Supplementary Material for Expandable Subspace Ensemble for Pre-Trained Model-Based Class-Incremental Learning

Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye(✉), De-Chuan Zhan
National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China
{zhoudw,sunhl,yehj,zhandc}@lamda.nju.edu.cn

Abstract

*In the main paper, we propose ExpAndable Subspace Ensemble (*EASE*) for PTM-based CIL. To enable model updating without conflict, we train a distinct lightweight adapter module for each new task to create task-specific subspaces. These adapters span a high-dimensional feature space, enabling joint decision-making across multiple subspaces. As data evolves, the expanding subspaces render the old class classifiers incompatible with new-stage spaces. Correspondingly, we design a semantic-guided prototype complement strategy that synthesizes old classes' new features without using any old class instance.*

In this supplementary, we provide more details about EASE*, including more implementation details and experimental results.*

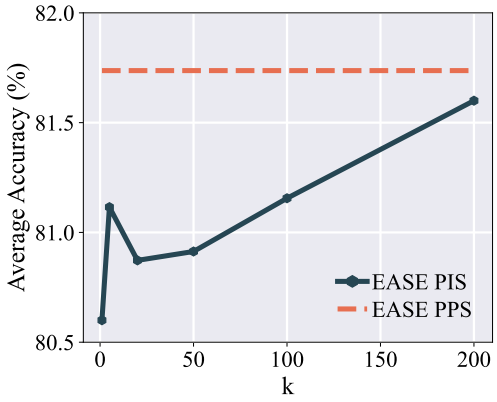
- *Section 1 introduces further analysis of* EASE*, including similarity calculation, subspace expansion, upper bound comparison, multiple runs, and running time comparison.*
- *Section 2 introduces the details of compared methods.*
- *Section 3 provides supplementary results of benchmark datasets to the main paper.*

1. Further Ablations

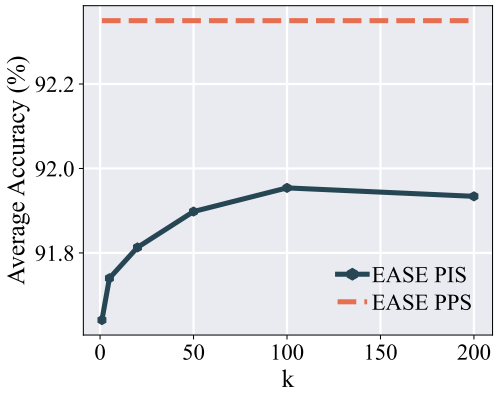
In this section, we conduct further analysis on EASE's components to investigate their effectiveness, *e.g.*, semantic-guided mapping and adapter-spanned subspaces. We also include the comparison about random seeds, running time, and the results of the upper bound.

1.1. Prototype-Prototype Similarity VS. Prototype-Instance Similarity

In the main paper, we formulate the prototype complement task as: given two subspaces (old and new) and two class sets (old and new), the target is to estimate old class prototypes in the new subspace $\hat{\mathbf{P}}_{o,n}$ using $\mathbf{P}_{o,o}$, $\mathbf{P}_{n,o}$, $\mathbf{P}_{n,n}$.



(a) ImageNet-R B0 Inc20



(b) CIFAR100 B0 Inc10

Figure 1. Experimental results on different similarity calculation methods. **Using prototype-prototype similarity shows better performance than prototype-instance similarity.**

Among them, $\mathbf{P}_{o,o}$ and $\mathbf{P}_{n,o}$ represent prototypes of old and new classes in the old subspace (which we call co-occurrence space), and $\mathbf{P}_{n,n}$ represents new classes prototypes in the new subspace.

During the complement process, we construct a class-wise similarity matrix in the old subspace:

$$\text{Sim}_{i,j} = \frac{\mathbf{P}_{o,o}[i]}{\|\mathbf{P}_{o,o}[i]\|_2} \frac{\mathbf{P}_{n,o}[j]^\top}{\|\mathbf{P}_{n,o}[j]\|_2}, \tag{1}$$

and then utilize it to reconstruct prototypes via class-wise similarity in the new subspace:

$$\hat{\mathbf{P}}_{o,n}[i] = \sum_j \text{Sim}_{i,j} \times \mathbf{P}_{n,n}[j]. \tag{2}$$

However, since we have the current dataset $\mathcal{D}^b$ in hand, apart from class-wise similarity, we can also measure the similarity of old class prototypes and new class instances.

$$\text{Sim}_{i,j} = \frac{\mathbf{P}_{n,o}[i]}{\|\mathbf{P}_{n,o}[i]\|_2} \frac{\phi(\mathbf{x}_j; \mathcal{A}_{old})^\top}{\|\phi(\mathbf{x}_j; \mathcal{A}_{old})\|_2}. \tag{3}$$

Different from prototype to prototype similarity in Eq. 1, Eq. 3 measures the similarity of an old class prototype to a new class instance in the same subspace. In the implementation, we can choose $\mathbf{x}_j$ in a subset containing k instances and obtain a similarity matrix of $|Y_{old}| \times k$. The choice of these k instances is based on the relative similarity. Similar to the reconstruction process in Eq. 2, we can build the prototype complement process via:

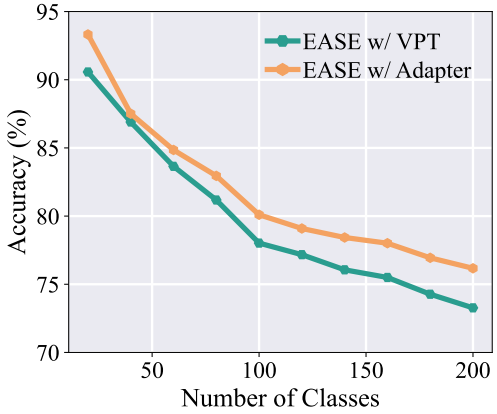
$$\hat{\mathbf{P}}_{o,n}[i] = \sum_j \text{Sim}_{i,j} \times \phi(\mathbf{x}_j; \mathcal{A}_{new}). \tag{4}$$

We call the prototype-instance similarity-based complement process in Eq. 4 as PIS (prototype-instance similarity) while calling the prototype-prototype similarity-based complement process in Eq. 2 as PPS (prototype-prototype similarity). In this section, we conduct experiments on CIFAR100 and ImageNet-R to compare these variations. We utilize ViT-B/16-IN21K as the backbone and keep other settings the same. We choose k in PIS among $\{1, 5, 20, 50, 100, 200\}$.

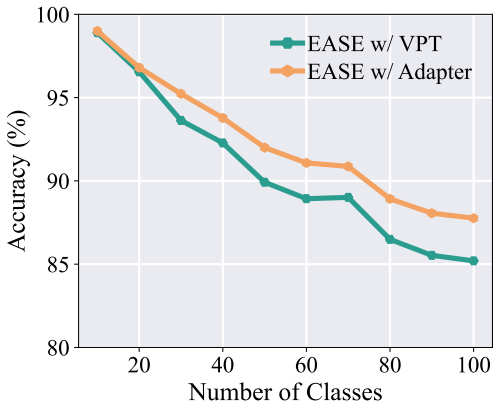
We report the experimental results in Figure 1. As shown in the figure, utilizing more instances (*i.e.*, with larger k) shows better performance. However, we find using prototype-instance similarity less effective than using prototype-prototype similarity, even consuming more resources.

1.2. Adapter VS. VPT

In the main paper, we build task-specific subspaces via adapter tuning [1]. However, apart from adapter tuning, there are other ways to tune the pre-trained model in a parameter-efficient manner, *e.g.*, visual prompt tuning [3] (VPT). In this section, we combine our method with different subspace build techniques and combine EASE with adapter and VPT, respectively. We conduct experiments on CIFAR100 and ImageNet-R. We keep other settings the same and only change the way of subspace building, and report results in Figure 2.



(a) ImageNet-R B0 Inc20



(b) CIFAR100 B0 Inc10

Figure 2. Experimental results on different subspace tuning methods. **Using adapter tuning shows better performance than VPT.**

As we can infer from the figure, using adapters to build subspaces shows better performance than using VPT, outperforming it by $2-3\%$ on these datasets. The main reason lies in the difference between VPT and adapter, where adapter tuning shows to be a stronger tuning method for pre-trained models. Hence, we choose adapter tuning as the way to build subspaces in EASE.

1.3. Comparison to Upper bound

In the main paper, we conduct inference using the completed prototypes. However, if we can save a subset of exemplars $\mathcal{E}$ from former classes, we do not need to complete former class prototypes and can directly calculate them via:

$$\boldsymbol{p}_{i,b} = \frac{1}{N} \sum_{j=1}^{|\mathcal{E}|} \mathbb{I}(y_j = i) \phi(\mathbf{x}_j; \mathcal{A}_b). \tag{5}$$

Table 1. Comparison to exemplar-based upper bound. EASE **does not use any exemplars while showing competitive performance**.

Method	Exemplars	ImageNet-R B0 Inc20 $\bar{\mathcal{A}}$	ImageNet-R B0 Inc20 $\mathcal{A}_B$	CIFAR B0 Inc10 $\bar{\mathcal{A}}$	CIFAR B0 Inc10 $\mathcal{A}_B$
Upper Bound	20 / class	**81.73**	76.08	92.32	**87.79**
EASE	**0**	**81.73**	**76.17**	**92.35**	87.76

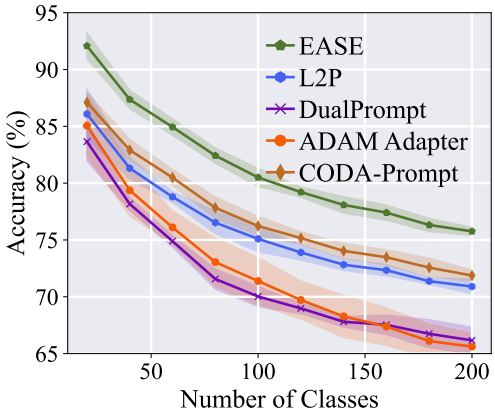


Figure 3. Results on ImageNet-R B0 Inc20 with multiple runs. EASE **consistently outperforms other methods by a substantial margin.**

We denote such a calculation process as the upper bound since the prototypes calculated via Eq. 5 are accurate estimations of the class center. In this section, we compare EASE to upper bound to show its effectiveness and report the results in Table 1.

As we can infer from the table, EASE shows competitive performance to the upper bound, achieving almost the same results without using any exemplars. Results verify the effectiveness of using semantic information to conduct prototype complement.

1.4. Multiple Runs

In the main paper, we conduct experiments on different datasets and follow [5] to shuffle class orders with random seed 1993. In this section, we also run the experiments multiple times using different random seeds, *i.e.*, {1993,1994,1995,1996,1997}. Hence, we can obtain five incremental results of different methods and report the mean and standard variance in Figure 3.

As we can infer from the figure, EASE consistently outperforms other methods by a substantial margin given various random seeds.

1.5. Running Time Comparison

In this section, we report the running time comparison of different methods. We utilize a single NVIDIA 4090 GPU

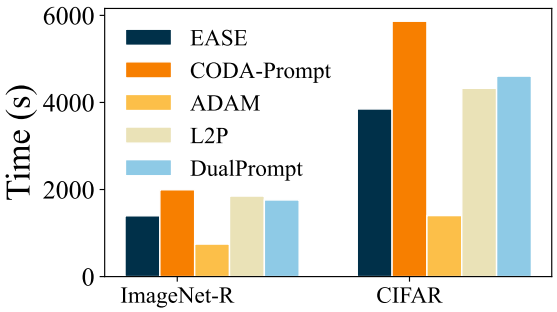


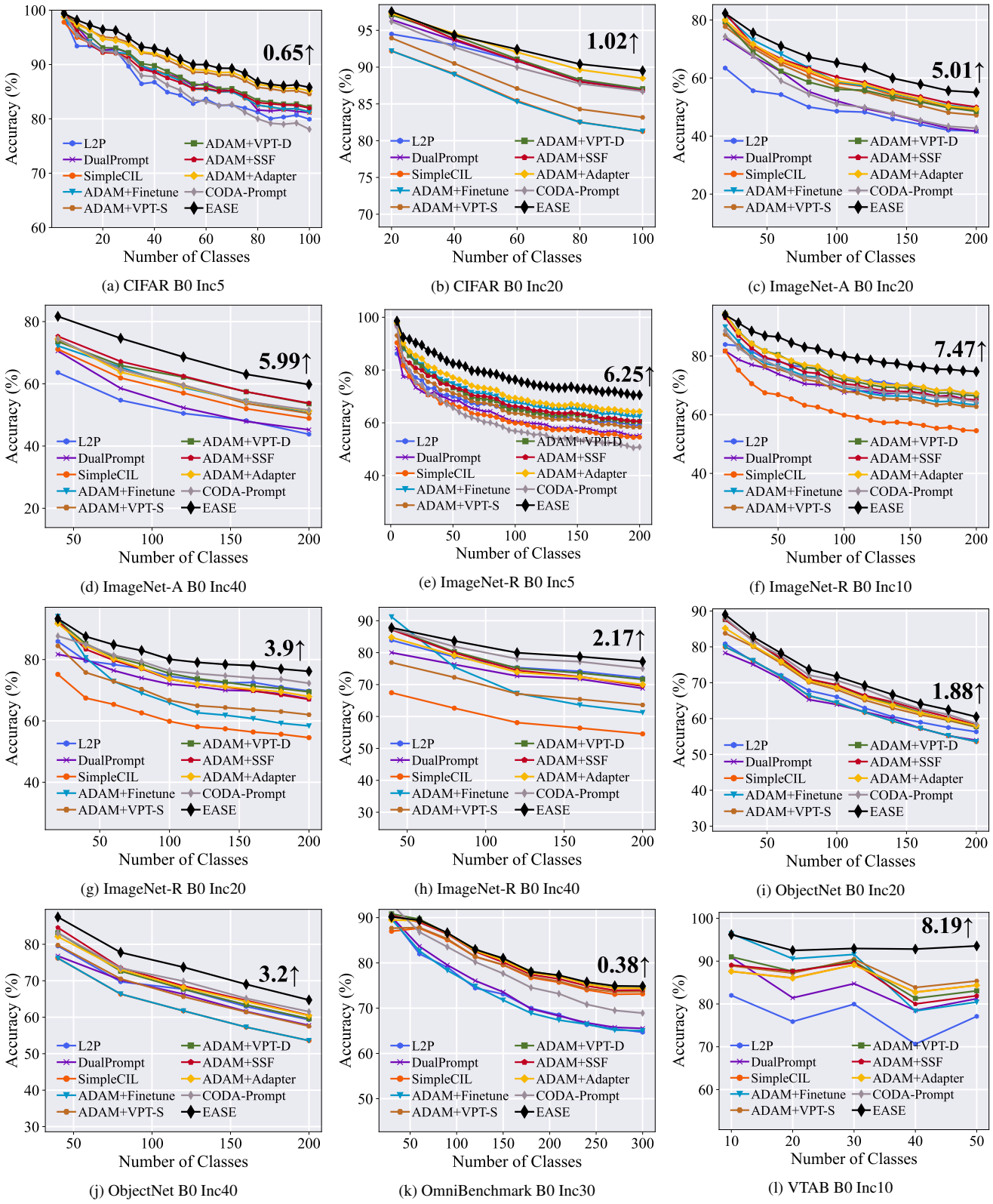
Figure 4. Running time comparison of different methods. EASE **utilizes less running time than CODA-Prompt, L2P, and DualPrompt while having better performance.**

to run the experiments and report the results in Figure 4. As we can infer from the figure, EASE requires less running time than CODA-Prompt, L2P, and DualPrompt, while having the best performance. Experimental results verify the effectiveness of EASE.

2. Introduction About Compared Methods

In this section, we introduce the details of compared methods adopted in the main paper. **All methods are based on the same pre-trained model for a fair comparison.** They are listed as:

- **Finetune**: with a pre-trained model as initialization, it finetunes the PTM with cross-entropy loss for every new task. Hence, it suffers sever catastrophic forgetting on former tasks.
- **LwF [4]**: aims to utilize knowledge distillation [2] to resist forgetting. In each new task, it builds the mapping between the last-stage model and the current model to reflect old knowledge in the current model.
- **SDC [11]**: utilizes a prototype-based classifier. During model updating, the feature drifts, and the old prototypes cannot represent former classes. Hence, it utilizes new class instances to estimate the drift of old classes.
- **L2P [9]**: is the first work introducing pre-trained vision-transformers into continual learning. During model updating, it freezes the pre-trained weights and utilizes visual prompt tuning [3] to trace the new task's features. It builds instance-specific prompts with a prompt pool, which is constructed via key-value mapping.
- **DualPrompt [8]**: is an extension of L2P, which extends the prompt into two types, *i.e.*, general and expert prompts. The other details are kept the same with L2P, *i.e.*, using the prompt pool to build instance-specific prompts.
- **CODA-Prompt [6]**: noticing the drawback of instance-specific prompt select, it aims to eliminate the prompt



(a) CIFAR B0 Inc5 (b) CIFAR B0 Inc20 (c) ImageNet-A B0 Inc20

(d) ImageNet-A B0 Inc40 (e) ImageNet-R B0 Inc5 (f) ImageNet-R B0 Inc10

(g) ImageNet-R B0 Inc20 (h) ImageNet-R B0 Inc40 (i) ObjectNet B0 Inc20

(j) ObjectNet B0 Inc40 (k) OmniBenchmark B0 Inc30 (l) VTAB B0 Inc10

Figure 5. Performance curve of different methods under different settings. All methods are initialized with **ViT-B/16-IN21K**. We annotate the relative improvement of EASE above the runner-up method with numerical numbers at the last incremental stage.

selection process by prompt reweighting. The prompt selection process is replaced with an attention-based prompt recombination.

- **SimpleCIL [14]**: explores prototype-based classifier with vanilla pre-trained model. With a PTM as initialization, it builds the prototype classifier for each class and utilizes a cosine classifier for classification.
- **ADAM [14]**: extends SimpleCIL by aggregating the pre-trained model and adapted model. It treats the first incremental stage as the only adaptation stage and adapts the PTM to extract task-specific features. Hence, the model can unify generalizability and adaptivity in a unified framework.

Above methods are exemplar-free, which do not require using exemplars. However, we also compare some exemplar-based methods in the main paper as follows:

- **iCaRL [5]**: utilizes knowledge distillation and exemplar replay to recover former knowledge. It also utilizes the nearest center mean classifier for final classification.
- **DER [10]**: explores network expansion in class-incremental learning. Facing a new task, it freezes the prior backbone to keep it in memory and initializes a new backbone to extract new features for the new task. With all historical backbones in the memory, it utilizes the concatenation as feature representation and learns a large linear layer as the classifier. The linear layer maps the concatenated features to all seen classes, requiring exemplars for calibration. DER shows impressive results in class-incremental learning, while it requires large memory costs for saving all historical backbones.
- **FOSTER [7]**: to alleviate the memory cost of DER, it proposes to compress backbones via knowledge distillation. Hence, only one backbone is kept throughout the learning process, and it achieves feature expansion with low memory cost.
- **MEMO [13]**: aims to alleviate the memory cost of DER from another aspect. It decouples the network structure into specialized (deep) and generalized (shallow) layers and extends specialized layers based on the shared generalized layers. Hence, the memory cost for network expansion decreases from a whole backbone to generalized blocks. In the implementation, we follow [13] to decouple the vision transformer at the last transformer block.

In the experiments, we reimplement the above methods based on their source code and PyCIL [12].

3. Full Results

In this section, we show more experimental results of different methods. Specifically, we report the incremental performance of different methods with ViT-B/16-IN21K in Figure 5. As shown in these results, EASE consistently outperforms other methods on different datasets by a substantial margin.

Algorithm 1 EASE for CIL

Input: Incremental datasets: $\{\mathcal{D}^1, \mathcal{D}^2, \cdots, \mathcal{D}^B\}$, Pre-trained embedding: $\phi(\mathbf{x})$;
Output: Incrementally trained model;

1: **for** $b = 1, 2 \cdots, B$ **do**
2: Get the incremental training set $\mathcal{D}^b$;
3: Initialize a new adapter $\mathcal{A}_b$;
4: Optimize the subspace via Eq. 5;
5: Extract the prototypes of $\mathcal{D}^b$ for all adapters via Eq. 7;
6: Complete the prototypes for former classes via Eq. 9;
7: Construct the prototypical classifier via Eq. 10;
8: Test the model via Eq. 12;
return the updated model;

4. Pseudo Code

We summarize the training pipeline of EASE in Algorithm 1. We initialize and train an adapter for each incoming task to encode the task-specific information (Line 4). Afterward, we extract the prototypes of the current dataset for all adapters and synthesize the prototypes of former classes (Line 6). Finally, we construct the full classifier and reweight the logit for prediction (Line 8). Since we are using the prototype-based classifier for inference, the classifier W in Eq. 5 will be dropped after each learning stage.

References

[1] Shoufa Chen, GE Chongjian, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, 2022. 2

[2] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[3] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727, 2022. 2, 3

[4] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017. 3

[5] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017. 3, 5

[6] 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. 3

[7] 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. 5

[8] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vin-

cent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, pages 631–648, 2022. 3

[9] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, pages 139–149, 2022. 3

[10] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, pages 3014–3023, 2021. 5

[11] Lu Yu, Bartlomiej 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. 3

[12] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: a python toolbox for class-incremental learning. *SCIENCE CHINA Information Sciences*, 66(9):197101–, 2023. 5

[13] 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. 5

[14] Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *arXiv preprint arXiv:2303.07338*, 2023. 5