Appendices to “eTag: Class-Incremental Learning via Embedding Distillation and Task-Oriented Generation”

Libo Huang[1], Yan Zeng[2], Chuanguang Yang[1], Zhulin An[1*], Boyu Diao[1], Yongjun Xu[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences
[2] School of Mathematics and Statistics, Beijing Technology and Business University
{www.huanglibo, yanazeng013}@gmail.com, {yangchuanguang, anzhulin, diaoboyu2012, xyj}@ict.ac.cn

In this Appendix, we provide further details about,

A. Experimental settings, including the explanations of the dataset, evaluation metric, comparison method, and implementation details.
B. Additional overall evaluation, including the experiments on ImageNet-100, ImageNet-1000, and Tiny-ImageNet compared with the recent exemplar-free and sample-generative CIL methods.
C. Additional Ablation Study, including the analysis of SS, architecture generalization, Tag’s effectiveness, time consumption, and memory budgets.

Appendix A Experimental Settings

In this section, we provide the details of datasets, evaluation metrics, comparison methods, and implementation.

A.1 Datasets

The experiments are conducted on four benchmark datasets: CIFAR-100 (Liu et al. 2020), ImageNet-100 (Wang et al. 2022), ImageNet-1000 (Deng et al. 2009), and Tiny-ImageNet (Zhu et al. 2021a). CIFAR-100 consists of $60,000$ images belonging to 100 classes, with 500 images per class for training and 100 images per class for testing. The size of each image is 32×32 pixels, and they are padded with 4 pixels at first (Masana et al. 2022). During training, 32×32 crops are randomly sampled, while center crops are used for testing. ImageNet-100 is a subset of ImageNet-1000, containing 100 randomly selected classes. Each class has more than $1,000$ images for training and 500 for testing. Each image is first resized to 256×256 pixels before randomly sampling 224×224 crops during training. For testing, the original center crop with 224×224 pixels is used. Tiny-ImageNet is a dataset similar to ImageNet-1000, but it contains smaller images measuring 64×64 pixels. The dataset consists of a total of 200 classes. Each class has 500 training images and 50 test images.

A.2 Evaluation Metric

We employ three metrics: *average incremental accuracy* (A), *average forgetting measure* (F), and classification *accuracies* (ACC) for evaluation, following previous works (Hou et al. 2019; Masana et al. 2022). The metric A measures the overall performance of the CIL model on all learned tasks, where higher values indicate better performances. The metric F measures the amount of overall forgetting during CIL, where lower values indicate less forgetting. ACC is a more detailed metric evaluated on each learned task separately (Yu et al. 2020).

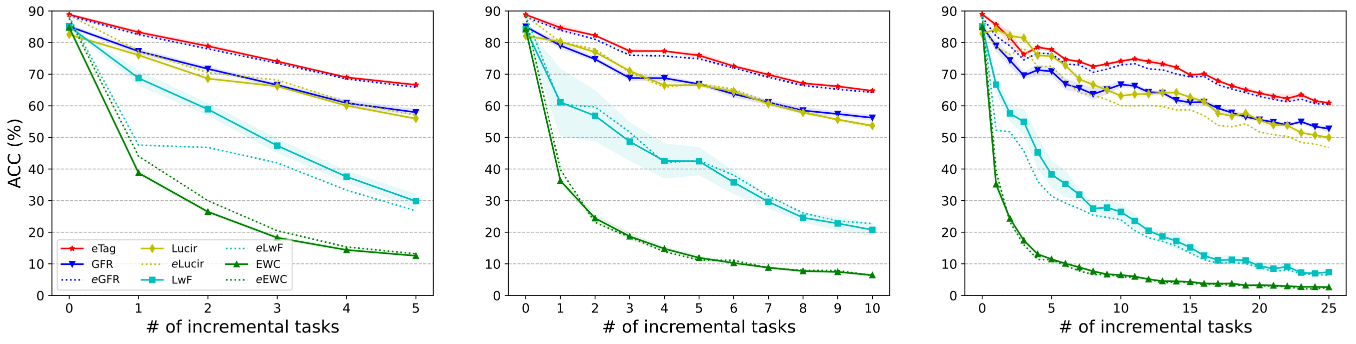
A.3 Comparison Method

We provide the details about the Comparison methods of the main paper. The Fine method updates the network only for the new task, while the Joint method assumes that all previous task data are available during training for each task (Chen and Liu 2018). Methods like EWC (Kirkpatrick et al. 2017), MAS (Aljundi et al. 2018), LwF (Li and Hoiem 2017), LwM (Dhar et al. 2019), and GFR (Liu et al. 2020) are trained without exemplars, whereas PASS (Zhu et al. 2021b) stores prototypes and their variances, while IL2M (Belouadah and Popescu 2019) and Lucir (Hou et al. 2019) store 20 exemplars for each class. Multi-head networks are used for EWC, MAS, LwF, and LwM; the highest probability result is taken during inference. For GFR, the same VAE generator as used in the eTag is employed to produce features. Other experimental parameters are consistent with those in (Masana et al. 2022) paper, and all experiments are repeated three times with different random seeds.

A.4 Implementation Details

The models used in our experiments are implemented using the PyTorch framework (Paszke et al. 2017) and trained on V-100 GPUs. For CIFAR-100 and Tiny-ImageNet, we modify the first convolutional layer of ResNet-18 (He et al. 2016) with 3×3 kernels, while for ImageNet-100 and ImageNet-1000, we use the original ResNet-18. Models are trained for 100 epochs on CIFAR-100 and Tiny-ImageNet, while 70 on ImageNet-100 and ImageNet-1000. The generator is trained for 100 epochs. We use the Adam optimizer (Kinga, Adam et al. 2015) with a batch size of 128, and train all models from scratch. The initial learning rate is set to $1e-3$ for classification and $1e-4$ for the generator, and is divided by 10 for every 30 epoch (Masana et al. 2022).

[*]Zhulin An is the corresponding author.



(a) 5 incremental tasks (b) 10 incremental tasks (c) 25 incremental tasks

Figure A.1: Detailed ACCs on ImageNet-100 with 5, 10, and 25 class-incremental tasks. Results are averaged after 3 runs.

Table B.1: Average incremental accuracy on ImageNet-1000.

Method	GFR	DER	Foster	eTag
$A(\uparrow)$	63.42%	66.73%	68.34%	70.28%

Appendix B Additional Overall Evaluation

In this section, we provide additional experiments about the overall evaluation, including the experiments on ImageNet-100, ImageNet-1000, and Tiny-ImageNet compared with the recent exemplar-free and sample-generative CIL methods.

B.1 Experiments on ImageNet-100

Fig.A.1 demonstrates the detailed ACCs on ImageNet-100. Similar to the conclusion drawn in Sec.4.1 of the main paper, directly plugging in the embedding distillation in these SOTA methods is not constantly helpful for incremental tasks. For instance, in the case of 25 incremental tasks, eLucir exhibits lower accuracy than Lucir. It may be due to two reasons. Firstly, the embedding is constructed with the conventional CE loss, which differs from the final classifier of eLucir, a variant of CE loss with cosine normalization, causing confusion between the intermediate embeddings and the final prediction. Secondly, the embedding is distilled with conventional KD loss, which contradicts that of eLucir, i.e., the correlations between the parameters of the final classifier and output features. Hence, we hold that ensuring the consistency between the classification and distillation metrics may matter for class incremental learning.

B.2 Experiments on ImageNet-1000

Following the same setting in DER (Yan, Xie, and He 2021) and FOSTER (Wang et al. 2022), we also conduct experiments on ImageNet-1000, using 10 equally divided CIL tasks with each task containing 100 different classes. DER and FOSTER are exemplar-based CIL methods, meaning they store exemplars for each learned class. As shown in Tab.B.1, eTag outperforms GFR, DER, and FOSTER by 6.86%, 3.55%, and 1.94%, respectively, validating the effectiveness of eTag on the large-scale dataset.

Table B.2: Average incremental accuracy of various exemplar-free CIL methods on CIFAR-100 and Tiny-ImageNet. We reproduced the results under the same implementations as Appendix-A.4, and also copied the results from the recently public papers (Petit et al. 2023; Zhu et al. 2022; Gao et al. 2022) indicated with †. Bold and underline indicate the **best** and the <u>second best</u> results, respectively.

Method	CIFAR-100			Tiny-ImageNet		
	5 tasks	10 tasks	20 tasks	5 tasks	10 tasks	20 tasks
IL2A	62.74	58.94	56.28	50.02	47.52	41.34
IL2A†	66.00	60.30	57.90	47.30	44.70	40.00
R-dfcil	61.83	58.72	56.32	49.98	47.84	43.76
R-dfcil†	64.78	61.71	49.95	48.91	47.60	40.85
SSRE	61.40	57.06	55.62	50.44	48.20	40.05
SSRE†	65.88	65.04	<u>61.70</u>	50.39	48.93	48.17
Fetril	63.29	61.36	59.48	50.71	47.96	45.03
Fetril†	<u>66.30</u>	<u>65.20</u>	61.50	<u>54.80</u>	<u>53.10</u>	**52.20**
eTag	**67.99**	**65.50**	**63.07**	**56.81**	**54.64**	<u>51.25</u>

B.3 Compared with Exemplar-free Methods

The exemplar-free CIL methods broadly belong to the data-free CIL methods, although most require storing prototypes, i.e., class mean features, for each learned class. We conduct experiments with several exemplar-free CIL methods, IL2A (Zhu et al. 2021a), R-dfcil (Gao et al. 2022), SSRE (Zhu et al. 2022), and Fetril (Petit et al. 2023), under the same setting on CIFAR-100 and Tiny-ImageNet datasets (Zhu et al. 2021a, 2022; Petit et al. 2023).

Similar to (Zhu et al. 2021a, 2022; Petit et al. 2023), we train the model on half of the classes for the initial task, and equal classes in the rest tasks. We reproduce these exemplar-free methods under our configuration detailed in Appendix-A.4 and copy their recent public papers' results.

As shown in Tab.B.2, all these recent CIL methods achieve considerable results without exemplars, and eTag outperforms them in most cases on CIFAR-100 and Tiny-ImageNet with 5, 10, and 20 tasks. This is mainly attributed to our proposed embedding distillation and task-oriented generation, which respectively transfer rich dark knowledge underlying the intermediate blocks of the backbone and generate appropriate final features for the top classifier.

Table B.3: Average incremental accuracy of various generative CIL methods on CIFAR-100.

Method	BI-R	GC	ABD	eTag
$A(\uparrow)$	34.38($\pm$0.21)	49.55($\pm$0.06)	50.17($\pm$0.43)	54.36($\pm$0.22)

Table C.1: Results same as Tab.2 from the main paper.

Baselines	B0	B1	B2	B3	B4	eTag
GE	T	N	T	N	N	T
KD	E	N	N	E	E	E
SS	-	-	-	✓	-	✓
$A(\uparrow)$	58.38	57.30	59.82	61.18	56.89	64.10

Table C.2: Effects of different SS augmented tasks.

SS Type	None	jigsaw puzzles	color channel permutation	random rotations
$A(\uparrow)$	58.38	59.11	56.66	64.10

Table C.3: Various backbone results on CIFAR-100.

Backbone	VGG-11	ResNet-18	TinyViT-5M	TinyViT-11M
$A(\uparrow)$	59.34	64.10	69.52	72.36

Table C.4: Tag versus fixed prototype.

Baseline	KD	GE	$A(\uparrow)$
eTag	E	Tag	74.10
ePASS	E	Fix	59.42
A	N	Tag	60.54
B	N	Fix	56.93

B.4 Compared with Sample-Generative Methods

Following the same setting in GC (van de Ven, Li, and Tolias 2021), we conduct the experiment on CIFAR-100 with 10 equally divided tasks. As shown in Tab.B.3, eTag achieves the best incremental accuracy among the compared methods. This is mainly attributed to eTag's task-oriented feature generation, while the other methods use sample generation.

Appendix C Additional Ablation Study

In this section, we provide additional ablation studies about different SS, generative architecture, Tag's effectiveness, time consumption, and memory budget.

C.1 SS Analysis.

We analyze the effects of the self-supervised (SS) tasks used in eTag: i) effects of whether to use SS or not; and ii) effects of different SS tasks.

Regarding i) effects of whether to use SS or not, we compare our eTag with five baselines, B0, B1, B2, B3, and B4. B0 is similar to eTag only without SS, B1 uses naïve feature generation and naïve knowledge distillation (KD). B2 replaces B1's naïve feature generation with the task-oriented generation, while B4 replaces B1's naïve KD with embedding KD. B3 improves B4 by using the SS task.

The results in Tab.C.1 demonstrate the efficacy of SS tasks in improving incremental learning performance, particularly indicating that SS and embedding distillation are a unified technique for performance-enhancing in eTag. For example, the ACC of B4 (with naïve feature generation and embedding KD) is 56.89, worse than that of B1 (with naïve feature generation and naïve KD) at 57.30. It reaches a similar comparison result for the "B0 versus B2", saying that without SS, naïve distillation is better than embedding distillation, which is consistent with the finding in Tian, Krishnan, and Isola (2019). This may be because using knowledge distillation only for incremental learning always incurs a data drift problem (Wu et al. 2018), i.e., the training data of the previous network (i.e., the teacher in the KD community) is from the old tasks while the training data during distillation is from the current task. Therefore, using KD in CIL is quite prone to transfer variant features or embeddings. In contrast, SS can introduce invariant representations (Zhu et al. 2021b; Xu et al. 2020), such as rotation-invariant embeddings that are independent of the task sequence.

Overall, the comparison pair "B1 versus B2" reflects the effectiveness of task-oriented feature generation; the comparison pair "B1 versus B3" reflects the effectiveness of embedding distillation; and the comparison pairs "B3 versus B4" and "B0 versus eTag" along with the above analyses reflects the effectiveness of SS.

Regarding ii) effects of different SS tasks, we conduct experiments with different SS tasks, including random rotations, jigsaw puzzles (Noroozi and Favaro 2016), and color channel permutation (Zhang, Isola, and Efros 2016). The results in Tab.C.2 suggest that random rotations are more effective in transferring dark knowledge from history networks to the current one than other SS tasks. Without using SS tasks, the performance of eTag drops significantly from 64.10% to 58.38%, highlighting the importance of incorporating SS tasks in incremental learning.

C.2 Architecture Generalization Analysis

The proposed eTag framework mainly refers to generator and backbone architectures. The generator can be generalized as it takes low-dimension features as input. In Fig.1(d), a 5-layer fully connected (FC) generator is used, while other experiments in the paper employ a 3-layer FC generator. They outperform DGR and other baselines, demonstrating the generality of the generator architecture. Additionally, the backbone can be generalized to other modern convolutional neural networks, including TinyViT, as mentioned in Footnote 2 of the paper and Yang et al. (2021). We conducted experiments on four equally divided tasks of CIFAR-100. As shown in Tab.C.3, eTag based on VGG and TinyViT also works, and performances vary by the number of parameters.

C.3 Tag's Effectiveness Analysis

We conducted experiments on four equally divided tasks of CIFAR-100. As shown in Tab.C.4, under the same embedding knowledge distillation ($KD = E$), eTag (using Tag to GEnerate old features achieved 64.10) outperforms ePASS

Table C.5: Training and testing time (in second) of baselines.

Method	Training	Testing	$A(\uparrow)$
PodNet	4793.09s	11.80s	58.05
ABD	13131.21s	14.31s	57.31
B0	6406.54s	12.78s	58.38
B1	3050.58s	12.12s	57.30
B2	3131.67s	13.13s	59.82
B3	12814.07s	12.85s	61.18
B4	3102.17s	12.77s	56.89
eTag	12858.18s	13.87s	64.10

Table C.6: Training parameters (in million (M)) and memory budgets (in megabytes (MB)) of different methods built upon ResNet-18 for CIFAR-100 and ImageNet-100.

Method	Parameters (M)		Memory Budget (MB)	
	CIFAR-100	ImageNet-100	CIFAR-100	ImageNet-100
Joint/Fine	11.22	11.38	42.80	43.41
eIL2M; IL2M	41.79; 11.27	42.19; 11.43	48.84; 48.84	330.52; 330.52
eLucir; Lucir	41.79; 11.27	42.19; 11.43	48.84; 48.84	330.52; 330.52
ePASS; PASS	41.74; 11.22	42.14; 11.38	42.82; 42.82	43.61; 43.61
eGFR; GFR	43.20; 12.68	43.60; 12.84	48.65; 48.65	48.98; 48.98
eTag	43.20	43.60	45.38	46.00

(using Fixed prototype to GEnerate old features achieved 59.42) by 4.68%. Under the same naïve knowledge distillation ($KD = N$), baselines A (using Tag to GEnerate old features achieved 60.54) outperforms B (using Fixed prototype to GEnerate old features achieved 56.93) by 3.61%, verifying the effectiveness of Tag.

C.4 Time Consumption Analysis

To evaluate the training and testing time of each component of eTag, we conduct experiments with PodNet (Douillard et al. 2020), ABD (Smith et al. 2021), and all baselines, including B0, B1, B2, B3, and B4. PodNet transfers vanilla feature maps following FitNet (Romero et al. 2015) while ABD is a recently proposed data-free CIL method by replaying the generated old samples. As shown in Tab.C.5, we can find that the extra training consumption of eTag lies in two parts, the auxiliary classifiers and the constructed SS tasks. On the one hand, B0 mainly incorporates the auxiliary classifiers from B1, which increases the training time from 3050.58s to 6406.54s. Note that B2 replaces naive feature generation with task-oriented feature generation from B1, resulting in a minor and negligible increase in training time. On the other hand, eTag builds upon B0 by incorporating the constructed SS tasks, leading to a further increase in training time from 6406.54s to 12858.18s. Compared with the recent data-free CIL method, ABD (Smith et al. 2021), eTag's training time is within an acceptable range, i.e., 13131.21s versus 12858.18s. Though the training time of eTag is higher than most of the comparisons, our accuracy outperforms much better than others. Note further that we could discard eTag's auxiliary classifiers during inference. Thus eTag performs similar testing time as others, i.e., around 13s.

C.5 Memory Budgets Analysis

Following the recent advocacy of the CIL research community (Zhou et al. 2023), we list the training parameters and memory budgets of various methods upon ResNet-18 for CIFAR-100 and ImageNet-100. As shown in Tab.C.6, we observe that: a) Our method and modified methods eXX cost a larger number of parameters, than those original baseline methods. This majorly is because our embedding distillation-like methods have adopted an auxiliary classifier behind each block, and the modified methods eXX engage a similar architecture as eTag. b) Regarding memory budges, eTag needs comparable resources to other baselines for CIFAR-100, whereas for ImageNet-100, eTag, GFR, and PASS need far fewer resources than IL2M and Lucir. This majorly is because the former methods are data-free while the latter are exemplar-based. Date-free methods are not prone to significantly increase budget memory as the number of classes or tasks increases.

References

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 139–154.

Belouadah, E.; and Popescu, A. 2019. IL2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 583–592.

Chen, Z.; and Liu, B. 2018. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3): 1–207.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 248–255. Ieee.

Dhar, P.; Singh, R. V.; Peng, K.-C.; Wu, Z.; and Chellappa, R. 2019. Learning without memorizing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5138–5146.

Douillard, A.; Cord, M.; Ollion, C.; Robert, T.; and Valle, E. 2020. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of European Conference on Computer Vision (ECCV)*, 86–102. Springer.

Gao, Q.; Zhao, C.; Ghanem, B.; and Zhang, J. 2022. R-dfcil: Relation-guided representation learning for data-free class incremental learning. In *Proceedings European Conference on Computer Vision (ECCV)*, 423–439. Springer.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z.; and Lin, D. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 831–839.

Kinga, D.; Adam, J. B.; et al. 2015. A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, volume 5, 6. San Diego, California;.

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. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13): 3521–3526.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12): 2935–2947.

Liu, X.; Wu, C.; Menta, M.; Herranz, L.; Raducanu, B.; Bagdanov, A. D.; Jui, S.; and de Weijer, J. v. 2020. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 226–227.

Masana, M.; Liu, X.; Twardowski, B.; Menta, M.; Bagdanov, A. D.; and 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.

Noroozi, M.; and Favaro, P. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, 69–84. Springer.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. *Advances in Neural Information Processing Systems Workshop (NIPSW)*.

Petit, G.; Popescu, A.; Schindler, H.; Picard, D.; and Delezoide, B. 2023. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3911–3920.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2015. Fitnets: Hints for thin deep nets. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Smith, J.; Hsu, Y.-C.; Balloch, J.; Shen, Y.; Jin, H.; and Kira, Z. 2021. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 9374–9384.

Tian, Y.; Krishnan, D.; and Isola, P. 2019. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*.

van de Ven, G. M.; Li, Z.; and Tolias, A. S. 2021. Class-incremental learning with generative classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3611–3620.

Wang, F.-Y.; Zhou, D.-W.; Ye, H.-J.; and Zhan, D.-C. 2022. Foster: Feature boosting and compression for class-incremental learning. In *European conference on computer vision (ECCV)*, 398–414. Springer.

Wu, C.; Herranz, L.; Liu, X.; van de Weijer, J.; Raducanu, B.; et al. 2018. Memory replay gans: Learning to generate new categories without forgetting. *Advances in Neural Information Processing Systems (NIPS)*, 31.

Xu, G.; Liu, Z.; Li, X.; and Loy, C. C. 2020. Knowledge distillation meets self-supervision. In *European Conference on Computer Vision (ECCV)*, 588–604. Springer.

Yan, S.; Xie, J.; and He, X. 2021. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3014–3023.

Yang, C.; An, Z.; Cai, L.; and Xu, Y. 2021. Hierarchical self-supervised augmented knowledge distillation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1217–1223.

Yu, L.; Twardowski, B.; Liu, X.; Herranz, L.; Wang, K.; Cheng, Y.; Jui, S.; and Weijer, J. v. d. 2020. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6982–6991.

Zhang, R.; Isola, P.; and Efros, A. A. 2016. Colorful image colorization. In *European Conference on Computer Vision (ECCV)*, 649–666. Springer.

Zhou, D.-W.; Wang, Q.-W.; Qi, Z.-H.; Ye, H.-J.; Zhan, D.-C.; and Liu, Z. 2023. Deep class-incremental learning: A survey. *arXiv preprint arXiv:2302.03648*.

Zhu, F.; Cheng, Z.; Zhang, X.-y.; and Liu, C.-l. 2021a. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 14306–14318.

Zhu, F.; Zhang, X.-Y.; Wang, C.; Yin, F.; and Liu, C.-L. 2021b. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5871–5880.

Zhu, K.; Zhai, W.; Cao, Y.; Luo, J.; and Zha, Z.-J. 2022. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9296–9305.