Improving Plasticity in Online Continual Learning via Collaborative Learning

Supplementary Material

8. Extra Experiments

Impact of the number of augmentation stages in DC. As mentioned in Sec. 4, DC comprises three augmentation stages, including one geometric distortion stage and two RandAugment stages. In this ablation study, we aim to investigate how more or less augmentation stages will impact the final performance. In this experiment, we generalize the number of augmentation stages in CCL-DC from 0 (which is equivalent to CCL without DC) to N. For $N \geq 1$, we apply one geometric distortion stage and $N-1$ RandAugment stages. As shown in Fig. 6, CCL-DC performs better when the number of augmentation stages increases. However, the training time and memory footprint also increase with more augmentation stages. Thus, for a trade-off, we set the number of stages to 3 in the main paper.

T-SNE visualization. Another advantage of CCL-DC is its ability to enhance the feature discrimination of continual learners. Fig. 10 illustrates the t-SNE visualization of the memory data's embedding space at the end of the training. We can see that the feature representation of the method with CCL-DC is more discriminative compared with the baseline.

Classification loss curve on other baselines. In Sec. 6, we present the classification loss curve of the model during training and illustrate how CCL-DC can assist the model in descending deeper into the loss landscape. We present the classification curve for the remaining baselines in Fig. 11. With improved plasticity, for every baseline method, CCL-DC can improve the training by descending deeper at the end of each task.

Independent network performance. Although the ensemble method gives extra performance at inference time, by averaging the logit output of two networks in CCL-DC, it also doubles the computation. In some cases, computational efficiency becomes more crucial during inference. Continual learners trained with CCL-DC are also able to do inference independently, albeit with a slight performance drop compared with ensemble inference. Table 9 shows the accuracy achieved through independent inference. It is evident that the performance loss in independent inference, when compared to ensemble inference, is minimal (approximately 1%).

Performance with NCM classifier. Besides t-SNE, we can evaluate the feature discrimination using the clustering methods. We remove the final FC classifier and use Nearest-Class-Mean (NCM) classifier with intermediate representations. Table 11 demonstrates that CCL-DC can greatly enhance the NCM accuracy, which evidences the capability of CCL-DC in improving feature discrimination.

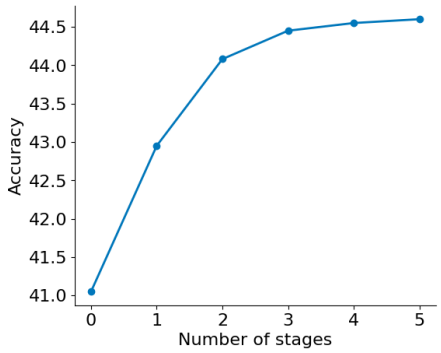


Figure 6. The performance of ER incorporating CCL-DC with varying numbers of augmentation stages on CIFAR-100 (M =2k). All numbers are averaged over 10 runs.

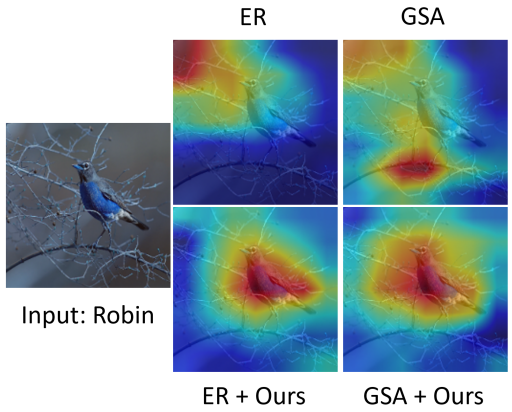


Figure 7. GradCAM++ visualization on the training set of ImageNet-100 (M =5k). Shortcut learning exists in the baseline methods despite making correct predictions.

GradCAM++ visualization. Shortcut learning is another commonly observed issue that hinders the generalization capability of continual learners [44]. In Fig. 7, we use GradCAM++ on the training set of ImageNet-100 (M =5k) at the end of the training of ER and GSA. Although both ER and GSA make correct predictions, we observed that they focus on irrelevant objects, which indicates a tendency toward shortcut learning. Also, we can see that by integrating CCL-DC, the shortcut learning can be greatly alleviated.

Dataset	CIFAR10		CIFAR100			Tiny-ImageNet			ImageNet-100
Memory Size M	500	1000	1000	2000	5000	2000	5000	10000	5000
ER + Ours (Ind.)	65.66±2.35	73.37±1.70	32.97±1.06	43.58±1.05	52.96±1.16	16.32±1.58	28.68±1.20	37.14±0.93	41.82±1.54
ER + Ours (Ens.)	**66.43±2.48**	**74.10±1.71**	**33.43±1.06**	**44.45±1.04**	**53.81±1.16**	**16.56±1.63**	**29.39±1.23**	**37.73±0.85**	**43.11±1.49**
DER++ + Ours (Ind.)	68.15±1.40	73.56±1.12	33.81±0.90	42.79±1.38	52.04±0.81	**11.11±1.53**	21.47±1.93	27.37±2.64	44.22±1.25
DER++ + Ours (Ens.)	**68.79±1.42**	**74.25±1.10**	**34.36±0.89**	**43.52±1.35**	**52.95±0.86**	10.99±1.39	**21.68±1.94**	**28.01±2.46**	**45.70±1.32**
ER-ACE + Ours (Ind.)	69.35±1.24	74.86±1.06	36.34±1.08	44.15±1.05	52.94±0.44	17.99±1.56	25.69±2.00	31.69±1.69	43.92±1.71
ER-ACE + Ours (Ens.)	**70.08±1.38**	**75.56±1.14**	**37.20±1.15**	**45.14±1.00**	**53.92±0.48**	**18.32±1.49**	**26.22±2.01**	**32.23±1.70**	**45.15±1.94**
OCM + Ours (Ind.)	73.00±0.88	76.66±1.38	34.02±1.22	42.39±1.36	50.19±1.36	22.53±1.28	32.16±0.96	38.02±0.94	41.71±1.07
OCM + Ours (Ens.)	**74.14±0.85**	**77.66±1.46**	**35.00±1.15**	**43.34±1.51**	**51.43±1.37**	**23.36±1.18**	**33.17±0.97**	**39.25±0.88**	**43.19±0.98**
GSA + Ours (Ind.)	68.10±1.58	74.78±1.27	35.14±1.40	43.84±1.34	54.29±1.10	16.53±1.62	27.57±1.61	36.12±1.59	43.27±1.05
GSA + Ours (Ens.)	**68.91±1.68**	**75.78±1.16**	**35.56±1.39**	**44.74±1.32**	**55.39±1.09**	**16.70±1.66**	**28.11±1.70**	**37.13±1.75**	**44.28±1.16**
OnPro + Ours (Ind.)	73.65±2.16	77.84±1.33	34.20±1.12	41.18±0.83	49.18±0.81	21.22±1.05	31.13±0.71	37.30±0.93	46.84±1.33
OnPro + Ours (Ens.)	**74.49±2.14**	**78.64±1.42**	**34.76±1.12**	**41.89±0.82**	**50.01±0.85**	**21.81±1.02**	**32.00±0.72**	**38.18±1.02**	**47.93±1.26**

Table 9. Comparison of the final average accuracy achieved through independent inference and the use of the ensemble method on four benchmark datasets with difference memory buffer size M. All values are averages of 10 runs.

Dataset	CIFAR10		CIFAR100			Tiny-ImageNet			ImageNet-100
Memory Size M	500	1000	1000	2000	5000	2000	5000	10000	5000
ER	33.16±3.50	20.94±6.79	**32.65±1.78**	**22.20±2.26**	**13.29±1.98**	**58.38±1.69**	**46.87±1.60**	**40.77±2.45**	**23.38±2.10**
ER + Ours	**30.22±3.75**	**19.85±2.55**	43.28±1.67	29.35±1.50	16.88±1.99	69.56±1.54	53.13±0.85	42.63±0.80	28.48±1.50
DER++	**24.21±2.75**	**18.42±1.84**	**34.49±4.39**	**25.55±3.26**	20.01±2.88	**62.03±2.83**	**51.57±4.60**	**49.51±3.04**	28.77±4.10
DER++ + Ours	25.08±2.88	18.47±3.12	42.76±1.31	31.13±2.41	**18.45±2.89**	72.59±1.29	57.71±1.80	50.31±2.34	**27.22±2.17**
ER-ACE	**12.72±3.56**	**10.66±2.48**	**12.67±1.62**	**9.11±0.78**	**5.92±1.09**	**19.12±0.63**	**17.14±0.67**	15.59±1.24	**14.11±1.19**
ER-ACE + Ours	22.86±2.23	16.07±2.38	35.85±1.12	25.84±1.96	14.21±0.85	24.10±2.00	19.14±1.91	**15.14±1.60**	26.02±2.33
OCM	13.68±4.25	11.63±2.62	**14.99±1.55**	**9.16±1.75**	**3.76±1.16**	**26.12±1.63**	**19.74±1.30**	15.92±1.47	**3.25±0.90**
OCM + Ours	**11.59±2.24**	**9.18±2.03**	16.69±2.36	10.07±1.37	3.99±0.78	26.16±1.90	19.99±1.96	**15.56±1.06**	8.91±1.18
GSA	**25.45±2.86**	**16.42±3.59**	**33.97±2.55**	**22.74±1.83**	**12.31±2.35**	**27.23±2.01**	**23.61±2.26**	**20.58±2.09**	**24.53±1.59**
GSA + Ours	28.47±3.08	19.00±2.08	42.41±1.44	31.09±1.86	16.77±1.87	64.86±1.19	48.23±1.28	35.79±1.46	32.77±2.07
OnPro	**17.94±3.69**	**14.20±2.60**	**16.76±2.47**	**12.42±1.39**	**6.72±0.94**	**28.01±1.59**	23.52±1.75	20.32±1.70	**7.59±1.17**
OnPro + Ours	19.89±4.01	14.62±2.75	28.93±2.19	20.23±1.03	10.55±1.89	28.21±1.58	**20.86±1.13**	**16.17±0.63**	9.90±1.93

Table 10. Forgetting Measure (%, lower is better) on four benchmark datasets with difference memory buffer size M, with and without our proposed CCL-DC scheme. The result of our method is given by the ensemble of two peer models. All values are averages of 10 runs.

Method	NCM Acc. ↑	Logit Acc. ↑
ER	36.56±0.60	31.89±1.45
ER + Ours	44.76±0.55	44.45±1.04
ER-ACE	34.91±1.02	34.21±1.53
ER-ACE + Ours	45.62±1.04	45.14±1.00
OnPro	34.32±0.95	33.33±0.93
OnPro + Ours	42.82±0.67	41.89±0.82

Table 11. Final average accuracy on CIFAR-100 (M =2k), with and without NCM classifier.

9. Counterintuitive performance of ER-ACE

As shown in Table 4, ER-ACE suffers from counterintuitive performance on plasticity, especially when the task number is large (*e.g.*, TinyImageNet experiments). This is because ER-ACE employs Asymmetric Cross-Entropy loss (ACE) during the training of batch images. ACE manually masks out the old classes for batch image training, which reduces the feature drift of old classes and enhances ER-ACE's stability, as stated in the original paper. However, ACE cuts the gradient for old classes in classification loss, which limits the optimizer's maneuverability in the final classification layer. This loss of maneuverability is significant when there are many tasks involved, and thus we may observe the LA close to 0 in the later stages of training. Despite the low plasticity, ER-ACE has a good overall performance because: (1) Memory replay partially compensates for this loss in terms of plasticity (Learner can still learn from samples in the memory buffer), and (2) ER-ACE has higher stability. Additionally, we witness a major stability drop in ER-ACE when incorporating CCL-DC. As indicated in the Algorithm 1, although we also use ACE in the classification loss of DC, we do not perform masking in the distillation loss. The distillation between probability distributions of

peer models retrieves some plasticity, but it also leads to extra feature drift, which hurts the stability to some extent.

10. Experiment Details

Dataset As mentioned in Sec. 5, we use four datasets to evaluate the effectiveness of our method. The original datasets are split into several tasks of disjoint classes. The detailed information about dataset split and task allocation is as follows:

CIFAR-10 [25] has 10 classes with 50,000 training samples and 10,000 test samples. Images are sized at 32×32. In our experiments, it is split into five non-overlapping tasks with two classes per task.

CIFAR-100 [25] has 100 classes with 50,000 training samples and 10,000 test samples. The images are 32×32 in size. It is split into 10 disjoint tasks with 10 classes per task.

TinyImageNet [26] has 200 classes with 100,000 training samples and 10,000 test samples. Images are sized at 64×64. It is split into 100 non-overlapping tasks with two classes per task.

ImageNet-100 [22] is the subset of ImageNet-1k [15] containing 100 classes. We follow [11] for the class selection. The images are 224×224 in size. It is split into 10 disjoint tasks with 10 classes per task.

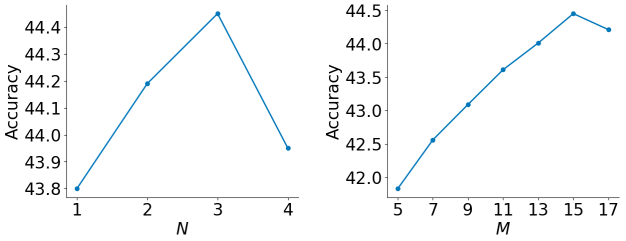
Task Sequence. In online CL, some work uses a fixed task sequence throughout all runs to evaluate the performance, for the sake of fair comparison. However, we found that the evaluation heavily depends on the task order. For fair comparison, we randomize the allocation of classes to tasks and the sequence of tasks using 10 fixed random seeds (for 10 runs in our experiments). This ensures our evaluation result is not biased to task difficulty.

Data augmentation for baseline methods. Data augmentation has been demonstrated to be successful in improving the performance of online CL [6, 19, 35]. However, methods benefit differently from different augmentation intensities, and some methods may gain more performance with simple augmentations instead of complicated ones. Thus, to achieve optimal performance for comparison, we involve two different augmentation strategies for baseline methods:

1. Partial augmentation strategy. The partial augmentation is a strategy with weak augmentation. It comprises random cropping with $p = 0.5$, followed by random horizontal flip with $p = 0.5$.

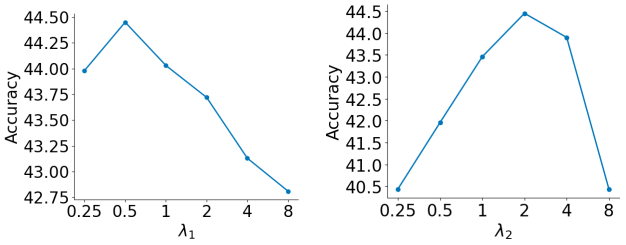
2. Full augmentation strategy. The full strategy is a superset of the partial strategy. It consists of random cropping, horizontal flipping, color jitter, and random grayscale. The parameter of color jitter is set to (0.4, 0.4, 0.4, 0.1) with $p = 0.8$, while the probability of random grayscale is 0.2.

For fair comparison, the models trained with CCL-DC also employ the same data augmentation strategy in the baseline loss part, as illustrated in Algorithm 1.



(a) Performance when $M = 15$ (b) Performance when $N = 3$

Figure 8. The impact of N and M in RandAugment on the performance for ER + *Ours* on CIFAR-100 (M =2k). As shown in the figure, the best performance is achieved with $N = 3$ and $M = 15$. All numbers are averaged over 10 runs.



(a) Performance when $\lambda_2 = 2$ (b) Performance when $\lambda_1 = 0.5$

Figure 9. The impact of λ_1 and λ_2 on the performance for ER + *Ours* on CIFAR-100 (M =2k). As shown in the figure, the best performance is achieved with $\lambda_1 = 0.5$ and $\lambda_2 = 2$. All numbers are averaged over 10 runs.

Hyperparameter search for baselines. For hyperparameters in the baseline methods, as indicated in Sec. 5, we perform a hyperparameter search on CIFAR-100 (M =2k) for the baseline methods. Table 12 shows the exhaustive list of the grid search. Note that we used the hyperparameters from the original OCM paper to reduce the hyperparameter search space due to computational constraints. For fair comparison, after finding the optimal hyperparameters for the baseline methods, we apply the same hyperparameters when incorporating CCL-DC.

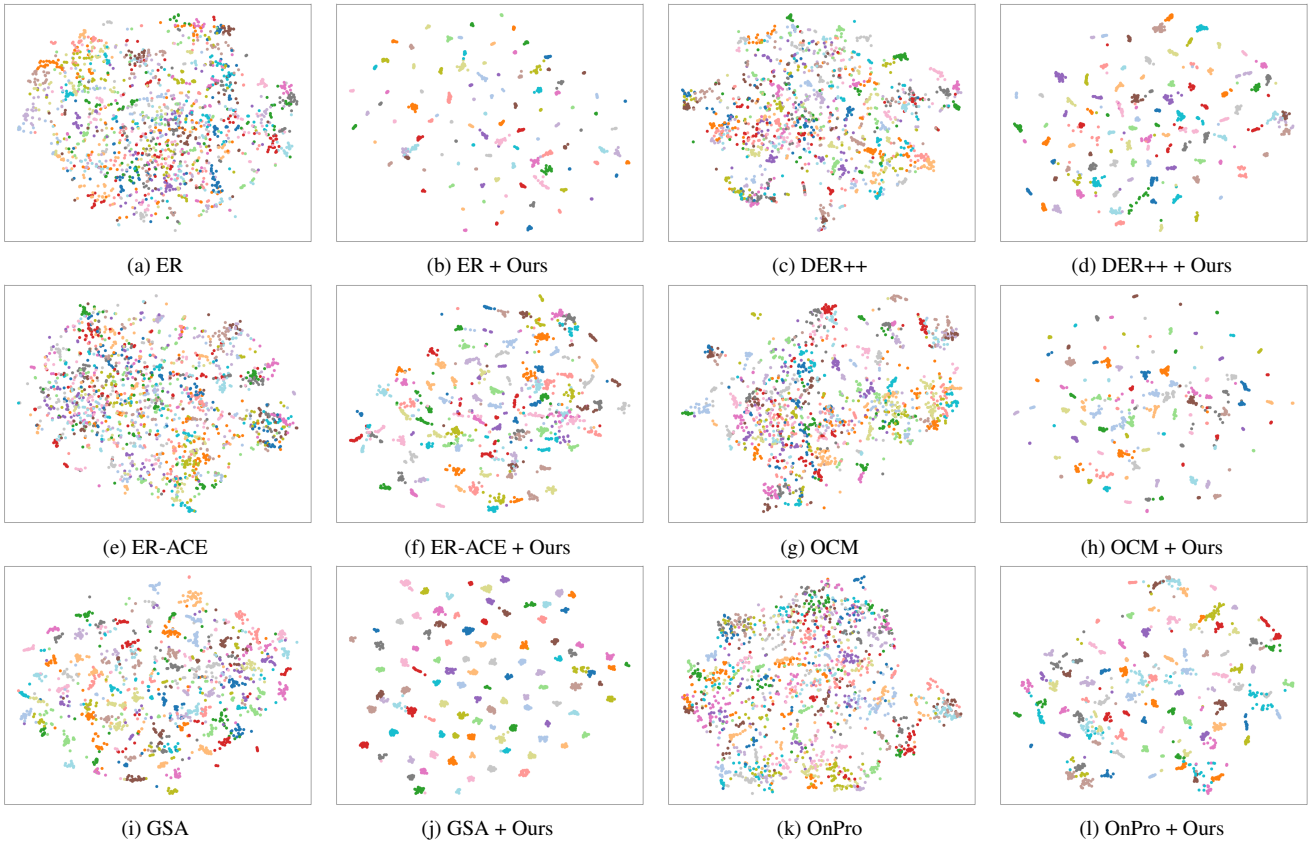
Hyperparameter search for CCL-DC. CCL-DC also has four unique hyperparameters, including N, M in RandAugment of DC, and λ_1, λ_2 in Eq. 9 and Eq. 10.

In CCL-DC, we use RandAugment to generate samples with different difficulty. Thus, two additional hyperparameters in RandAugment (N and M) are involved in CCL-DC. Since the transformation intensity (N and M) is highly related to the dataset instead of the baseline method,

We conduct a hyperparameter search for every dataset with ER + CCL-DC and apply the same hyperparameter across all baseline methods when incorporating CCL-DC. We searched in 4 settings of N and 7 settings of M (*i.e.*, $N = \{1, 2, 3, 4\}$ and $M = \{5, 7, 9, 11, 13, 15, 17\}$). With our grid search, we find that ($N = 3, M = 15$) is the best for CIFAR-10 and CIFAR-100. ($N = 1, M = 11$) achieves the best results on Tiny-ImageNet and ($N = 3, M = 11$) is the best for ImageNet-100. We also visualize some of the experimental results on CIFAR-100, as shown in Fig. 8.

CCL-DC also comprises two hyperparameters λ_1 and λ_2 in Eq. 9 and Eq. 10. Similar to the hyperparameter search strategy we do for baseline hyperparameters, for each baseline method with CCL-DC, we initiate another hyperparameter search for λ_1 and λ_2 on CIFAR-100 (M =2k) and apply the hyperparameter to all of the settings. We searched from $\lambda_1, \lambda_2 = \{0.25, 0.5, 1, 2, 4, 8\}$. We visualize some experimental results in Fig 9.

Hardware and Computation. All the experiments in our work are conducted on NVIDIA A100 GPUs. Fig. 12 shows the training time for each method with and without CCL-DC on CIFAR-100 (M =2k).



(a) ER (b) ER + Ours (c) DER++ (d) DER++ + Ours

(e) ER-ACE (f) ER-ACE + Ours (g) OCM (h) OCM + Ours

(i) GSA (j) GSA + Ours (k) OnPro (l) OnPro + Ours

Figure 10. T-SNE visualization of memory data at the end of training on CIFAR-100 (M =2k).

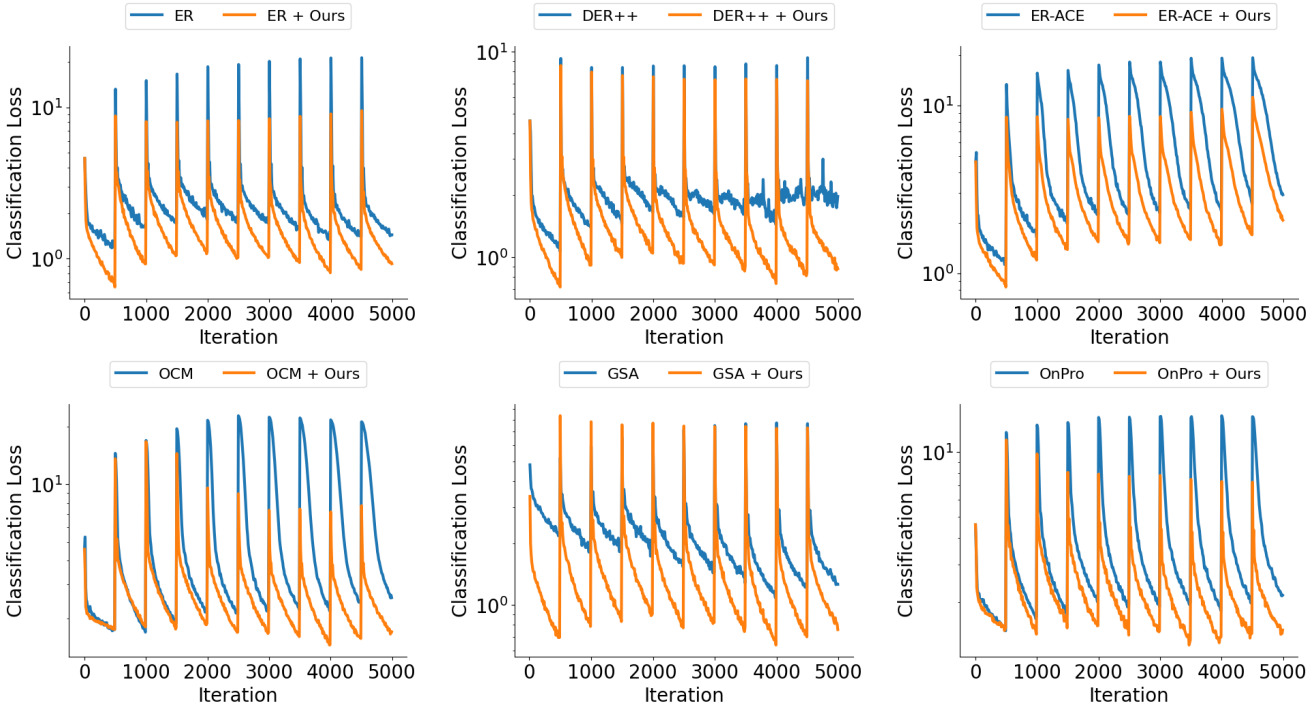


Figure 11. Classification loss curve on CIFAR-100 (M =2k). The curve is calculated on all training samples of the *current* task. Since there are 10 tasks in total, the curve has 10 peaks.

Method	HP	Values
ER	optimizer	[SGD, AdamW]
	lr	[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]
	weight decay	[0, 1e-4]
	momentum (for SGD)	[0, 0.9]
	aug. strat.	[partial, full]
DER++	optimizer	[SGD, AdamW]
	lr	[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]
	weight decay	[0, 1e-4]
	momentum (for SGD)	[0, 0.9]
	aug. strat.	[partial, full]
	alpha	[0.1, 0.2, 0.5, 1.0]
	beta	[0.5, 1.0]
ER-ACE	optimizer	[SGD, AdamW]
	lr	[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]
	weight decay	[0, 1e-4]
	momentum (for SGD)	[0, 0.9]
	aug. strat.	[partial, full]
OCM	optimizer	[AdamW]
	lr	[0.001]
	weight decay	[1e-4]
	aug. strat.	[partial, full]
GSA	optimizer	[SGD, AdamW]
	lr	[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]
	weight decay	[0, 1e-4]
	momentum (for SGD)	[0, 0.9]
	aug. strat.	[partial, full]
OnPro	optimizer	[SGD, AdamW]
	lr	[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]
	weight decay	[0, 1e-4]
	momentum (for SGD)	[0, 0.9]
	aug. strat.	[partial, full]

Table 12. Exhaustive list of hyperparameters searched on CIFAR-100 (M =2k).

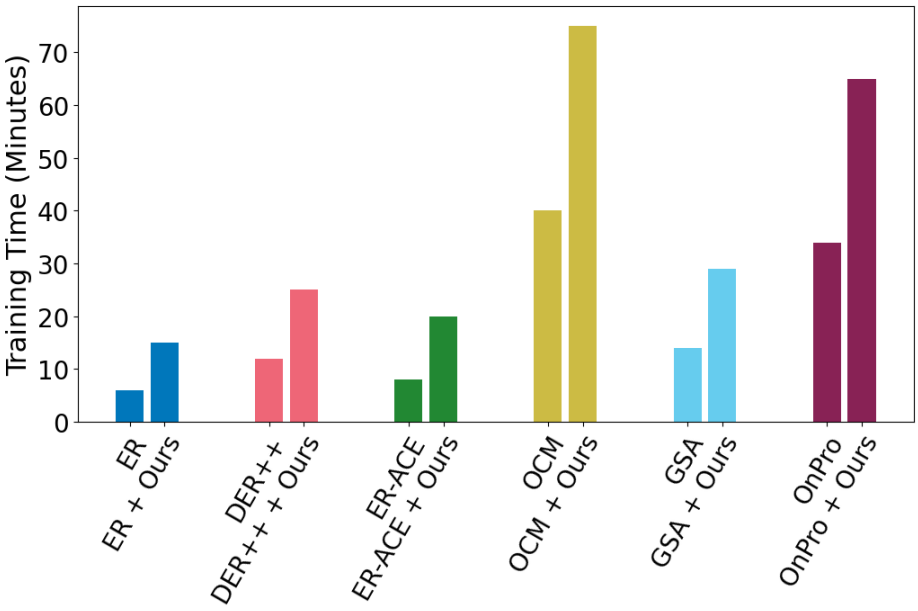


Figure 12. Training time of each method on CIFAR-100 (M =2k).