present the incremental performance of various configurations on ImageNet-A B0 Inc20 in Figure 5a. In the figure, ‘Baseline’ denotes training task-specific adapters for each task and predicting using all adapters during inference, selecting the maximum logit as the final prediction. ‘w/ entropy-based adapter selection’ means selecting the task-specific adapter based on entropy and using its output for prediction, which proves to be an effective strategy for choosing the appropriate adapter. Furthermore, ‘w/ orth loss’ introduces an orthogonality loss during training to enhance task-specific knowledge learning, and the results show that this addition improves performance. Finally, ‘w/ universal adapter’ ensembles the outputs from a universal adapter, which captures general knowledge shared across tasks, enabling the model to better handle all encountered tasks. The ablation study confirms that each component in TUNA contributes to improving CIL performance.

5.4. Further Analysis

Different inference strategies: To validate our proposed inference strategy, we conduct experiments on ImageNet-A

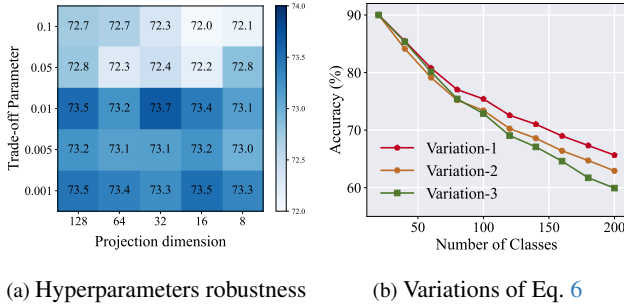


Figure 6. Further analysis on parameter robustness and orthogonal loss implementation.

B0 Inc20 using three inference strategies: Variation-1 (our strategy), Variation-2 (task-specific adapter selection based on entropy), and Variation-3 (sole reliance on the universal adapter). As shown in Figure 5b, Variation-1 consistently outperforms the others across all tasks. Variation-2 fails to leverage shared knowledge between tasks, while Variation-3 lacks the granularity to capture task-specific nuances, resulting in suboptimal performance.

Parameter robustness: TUNA involves two hyperparameters, the projection dim r in the adapter and the trade-off parameter λ in Eq. 7. To evaluate their robustness, we conduct experiments on ImageNet-A B0 Inc20 by varying these parameters. Specifically, we choose r among $\{8, 16, 32, 64, 128\}$, and λ among $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. We report the average performance in Figure 6a. The results demonstrate that the performance remains stable across different parameter values.

Different orthogonal loss: In Eq. 6, we force the current adapter’s up projection weight to be orthogonal to previous adapters’ up projection weights, we call it Variation-1. Additionally, we can extend this orthogonality constraint to the current adapter’s down projection weight relative to previous adapters’ down projection weights, termed Variation-2, or apply it to both the up projection and down projection weights simultaneously, denoted as Variation-3. We conduct experiments on ObjectNet B0 Inc20 setting to compare different losses. As we can see from Figure 6b, with other settings the same, we find Variation-1 performs the best among these variations. This is likely because the up projection weight plays a more critical role in capturing task-specific features, and enforcing orthogonality on it alone is sufficient to reduce task interference. In contrast, the down projection weight primarily projects input features into a lower-dimensional space, and overly restricting it may hinder the model’s ability to encode task-specific information. Additionally, applying constraints to both weights simultaneously may introduce excessive rigidity, reducing flexibility and risking underfitting. Thus, focusing orthogonality constraints solely on the up projection weight offers a more balanced and efficient approach for continual learning.

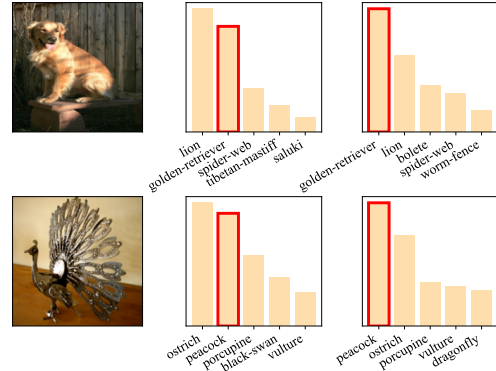


Figure 7. Visualizations of the predictions on ImageNet-R. The original images are depicted in the first column, followed by the top-5 prediction probability produced by task-specific adapter, and the probabilities generated by the universal adapter in the last column. The ground-truth class is highlighted with red boxes.

Visualizations: To explore why combining task-specific and universal adapters boosts performance, we visualize predictions from each adapter separately using ImageNet-R images and the model trained under the B0 Inc20 setting. Figure 7 shows the original images in the first column, top-5 predictions from task-specific adapters in the second column, and top-5 predictions from the universal adapter in the third column. Task-specific adapters, focusing on limited information, often misclassify similar classes, such as predicting a golden retriever as a lion or a peacock as an ostrich. In contrast, the universal adapter, which integrates cross-task knowledge, captures shared features and refines predictions, increasing the chances of correct classification. This synergy enhances overall performance.

6. Conclusion

Incremental learning is crucial for practical systems. This paper introduces a novel method that integrates Task-Specific and Universal Adapters(TUNA) for pre-trained model-based CIL. Specifically, we train task-specific adapters to capture distinct features for their tasks. We also introduce an adapter fusion mechanism to create a universal adapter that encapsulates shared knowledge across tasks. During inference, we employ an entropy-based selection to choose the most suitable task-specific adapter and then ensemble its predictions with those from the universal adapter. Extensive experiments verify TUNA’s effectiveness.

Limitations and future works: The process of selecting the optimal task-specific adapter requires multiple forward passes through the model, resulting in increased computational time. Future works include designing methods to speed up the algorithm.

Acknowledgments

This work is supported by the NSFC (62376118), CCF-Tencent Rhino-Bird Open Research Fund (RAGR20240101), Fundamental Research Funds for the Central Universities (14380021), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

- [1] Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hoyjun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *ICCV*, pages 844–853, 2021. 2
- [2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018. 2
- [3] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *NeurIPS*, pages 11816–11825, 2019. 2
- [4] Alexandru Barbu, Dheeraj Dwivedi, Chen Wang, Trevor Darrell, and Rob Fergus. Objectnet: A large-scale bias measurement dataset for object recognition models. In *NeurIPS*, page 9453–9463, 2019. 6
- [5] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, pages 532–547, 2018. 2
- [6] Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *AAAI*, pages 6993–7001, 2021. 2
- [7] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, pages 16664–16678, 2022. 1, 3
- [8] Shuo Chen, Chen Gong, Jun Li, Jian Yang, Gang Niu, and Masashi Sugiyama. Learning contrastive embedding in low-dimensional space. In *NeurIPS*, pages 6345–6357, 2022. 1
- [9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 1
- [10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, and Sylvain Gelly. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 3
- [11] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pages 86–102, 2020. 2
- [12] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *CVPR*, pages 9285–9295, 2022. 2
- [13] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 1
- [14] Kai Gan, Bo Ye, Min-Ling Zhang, and Tong Wei. Semi-supervised clip adaptation by enforcing semantic and trapezoidal consistency. In *ICLR*, 2025. 1
- [15] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *ICML*, pages 1319–1327, 2013. 5
- [16] Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost Van De Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. In *NeurIPS*, pages 6582–6595, 2023. 2
- [17] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021. 1
- [18] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, pages 8320–8329, 2021. 5, 6
- [19] Dan Hendrycks, Norman Mu, Andrew Ilyas, Steven Basart, Colin Raffel, and Dawn Song. Unnatural adversarial examples. In *ICLR*, 2021. 5, 6
- [20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2
- [21] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *ECCV*, pages 437–452, 2018. 2
- [22] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022. 4
- [23] Zhiyuan Hu, Yunsheng Li, Jiancheng Lyu, Dashan Gao, and Nuno Vasconcelos. Dense network expansion for class incremental learning. In *CVPR*, pages 11858–11867, 2023. 2
- [24] Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. Emr-merging: Tuning-free high-performance model merging. In *NeurIPS*, pages 122741–122769, 2024. 4
- [25] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727, 2022. 1, 4
- [26] Dahui Jung, Dongyoon Han, Jihwan Bang, and Hwanjun Song. Generating instance-level prompts for rehearsal-free continual learning. In *ICCV*, pages 11847–11857, 2023. 2
- [27] Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *CVPR*, pages 16071–16080, 2022. 2
- [28] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 6

- [29] Janghyeon Lee, Hyeong Gwon Hong, Donggyu Joo, and Junmo Kim. Continual learning with extended kronecker-factored approximate curvature. In *CVPR*, pages 9001–9010, 2020. 2
- [30] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *NIPS*, pages 4652–4662, 2017. 2
- [31] Lan Li, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Addressing imbalanced domain-incremental learning through dual-balance collaborative experts. In *ICML*, 2025. 2
- [32] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017. 2
- [33] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *NeurIPS*, pages 32033–32046, 2022. 1
- [34] Yaoyao Liu, Yingying Li, Bernt Schiele, and Qianru Sun. Online hyperparameter optimization for class-incremental learning. In *AAAI*, pages 8906–8913, 2023. 2
- [35] Zilin Luo, Yaoyao Liu, Bernt Schiele, and Qianru Sun. Class-incremental exemplar compression for class-incremental learning. In *CVPR*, pages 11371–11380, 2023. 2
- [36] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. In *NeurIPS*, pages 17703–17716, 2022. 4
- [37] Mark D. McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *NeurIPS*, pages 12022–12053, 2023. 2, 6
- [38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019. 6
- [39] Federico Pernici, Matteo Bruni, Claudio Baccchi, Francesco Turchini, and Alberto Del Bimbo. Class-incremental learning with pre-allocated fixed classifiers. In *ICPR*, pages 6259–6266, 2021. 2
- [40] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 5533–5542, 2017. 1, 2, 6, 7
- [41] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *NeurIPS*, pages 348–358, 2019. 2
- [42] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *CVPR*, pages 1591–1600, 2021. 2
- [43] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *CVPR*, pages 11909–11919, 2023. 1, 2, 3, 6
- [44] Hai-Long Sun, Da-Wei Zhou, Hanbin Zhao, Le Gan, De-Chuan Zhan, and Han-Jia Ye. Mos: Model surgery for pre-trained model-based class-incremental learning. In *AAAI*, pages 20699–20707, 2024. 6
- [45] Yuwen Tan, Qin hao Zhou, Xiang Xiang, Ke Wang, Yuchuan Wu, and Yongbin Li. Semantically-shifted incremental adapter-tuning is a continual vitransformer. In *CVPR*, pages 23252–23262, 2024. 1, 6
- [46] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, pages 398–414, 2022. 2, 7
- [47] Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. In *NeurIPS*, pages 69054–69076, 2023. 3
- [48] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam’s razor for domain incremental learning. In *NeurIPS*, pages 5682–5695, 2022. 1
- [49] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer G Dy, and Tomas Pfister. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, pages 631–648, 2022. 1, 2, 3, 6
- [50] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer G Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, pages 139–149, 2022. 1, 2, 3, 6
- [51] Ross Wightman. Pytorch image models. <https://github.com/rwightman/pytorch-image-models>, 2020. 6
- [52] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 2
- [53] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021. 2, 7
- [54] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *ICLR*, 2024. 4
- [55] Bo Ye, Kai Gan, Tong Wei, and Min-Ling Zhang. Bridging the gap: Learning pace synchronization for open-world semi-supervised learning. In *IJCAI*, pages 5362–5370, 2024. 1
- [56] Lu Yu, Bartłomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *CVPR*, pages 6982–6991, 2020. 2
- [57] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, pages 3987–3995, 2017. 2

- [58] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *ICCV*, pages 19148–19158, 2023. [2](#), [6](#)
- [59] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pages 13208–13217, 2020. [2](#)
- [60] Hanbin Zhao, Hui Wang, Yongjian Fu, Fei Wu, and Xi Li. Memory-efficient class-incremental learning for image classification. *TNNLS*, 33(10):5966–5977, 2021. [2](#)
- [61] Bowen Zheng, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Multi-layer rehearsal feature augmentation for class-incremental learning. In *ICML*, pages 61649–61663, 2024. [2](#)
- [62] Bowen Zheng, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Task-agnostic guided feature expansion for class-incremental learning. In *CVPR*, pages 10099–10109, 2025. [2](#), [7](#)
- [63] Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *ICLR*, 2023. [7](#)
- [64] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *IJCV*, 133(3):1012–1032, 2024. [1](#), [2](#), [3](#), [6](#)
- [65] Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *CVPR*, pages 23554–23564, 2024. [2](#), [3](#), [6](#)
- [66] Da-Wei Zhou, Yuanhan Zhang, Yan Wang, Jingyi Ning, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Learning without forgetting for vision-language models. *TPAMI*, 47(6): 4489 – 4504, 2025. [1](#)
- [67] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5871–5880, 2021. [2](#)