Class-Incremental Exemplar Compression for Class-Incremental Learning
Supplementary Materials

Zilin Luo[1] Yaoyao Liu[2] Bernt Schiele[2] Qianru Sun[1]

[1]Singapore Management University
[2]Max Planck Institute for Informatics, Saarland Informatics Campus

zilin.luo.2021@phdcs.smu.edu.sg {yaoyao.liu, schiele}@mpi-inf.mpg.de qianrusun@smu.edu.sg

These supplementary materials include implementation details (§1), dataset details (§2), SOTA 95% CI results (§3), learned PAU results (§4), compression footprint results (§5), sensitivity analysis results (§6), overhead analysis results (§7), and hardware information (§8).

1. Implementation Details

Downsampling Method. Supplementary to Section 4.1 "**Compression with BBox**". For generating compressed exemplars, we adopted a simple downsampling method (to apply on non-discriminative pixels) called Nearest Neighbor Interpolation [11]. Specifically, it replaces the pixel value with that of the nearest pixel. This can be achieved by calling the `cv2.resize()` function with the `INTER_NEAREST` flag in the standard image processing library OpenCV [2].

Mitigating the Effect of Compression Artifacts. Supplementary to Section 4.1 "**Compression Artifacts**". We mitigated the effect of compression artifacts by applying a augmentation method to new-class data $\mathcal{D}_i$ in each learning phase. Specifically, there were three steps. 1) For all images in $\mathcal{D}_i$, we generated the CAM-based bounding boxes at the beginning of training and updated them once per 40 epochs. 2) In each epoch, we randomly selected a subset of $\mathcal{D}_i$. The proportion of the subset was adjusted according to the training progress—0 at the beginning and increased by 0.1 every 40 epochs. 3) Before feeding each image in the subset (into the CIL model), we downsampled (with ratio η) the pixels outside the bounding box (obtained in step 1).

Compressing $\mathcal{D}_i$ into $\tilde{\mathcal{D}}_i(\phi_i)$. Supplementary to Section 4.2 "**2) Mask-level Optimization**". Different from the compressed pipeline introduced in Section 4.1, compression in the inner-level optimization should involve only differentiable operations to enable gradient descent. To achieve this, we skipped the steps of adding the threshold and bounding box and obtained the compressed images by applying continuously valued masks as follows,

$$A(\phi_i) = \omega_{i,y}^{\top} F(x; \theta_i, \phi_i), \tag{S1a}$$

$$\mathcal{M}^{\text{CAM}}(\phi_i) = \frac{A(\phi_i) - \min\left(A(\phi_i)\right)}{\max\left(A(\phi_i)\right) - \min\left(A(\phi_i)\right)}, \tag{S1b}$$

$$\tilde{x}(\phi_i) = \mathcal{M}^{\text{CAM}}(\phi_i) \odot x + (1 - \mathcal{M}^{\text{CAM}}(\phi_i)) \odot x_\eta. \tag{S1c}$$

Comparing with Other Compression-based Methods. Supplementary to Table 4. The code of plugging MRDC [15] into PODNet [5] (which shows the best results in its original paper) was not released by authors. So we rerun the results of MRDC when plugging it into LUCIR [6]. The experiments are conducted on the LFH setting. For a fair comparison, we apply weight transfer operations [13] in all these experiments following Mnemonics [8].

2. Dataset Details

Supplementary to Section 4.1 "**Datasets**". We show the details about three datasets in Table S1. We elaborate the image preprocessing methods applied on the three datasets in Table S2. Please note that for image preprocessing, we strictly followed [5–8, 12, 14, 16] for a fair comparison.

3. More Results Comparing with the SOTA

Supplementary to Table 1. In Table S3, we report the 95% confidence intervals corresponding to the numbers in the Table 1 of the main paper.

4. Results of Learned PAUs

Supplementary to Section 5.2 "Results and Analyses". Figure S1 shows the activation distances in different network blocks and phases. We measure the distance by $\int_{-3}^{3} |f_{\text{PAU}}(x) - f_{\text{ReLU}}(x)| dx$ and use 601 interpolated points to approximate the integration value. The learned PAUs in the last block have larger distances than those in shallow

Dataset	#Classes	#Training images	#Test images	Avg. size
Food-101 [1]	101	75,750	25,250	475×496
ImageNet-100 [12]	100	129,395	5,000	407×472
ImageNet-1000 [4]	1,000	1,281,167	50,000	406×474

Table S1. Details of the three datasets. The "Avg. size" colume shows the average height×width of images of each dataset.

Training transformation	Test transformation	CAM transformation

`RandomResizedCrop(224),`
`RandomHorizontalFlip(0.5),`
`ColorJitter(63/255),`
`ToTensor(),`
`Normalize().`
	`Resize(256),`
	`CenterCrop(224),`
	`ToTensor(),`
	`Normalize().`
		`Resize((224,224)),`
		`ToTensor(),`
		`Normalize().`

Table S2. Uniform image preprocessing methods for the three datasets. "CAM transformation" denotes the preprocessing method for generating CAM [18]. The mean and standard deviation parameters of `Normalize()` are omitted. Note that FOSTER [14] additionally applies AutoAugment [3] in the training transformation, and we followed it for fair comparison.

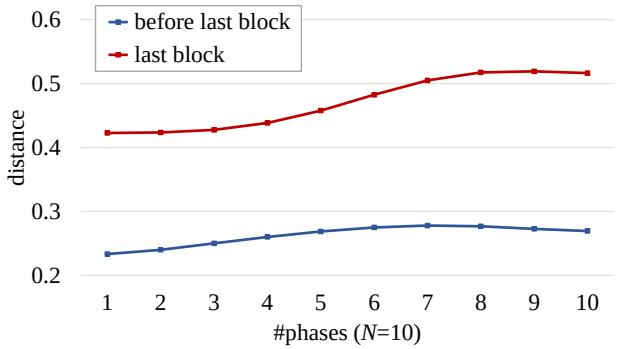


Figure S1. The average activation distances between ReLU [10] and the learned PAUs [9] before and in the last block over all phases. The experimental setting is N=10 (LFS) on ImageNet-100. The curves are smoothed with Gaussian ($\sigma = 2$).

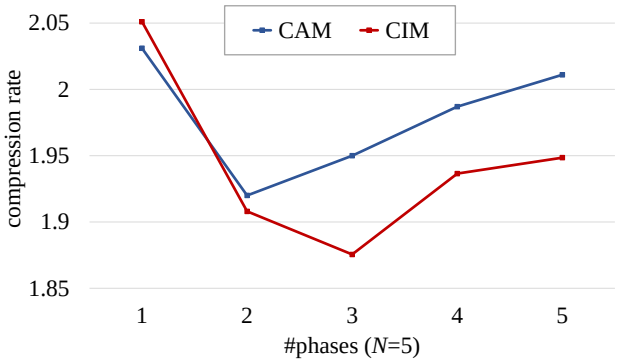


Figure S2. The compression rates resulted from CAM and CIM. The experimental setting is N=5 (LFS) on ImageNet-100.

blocks. The last block mainly encodes high-level semantic information (e.g., "body" of "dog"), this suggests that the learned PAUs are adjusted to focus on the most discriminative semantics. Shallow blocks learn to capture low-level features that are shareable between classification and mask generation. Therefore, the learned PAUs in these shallow blocks have little adjustment. This motivates the variant in Row 10 of Table 3 in the main paper.

5. Results of Compression Footprints.

Supplementary to Section 5.2 "Results and Analyses".. Figure S2 provides the compression footprints along incremental phases. Comparing with CAM, CIM learns to produce more conservative compression footprints in later phases. Our explanation is that more visual cues are required to classify more classes.

6. Results of Sensitivity Analyses

Supplementary to Section 5.1 "Implementation Details". In Figure S3, we show the sensitivity analyses for CAM threshold τ, downsampling ratio η, mask-level learning rates β_1, β_2 and two regularization weights μ, μ' on ImageNet-100 [12]. For τ and η, we also show the computation overheads.

Method	*Learning from Scratch (LFS)*						*Learning from Half (LFH)*					
	Food-101			*ImageNet-100*			*Food-101*			*ImageNet-100*		
	N=5	10	20	5	10	20	5	10	25	5	10	25
iCaRL [12]	0.58	1.11	1.50	0.98	0.64	-	1.24	0.96	0.42	0.86	1.48	1.37
WA [17]	0.43	0.36	2.04	0.65	0.75	-	0.38	0.30	1.54	1.33	-	-
PODNet [5]	0.92	1.91	1.83	1.08	1.87	-	0.80	1.37	1.46	0.29	1.05	2.77
AANets [7]	0.83	1.56	0.74	1.31	1.07	2.01	1.12	0.48	1.22	0.53	0.74	0.81
DER [16]	0.47	0.62	0.94	0.46	0.39	-	0.56	0.68	-	0.51	-	-
DER *w/* ours	0.28	0.75	1.10	0.32	0.24	0.65	0.17	0.52	-	0.41	0.53	-
FOSTER [14]	0.34	0.43	0.89	0.51	0.53	-	0.34	0.32	0.60	0.07	-	0.38
FOSTER *w/* ours	0.45	0.22	1.25	0.36	0.42	0.54	0.26	0.17	0.52	0.44	0.18	0.61

Table S3. The 95% confidence intervals (%) for the results in Table 1.

7. Space and Computation Overheads

Supplementary to Section 4.2 "Limitations". We elaborate on the space overhead by taking ResNet-18 as an example. We add 17 PAUs to it, each with 10 optimizable parameters. So we use 170 extra parameters in total. This is negligible compared to 11 million of network parameters. Besides, we save bbox along with exemplars in the memory. Each bbox takes around 0.01% memory of a 224×224 RGB image. For computation overhead, our method needs around 60% extra computations over baseline CIL training, caused by two factors: 1) BOP between CIL and CIM models; and 2) training on a large number of compressed exemplars.

8. Hardware Information

– **CPU**: AMD EPYC 7F72 24-Core Processor
– **GPU**: $4\times$ NVIDIA GeForce RTX 3090
– **Mem**: $8\times$ DDR4-3200 ECC RDIMM - 32GB

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014. 2

[2] Gary Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 2000. 1

[3] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019. 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[5] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, 2020. 1, 3

[6] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, 2019. 1

[7] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *CVPR*, 2021. 1, 3

[8] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, 2020. 1

[9] Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In *ICLR*, 2019. 2

[10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 2

[11] J Anthony Parker, Robert V Kenyon, and Donald E Troxel. Comparison of interpolating methods for image resampling. *IEEE Transactions on Medical Imaging*, 1983. 1

[12] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, 2017. 1, 2, 3

[13] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019. 1

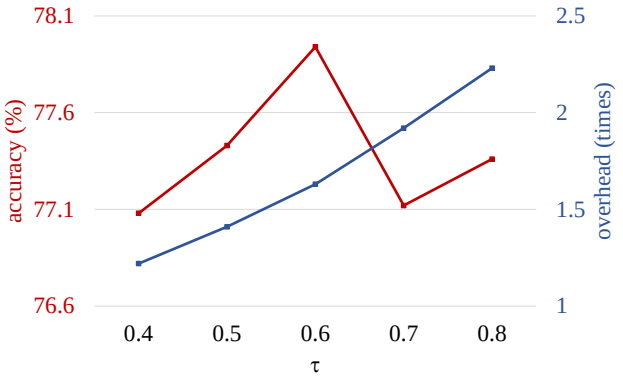
[14] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. *arXiv*, 2022. 1, 2, 3

[15] Liyuan Wang, Xingxing Zhang, Kuo Yang, Longhui Yu, Chongxuan Li, Lanqing Hong, Shifeng Zhang, Zhenguo Li, Yi Zhong, and Jun Zhu. Memory replay with data compression for continual learning. In *ICLR*, 2022. 1

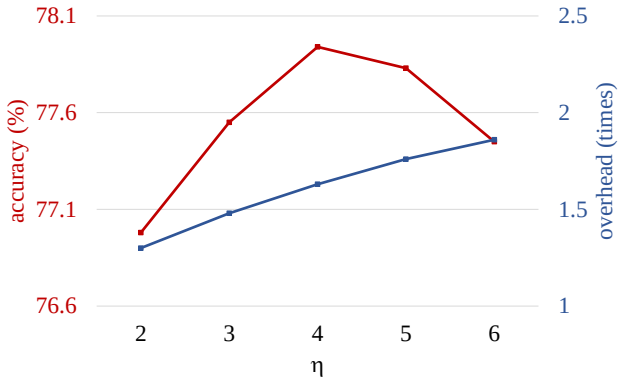
[16] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, 2021. 1, 3

[17] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, 2020. 3

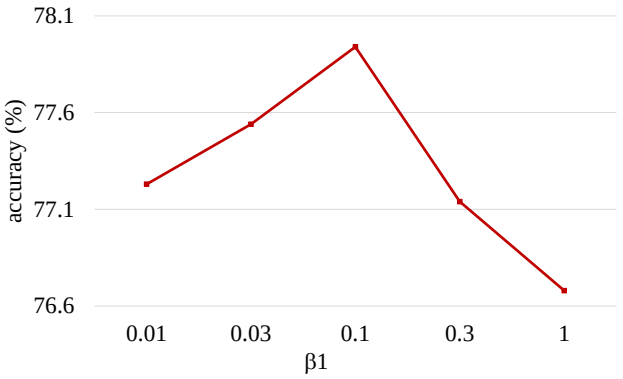
[18] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016. 2



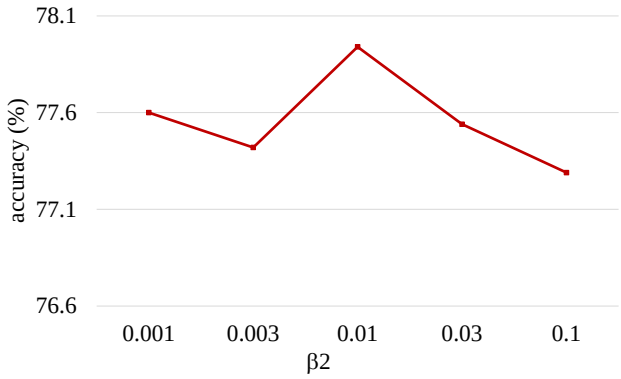
(a) Masking threshold τ.



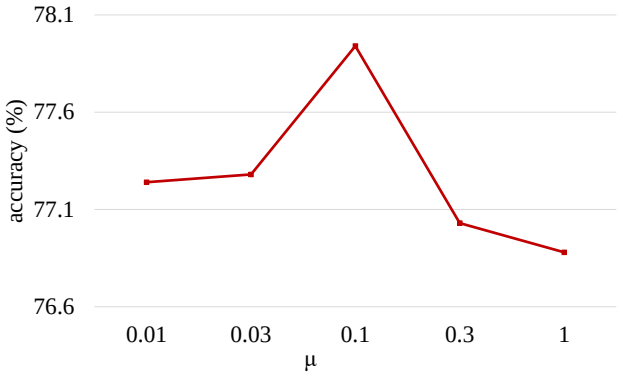
(b) Downsampling ratio η.



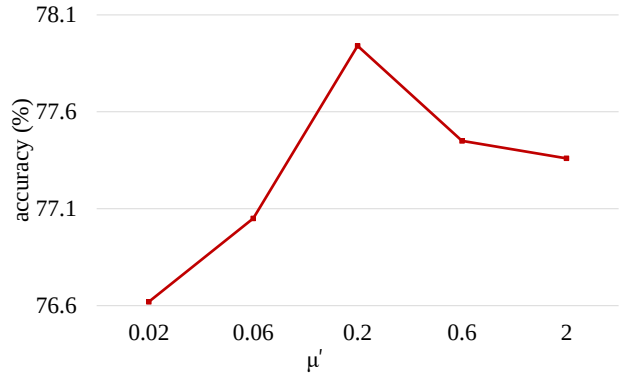
(c) Inner-level learning rate β_1.



(d) Outer-level learning rate β_2.



(e) ℓ_2 regularization weight μ.



(f) CE regularization weight μ'.

Figure S3. Results of hyperparameter sensitivity.