Dynamic Integration of Task-Specific Adapters for Class Incremental Learning

Supplementary Material

1. Methodology Supplementary

1.1. Analysis of knowledge of retention and reproduction:

As mentioned in the main manuscript, for an input token $\mathbf{p} \in \mathbb{R}^m$, consider an adapter without an activation function, with weights $W = \mathbf{W}_{\text{down}}\mathbf{W}_{\text{up}} \in \mathbb{R}^{m\times n}$, $\mathbf{W}_{\text{down}} \in \mathbb{R}^{m\times r}$, $\mathbf{W}_{\text{up}} \in \mathbb{R}^{r\times n}$, and a matrix rank of r. The weight matrix can be decomposed using SVD as follows:

$$\mathbf{W} = \mathbf{U}\text{diag}(\sigma)\mathbf{V}, \tag{1}$$

$$\mathbf{U}^\top = \left[\mathbf{u}_i{}^\top\right]_{i=1}^r \in \mathbb{R}^{r\times m}, \mathbf{V} = \left[\mathbf{v}_i{}^\top\right]_{i=1}^r \in \mathbb{R}^{r\times n}. \tag{2}$$

And the output $\mathbf{o}$ can be formulated as:

$$o = \mathcal{A}(\mathbf{p}) = \mathbf{W}^\top\mathbf{p} = \sum \rho_i^{'}\mathbf{v}_i = \mathbf{V}^\top g(\mathbf{p}), \tag{3}$$

$$g(\mathbf{p}) = \text{diag}(\sigma)^\top\mathbf{U}^\top\mathbf{p} = [\rho_i^{'}]_{i=1}^r \in \mathbb{R}^r. \tag{4}$$

After introducing non-linear activation function ReLU, for each input $\mathbf{p} \in \mathbb{R}^m$, the output of the adapter can be reformulated as:

$$\mathcal{A}(\mathbf{p}) = \mathbf{W}_{\text{up}}^T\text{ReLU}(\mathbf{W}_{\text{down}}^T\mathbf{p}), \tag{5}$$

We perform SVD decomposition on $\mathbf{W}_{\text{up}}$ as the above Eq. 2:

$$\mathbf{W}_{\text{up}} = \mathbf{U}_{\text{up}}\text{diag}(\sigma_{\text{up}})\mathbf{V}_{\text{up}}, \tag{6}$$

$$\mathbf{W}_{\text{up}} = \sum \mathbf{u}_{(i,\text{up})}\sigma_{(i,\text{up})}\mathbf{v}_{(i,\text{up})}^\top. \tag{7}$$

The output $\mathbf{o}$ can be reformulated as:

$$\begin{aligned} o = \mathcal{A}(\mathbf{p}) &= \mathbf{W}_{\text{up}}^T\text{ReLU}(\mathbf{W}_{\text{down}}^T\mathbf{p}) \\ &= \sum \left(\mathbf{u}_{(i,\text{up})}\sigma_{(i,\text{up})}\mathbf{v}_{(i,\text{up})}^\top\right)^\top \text{ReLU}(\mathbf{W}_{\text{down}}^T\mathbf{p}) \\ &= \sum \mathbf{v}_{(i,\text{up})}\left(\sigma_{(i,\text{up})}\mathbf{u}_{(i,\text{up})}^\top\text{ReLU}(\mathbf{W}_{\text{down}}^T\mathbf{p})\right) \\ &= \sum \rho_i^{'}\mathbf{v}_{(i,\text{up})} = \mathbf{V}_{\text{up}}^\top g(\mathbf{p}), \end{aligned} \tag{8}$$

$$g(\mathbf{p}) = \text{diag}(\sigma_{(i,\text{up})})^\top\mathbf{U}_{(i,\text{up})}^\top\text{ReLU}(\mathbf{W}_{\text{down}}^T\mathbf{p}) \in \mathbb{R}^r. \tag{9}$$

From Eq. 9, we observe that after introducing the nonlinear activation, we obtain an expression similar to Eq. 4, with the function g replaced by a nonlinear function.

1.2. Training Pipeline

The training pipeline for the proposed DIA method follows previous approaches [6, 9, 10] and is composed of two stages: new task learning and classifier alignment for each task t, as illustrated in Algorithm.1.

Algorithm 1 Training Pipeline for Task t

Input: Training dataset D^t; TSAI module $f(\cdot;\theta_b)$; Cosine classifier $\phi(\cdot;\mathbf{W}_{cls})$;
Parameter: $\theta_b = \{\theta_{ptm}, \theta^o, \theta^n\}$, $\mathbf{W}_{cls} = \{\mathbf{W}_{cls}^o, \mathbf{W}_{cls}^n\}$;
Initialization: Initialize θ^n and $\mathbf{W}_{cls}^n$. Freeze parameters θ_{ptm} and θ^o;

```
1:  # New Task Learning.
2:  while not converged do
3:      for $\{I_i^t, y_i^t\} \in D^t$ do
4:          Compute logits $\xi_{y_i^t} = \phi\left(f(I_i^t;\theta_b);\mathbf{W}_{cls}^n\right)$;
5:          Compute CE loss $\mathcal{L}_{CE}(\xi_{y_i^t}; s, \mathfrak{m})$;
6:          Compute patch-level distillation loss $\mathcal{L}_{pld}$;
7:          Backward with objective $\mathcal{L}_{obj} = \mathcal{L}_{CE} + \lambda\mathcal{L}_{PDL}$;
8:      end for
9:  end while
10: For each class $c_k^t \in C^t$ in task $t$, we compute its class
    prototype $\boldsymbol{\mu}_k^t$ and store it in memory;
11: # Classifier alignment.
12: while not converged do
13:     for Training batch within $D^t$ do
14:         Sample $N$ class prototypes $\{\mu_{c_i}\}_{i=1}^N, c_i \in C^{1:t}$;
15:         Construct class feature $\hat{\mu}_{i,c}$ using PFR method
            with the training batch;
16:         Fine-tune the classifier $\phi(\cdot;\mathbf{W}_{cls}^o, \mathbf{W}_{cls}^n)$;
17:     end for
18: end while
```

Method	Backbone	ImageNet-R		Cifar-100	
		$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$	$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$
RAPF	CLIP	**80.28**	<u>85.58</u>	79.04	<u>86.19</u>
P-Fusion	CLIP	79.10	-	<u>85.50</u>	-
DIA-r16	ViT-B16	<u>79.82</u>	**86.04**	**90.38**	**94.36**

Table 1. Comparison with CLIP-based CIL methods with their reported accuracy. The best results are marked in **bold**, and the second are marked in <u>underline</u>.

For incremental task t, the TSAI module is parameterized by $\theta_b = \{\theta_{ptm}, \theta^o, \theta^n\}$, where θ_{ptm} denotes the parameters of the pre-trained model (PTM), θ^o refers to the parameters of the adapters and signature vectors for old tasks, and θ^n corresponds to the parameters for the new task t. The cosine classifier $\phi(\cdot;\mathbf{W}_{cls})$ comprises two parts: the old class classifier $\phi^o(\cdot;\mathbf{W}_{cls}^o)$ and the new class classifier $\phi^n(\cdot;\mathbf{W}_{cls}^n)$.

New Task Learning: During new task learning, for each

Method	Params	Flops	ImageNet-R		ImageNet-A		CUB-200		Cifar-100	
			$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$	$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$	$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$	$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$
DIA-MLP	0.17M	17.91B	79.03	85.61	61.69	71.58	86.73	93.21	90.8	94.28
DIA-MHSA	0.51M	18.56B	78.50	85.05	57.27	69.12	83.29	90.56	90.33	94.17
DIA-MIX	0.69M	18.9B	79.69	85.41	58.21	69.93	85.68	92.38	90.85	94.30

Table 2. Ablation experiments on the adapter structure with 10 incremental tasks.

Method	Params	Flops	ImageNet-R		ImageNet-A		CUB-200		Cifar-100	
			$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$	$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$	$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$	$\mathcal{A}^{10}\uparrow$	$\bar{\mathcal{A}}^{10}\uparrow$
DIA-r08	0.17M	17.91B	79.03	85.61	61.69	71.58	86.73	93.21	90.8	94.28
DIA-r16	0.31M	18.18B	79.82	86.04	57.74	69.84	86.41	92.39	90.38	94.36
DIA-r64	1.19M	19.20B	79.10	85.62	59.91	70.90	87.19	92.26	90.56	94.51

Table 3. Ablation experiments on the adapter rank with 10 incremental tasks.

image $\{I_i^t, y_i^t\} \in D^t$, where $y_i^t \in c_k^t \in C^t$, we first extract features $f(I_i^t;\theta_b)$ using TSAI module $f(\cdot;\theta_b)$, then compute the output logits

$$\xi_{y_i^t} = \phi^n\left(f(I_i^t;\theta_b); \mathbf{W}_{cls}^n\right), \tag{10}$$

with the cosine classifier of new task $\phi^n(\cdot;\mathbf{W}_{cls}^n)$. We optimize the classification results using a variant of the CE loss $\mathcal{L}_{CE}$ and ensure feature consistency with patch-level distillation loss (PDL) $\mathcal{L}_{pdl}$.

Following methods [4, 5], we define $\mathcal{L}_{CE}$ as:

$$\mathcal{L}_{CE}(\xi_{y_i^t}; s, \mathfrak{m}) = -\log \frac{e^{s(\xi_{y_i^t}-\mathfrak{m})}}{e^{s(\xi_{y_i^t}-\mathfrak{m})} + \sum_c^{C^t - c_i^t} e^{s(\xi_c)}}. \tag{11}$$

Here, s is a scaling factor that adjusts the magnitude of cosine similarity, ensuring that the Softmax function produces a more discriminative probability distribution. $\mathfrak{m}$ introduces an additional angular separation between classes. When $s = 0$ and $\mathfrak{m} = 0$, $\mathcal{L}_{CE}(\cdot; 0, 0)$ reduces to the standard cross-entropy loss function. When $s \neq 0$ and $\mathfrak{m} = 0$, $\mathcal{L}_{CE}(\cdot; s, 0)$ adopts a common format used in cosine classifiers, adjusting the scale of cosine similarity via s.

The loss function during training is defined as:

$$\mathcal{L}_{obj} = \mathcal{L}_{CE} + \lambda\mathcal{L}_{pdl}, \tag{12}$$

where λ is a hyperparameter that controls the strength of the regularization.

Classifier Alignment: To further refine the classification layer, we perform classifier alignment after the new task learning stage, following methods [9, 10] (see Algorithm 1). Specifically, during the new task learning phase, only the classifiers corresponding to the current task's classes are trained alongside the TSAI module. Once new task learning is completed, we compute the class prototype $\boldsymbol{\mu}_k^t = \frac{1}{N_k^t}\sum_i^{N_k^t} f(I_i^t;\theta_b)$ for each class $c_k^t \in C^t$ in the current task, where N_k^t denotes the number of images for class c_k^t.

During the classifier alignment stage for task t, we randomly select N (Set to 32 in our implementation) class prototypes $\{\mu_{c_i}\}_{i=1}^N, c_i \in C^{1:t}$ in each training batch and generate pseudo-features $\{\hat{\mu}_{c_i}\}_{i=1}^N$ using the PFR method. These pseudo-features, combined with training samples, are used as inputs to the classifier $\phi(\cdot;\mathbf{W}_{cls}^o, \mathbf{W}_{cls}^n)$, which is then fine-tuned using $\mathcal{L}_{CE}(\cdot;0,0)$.

2. Supplementary Experiments

This section provides additional experiments, including comparisons with CLIP-MoE [7], hyperparameter ablation studies, and model structure ablation studies.

2.1. Comparative Experiments

We conduct comparative experiments with the latest methods that use a CLIP backbone: RAPF [3] and P-Fusion [1]. As shown in Table 1, despite the advantage of a more powerful CLIP backbone and access to additional semantic information, RAPF [3] and P-Fusion [1] are outperformed by our method on the CIFAR-100 dataset by a significant margin. Furthermore, our approach achieves comparable performance to these methods on the Imagenet-R dataset.

2.2. Structure Ablation

We explore the impact of inserting the TSAI module in parallel within both the MHSA and MLP structures of the transformer block. Specifically, we integrate TSAI in parallel with the three QKV projection layers of MHSA. As

Ablation	ImageNet-R		Cifar100	
	$\mathcal{A}^{10}$	$\bar{\mathcal{A}}^{10}$	$\mathcal{A}^{10}$	$\bar{\mathcal{A}}^{10}$
DIA w SDC	75.13	83.63	89.13	93.46
DIA w LDC	76.43	84.08	88.66	93.66
DIA w PFR	79.03	85.61	90.80	94.29

Table 4. Ablation experiments on the feature shift techniques with 10 incremental tasks.

shown in Table 2, despite tripling the number of trainable parameters per task, DIA-MHSA still slightly underperforms compared to DIA-MLP. We attribute this to the multi-head attention mechanism's role in capturing input sequence dependencies, where its complex structure may be disrupted by the addition of adapters, thereby increasing optimization difficulty. The experimental results further validate the rationality of our current model structure.

2.3. Adapter Rank Ablation

We evaluate the model's accuracy across four datasets by varying the adapter down-projection dimensions to 8, 16, and 64 in Table 3. The results show that increasing the number of parameters does not lead to significant accuracy improvements, with the r64 configuration yielding less than a one-point gain over r08. This demonstrates that the importance of model architecture and training strategies outweighs that of merely increasing the number of parameters.

2.4. Feature Drift Discussion

We further perform ablation studies to evaluate the impact of feature shift techniques on model alignment, as summarized in Table 4 The results demonstrate that compared to SDC [8], which calculates class prototype shifts using feature gaps, or LDC [2], which uses MLP to learn mappings from old to new feature spaces, our proposed PFR method better captures the distribution of old class features in the new task's feature space, achieving superior accuracy on both ImageNet-R and CIFAR-100 datasets.

References

[1] Haoran Chen, Zuxuan Wu, Xintong Han, Menglin Jia, and Yu-Gang Jiang. Promptfusion: Decoupling stability and plasticity for continual learning, 2024. 2

[2] Alex Gomez-Villa and et al. Exemplar-free continual representation learning via learnable drift compensation. In *Proc. ECCV*, 2025. 3

[3] Linlan Huang, Xusheng Cao, Haori Lu, and Xialei Liu. Class-incremental learning with clip: Adaptive representation adjustment and parameter fusion, 2024. 2

[4] Can Peng, Kun Zhao, Tianren Wang, Meng Li, and Brian C Lovell. Few-shot class-incremental learning from an open-set perspective. In *European Conference on Computer Vision*, pages 382–397. Springer, 2022. 2

[5] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition, 2018. 2

[6] Shaokun Wang, Weiwei Shi, Yuhang He, Yifan Yu, and Yihong Gong. Non-exemplar class-incremental learning via adaptive old class reconstruction. In *Proceedings of the ACM International Conference on Multimedia*, page 4524–4534, New York, NY, USA, 2023. Association for Computing Machinery. 1

[7] Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23219–23230, 2024. 2

[8] Lu Yu and et al. Semantic drift compensation for class-incremental learning. In *Proc. CVPR*, 2020. 3

[9] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19091–19101, 2023. 1, 2

[10] 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 (CVPR)*, pages 5871–5880, 2021. 1, 2