Online Prototype Learning for Online Continual Learning —Supplementary Material—

Yujie Wei[1] Jiaxin Ye[1] Zhizhong Huang[2] Junping Zhang[2] Hongming Shan[1,3,4*]
[1] Institute of Science and Technology for Brain-inspired Intelligence, Fudan University
[2] Shanghai Key Lab of Intelligent Information Processing, School of Computer Science Fudan University
[3] MOE Frontiers Center for Brain Science, Fudan University
[4] Shanghai Center for Brain Science and Brain-inspired Technology
{yjwei22, jxye22}@m.fudan.edu.cn, {zzhuang19, jpzhang, hmshan}@fudan.edu.cn

A. Difference from PCL

PCL [8] bridges instance-level contrastive learning with clustering based on unsupervised representation learning. We discuss the differences between PCL and OPE in the following three parts.

(1) Difference in learning settings. PCL is an unsupervised contrastive learning method while OPE explicitly leverages class labels to compute online prototypes. Thus, OPE belongs to the supervised setting.

(2) Difference in prototype calculation. At each time step (iteration), PCL uses all samples of classes to obtain prototypes by performing K-means clustering. In contrast, OPE just utilizes a mini-batch of training data to calculate online prototypes.

(3) Difference in contrastive form (most significant differences). The anchor of OPE as well as its positive and negative samples are online prototypes, which means no instance is involved, while PCL takes instance-level representation as the anchor and cluster centers as the positive and negative samples. Specifically, OPE regards an online prototype and its augmented view as a positive pair; online prototypes of different classes are regarded as negative pairs. PCL clusters samples M times, then regards a representation $\mathbf{z}$ of one image (instance) and its cluster center $\mathbf{c}$ as a positive pair; $\mathbf{z}$ and other cluster centers as negative pairs, formally defined as:

$$\mathcal{L}_{\mathrm{PCL}} = -\sum_{i=1}^{2N}\left(\frac{1}{M}\sum_{m=1}^{M}\log\frac{\exp(\frac{\mathbf{z}_i^{\mathrm{T}}\mathbf{c}_i^m}{\tau^m})}{\sum_{j=0}^{r}\exp(\frac{\mathbf{z}_i^{\mathrm{T}}\mathbf{c}_j^m}{\tau^m})}\right), \quad \text{(S1)}$$

where N is the batch size, r is the number of negative samples, and τ^m is the temperature hyper-parameter.

*Corresponding author

Method	$M = 0.1k$	$M = 0.2k$	$M = 0.5k$
ER	19.4±0.6	20.9±0.9	26.0±1.2
ER with KD	17.0±2.7	17.3±2.1	17.6±0.8

Table S1. Average Accuracy with knowledge distillation [10] (KD) for ER on CIFAR-10. All results are the average of 5 runs.

	ER	SCR	DVC	OCM	**OnPro**
IN-100	9.6±3.5	12.9±2.2	11.7±2.9	16.4±3.6	**18.6**±2.3
IN-1k	5.6±4.5	4.7±0.2	0.1±0.1	5.5±0.1	**6.0**±0.2

Table S2. Average Accuracy on ImageNet-100 ($M = 1k$) and ImageNet-1k ($M = 5k$). All results are the average of 3 runs.

In addition, at each iteration, PCL needs to cluster all samples M times, which is very expensive for training, while OPE only needs to compute online prototypes once.

B. Extra Experimental Results

B.1. More Visual Explanations

To further demonstrate the shortcut learning in online CL, we randomly select several images from all (ten) classes in the training set of CIFAR-10 and provide their visual explanations by GradCAM++ [2], as shown in Fig. S1. The results confirm that shortcut learning is widespread in online CL. Although ER [3] and DVC [5] predict the correct class, they still focus on some oversimplified and object-unrelated features. In contrast, our OnPro learns representative features of classes.

B.2. Knowledge Distillation on ER

As analyzed in the main paper, it is hard to distill useful knowledge due to shortcut learning. To demonstrate this, we apply the knowledge distillation (KD) in [10] to ER, and the results are shown in Table S1. The performance of ER

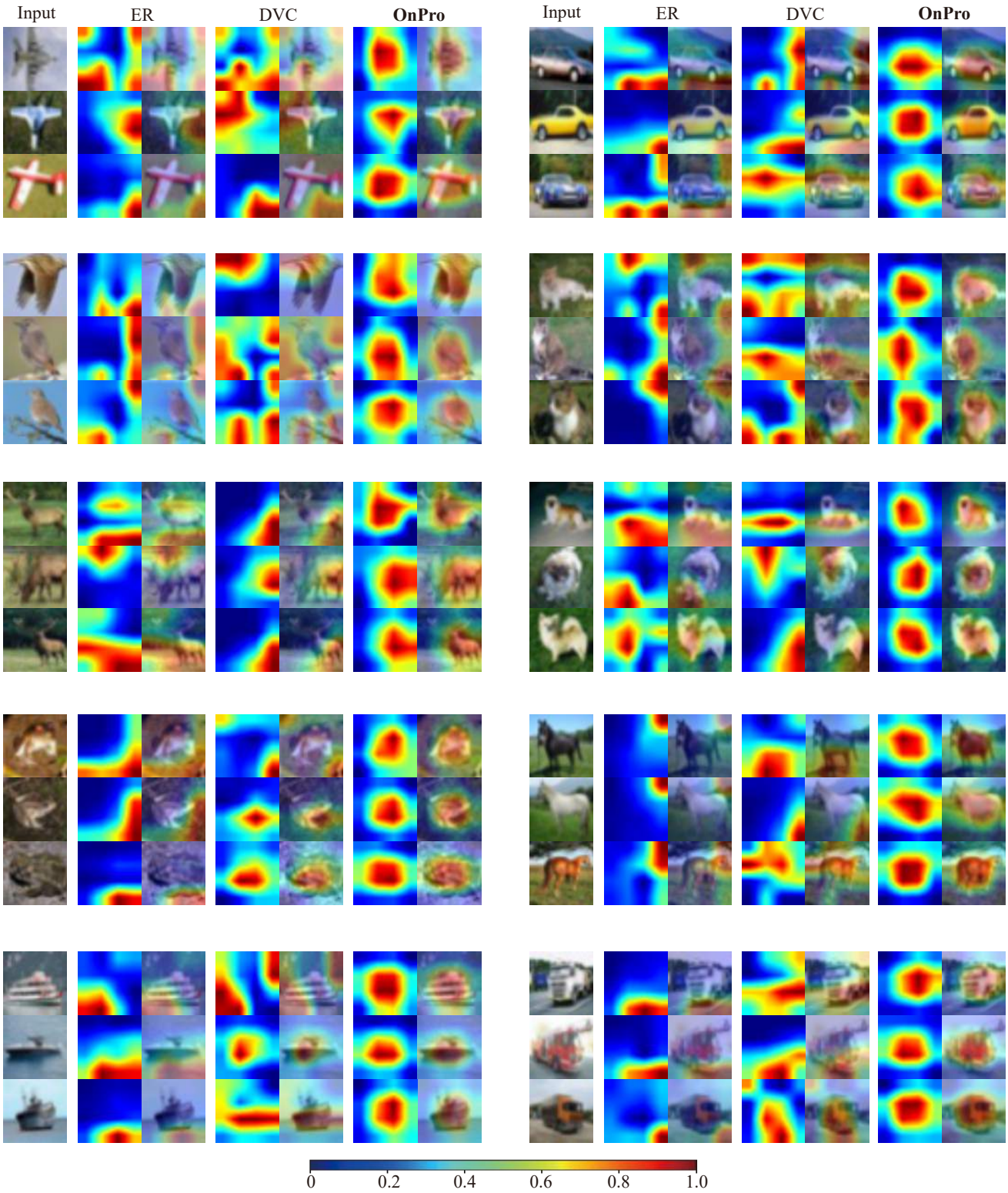

Figure S1. More visual explanations by GradCAM++ on the training set of CIFAR-10 (image size 32×32).

(a) OCM (b) OnPro

Figure S2. t-SNE visualization of all classes in the test set of CIFAR-10 ($M = 0.2k$).

Method	CIFAR-10		CIFAR-100	
	Accuracy ↑	Forgetting ↓	Accuracy ↑	Forgetting ↓
$\mathcal{L}_{\text{CE}}(\text{both})$	48.5±2.2	46.6±2.4	20.4±0.6	41.0±0.6
$\mathcal{L}_{\text{CE}}(\text{sepa})$	53.2±2.1	38.9±2.3	18.8±0.6	48.1±0.8
OnPro (**ours**)	**57.8**±1.1	**23.2**±1.3	**22.7**±0.7	**15.0**±0.8

Table S3. Ablation studies about $\mathcal{L}_{\text{CE}}$ on CIFAR-10 ($M = 0.1k$) and CIFAR-100 ($M = 0.5k$). $\mathcal{L}_{\text{CE}}(\text{both})$ means calculating X and X^{b} in one CE loss, while $\mathcal{L}_{\text{CE}}(\text{sepa})$ is calculating X and X^{b} separately in two CE losses. All results are the average of 15 runs.

decreases after using KD, and a larger memory bank does not result in significant performance gains.

B.3. Experiments on Larger Datasets

We conduct extra experiments on ImageNet-100 and ImageNet-1k. ImageNet-100 is a subset of ImageNet-1k with randomly sampled 100 classes; we follow [7] to use the fixed random seed (1993) for dataset generation. We set the number of tasks to 50, the batch size and the buffer batch size to 10, and the memory bank size to 1k for ImageNet-100 and 5k for ImageNet-1k. For a fair comparison, all methods use the same data augmentations, including resized-crop, horizontal-flip, and gray-scale. The mean Average Accuracy over 3 runs are reported in Table S2, suggesting: (i) on larger datasets, our OnPro still achieves the best performance and is more stable (lower STD); and (ii) the performance on larger datasets varies greatly. For example, on ImageNet-1k, DVC fails, ER is unstable (large STD), and SCR performs even worse than ER.

B.4. Visualization of All Classes

To demonstrate the impact of our OnPro on classification, we provide the visualization of OnPro and OCM for all classes in the test set on CIFAR-10 ($M = 0.2k$), as shown in Fig. S2. It is intuitive that the closer the prototypes of the two classes are, the more confused these two classes become. Obviously, OCM does not avoid class confusion, especially for the three animal classes of `Bird`, `Cat`, and `Dog`, while OnPro achieves clear inter-class dispersion. Furthermore, compared to OCM, OnPro can perceive semantically similar classes and present their relationships in the embedding space. Specifically, for the two classes of `Automobile` and `Truck`, their distributions are adjacent in OnPro because they have more similar semantics compared to other classes. However, OCM cannot capture the semantics relationships, causing the two classes to be relatively far apart. The results suggest that OnPro can achieve an equilibrium status that separates all seen classes well by learning representative and discriminative features with online prototypes.

C. Extra Ablation Studies

C.1. Class Balance on Cross-Entropy Loss

In Table S3, we find that the way to calculate the cross-entropy (CE) loss can significantly affect the performance of OnPro, where $\mathcal{L}_{\text{CE}}(\text{both}) = l(y \cup y^{\text{b}}, \varphi(f(x \cup x^{\text{b}})))$ and $\mathcal{L}_{\text{CE}}(\text{sepa}) = l(y, \varphi(f(x))) + l(y^{\text{b}}, \varphi(f(x^{\text{b}})))$. Here we omit aug for simplicity. Both $\mathcal{L}_{\text{CE}}(\text{both})$ and $\mathcal{L}_{\text{CE}}(\text{sepa})$ degrade the performance because adding the data of new classes will bring serious class imbalance, causing the classifier to easily overfit to new classes and forget previous knowledge.

C.2. Effects of the APF Ratio α

Encouraging the model to have a tendency to focus on confused classes is helpful for mitigating catastrophic forgetting. However, excessive focus on these classes may dis-

α	0	0.10	0.25	0.50	0.75	0.9
CIFAR-10	62.9±2.5	63.2±2.0	**65.5**±1.0	65.4±2.7	64.6±1.8	64.1±2.0
CIFAR-100	22.0±1.5	**22.7**±0.7	22.1±1.1	21.7±1.2	21.3±1.3	21.1±1.1

Table S4. Effects of the APF ratio α on CIFAR-10 ($M = 0.2k$) and CIFAR-100 ($M = 0.5k$). All results are the average of 5 runs.

Method	$M = 0.1k$	$M = 0.2k$	$M = 0.5k$
ER-Rot	30.1±1.9	34.1±3.0	42.8±4.1
ASER-Rot	30.7±3.5	35.8±0.8	43.8±2.1
SCR-Rot	35.8±3.3	46.4±2.4	59.8±2.6
DVC-Rot	45.3±4.3	58.5±2.8	66.7±2.1
OCM	47.5±1.7	59.6±0.4	70.1±1.5
OnPro (**ours**)	**57.8**±1.1	**65.5**±1.0	**72.6**±0.8

Table S5. Average Accuracy using Rotation augmentation (Rot) on CIFAR-10. All results are the average of 5 runs.

rupt the established equilibrium. Therefore, we study the trade-off factor α on CIFAR-10 ($M = 0.2k$) and CIFAR-100 ($M = 0.5k$), and the results are shown in Table S4. On the one hand, when α is too small, the APF reduces to the random selection and takes little account of easily misclassified classes. On the other hand, too large α causes focusing too much on confused classes and ignoring general cases. Based on the experimental results, we set $\alpha = 0.25$ on CIFAR-10 and $\alpha = 0.1$ on CIFAR-100 and TinyImageNet.

C.3. Effects of Rotation Augmentation

As mentioned in the main paper, besides resized-crop, horizontal-flip, and gray-scale, OCM and OnPro use Rotation augmentation (Rot) like [10]. To explore the effects of Rot, we employ it for some SOTA baselines, as shown in Table S5. We find that using Rot can improve the performance of baselines except for SCR. However, they are still inferior to OnPro.

C.4. Effects of Projection Head g

We employ a projection head g to get representations, which is widely-used in contrastive learning [4]. For baselines, SCR [9], DVC [5], and OCM [6] also use a projection head to get representations. To explore the effects of the projector g in OnPro, we conduct the experiment in Table S6. The result shows that projector g can only bring a slight performance improvement, and also illustrates that the performance of OnPro comes mainly from our proposed components.

C.5. Effects of Memory Bank Batch Size m

Fig. S3 shows the effects of memory bank batch size. We can observe that the performance of OnPro improves as the memory bank batch size increases. However, the training time also grows with larger memory bank batch sizes.

Method	$M = 0.1k$	$M = 0.2k$	$M = 0.5k$
no Projector	56.1±4.7	63.3±1.9	71.0±1.5
OnPro (**ours**)	**57.8**±1.1	**65.5**±1.0	**72.6**±0.8

Table S6. Average Accuracy without projector g on CIFAR-10. All results are the average of 5 runs.

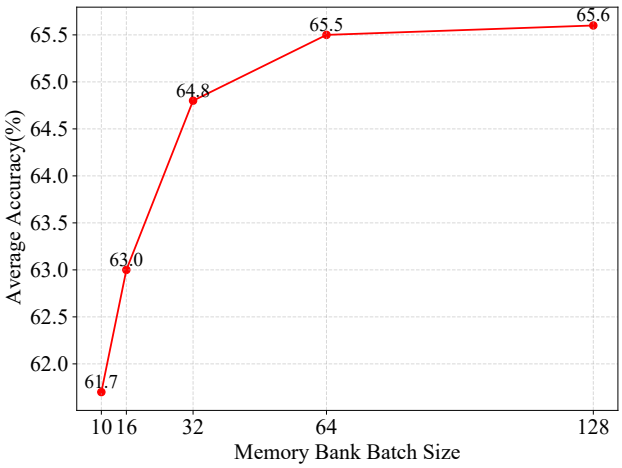

Figure S3. The performance of OnPro on CIFAR-10 ($M = 0.2k$) with different memory bank batch sizes.

Following [6], we set the memory bank batch size to 64.

D. Training Algorithms of OnPro and APF

The training procedures of the proposed OnPro and APF are presented in Algorithms 1 and 2, respectively. The source code will be made publicly available upon the acceptance of this work.

E. Implementation Details about Baselines

The hyperparameters of OnPro are given in the main paper. Here we discuss in detail how each baseline is implemented.

For all baselines, we follow their original paper and default settings to set the hyperparameters. We set the random seed to 0 and run the experiment 15 times in the same program to get the results.

For iCaRL, AGEM, and ER, we use the SGD optimizer and set the learning rate to 0.1. We uniformly randomly select samples to update the memory bank and replay.

For DER++, we use the SGD optimizer and set the learning rate to 0.03. We fix α to 0.1 and β to 0.5.

Algorithm 1: Training Algorithm of OnPro

Input: Data stream $\mathfrak{D}$, encoder f, projector g, classifier φ, and data augmentation aug.
Initialization: Memory bank $\mathcal{M} \leftarrow \{\}$,
for t=*1 to* T **do**
  for *each mini-batch* X *in* $\mathcal{D}_t$ **do**
    $X^{\mathrm{b}} \leftarrow \mathrm{APF}\ (\mathcal{M})$
    $\widehat{X}, \widehat{X}^{\mathrm{b}} \leftarrow \mathrm{aug}(X, X^{\mathrm{b}})$
    $\mathbf{z}, \mathbf{z}^{\mathrm{b}} = g(f(X \cup \widehat{X})), g(f(X^{\mathrm{b}} \cup \widehat{X}^{\mathrm{b}}))$
    Compute online prototypes $\mathcal{P}$ and $\mathcal{P}^{\mathrm{b}}$ ▷ Eq. (2) in the main paper
    $\mathcal{L}_{\mathrm{OnPro}} \leftarrow \mathcal{L}_{\mathrm{OPE}}(\mathcal{P}, \mathcal{P}^{\mathrm{b}}) + \mathcal{L}_{\mathrm{INS}}(\mathbf{z}, \mathbf{z}^{\mathrm{b}}) + \mathcal{L}_{\mathrm{CE}}(\varphi(f(\widehat{X}^{\mathrm{b}})))$
    $\theta_f, \theta_g \leftarrow \mathcal{L}_{\mathrm{OnPro}}$
    $\mathcal{M} \leftarrow \mathrm{Update}(\mathcal{M}, X)$
  end
end

Algorithm 2: Algorithm of APF

Input: $\mathcal{M}$, and online prototypes $\{\mathbf{p}_i^{\mathrm{b}}\}_{i=1}^{K^{\mathrm{b}}}$ of previous time step.
Output: X^{b}
Initialization: $\mathcal{S} \leftarrow \{\}$, $n_{\mathrm{APF}} = \alpha \cdot m$,
$P \leftarrow$Compute probability $P_{i,j}$ for each class pair using $\mathbf{p}_i^{\mathrm{b}}$ and $\mathbf{p}_j^{\mathrm{b}}$ ▷ Eq. (6) in the main paper
for *each* $P_{i,j}$ *in* P **do**
  $X_i, X_j \leftarrow$ sample $\lfloor P_{i,j} \cdot n_{\mathrm{APF}} + 0.5 \rfloor$ images from class i and class j
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathrm{Mixup}(X_i, X_j)$
end
$X_{\mathrm{base}} \leftarrow$ the remaining $m - n_{\mathrm{APF}}$ samples are uniformly randomly selected from $\mathcal{M}$
$X^{\mathrm{b}} \leftarrow \mathcal{S} \cup \mathrm{Mixup}(X_{\mathrm{base}}, X_{\mathrm{base}})$

For PASS, we use the Adam optimizer and set the learning rate to 0.001. The weight decay is set to 2e-4. We set the loss weights λ and γ to 10 and fix the temperature as 0.1.

For GSS, we use the SGD optimizer and set the learning rate to 0.1. The number of batches randomly sampled from the memory bank to estimate the maximal gradients cosine similarity score is set to 64, and the random sampling batch size for calculating the score is also set to 64.

For MIR, we use the SGD optimizer and set the learning rate to 0.1. The number of subsamples is set as 100.

For GDumb, we use the SGD optimizer and set the learning rate to 0.1. The value for gradient clipping is set to 10. The minimal learning rate is set to 0.0005, and the epochs to train for the memory bank are 70.

For ASER, we use the SGD optimizer and set the learning rate to 0.1. The number of nearest neighbors to perform ASER is set to 3. We use mean values of Adversarial SV and Cooperative SV, and set the maximum number of samples per class for random sampling to 1.5. We use the SV-based methods for memory update and retrieval as given in the original paper.

For SCR, we use the SGD optimizer and set the learning rate to 0.1. We set the temperature to 0.07 and employ a linear layer with a hidden size of 128 as the projection head.

For CoPE, we use the SGD optimizer and set the learning rate to 0.001. We set the temperature to 1. The momentum of the moving average updates for the prototypes is set to 0.99. We use dynamic buffer allocation instead of a fixed class-based memory as given in the original paper.

For DVC, we use the SGD optimizer and set the learning rate to 0.1. The number of candidate samples for retrieval is set to 50. For CIFAR-100 and TinyImageNet, we set loss weights $\lambda_1 = \lambda_2 = 1$, $\lambda_3 = 4$. For CIFAR-10, $\lambda_1 = \lambda_2 = 1$, $\lambda_3 = 2$.

For OCM, we use the Adam optimizer and set the learning rate to 0.001. The weight decay is set as 0.0001. We set the temperature to 0.07 and employ a linear layer with a hidden size of 128 as the projection head. λ is set to 0.5. We set α to 1 and β to 2 for contrastive loss and set α to 0 and β to 2 for supervised contrastive loss as given in the original paper of OCM.

We refer to the links in Table S7 to reproduce the results.

Baseline	Link
iCaRL	https://github.com/srebuffi/iCaRL
DER++	https://github.com/aimagelab/mammoth
PASS	https://github.com/Impression2805/CVPR21_PASS
AGEM	https://github.com/facebookresearch/agem
GSS	https://github.com/rahafaljundi/Gradient-based-Sample-Selection
MIR	https://github.com/optimass/Maximally_Interfered_Retrieval
GDumb	https://github.com/drimpossible/GDumb
ASER and SCR	https://github.com/RaptorMai/online-continual-learning
CoPE	https://github.com/Mattdl/ContinualPrototypeEvolution
ER and DVC	https://github.com/YananGu/DVC
OCM	https://github.com/gydpku/OCM

Table S7. Baselines with source code links.

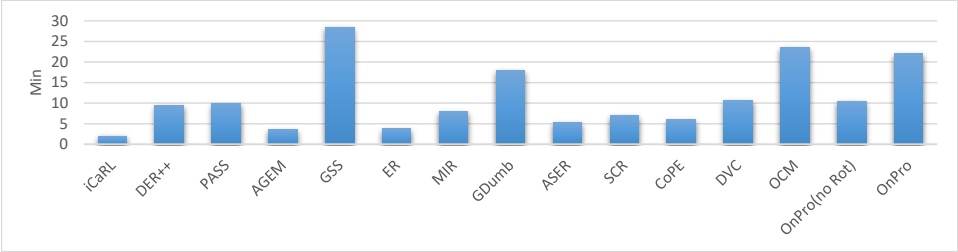

Figure S4. Training time of each method on CIFAR-10.

F. Execution Time

Fig. S4 shows the training time of all methods on CIFAR-10. OnPro is faster than OCM [6] and GSS [1]. We find that rotation augmentation (Rot) is the main reason for the increase in training time. When rotation augmentation is not used, the training time of OnPro is significantly reduced and is close to most of the baselines. Furthermore, OnPro achieves the best performance compared to all baselines.

References

[1] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 6

[2] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847, 2018. 1

[3] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv:1902.10486*, 2019. 1

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607, 2020. 4

[5] Yanan Gu, Xu Yang, Kun Wei, and Cheng Deng. Not just selection, but exploration: Online class-incremental continual learning via dual view consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7442–7451, 2022. 1, 4

[6] Yiduo Guo, Bing Liu, and Dongyan Zhao. Online continual learning through mutual information maximization. In *International Conference on Machine Learning*, pages 8109–8126, 2022. 4, 6

[7] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019. 3

[8] Junnan Li, Pan Zhou, Caiming Xiong, and Steven C. H. Hoi. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2021. 1

[9] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3589–3599, 2021. 4

[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*, pages 5871–5880, 2021. 1, 4