Supplementary Material
CBA: Improving Online Continual Learning via Continual Bias Adaptor

Quanziang Wang[1] Renzhen Wang[1*] Yichen Wu[2] Xixi Jia[3] Deyu Meng[1,4*]

[1] Xi'an Jiaotong University [2] City University of Hong Kong [3] Xidian University

[4] Macau University of Science and Technology

quanziangwang@gmail.com, {rzwang, dymeng}@xjtu.edu.cn

A. Details of the Bi-level Optimization

In this section, we first review the proposed bi-level learning framework to optimize the parameters of the classifier network and the Continual Bias Adaptor (CBA) module. Formally, the bi-level optimization problem can be formulated as

$$\begin{aligned} \min_{\omega} \quad & \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta(\omega)}\right) \\ \text{s.t.} \quad & \theta(\omega) = \arg\min_{\theta} \mathcal{L}^{trn}\left(\mathcal{B}^{trn}; \mathcal{F}_{\theta,\omega}\right). \end{aligned} \tag{8}$$

We use a nested gradient-optimization-based method to update the classifier network parameter θ and the CBA parameter ω. Specifically, in the inner loop, the updating formulation of θ at iteration step k can be expressed as

$$\theta^{k+1}(\omega) = \theta^{k} - \alpha \cdot \nabla_{\theta} \mathcal{L}^{trn}\left(\mathcal{B}^{trn}; \mathcal{F}_{\theta^k,\omega^k}\right), \tag{9}$$

where $\alpha > 0$ is the learning rate of the inner loop.

Then we present details of the updating formulation of the CBA parameter ω in the outer loop. The updating of ω can be represented as

$$\omega^{k+1} = \omega^{k} - \beta \cdot \nabla_{\omega} \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta^{k+1}(\omega)}\right), \tag{10}$$

where $\beta > 0$ is the learning rate. In Eq. (10), the derivation term can be represented as

$$\begin{aligned} & \nabla_{\omega} \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta^{k+1}(\omega)}\right) \\ =& \frac{\partial \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta^{k+1}(\omega)}\right)}{\partial \theta^{k+1}(\omega)} \frac{\partial \theta^{k+1}(\omega)}{\partial \omega} \\ =& \frac{\partial \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta^{k+1}(\omega)}\right)}{\partial \theta^{k+1}(\omega)} \frac{\partial}{\partial \omega}\left(-\alpha \frac{\partial \mathcal{L}^{trn}\left(\mathcal{B}^{trn}; \mathcal{F}_{\theta^k,\omega^k}\right)}{\partial \theta}\right) \\ =& -\alpha \frac{\partial \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta^{k+1}(\omega)}\right)}{\partial \theta^{k+1}(\omega)} \frac{\partial^2 \mathcal{L}^{trn}\left(\mathcal{B}^{trn}; \mathcal{F}_{\theta^k,\omega^k}\right)}{\partial \omega \partial \theta}. \end{aligned} \tag{11}$$

Obviously, the update of ω introduces a second-order derivation that can be easily implemented by PyTorch [31].

*Corresponding author

To alleviate the calculation burden of this second-order derivation in Eq. (11), we assume that this derivation only depends on the parameter of the last linear classification layer rather than the whole classifier network like ResNet-18. The linear classification layer only introduces a small number of parameters which can significantly speed up the computation and makes our bi-level optimization efficient and suitable for online CL. The detailed experiments are shown in Appendix D.4.

B. Proof of Theorem 1

Proof. The bi-level learning framework is represented as Eq. (8). If the inner optimization problem is approximated by one-step gradient descent by Eq. (9), the $(k+1)$th iteration becomes

$$\begin{aligned} \min_{\omega} \quad & \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta^{k+1}(\omega)}\right) \\ \text{s.t.} \quad & \theta^{k+1}(\omega) = \theta^{k}(\omega) - \alpha \frac{\partial \mathcal{L}^{trn}\left(\mathcal{B}^{trn}; \mathcal{F}_{\theta^k,\omega}\right)}{\partial \theta^{k}}. \end{aligned} \tag{12}$$

For simplicity in this proof, we omit the dataset of each loss function, *i.e.*, the training loss of the inner and outer loop can be reformulated as $\mathcal{L}^{buf}\left(f_{\theta(\omega)}\right)$ and $\mathcal{L}^{trn}\left(\mathcal{F}_{\theta,\omega}\right)$, respectively.

Assume that the outer-loop training loss function $\mathcal{L}^{buf}\left(f_{\theta(\omega)}\right)$ is η gradient Lipschitz w.r.t θ, then we have

$$\begin{aligned} & \mathcal{L}^{buf}\left(f_{\theta^{k+1}(\omega)}\right) \\ \leq & \mathcal{L}^{buf}\left(f_{\theta^{k}(\omega)}\right) + \left\langle \frac{\partial \mathcal{L}^{buf}\left(f_{\theta^{k}(\omega)}\right)}{\partial \theta^{k}}, \theta^{k+1}(\omega) - \theta^{k}(\omega) \right\rangle \\ & + \frac{\eta}{2}\left\| \theta^{k+1}(\omega) - \theta^{k}(\omega) \right\|^2 \\ = & \mathcal{L}^{buf}\left(f_{\theta^{k}(\omega)}\right) + \left\langle \frac{\partial \mathcal{L}^{buf}\left(f_{\theta^{k}(\omega)}\right)}{\partial \theta^{k}}, -\alpha \frac{\partial \mathcal{L}^{trn}\left(\mathcal{F}_{\theta^k,\omega}\right)}{\partial \theta^{k}} \right\rangle \\ & + \frac{\eta}{2}\left\| \alpha \frac{\partial \mathcal{L}^{trn}\left(\mathcal{F}_{\theta^k,\omega}\right)}{\partial \theta^{k}} \right\|^2. \end{aligned} \tag{13}$$

The update of ω by SGD as

$$\omega^{k+1} = \omega^k - \beta \frac{\partial \mathcal{L}^{buf}\left(f_{\theta(\omega)}\right)}{\partial \omega^k},$$

which aims to decrease the objective $\mathcal{L}^{buf}\left(f_{\theta(\omega)}\right)$ such that $\mathcal{L}^{buf}\left(f_{\theta^{k+1}(\omega)}\right) \leq \mathcal{L}^{buf}\left(f_{\theta^k(\omega)}\right)$. According to Eq. (13), one obtains that

$$\begin{aligned} &\left\langle \frac{\partial \mathcal{L}^{buf}\left(f_{\theta^k(\omega^{k+1})}\right)}{\partial \theta^k}, \frac{\partial \mathcal{L}^{trn}\left(\mathcal{F}_{\theta^k,\omega^{k+1}}\right)}{\partial \theta^k} \right\rangle \\ &\geq \frac{\alpha\eta}{2} \left\| \frac{\partial \mathcal{L}^{trn}\left(\mathcal{F}_{\theta^k,\omega^{k+1}}\right)}{\partial \theta^k} \right\|_2^2, \end{aligned} \tag{14}$$

where $\alpha > 0$ is the inner-loop learning rate and $\eta > 0$ is the Lipschitz constant. □

C. Details of Experiments

C.1. Comparison Methods

We herein detail the four baselines: ER, DER++ [7], RAR [41], and CLSER [4]; and five state-of-the-art comparison methods presented in the main text: iCaRL [33], LUCIR [20], BiC [39], ER-ACE [8], and SS-IL [1].

- **ER** is the most commonly used baseline in the continual learning problem, which has been introduced in Sec. 3.1.
- **DER++ [7]** additionally saves the output logits in the memory buffer and utilizes an additional logit distillation to further prevent forgetting the previously learned knowledge, which is another simple yet strong and effective baseline.
- **RAR [41]** applies random augmentation to prevent overfitting on the memory buffer samples, which is also a plug-and-play strategy that can be used for many rehearsal-based CL methods. In our experiments, we apply RAR on DER++ which is also noted as RAR for simplicity.
- **CLSER [4]** involves a stable network and a plastic network to consolidate the previous knowledge and learn recent experiences, respectively.
- **iCaRL [33]** replaces the linear classifier as the nearest class mean (NCM) classifier during inference and chooses samples that are close to class means as the memory buffer. This buffer management strategy facilitates the calculation of more accurate class means. So we still use this strategy for a better prediction performance, although the greedy algorithm does not fully meet the online setting.

method	cifar10			
	M=0.2k		M=0.5k	
	ACC	FM	ACC	FM
ER	52.99	29.34	57.80	21.21
ER-CBA (ours)	**57.25**	**16.38**	**58.13**	**12.39**
Gains	+ 4.26	-12.96	+ 0.33	- 8.82
DER++ [7]	57.13	25.85	59.31	25.31
DER-CBA (ours)	**57.78**	**12.87**	**60.64**	**11.30**
Gains	+ 0.65	-12.98	+ 1.33	-14.01
RAR [41]	53.90	31.91	56.55	27.66
RAR-CBA (ours)	**56.12**	**16.68**	**58.25**	**12.70**
Gains	+ 2.22	-15.23	+ 1.70	-14.96
CLSER [4]	54.64	25.47	59.31	20.07
CLSER-CBA (ours)	**57.88**	**18.36**	**60.62**	**13.14**
Gains	+ 3.24	- 7.11	+ 1.31	- 6.93

Table 6: ACC and FM of our method applied on the four baselines under the '*Blurry-30*' settings on Split CIFAR-10.

- **LUCIR [20]** proposes a weight normalization strategy to rebalance the new and old samples. It is also coupled with a cosine constraint and a margin ranking loss for negative sample learning.
- **BiC [39]** splits a small balanced validation set from the whole training data used for the second stage training. Following the origin paper, the ratio of train/validation split on the examples is also set as 9:1 in our experiments. In the second stage, we train the bias correction layer for 250 epochs by the Adam optimizer with a learning rate of 0.001.
- **ER-ACE [8]** separates the losses on the incoming new task data and the buffered old task data and only considers news classes for the denominator of CE loss on the incoming data. This asymmetric update pushes new classes to adapt to the older ones.
- **SS-IL [1]** proposes a separated softmax blocking the flow of the gradients between the old and new classes, which can avoid the imbalance penalization for the new and old samples. And an additional task-specific distillation is used on the rehearsal data.

C.2. Experiment Details

Details of the datasets. For CIFAR-10 and CIFAR-100 [24], both of them consist of 50,000 32×32 RGB images for training and 10,000 test images, whereas CIFAR-10 only contains 10 classes and CIFAR-100 contains 100 classes. We evenly split them into 5 tasks and 10 tasks in order, respectively, which are noted as Split CIFAR-10 and Split CIFAR-100. In addition, for the tiny-ImageNet, there are 100,000 color images of 200 different classes cropped as 64×64. Similarly, we split the tiny-ImageNet into 10

Method	$a_{1,5}$	$a_{2,5}$	$a_{3,5}$	$a_{4,5}$	$a_{5,5}$	ACC
SS-IL [1]	68.26	31.84	40.01	59.25	6.73	41.22
ER-ACE [8]	48.35	32.33	47.04	53.44	50.60	46.35
ER	28.27	26.12	31.95	46.81	78.48	42.32
ER-CBA (ours)	44.29	30.40	37.41	48.74	66.23	45.41
DER++ [7]	31.48	20.56	34.16	51.96	79.02	43.44
DER-CBA (ours)	62.88	33.25	36.98	48.69	60.39	48.44
RAR [41]	32.17	30.79	35.59	49.70	79.61	45.57
RAR-CBA (ours)	65.42	32.51	43.83	44.70	55.80	48.45
CLSER [4]	33.05	28.66	36.58	46.31	77.79	44.48
CLSER-CBA (ours)	47.48	34.95	42.82	51.85	71.08	49.63

Table 7: The t-th task accuracy $a_{t,T}(t = 1, \cdots, T)$ after the final task (*i.e.*, the T-th task) training on Split CIFAR-10 with buffer size $M = 0.5k$.

Method	$a_{1,10}$	$a_{2,10}$	$a_{3,10}$	$a_{4,10}$	$a_{5,10}$	$a_{6,10}$	$a_{7,10}$	$a_{8,10}$	$a_{9,10}$	$a_{10,10}$	ACC
SS-IL [1]	26.96	18.90	30.94	17.68	25.42	26.41	27.64	22.10	24.64	35.32	25.60
ER-ACE [8]	27.91	27.22	31.77	21.78	27.82	30.10	31.73	24.96	26.16	15.90	26.54
ER	19.26	19.21	22.84	13.35	17.01	20.58	21.91	14.92	14.20	64.01	22.73
ER-CBA (ours)	25.92	22.35	34.82	27.81	28.44	35.77	32.83	30.04	17.83	20.05	27.59
DER++ [7]	12.81	8.76	17.60	5.93	12.03	11.38	19.22	9.13	6.69	72.52	17.61
DER-CBA (ours)	43.40	24.60	29.07	16.33	23.66	24.06	27.92	16.99	11.96	46.72	26.47
RAR [41]	9.31	6.44	12.60	3.98	7.53	10.37	13.70	6.80	6.45	68.49	14.57
RAR-CBA (ours)	40.58	22.59	26.15	12.65	17.90	21.96	26.59	15.99	10.91	46.58	24.19
CLSER [4]	21.56	19.45	25.35	11.78	19.33	17.44	23.99	15.73	10.73	66.48	23.18
CLSER-CBA (ours)	26.21	22.02	36.89	27.87	33.85	36.80	35.60	34.12	18.17	19.36	29.09

Table 8: The t-th task accuracy $a_{t,T}(t = 1, \cdots, T)$ after the final task (*i.e.*, the T-th task) training on Split CIFAR-100 with buffer size $M = 5k$.

tasks (*i.e.*, Split Tiny-ImageNet), and each task contains 20 disjoint classes.

Details of experiment settings. Following [7], as we aforementioned in Sec. 4.1, we use ResNet18 as our backbone and optimized by SGD optimizer with a learning rate of 0.03. And the CBA module is updated by Adam optimizer where the learning rate is 0.001 for Split CIFAR-10 and 0.01 for Split CIFAR-100 and Split Tiny-ImageNet. The batch size is set as 32 in all experiments. Besides, the weight of the logit distillation loss and the rehearsal loss of the baseline DER++ [7] are fixed at 0.2 and 0.5, respectively.

D. Additional Experiments

D.1. Results under '*Blurry-30*' Setting

Since task blurry is a more challenging setting in online CL, we further investigate another blurry setting where the 30% data of each task may appear in the other tasks, *i.e.*, the '*Blurry-30*' setting. Table 6 compiles the outcomes of our method applied on the four baselines (*i.e.*, ER, DER++, RAR, and CLSER) on Split CIFAR-10 with the buffer sizes $M = 0.2k$ and $M = 0.5k$. Similar to the findings observed under '*Blurry-10*' in Table 3 of the main text, our CBA module can consistently improve the average accuracy and significantly alleviate forgetting of the baselines across different buffer sizes. These results further demonstrate that our proposed CBA is a flexible and adaptable module that can be easily plugged into most rehearsal-based methods under various settings.

D.2. Accuracy of Each Task

In Table 2 of the main text, we present the accuracy of each task after the whole training process based on ER, which demonstrates the proposed CBA module can help the model adapt to distribution shift by consolidating the previously learned knowledge. To further illustrate it, we herein provide more results of our CBA module applied to the four baselines on Split CIFAR-10 ($M = 0.5k$, in Table 7), Split CIFAR-100 ($M = 5k$, in Table 8), and Split Tiny-ImageNet ($M = 5k$, in Table 9), respectively.

Method	$a_{1,10}$	$a_{2,10}$	$a_{3,10}$	$a_{4,10}$	$a_{5,10}$	$a_{6,10}$	$a_{7,10}$	$a_{8,10}$	$a_{9,10}$	$a_{10,10}$	ACC
SS-IL [1]	16.38	16.14	16.99	25.04	19.62	19.04	17.39	16.66	12.49	25.59	18.53
ER-ACE [8]	21.65	20.38	23.25	24.94	21.36	18.39	16.38	17.85	12.74	13.44	19.04
ER	14.38	14.44	15.39	18.29	15.53	13.14	11.25	11.87	4.80	51.15	17.02
ER-CBA (ours)	21.58	18.84	24.44	25.30	20.79	18.26	18.83	18.89	11.33	23.75	20.20
DER++ [7]	7.43	9.20	9.36	10.95	9.35	7.02	5.08	7.18	1.65	55.88	12.31
DER-CBA (ours)	36.04	21.74	22.85	24.36	17.68	13.50	12.76	12.03	6.60	31.40	19.90
RAR [41]	5.32	6.48	7.42	8.54	8.19	4.52	3.64	4.67	1.18	53.97	10.39
RAR-CBA (ours)	30.71	19.69	20.85	20.45	15.63	11.76	9.28	9.39	5.23	29.92	17.29
CLSER [4]	13.80	14.05	15.99	19.43	15.60	13.26	11.64	11.37	5.84	51.81	17.28
CLSER-CBA (ours)	23.04	20.16	23.37	25.17	22.30	21.38	22.56	22.84	10.55	24.83	21.62

Table 9: The t-th task accuracy $a_{t,T}(t = 1, \cdots, T)$ after the final task (*i.e.*, the T-th task) training on Split Tiny-ImageNet with buffer size $M = 5k$.

The results in Table 7-9 show that the proposed method can consistently improve the four baselines ER, DER++, RAR, and CLSER. Among these four baselines, DER++ achieves the worst performance on Split CIFAR-100 and Split Tiny-ImageNet datasets under the online CL settings, as it suffers from severe task-recency bias with higher accuracy on the new task yet lower accuracy on old tasks. However, our method outperforms DER++ by a large margin, which indicates that it can significantly ameliorate the bias of the baseline. Although RAR and CLSER perform better than DER++, the two baselines still tend to classify samples of the old tasks as those of the new task, while our proposed CBA method can alleviate this tendency by consolidating the old tasks.

We also compare the results with two state-of-the-art methods, *i.e.*, SS-IL, and ER-ACE. As shown in Table 7, SS-IL fails to learn the new task as it pays too much attention to the previous tasks. The same finding can also be observed in Table 8 and Table 9 for ER-ACE. Due to suppressing the new task performance in SS-IL and ER-ACE, the two methods achieve a lower FM in Table 1 (referring to the definition of FM in Sec. 4.1). However, our method not only retains the learned knowledge of the previous tasks but also focuses on learning the new task knowledge which leads to better performance.

D.3. Results of Small Buffer Sizes

In Table 1 of the main text, we have reported the results of our method under the three datasets with various buffer sizes and demonstrated the effectiveness of the proposed CBA module. In this section, we further explore the performance of our method under some extreme scenarios, *i.e.*, the storage spaces are limited, leading to small buffer sizes in the CL process. In Table 10, we further test the performance with smaller buffer sizes, *i.e.*, on Split CIFAR-10 with M=100 and 50, and on Split CIFAR-100/Tiny-ImageNet with M=200. Our method consistently outperforms the baseline ER for both ACC and FM, demonstrating that the proposed bi-level optimization framework remains effective in alleviating task-recency bias even with extremely small buffer sizes.

Dataset	M	ER		ER-CBA	
		ACC ↑	FM ↓	ACC ↑	FM ↓
Split CIFAR-10	200	35.21	50.28	**42.32**	**40.80**
	100	32.89	64.18	**35.00**	**59.14**
	50	23.06	72.51	**23.73**	**70.08**
Split CIFAR-100	200	10.10	49.16	**13.14**	**33.61**
Split Tiny-ImageNet	200	6.52	43.12	**7.88**	**32.81**

Table 10: Ablation analysis on small buffer sizes.

D.4. Computation and GPU Memory

Table 11 lists the training cost on NVIDIA GeForce RTX 2080 Ti, and it shows that our approach only increases approximately 100 Mb GPU memory and takes no more than 2 seconds extra training time compared with the baselines since it only needs to unroll the gradient of the linear classification layer to update a few hyper-parameters ϕ in the outer-loop update (lines 426-450 in the main text). Besides, our approach involves no extra overhead compared with the baselines in the test stage. We will add a related discussion in the revision.

Method	Params ($\times 10^6$)	GPU memory (Mb)	Training time† (s)
ER	11.174	1517	27.87
ER-CBA	11.179	1615	29.74
DER++ [7]	11.174	1665	39.52
DER-CBA	11.179	1779	40.03

Table 11: Computation and memory on Split CIFAR-10 with M=0.2k. †: the average training time for each task.

D.5. Effect of Batch Sizes

In this section, we perform an additional ablation study about the effectiveness of different batch sizes to show that the performance gain of our method does not come from the extra batch of replaying data in the outer-loop update but from our advanced model. To this end, we show in detail the batch sizes used in the inner-loop ($\mathcal{B}^{trn}$) and outer-loop ($\mathcal{B}^{buf}$) for each setup in Table 12, where (a) the baseline ER only use replay data in $\mathcal{B}^{trn}$ while (b) the proposed ER-CBA uses another replaying batch in $\mathcal{B}^{buf}$. In order to address this concern, we keep the same amount of replay data with ER (only 32) and introduce two variants (ER-CBA*) in which the inner- and outer-loop updates share a single batch of 32 samples from the memory buffer. Specifically, we select 16 and 8 samples as $\mathcal{B}^{buf}$ for the outer loop, as shown in Table 12 (c) (16+16) and (d) (24+8). The results show that even with the same amount of replay batch with ER, ER-CBA* does not degrade performance compared with ER-CBA, indicating the effectiveness of our method does not attribute to the additional replay batch.

	Method	$\mathcal{B}^{trn}$		$\mathcal{B}^{buf}$	ACC ↑	FM ↓
		New task	Memory buffer			
(a)	ER	32	32	/	35.11	50.28
(b)	ER-CBA (paper)	32	32	32	37.27	41.39
(c)	ER-CBA*	32	16	16	37.81	37.25
(d)	ER-CBA*	32	24	8	38.12	36.74

Table 12: Effect of batch sizes on Split CIFAR-10 with M=0.2k.