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

Abstract

Online continual learning (CL) aims to learn new knowledge and consolidate previously learned knowledge from non-stationary data streams. Due to the time-varying training setting, the model learned from a changing distribution easily forgets the previously learned knowledge and biases toward the newly received task. To address this problem, we propose a Continual Bias Adaptor (CBA) module to augment the classifier network to adapt to catastrophic distribution change during training, such that the classifier network is able to learn a stable consolidation of previously learned tasks. In the testing stage, CBA can be removed which introduces no additional computation cost and memory overhead. We theoretically reveal the reason why the proposed method can effectively alleviate catastrophic distribution shifts, and empirically demonstrate its effectiveness through extensive experiments based on four rehearsal-based baselines and three public continual learning benchmarks[1].

1. Introduction

Continual learning (CL) [40, 48] focuses on designing a model that can learn continuously from streaming data and accumulate new knowledge while consolidating previously learned knowledge. In the context of CL, the data distribution of streaming tasks is in general non-stationary and changes over time, which violates the independent and identically distributed (i.i.d) assumption that is commonly adopted in machine learning. Therefore, continual learning suffers from the notorious catastrophic forgetting problem [16], where the model severely forgets the previously learned knowledge after being trained on a new task.

Traditional offline CL stores all training batches of the current task and the model is trained on these samples for multiple epochs with repeat shuffle. However, the availability of previously learned batches might be restricted due to privacy concerns [14] or memory limitations. In this paper, we mainly focus on a more challenging and realistic setting, online CL [28], where samples from each task can be trained only single-pass (*i.e.*, one epoch) and the past batches are not accessible.

In online CL, the distribution of the training data changes over time and it differs not only from the joint distribution of all tasks (as in offline CL) but also from the distribution of the task they belong to. Therefore, online CL commonly causes an even more severe distribution shift, which further intensifies catastrophic forgetting. To alleviate this problem, rehearsal-based algorithms [35, 48] employed a small memory buffer to store examples of previous tasks so as to approximate the joint distribution of all seen training data, then collectively trained the model on the memory buffer with the current task. Along this line, DER++ [7] utilized an additional knowledge distillation to further reply logits of old task samples, and RAR [46] used random augmentation to address the overfitting problem of the small memory buffer. In another vein, a wide range of works attribute catastrophic forgetting to task-recency bias [30], *i.e.*, the classifiers tend to classify samples into currently received classes. They in turn proposed to improve the original linear classifier [1, 20, 44] or directly replace the classifier with the nearest classifier [30, 35] to mitigate the adverse effects of class imbalance between currently received classes and replayed classes. Despite the promising performance, almost all of these methods implicitly view task-recency bias as a label distribution shift and tackle it from the perspective of class imbalance problem, which makes these methods suboptimal in practice [8].

In fact, the target of online CL is to accomplish a stable consolidation of knowledge for all learned tasks, by training a classifier to fit the posterior probability $\mathbb{P}(Y|X)$, where X and Y represent stochastic variables of the input data and the corresponding label, respectively. According to Bayes's rule $\mathbb{P}(Y|X) \propto \mathbb{P}(X|Y)\mathbb{P}(Y)$, the posterior probability de-

*Corresponding author
[1]Code is available at https://github.com/wqza/CBA-online-CL

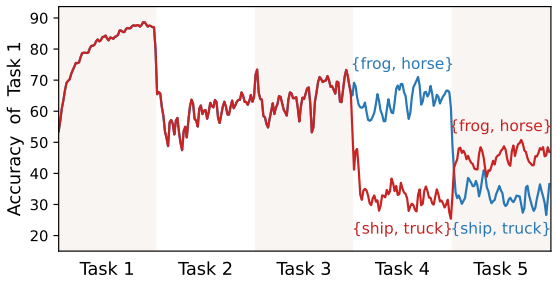


Figure 1: Comparing the accuracy change of the categories learned in 1st task. For the 4th and 5th tasks, the **red** corresponds to learning from $\{ship, truck\}$ to $\{frog, horse\}$, and the **blue** corresponds to learning from $\{frog, horse\}$ to $\{ship, truck\}$.

pends on both the prior probability $\mathbb{P}(Y)$ and the likelihood $\mathbb{P}(X|Y)$. It can be seen that any shift from $\mathbb{P}(Y)$ or $\mathbb{P}(X|Y)$ will lead to distribution change in $\mathbb{P}(Y|X)$. On the one hand, as aforementioned, the label distribution $\mathbb{P}(Y)$ shifts leading to severe forgetting when a new task comes in. On the other hand, $\mathbb{P}(X|Y)$ can also suffer from catastrophic shifts (dubbed feature distribution shift for simplicity) due to the time-varying data streams. To illustrate this, we conduct a toy experiment with Experience Replay (ER) [34] on CIFAR-10 [24], where we manipulate the incoming categories of the 4th task (represented by the red and blue lines, respectively), and keep tracking the accuracy of the categories of the 1st task as shown in Fig.1. Notably, $\mathbb{P}(X|Y)$ of the two lines are totally different, while their label distribution $\mathbb{P}(Y)$ maintains the same tendency. This disparity leads to a catastrophic change in the performance of categories learning in the 1st task. This validates the existence of feature distribution shifts and poses a challenging problem for online CL: how to accomplish a stable consolidation of past knowledge under these distribution shifts.

To tackle this challenge, based on rehearsal-based methods, this paper proposes a bi-level learning framework to adapt the posterior distribution shift during each training step directly. Specifically, we introduce a nonlinear transformation module, Continual Bias Adaptor (CBA), to dynamically capture the catastrophic distribution change for posterior probability $\mathbb{P}(Y|X)$. Simultaneously, the original classifier is adjusted to fit an implicit posterior probability which tends to be a stable consolidation of previously learned knowledge across different tasks. Intrinsically, the proposed method implicitly aligns the distribution between training and memory buffer data at each iteration. During the testing stage, the CBA module can be simply removed and the original classifier can achieve a good performance on all seen tasks. To summarize, our main contributions are three-fold:

- We propose a bi-level learning framework to model the posterior distribution shift in an online manner. We theoretically investigate the proposed method from gradient alignment and reveal the reason why it can effectively alleviate catastrophic distribution shifts.
- We propose a CBA module that can plug in most of the rehearsal-based methods during training, and be removed in the test stage so that it involves no computation cost and memory overhead in inference.
- We evaluate the performance based on four rehearsal-based baselines with extensive experiments over various benchmarks. We show that the proposed method can effectively consolidate previously learned knowledge. This is also demonstrated by task-blurry online CL and offline CL settings.

2. Related Work

Continual learning settings. Based on different task construction manners, continual learning mainly falls into three categories [40, 48]: Task-incremental learning (Task-IL), Domain-incremental learning (Domain-IL), and Class-incremental learning (Class-IL). Specifically, Task-IL assumes the task identity is provided, which means the model can use the task index in both the training and testing stage. Domain-IL focuses on the concept drift where the domain of each task is changing but with the same label space [17, 29, 39]. This paper concentrates on the more challenging Class-IL, where the task indices are absent during testing [7, 22, 35, 49].

From the training perspective, CL can be divided into offline and online CL. Offline CL could preserve all samples of the current task, permitting repeated training on them [4, 7, 9, 11, 35]. As for online CL, samples of each task cannot be stored and each sample can only be seen once, except for those examples saved in the memory buffer [23, 28]. In this paper, we mainly focus on online CL, which is more demanding and realistic than offline CL.

Rehearsal-based methods in online CL: The objective of online CL is to learn models with a stronger ability to quickly fit new tasks while retaining knowledge of old tasks [8, 13, 23, 41, 45]. Experience replay (ER), the most commonly used baseline, trains the new incoming samples together with old samples stored in the memory buffer. Its variants attempt to combine replay strategies with other techniques, such as knowledge distillation and random augmentation. For example, DER++ [7] utilized the knowledge distillation technique to further replay logits of memory buffer data. RAR [46] adopted random augmentation to alleviate the overfitting of the tiny memory buffer, and CLSER [4] constructed a plastic model and a stable model for recent and structural knowledge distillation. Different from these methods, some studies emphasize maximizing

the utility and benefits of the memory buffer samples. Instead of randomly sampling, GSS [3] selected the samples stored in the memory buffer according to the cosine similarity of gradients. MIR [2] chose maximally interfering samples whose prediction will be most negatively impacted by the foreseen parameters update. OCS [45] picked the most representative data of the current task while minimizing interference to previous tasks. Unlike these methods, our method focuses on alleviating distribution shifts and can plug in most current rehearsal-based approaches.

Task-recency bias in online CL: Task-recency bias [20, 48] in online CL means that classifiers tend to mistakenly classify previously learned classes as newly encountered ones. A popular perspective is that the linear classifier is particularly susceptible to task-recency bias. To address this, iCaRL [35] proposed to replace the linear classifier with nearest class mean (NCM) classifiers. Similarly, SCR [30] and Co^{2}L [10] employed the NCM classifier where the feature extractor is learned from contrastive learning. A wide range of works tackles the task-recency bias as a class imbalance problem [1, 9, 20, 44, 47]. For example, LUCIR [20] added weight normalization on the linear classifier. BiC [44] proposed a bias correction layer turned on a held-out validation set. SS-IL [1] separated the softmax to mitigate the imbalanced penalization of the old class outputs. On the flip side, ER-ACE [8] pointed out that task-recency bias can also arise from feature interference and designed an asymmetric loss to address this problem. Different from these methods, our proposed method relaxes the assumption on label/feature distribution shift by directly modeling the posterior distribution shift. Besides, some methods addressing class imbalance may potentially be explored for tackling task-recency bias in online CL [25, 31, 42, 43]. However, many of these approaches face challenges in generalization to CL due to unstable data distribution.

3. Method

3.1. Preliminaries

Suppose we have N sequential tasks $\{\tau_1, \tau_2, \cdots, \tau_N\}$, each task $\tau_i = \{(x_j^i, y_j^i)\}_{j=0}^{N_i}$, where N_i is the total number of training samples in τ_i. In online CL, the classification model f_θ can only train single-pass on samples of each task τ_i (*i.e.*, train with one epoch), and the previously seen batches are not accessible. Let τ_t represent the current learning task, $\mathcal{M}$ denotes the tiny memory buffer which stores the samples of previous tasks $\{\tau_1, \cdots, \tau_{t-1}\}$. The most representative rehearsal-based method Experience Replay (ER) [34, 37] jointly trains current task τ_t with examples sampled from the buffer $\mathcal{M}$, and its training objective function $\mathcal{L}$ is:

$$\mathcal{L}^{trn}\left(\mathcal{B}^{trn}; f_\theta\right) = \frac{1}{|\mathcal{B}^{trn}|} \sum_{x,y \in \mathcal{B}^{trn}} L(f_\theta(x), y), \quad (1)$$

where the training batch $\mathcal{B}^{trn} = \mathcal{B}^t \cup \mathcal{B}^{buf}$ consists of a batch of incoming new samples $\mathcal{B}^t \subset \tau_t$ and a batch sampling from the memory buffer $\mathcal{B}^{buf} \subset \mathcal{M}$, and L denotes the cross-entropy loss.

Note that the memory buffer $\mathcal{M}$ is updated by reservoir sampling after training each batch $\mathcal{B}_t$, which means $\mathcal{M}$ not only includes samples of previous tasks but also contains the samples of the current task τ_t. Therefore, the memory buffer $\mathcal{M}$ is a relatively balanced set containing samples of all seen tasks.

3.2. Framework Formulation

Our focus in this work is to alleviate catastrophic forgetting of rehearsal-based models under a challenging online CL setting. As aforementioned, current rehearsal-based models commonly suffer from catastrophic distribution changes due to time-varying data streams. To address the problem, we relax the assumption on the label or feature distribution shift in comparison to prior methods and propose to directly model the catastrophic distribution change for posterior probability $\mathbb{P}(Y|X)$, making the original f_θ learn a stable consolidation of knowledge for all past tasks.

The main methodology is to design a Continual Bias Adaptor (CBA) g_ω that: 1) dynamically augments the classifier network f_θ to produce more diverse posterior distributions by only tuning the parameters of g_ω (ω can be viewed as hyper-parameters) to adapt to catastrophic posterior change; 2) makes the original classifier f_θ fit an implicit posterior that tends to achieve a stable consolidation of knowledge of previously learned tasks. For a training example x^{trn}, its posterior probability is adjusted by the augmented classifier network $\mathcal{F}_{\theta,\omega} = g_\omega \circ f_\theta$ in an online manner, which can be formulated as

$$\tilde{y}^{trn} = \mathcal{F}_{\theta,\omega}(x^{trn}) = g_\omega \circ f_\theta(x^{trn}), \quad (2)$$

where $\circ$ is function composition operator, and g_ω is designed as a light-weight network. Thus, the augmented classifier network $\mathcal{F}_{\theta,\omega}$ is firstly updated to learn new knowledge from the training data $\mathcal{B}^{trn} = \mathcal{B}^t \cup \mathcal{B}^{buf}$ that minimizes the rehearsal-based empirical risk, *i.e.*,

$$\theta^*(\omega) = \arg\min_\theta \mathcal{L}^{trn}\left(\mathcal{B}^{trn}; \mathcal{F}_{\theta,\omega}\right), \quad (3)$$

where ω is actually a hyper-parameter of the optimal θ^*. Note that the training loss function $\mathcal{L}^{trn}$ can be adopted by different rehearsal-based loss functions like ER and its variants such as DER++, and CLSER. Here we take the ER formulation as an example for simplicity.

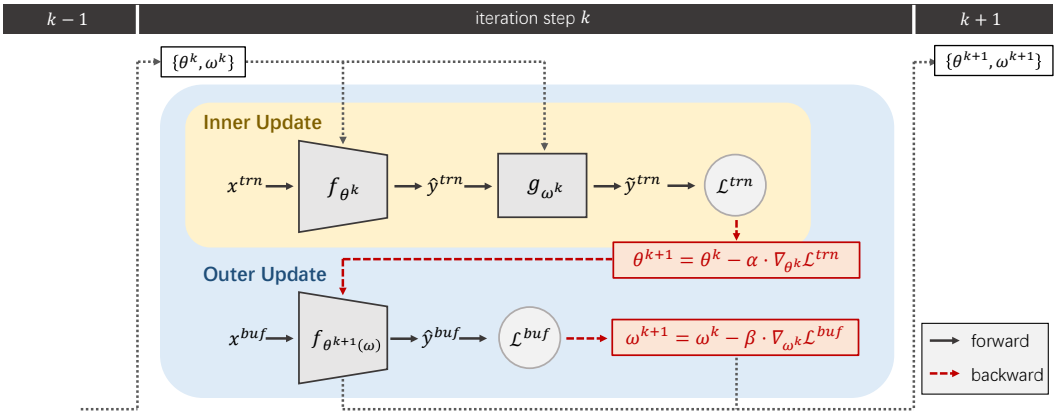


Figure 2: Method overview. At the iteration step k, for the inner loop, the forward process computes the training loss in Eq. (3) and the backward process updates the classification model parameter $\theta(\omega)$ by Eq. (5). And for the outer loop, the forward process computes the loss in Eq. (4) and the backward process updates the CBA parameter ω by Eq. (6)

On the other hand, our ultimate objective is to protect the original classifier network f_θ from catastrophic distribution shift, while accomplishing a stable consolidation of knowledge across different tasks. To this end, we further keep tracking the performance of the classifier network to prevent catastrophic forgetting, which requires that $f_{\theta^*(\omega)}$ returned by minimizing the rehearsal-based empirical risk Eq. (3), should maximize the performance of previously seen data. Since accessing all of this historical data is unfeasible, it can be approximated by the empirical risk over the memory buffer data, *i.e.*,

$$\begin{aligned} \omega^* &\triangleq \arg\min_{\omega} \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta^*(\omega)}\right), \\ &= \arg\min_{\omega} \frac{1}{|\mathcal{B}^{buf}|} \sum_{x,y \in \mathcal{B}^{buf}} L\left(f_{\theta^*(\omega)}(x), y\right). \end{aligned} \tag{4}$$

This formulation attempts to find the optimal parameters such that the classifier network returned by optimizing Eq. (3), should also have a good performance on the memory buffer data which acts as a stable consolidation of knowledge from the learned tasks.

In fact, Eq. (3) and Eq. (4) formulate a bi-level learning framework. In the inner loop Eq. (3), the augmented classifier network $F_{\theta,\omega}$ is updated to learn new knowledge and rehearse old knowledge from $\mathcal{B}^{trn}$. In the outer loop Eq. (4), the parameters of g_ω are updated from $\mathcal{B}^{buf}$ to consolidate the previously learned knowledge, which has been updated in the inner loop, against catastrophic posterior change.

We herein detail the proposed CBA module and bi-level optimization algorithm in the following aspects:

Continual Bias Adaptor: The proposed CBA module g_ω is designed to be a nonlinear transformation that takes the outputs (logits) of the original classifier network f_θ as its inputs. As such, the original classifier network f_θ is augmented by cascading the CBA module after its classification layer, as shown in Fig. 2. To capture catastrophic distribution change during continual learning, we parameterize g_ω as a multi-layer perceptron (MLP) network with a single hidden layer containing 256 nodes. Each hidden node is equipped with a ReLU [32] activation function, and the output employs the Softmax activation function to guarantee a posterior probability output. Albeit simple, this network is known as a universal approximator, capable of fitting almost any continuous function [18] and thus can fit various posterior distribution changes. Additionally, we introduce a skip-layer connection within the CBA module, connecting the outputs of f_θ to the MLP outputs, which aids the model convergence and facilitates the gradient backward propagation, which has been verified in previous works [19, 21].

Note that the proposed CBA module is only used in the training stage. In the test stage, the test sample x^{tst} is predicted by f_θ, that is $\hat{y}^{tst} = f_\theta(x^{tst})$. This indicates that our method does not introduce any calculation overhead in the test stage. As a result, our method can be inferred at any time in the learning process of online CL [23].

The learning of CBA. The Eq. (3) and Eq. (4) in the bi-level optimization are nested with each other. Concretely, the solution $\theta^*(\omega)$ of Eq. (3) depends on the hyper-parameter ω and the optimum solution is obtained at ω^*. However, the required optimum ω^* is solved by Eq. (4) which relies on the best $\theta^*(\omega)$. Indeed, there is generally no closed-form solution to the bi-level optimization framework [6] and we approximately update the θ and ω using a gradient-optimization-based method following [36, 38].

(1) **Update θ.** Given the CBA parameter ω^k at iteration step k, the CBA parameter ω^k is fixed and we formulate

Algorithm 1 Training of CBA in Online CL

Input: new incoming sample batch $\mathcal{B}^t$, memory buffer $\mathcal{M}$
Output: classifier network parameter θ, continual bias adaptor (CBA) parameter ω

Initialize all network parameters as $\{\theta^0, \omega^0\}$.
while $\mathcal{B}^t \neq \emptyset$ **do**
  # Inner-loop optimization:
  $\mathcal{B}^{trn} = \mathcal{B}^t \cup \mathcal{B}^{buf}, \mathcal{B}^{buf} \subset \mathcal{M}$ # Inner training data
  Compute the inner-loop loss $\mathcal{L}^{trn}$ by Eq. (3)
  Update classifier network parameters θ by Eq. (5)
  # Outer-loop optimization:
  $\mathcal{B}^{buf} \subset \mathcal{M}$ # Outer training data
  Compute the outer-loop loss $\mathcal{L}^{buf}$ by Eq. (4)
  Update CBA parameters ω by Eq. (6)

a one-step stochastic gradient descent (SGD) to update the classifier network parameter θ^k in Eq. (3), which can be represented as

$$\theta^{k+1}(\omega) = \theta^k - \alpha \cdot \nabla_\theta \mathcal{L}^{trn}\left(\mathcal{B}^{trn}; \mathcal{F}_{\theta^k,\omega^k}\right), \quad (5)$$

where α is the inner-loop learning rate.

(2) **Update ω.** After the one-step updating of θ^k, we have obtained $\theta^{k+1}(\omega)$ which is a function of ω. Then we can optimize the ω in Eq. (4) based on the updated $\theta^{k+1}(\omega)$ as following:

$$\omega^{k+1} = \omega^k - \beta \cdot \nabla_\omega \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta^{k+1}(\omega)}\right), \quad (6)$$

where β is the outer-loop learning rate. Note that $\nabla_\omega \mathcal{L}^{buf}$ in Eq. (6) introduces a second-order derivative, which can be easily implemented by the automatic differentiation system like Pytorch [33], and the detailed calculation can be found in Appendix A. To alleviate the calculation burden of this derivation, we assume that ω is only correlated with the parameters of the linear classification layer. This allows us to only unroll the second-order derivation of the linear classification layer in Eq. (6). As the linear classification layer involves a small number of parameters compared to the whole classifier network, our proposed algorithm is more efficient than other bi-level optimization algorithms [15, 26, 36, 38]. A comprehensive discussion concerning computation and GPU memory utilization of our method is presented in Appendix D.4. The complete training process is detailed in Alg. 1 and more intrinsic discussion can be found in the subsequent paragraph.

3.3. Theoretical Analysis

In this section, we theoretically analyze the proposed CBA model and the bi-level optimization procedure. The following theorem claims that our algorithm inherently establishes gradient alignment between the loss on the corresponding training set $\mathcal{B}^{trn}$ and the memory buffer $\mathcal{B}^{buf}$.

Theorem 1. *Assume that the outer-loop loss $\mathcal{L}^{buf}(\cdot; f_\theta)$ is η gradient Lipschitz continuous, then the bi-level optimization Eq. (3) and Eq. (4) potentially guarantees an alignment between the gradient of the outer-loop loss and the inner-loop loss with respect to the classification model parameter θ, that is,*

$$\left\langle \frac{\partial \mathcal{L}^{buf}\left(\mathcal{B}^{buf}; f_{\theta^k(\omega)}\right)}{\partial \theta^k}, \frac{\partial \mathcal{L}^{trn}\left(\mathcal{B}^{trn}; \mathcal{F}_{\theta^k,\omega}\right)}{\partial \theta^k} \right\rangle \geq \frac{\alpha\eta}{2} \left\| \frac{\partial \mathcal{L}^{trn}\left(\mathcal{B}^{trn}; \mathcal{F}_{\theta^k,\omega}\right)}{\partial \theta^k} \right\|_2^2, \quad (7)$$

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

The detailed proof of Theorem 1 is shown in Appendix B. Furthermore, this theorem reveals two insights into our algorithm. On the one hand, this theorem explains why the CBA module adaptively assimilates the task-recency bias. We hope the angle between the gradient w.r.t the classifier network parameter θ on the training set **with** CBA and the gradient on the memory buffer **without** CBA is as small as possible. This means that the classifier network is expected to perform well on a balanced test set without the help of CBA, and the CBA only works during training to absorb the bias. On the other hand, Theorem 1 also shows why our method can mitigate forgetting effectively. From Eq. (7), our method is potentially close to some previous gradient-alignment-based CL works [12, 27], which have already verified that the gradient alignment between the training set and the memory buffer aids the model in avoid forgetting. Unfortunately, they do not take into account the negative impact of recency bias on the model. Intuitively, our proposed method achieves more accurate gradient alignment because CBA can prevent bias from disturbing the gradient of the training set.

4. Experiments

To validate the effectiveness of the proposed method, we compare our CBA to the state-of-the-art approaches on various datasets under online CL, blurry tasks, and offline CL settings. Moreover, for better comprehension of CBA, we also conduct extensive ablation experiments to analyze different components of our approach.

4.1. Experimental Settings

Experimental datasets. Following [7], we choose three common datasets in CL, the **Split CIFAR-10** and the longer task sequence **Split CIFAR-100** and **Split Tiny-ImageNet**. Concretely, Split CIFAR-10 contains five binary classification tasks, which are constructed by evenly splitting ten classes of CIFAR-10 [24]. Split CIFAR-100 and Split Tiny-ImageNet both include ten disjoint tasks, each of which has

Method	Split CIFAR-10				Split CIFAR-100				Split Tiny-ImageNet			
	M = 0.2k		M = 0.5k		M = 2k		M = 5k		M = 2k		M = 5k	
	ACC ↑	FM ↓	ACC ↑	FM ↓	ACC ↑	FM ↓	ACC ↑	FM ↓	ACC ↑	FM ↓	ACC ↑	FM ↓
iCaRL [35]	40.99	26.84	44.50	24.87	9.13	7.79	9.13	8.14	4.03	**4.93**	4.03	**5.15**
LUCIR [20]	23.59	35.59	24.63	31.89	8.28	16.07	12.31	14.02	4.47	20.40	5.29	20.28
BiC [44]	27.71	66.45	35.47	47.92	16.32	36.70	20.89	32.33	5.43	40.14	7.50	38.52
ER-ACE [8]	41.49	20.84	46.35	18.98	24.95	**7.67**	26.54	**7.25**	17.89	7.04	19.04	6.90
SS-IL [1]	37.92	15.64	41.22	11.46	24.90	9.85	25.60	10.23	17.91	7.93	18.53	8.26
ER	35.21	50.28	42.32	40.80	20.84	35.88	22.73	33.92	14.39	32.59	17.02	31.02
ER-CBA (ours)	37.27	41.39	45.41	29.36	25.67	10.21	27.59	8.74	18.06	13.16	20.20	10.16
Gains	+ 2.06	- 8.89	+ 3.09	-11.44	+ 4.83	-25.67	+ 4.86	-25.18	+ 3.67	-19.43	+ 3.18	-20.86
DER++ [7]	40.17	41.84	43.44	40.63	16.87	44.46	17.61	44.53	11.81	39.88	12.31	39.93
DER-CBA (ours)	45.14	23.39	48.44	**16.75**	26.10	13.01	26.47	12.88	17.91	12.34	19.90	12.06
Gains	+ 4.97	-18.45	+ 5.00	-23.88	+ 9.23	-31.45	+ 8.86	-31.65	+ 6.10	-27.54	+ 7.59	-27.87
RAR [46]	40.25	40.43	45.57	36.16	14.64	45.54	14.57	47.02	10.28	40.07	10.39	40.56
RAR-CBA (ours)	43.28	**20.10**	48.45	17.42	23.87	13.10	24.19	14.00	16.70	11.48	17.29	12.56
Gains	+ 3.03	-20.33	+ 2.88	-18.74	+ 9.23	-32.44	+ 9.62	-33.02	+ 6.42	-38.59	+ 6.90	-28.00
CLSER [4]	42.01	42.33	44.48	36.83	22.48	34.80	23.18	34.35	15.34	32.48	17.28	31.79
CLSER-CBA (ours)	**44.31**	27.55	**49.63**	19.99	**26.90**	9.41	**29.09**	8.05	**19.31**	12.80	**21.62**	9.67
Gains	+ 2.30	-14.78	+ 5.15	-16.84	+ 4.42	-25.39	+ 5.91	-26.30	+ 3.97	-19.68	+ 4.34	-22.12

Table 1: Main results (ACC, higher is better, and FM, lower is better) on the three datasets with different memory buffer sizes. Our method applied on 4 baselines is shown with gray cells. **Bold** means the best results in all comparison methods and the gains of our method comparing the corresponding baselines are shown in red color.

	Method	$a_{1,5}$	$a_{2,5}$	$a_{3,5}$	$a_{4,5}$	$a_{5,5}$	ACC
Split CIFAR10 ($M = 0.5k$)	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

	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
Split CIFAR100 ($M = 5k$)	ER	19.26	19.21	22.84	13.35	17.01	20.58	21.91	14.92	14.20	64.01	22.72
	ER-CBA (ours)	25.92	22.35	34.82	27.81	28.44	35.77	32.83	30.04	17.83	20.05	27.58
Split Tiny-ImageNet ($M = 5k$)	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
	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

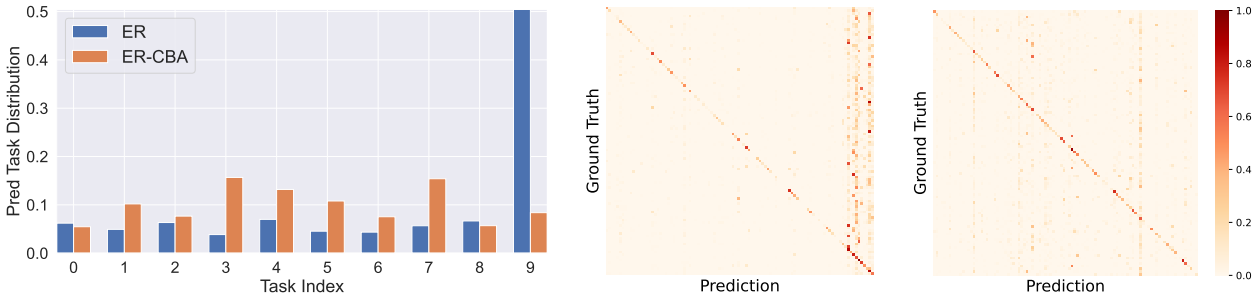
Table 2: Each task accuracy after the final task training $a_{t,T}(t = 1, \cdots, T)$ of ER and our ER-CBA on the three datasets. For Split CIFAR-10, task 5 is the new task, while task 10 is the incoming new task for Split CIFAR-100 and Split Tiny-ImageNet.

10 and 20 classes, respectively (see Appendix C for details of the three datasets).

Evaluation metrics. To comprehensively evaluate all comparison methods, we consider the following metrics:

- **Average Accuracy** (ACC ↑): calculate the average accuracy of the model trained on all tasks, *i.e.* $\text{ACC} = 1/T \sum_{t=1}^{T} a_{t,T}$, where $a_{i,j}$ represents the accuracy of the task i after training on the task j.
- **Forgetting Measure** (FM ↓): average of the decreasing from the best accuracy to the final accuracy, *i.e.* $\text{FM} = 1/T \sum_{t=1}^{T} a_t^* - a_{t,T}$, where the a_t^* is the best accuracy of task t in the whole training process.
- **Area Under the Curve of Accuracy** (ACC_{AUC} ↑): this metric is the area under the curve of the accuracy [23], *i.e.,* $\text{ACC}_{\text{AUC}} = \sum_i \bar{a}(i \cdot \triangle n) \cdot \triangle n$, where $\bar{a}(i)$ represent the average accuracy when the model training at step i, and $\triangle n$ is the interval training step which is 5 for faster evaluation in our experiments.

Implementation details. We adopt the commonly used ResNet-18 as our backbone [4, 7, 44], and train all methods using the Stochastic Gradient Descent (SGD) optimizer. For our CBA, we set the learning rate as 0.001 for Split CIFAR-10, and 0.01 for Split CIFAR-100 and Split Tiny-ImageNet. To verify our method consistently outperforms other baselines, each reported result in the online CL setting is an average of 10 repeated runs, and each result in the offline is an average of 5 runs. More details about the baselines and implementation are listed in Appendix C.



(a) Prediction Task Distribution of ER and ER-CBA. (b) Confusion Matrix of ER. (c) Confusion Matrix of ER-CBA.

Figure 3: (a): The prediction distribution of each task on Split CIFAR-100 with buffer size $M = 5k$. (b) and (c): The normalized confusion matrix of ER and ER-CBA, respectively.

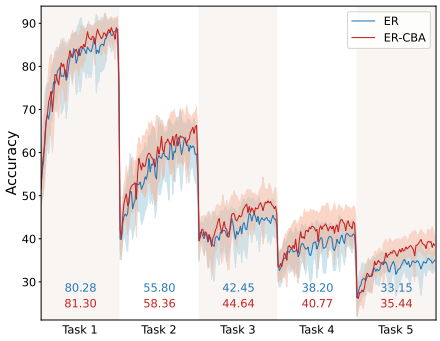


Figure 4: The average accuracy in the whole training process. The $\mathrm{ACC}_{\mathrm{AUC}}$ in each stage is shown in the figure, where the 1st row is the $\mathrm{ACC}_{\mathrm{AUC}}$ of the baseline ER and the 2nd row is that of our method ER-CBA.

4.2. Comparison on Disjoint Scenario

CBA enhanced current rehearsal-based methods. We first investigate the performance of CBA by plugging it into ER, DER, RAR, and CLSER. These four baselines adopt different replay strategies without the correction of the distribution shift. Table 1 shows that our proposed CBA improves the ACC of all these four rehearsal-based methods by up to 9.23% on these three widely used disjoint CL benchmarks. It can also be observed that the Forgetting Measure is reduced by up to 33.02%, indicating the CBA can significantly alleviate catastrophic forgetting. Additionally, our method achieves consistent improvement for the comparison methods under different memory buffer sizes, which verifies that our method has strong generalizability. Further results can be found in Appendix D.

To investigate the changing of average accuracy throughout the training process, we display the accuracy of the baseline ER and our ER-CBA in Fig. 4, which is evaluated on Split CIFAR-10 ($M = 0.2k$). As we can see, our method can surpass the baseline within just a few training iteration steps. In this figure, $\mathrm{ACC}_{\mathrm{AUC}}$ of each task is also calculated, revealing a consistent improvement by our method over the baseline ER. Moreover, our method maintains robust performance improvement even when constrained by small memory buffer sizes for each dataset, and further results are shown in Appendix D.

Analysis of other baselines Note that LUCIR employs a weight normalization prior, and BiC designs an additional linear layer, both attempting to rectify the distribution shift from a class imbalance view. However, they oversimplified the distribution shift issue, leading to suboptimal performance across these three benchmarks. iCaRL fails on the larger datasets Split CIFAR-100/Tiny-ImageNet, suggesting that the NCM classifier relies on accurate and well-separate class means, which is difficult to obtain in online CL on large datasets. Because of the worse accuracy of iCaRL on these two larger datasets, the lower FM in Table 1 does not make any sense. As for ER-ACE and SS-IL, although they exhibit decent ACC performance, they oversuppress the accuracy of new classes during training, which caused the lower FM metric.

4.3. Comparison on Blurry Scenario

Following [3, 5], we adopt the *Blurry-K* online CL setting. It simulates a practical situation where task boundaries are unclear, causing class overlaps across all tasks. Specifically, a fraction ($K\%$) of the training data from one task may appear in other tasks. Here we set $K = 10$ as an illustration, and the comparative results are summarized in Table 3.

It can be observed from Table 3 that our CBA module can vastly improve the performance of the corresponding baselines. For example, the ACC of CLSER-CBA is higher than the baseline CLSER by about 3.19% and 5.17% with memory buffer sizes $M = 0.2k$ and $0.5k$, respectively. And the FM of CLSER-CBA is lower than it of CLSER by about 8.46% and 14.97% with $M = 0.2k$ and $0.5k$, respectively.

Method	Split CIFAR-10			
	$M = 0.2k$		$M = 0.5k$	
	ACC ↑	FM ↓	ACC ↑	FM ↓
ER	44.15	40.33	50.37	30.29
ER-CBA (ours)	48.42	30.60	52.26	20.72
Gains	+ 4.27	- 9.73	+ 1.89	- 9.57
DER++ [7]	49.25	31.50	51.81	33.28
DER-CBA (ours)	51.38	23.75	52.11	17.13
Gains	+ 2.13	- 7.75	+ 0.30	-16.15
RAR [46]	47.20	37.62	48.46	37.78
RAR-CBA (ours)	**51.69**	**18.66**	52.25	**12.35**
Gains	+ 4.49	-18.96	+ 3.79	-25.43
CLSER [4]	47.57	35.77	49.52	33.54
CLSER-CBA (ours)	50.76	27.31	**54.69**	18.57
Gains	+ 3.19	- 8.45	+ 5.17	-14.97

Table 3: ACC and FM of our method applied on 4 baselines under the 'Blurry-10' settings on Split CIFAR-10.

Method	Split CIFAR-10			
	$M = 0.2k$		$M = 0.5k$	
	ACC ↑	FM ↓	ACC ↑	FM ↓
ER	55.80	47.77	66.06	33.02
ER-CBA (ours)	64.12	31.40	72.86	20.63
Gains	+ 8.32	-16.37	+ 6.80	-12.39
DER++ [7]	63.33	35.83	72.78	23.40
DER-CBA (ours)	65.64	31.78	74.37	20.06
Gains	+ 2.31	- 4.05	+ 1.59	- 3.34
RAR [46]	60.57	39.44	69.73	26.99
RAR-CBA (ours)	63.49	32.79	71.55	23.21
Gains	+ 2.92	- 6.65	+ 1.82	- 3.78
CLSER [4]	65.73	30.32	73.45	19.45
CLSER-CBA (ours)	**67.40**	**27.16**	**74.51**	**18.75**
Gains	+ 1.67	- 3.16	+ 1.06	- 0.70

Table 4: ACC and FM of our method applied on 4 baselines under the offline CL settings on Split CIFAR-10.

It shows that the proposed CBA module can be flexibly adapted to the classifier network without being perturbed by ambiguous task boundaries seriously, which also verifies the strength of our method.

4.4. Comparison under the Offline CL

To further verify the effectiveness and strength of our method, we extend our method to the offline CL setting, where each task can be trained over multiple epochs. As depicted in Table 4, these baselines perform better in the offline CL compared to the online context. Nevertheless, our method can also improve the corresponding baselines by correcting the distribution shift online. It is worth noting that CBA improves the ER significantly and makes it comparable with the other three baselines, which further illustrates the negative impact of task-recency bias on CL model performance and the strength of our method.

4.5. Discussion and Ablation Study.

To better understand the CBA, we conduct experiments and analyze the results to answer the following questions.

Question 1: Does CBA help the model adapt to distribution shift? To ascertain this, it is essential to investigate how the accuracy of each task changes after training on the final task, *i.e.*, $a_{t,T}(t = 1, \cdots, T)$. In Table 2, we exhibit $a_{t,T}$ and ACC of the baseline ER and ER-CBA on Split CIFAR-10 ($M = 0.5k$), Split CIFAR-100 ($M = 5k$), and Split Tiny-ImageNet ($M = 5k$), respectively. It shows that the CBA module can significantly improve the performance of previous tasks and reduce forgetting of the model. Additionally, our CBA can effectively trade off the accuracy of old and new tasks, which indicates that CBA can prevent the model from being disturbed by distribution shifts (caused by incoming new tasks).

Method	Details	Split CIFAR-10			
		$M = 0.2k$		$M = 0.5k$	
		ACC ↑	FM ↓	ACC ↑	FM ↓
ER	-	35.21	50.28	42.32	40.80
ER-CBA	$\mathcal{L}^{trn}$	31.95	51.76	36.66	44.55
ER-CBA	$\mathcal{L}^{trn} + \mathcal{L}^{buf}$	33.35	52.98	44.58	35.95
ER-CBA	64 hidden units	33.50	44.44	37.48	37.82
ER-CBA	1024 hidden units	**38.46**	**42.73**	**46.85**	**24.86**
ER-CBA	4 layers MLP	37.57	41.81	45.23	27.91
ER-CBA	ours	37.27	41.39	45.41	29.36

Table 5: The results of ablation study.

Question 2: How does the predictive distribution with and without the CBA module compare? To investigate this question, we calculate the prediction distribution of each task after training on the final task of Split CIFAR-100 ($M = 5k$) in Fig. 3 (a), and show the corresponding confusion matrix of ER and ER-CBA in Fig. 3 (b) and (c), respectively. Obviously, the baseline ER tends to predict test samples as new classes with high probability, which leads to significantly higher accuracy for new tasks than old tasks within ER. In contrast, our method can suppress the prediction probability of new tasks and automatically adapt to the changing distribution. Therefore, our method treats test samples from all seen classes more equally and achieves better overall performance, which reveals how CBA helps the model deal with the data distribution shift problem.

Question 3: How effective are the various components of the proposed model? We conduct a comprehensive ablation study to investigate the contribution of each key component of the proposed algorithm and the results on Split CIFAR-10 are shown in Table 5. Firstly, we verify the effectiveness of the bi-level optimization part. We replace the bi-level optimization procedure with an end-to-end training

parallel, that is, optimize the parameters θ of the classification model and the CBA parameter ω together. The bi-level optimization and the end-to-end training correspond to 1) have the rehearsal training loss Eq. (3) only; 2) sum the inner-loop training loss Eq. (3) and the outer-loop training loss Eq. (4). It can be noted that the end-to-end training cannot help the model alleviate the distribution shift and even hurt the performance of baseline ER (the 2nd and 3rd rows in Table 5), which demonstrate that the bi-level optimization is essential for our model training.

We also explore the effect of different MLP architectures in the CBA module where we use a two-layer MLP with 256 hidden units. Specifically, we test the performance of 1) a two-layer MLP with 64 and 1024 hidden units; 2) a four-layer MLP with 256 hidden units. As shown in the 4-6th rows of Table 5, the representation ability of the MLP with only 64 hidden units is not enough to fit the bias, which leads to lower performance than the baseline. And the MLP with 1024 hidden units performs even better than 256 hidden units, suggesting that the stronger CBA architecture might perform better even though it will introduce more parameters and slow down the training process. Since the performance of the complicated four-layer MLP with 256 hidden units is almost the same as the two-layer MLP, we choose the two-layer MLP with 256 hidden units as the base architecture of our CBA.

5. Conclusion

In this paper, we tackle online CL from the perspective of modeling the posterior distribution shift arising from time-varying data streams. We propose a Continual Bias Adaptor (CBA) module, which aims to augment the original classifier network to adapt to catastrophic distribution change during training, such that the classifier network is capable of learning a stable consolidation of the previously learned knowledge across all tasks. In the inference stage, CBA can be removed and requires no additional calculation burden or memory overhead. The effectiveness of the proposed method is demonstrated both theoretically and empirically. In theory, we explain the reason why our method can significantly alleviate catastrophic forgetting in online CL. Empirical results show that the proposed algorithm can be applied to many rehearsal-based baselines and consistently improve their performance, which verifies that our method can effectively consolidate the learned knowledge.

Acknowledgement

This research was supported by the National Key R&D Program of China (2020YFA0713900), the China NSFC projects under contracts 61906144, 61721002, 12226004, and the Macao Science and Technology Development Fund under Grant 061/2020/A2.

References

[1] Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 844–853, 2021.

[2] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 11849–11860. Curran Associates, Inc., 2019.

[3] 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.

[4] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations*, 2022.

[5] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8218–8227, 2021.

[6] Jonathan F Bard. *Practical Bilevel Optimization: Algorithms and Applications*, volume 30. Springer Science & Business Media, 2013.

[7] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, pages 15920–15930, 2020.

[8] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. *arXiv preprint arXiv:2104.05025*, 2021.

[9] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision*, pages 233–248, 2018.

[10] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co^{2}l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9516–9525, 2021.

[11] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision*, pages 532–547, 2018.

[12] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[13] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International*

Conference on Machine Learning, pages 1952–1961. PMLR, 2020.

[14] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[16] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[17] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):1–37, 2014.

[18] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1998.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[20] 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.

[21] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

[22] Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 16071–16080, 2022.

[23] Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. In *International Conference on Learning Representations*, 2022.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[25] Mingchen Li, Xuechen Zhang, Christos Thrampoulidis, Jiasi Chen, and Samet Oymak. Autobalance: Optimized loss functions for imbalanced data. *Advances in Neural Information Processing Systems*, 34:3163–3177, 2021.

[26] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[27] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[28] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

[29] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[30] 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*, pages 3589–3599, 2021.

[31] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314*, 2020.

[32] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010.

[33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[34] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[35] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[36] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343. PMLR, 2018.

[37] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[38] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in Neural Information Processing Systems*, 32, 2019.

[39] Christian Simon, Masoud Faraki, Yi-Hsuan Tsai, Xiang Yu, Samuel Schulter, Yumin Suh, Mehrtash Harandi, and Manmohan Chandraker. On generalizing beyond domains in cross-domain continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9265–9274, 2022.

[40] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[41] Quanziang Wang, Renzhen Wang, Yuexiang Li, Dong Wei, Kai Ma, Yefeng Zheng, and Deyu Meng. Relational experience replay: Continual learning by adaptively tuning task-wise relationship. *arXiv preprint arXiv:2112.15402*, 2021.

[42] Renzhen Wang, Kaiqin Hu, Yanwen Zhu, Jun Shu, Qian Zhao, and Deyu Meng. Meta feature modulator for long-tailed recognition. *arXiv preprint arXiv:2008.03428*, 2020.

[43] Renzhen Wang, Xixi Jia, Quanziang Wang, Yichen Wu, and Deyu Meng. Imbalanced semi-supervised learning with bias

adaptive classifier. In *The Eleventh International Conference on Learning Representations*, 2022.

[44] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.

[45] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. In *International Conference on Learning Representations*, 2022.

[46] Yaqian Zhang, Bernhard Pfahringer, Eibe Frank, Albert Bifet, Nick Jin Sean Lim, and Alvin Jia. A simple but strong baseline for online continual learning: Repeated augmented rehearsal. In *Advances in Neural Information Processing Systems*, 2022.

[47] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13208–13217, 2020.

[48] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Deep class-incremental learning: A survey. *arXiv preprint arXiv:2302.03648*, 2023.

[49] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, 34:14306–14318, 2021.