ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230448

R-STAR: Robust Self-Taught Task-Wise Reweighting for Rehearsal-Based Class Incremental Learning

Yutian Luo[a], Yizhao Gao[a], Haoran Wu[b], Ruitao Ma[b] and Zhiwu Lu[a;*]

[a]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
[b]China Unicom Research Institute, Beijing, China

Abstract. Class incremental learning (CIL) requires a model to learn the knowledge of new classes without overwriting that of old classes. The main challenge thus lies in catastrophic forgetting. Among all advances in addressing this challenge, rehearsal-based methods are the most widely-used due to their convenience and effectiveness. However, the (classification) scores bias between the old and new classes, known as the main cause of catastrophic forgetting for rehearsal-based methods, is still not fully addressed. Although some recent strategies are proposed to reduce the scores bias, they either take extra training time or sacrifice too much performance on the current task. In this paper, we propose a novel Robust Self-Taught Task-Wise Reweighting (R-STAR) method, which can act as a flexible and key component for improving existing rehearsal-based methods. Concretely, on top of the standard training process, it measures the forgetting degree of the model over the augmented buffer (for robust evaluation) on each task. Further, following the self-taught paradigm, it directly activates the task-wise forgetting degree into a reweighting ratio for scores bias reduction during the inference stage. Extensive experiments show that our R-STAR can improve most rehearsal-based methods with remarkable margins, but with (almost) no extra training cost or excessive performance sacrifice on the new task. Moreover, it also shows its advantages over existing scores bias correction strategies.

1 Introduction

Continual learning aims at enabling a model to learn in the way of human beings on continuously arriving data: new knowledge is continuously learned while no old knowledge is overwritten. Among all continual learning settings, the class incremental learning (CIL) setting meets most real application scenarios and is the most challenging one. Under the CIL setting, the model is required to learn about sequentially arriving tasks, where the data in each arriving task belong to unseen-before classes. Thus, the key challenge of the CIL lies in the catastrophic forgetting of old classes.

Over the past few years, various approaches have been proposed to solve this challenge. Among all sorts of existing approaches, rehearsal-based approaches are the most widely-used due to their convenience and effectiveness. Concretely, rehearsal-based approaches attempt to preserve the old knowledge by a continuously updated memory buffer where limited samples are selected from all previous tasks and then available in the new task. Among recent advances of rehearsal-based methods, most of them focus on designing a proper updating strategy of the buffer [26, 27, 19, 6] and exploiting the buffer data with extra regularization constraints [26, 7, 6, 24, 6]. Although these efforts indeed contribute to mitigating catastrophic forgetting, the main cause of this challenge for rehearsal-based CIL is still not fully addressed.

As some recent works [31, 1, 6] point out, the main cause of catastrophic forgetting for rehearsal-based CIL is the classification score bias. Specifically, the bias is due to data imbalance between old data in the memory buffer and new data in the current task. When the model learns on such unbalanced data, it tends to give higher classification scores for new classes. Regarding this, recent methods [31, 1, 6] propose new strategies to solve the scores bias. [31] train a linear model on a balanced dataset after each task and use it to correct the classification score bias in the last fully connected layer. Although the scores bias between all past classes and new classes is reduced, this method spends much more training time and its overall improvement is also limited. [1] propose to adopt the Separated-Softmax output layer and the task-wise knowledge distillation strategy. [8] propose a separated cross-entropy loss. These two methods aim at blocking the score gradients between the old and new classes. Despite their remarkable achievements in improving the scores of past classes, the data of the new task are only learned by classifying within new classes, which leads to a distinct degradation of the new task performance compared with other methods. Therefore, the overall improvements of these two methods are also affected.

In this paper, we thus devise a novel Robust Self-Taught Task-Wise Reweighting (R-STAR) method, which can act as a flexible and key component for improving existing rehearsal-based methods. The core idea of our R-STAR is to make the model aware of the scores bias and then reduce it with a task-wise reweighting ratio in a single self-taught step. Concretely, we first measure the scores bias with the task-wise forgetting degree evaluated on the memory buffer in the training stage. In this stage, an augmentation strategy is adopted to increase the robustness of scores bias evaluation when our R-STAR is combined with different rehearsal-based approaches. In the inference stage, we design an activation function to activate the recorded forgetting degree into a task-wise reweighting ratio. Thus, the model could teach itself with the guidance of the ratio and then modify the biased scores. Different from the original self-taught paradigm, our R-STAR takes only one self-taught step in the inference stage, which can help avoid the overfitting problem raised from repeated finetuning in the original self-taught paradigm. As a result, this choice is empirically found to further enhance the robustness of our R-STAR. To show the effectiveness of R-STAR, we conduct ex-

* Corresponding Author. Email: luzhiwu@ruc.edu.cn

tensive experiments on three commonly-used CIL benchmarks. The obtained results show that: (1) Our R-STAR is indeed effective for reducing score bias and consistently improves the performance of most rehearsal-based CIL methods. (2) Compared with other scores bias correction strategies, our R-STAR yields almost no extra training time or excessive performance sacrifice on the new task but could either further improve their performance or outperform them when combined with the same baseline method.

Overall, our main contributions are four-fold: (1) We devise a novel Robust Self-Taught Task-Wise Reweighting (R-STAR) method, which can act as a flexible and key component to reduce the scores bias and further improve existing rehearsal-based methods under the CIL setting. (2) We introduce a novel robust self-taught paradigm to reduce the scores bias. Without the overfitting problem in the original self-taught paradigm, it makes the model aware of the scores bias and then reduces this bias in a one-step self-taught way. (3) Our R-STAR outperforms existing scores bias correction strategies while leading to almost no extra training time or excessive performance sacrifice on the new task. (4) Extensive experiment results on four commonly-used CIL benchmark datasets show the effectiveness of our R-STAR.

2 Related Work

Rehearsal-Based Approaches. Among all approaches for continual learning, the rehearsal-based approaches are the most widely used due to their convenience and effectiveness. In order to tackle the catastrophic forgetting issue, rehearsal-based approaches [27, 11, 9, 2, 19, 31, 22, 24, 26, 7, 1, 5, 8, 6, 10, 4] attempt to preserve the old knowledge from all previous tasks by a memory buffer and replay it when the model learns on a new task. [27] propose to save a small subset of old data and replay it in new tasks. They also provide several basic memory update strategies which are widely adopted in later works. [19] propose inter-class separation which encourages a large margin to separate the old and new classes for a more harmonized classifier. [7] propose to match the logits saved throughout the optimization trajectory and further reinforce the regularization by aligning logits with their corresponding labels. [24] utilize the rehearsal data to train the pre-allocated classifier heads for better overall performance. In the very latest work, [6] extends the previous work [7] by modifying the past-future heads of saved logits and restricting the activation on the future and past classifier heads. This work combines the advantages of almost all the above-mentioned methods and makes remarkable improvements to [7] and performs as a very strong benchmark. In addition to these typical rehearsal-based methods, the pseudo-rehearsal methods [28, 14, 23, 32, 21, 20, 17] are proposed to avoid storing original images and privacy issues. This sort of methods usually adopt modified generated network to produce pseudo images and features [23, 32] for reviewing old knowledge. However, these works pay little attention to the scores bias, and our R-STAR thus could act as an effective component for them to reduce the scores bias and make further improvements.

Scores Bias Correction Methods. For the rehearsal-based CIL approaches, the main cause of catastrophic forgetting is classification score bias. As recent works [31, 1, 6] point out, the bias is due to data imbalance between the data in the rehearsal buffer and those in the current task. To deal with this problem, [19] propose to reduce the bias of norm weights in classifier between old classes and new classes by replacing the common softmax layer with cosine normalized layer. [31] construct a balanced dataset with the novel and buffer data. After each task, they retrain a linear model on the balanced batch and then adopt the linear model in the next task to correct the bias in the last fully-connected layer. Although this approach contributes to the scores bias reduction, it takes much extra training time and achieves limited performance. Recently, [1] propose a Separated-Softmax output layer and replace the general knowledge distillation loss with a task-wise distillation loss to block the score gradients between the old and new classes. Similarly, [8] adopt a separated cross-entropy loss to prevent the effects of the gradient on the past classes. These two methods perform well in preserving the learned knowledge. However, since most of the data in the current task are only learned by classification within new classes, the model actually sacrifices too much on the current task knowledge and has much lower performance compared with other rehearsal-based methods like [7]. Thus, their overall improvements are affected.

Self-Taught Learning Paradigm. The original concept of the self-taught learning paradigm is proposed by [25]. For better representation, they first train an auto-encoder with unlabelled data and then finetune the classifier layer of the auto-encoder repeatedly on limited labeled data in order that the bias between unlabelled data and labeled data is reduced. In this original self-taught paradigm, the model is required to teach itself to modify the biased class representation with few labeled data. Similar but not identical, in our proposed Robust Self-Taught Task Reweighting method the model is required to teach itself to reduce the scores bias among classes of all tasks with saved buffer data. Although they all utilize partial data for further modification, the key difference between our robust self-taught paradigm and the original self-taught paradigm is that we only make a one-step reweighting in the reference stage while the original self-taught needs to finetune repeatedly with the labeled data. This means that we could largely avoid the overfitting problem in the original self-taught learning paradigm, which actually enhances the robustness of our proposed R-STAR.

3 Methodology

3.1 Preliminary

We first give the formal problem formulation of class incremental learning. Supposing there are n sequentially arriving tasks, we assume $\{D_t\}_{t=1}^{n}$ as the data set of the n tasks, where D_t indicates the data for task T_t. All samples in D_t are denoted as $\{(x_t^i, y_t^i)\}_{i=1}^{|D|}$, where x_t^i denotes the i-th image in D_t, y_t^i denotes the ground truth label of the i-th image in D_t, and $|D|$ denotes the number of samples in each task. Consistent with $\{D_t\}_{t=1}^{n}$, we denote all the classes in n tasks as $\{C_t\}_{t=1}^{n}$, where C_t is composed of the classes contained in D_t and we use $|C|$ to denote the number of classes in each task. In the CIL problem, each C_t includes disjoint classes for the other $(n-1)$ class sets. In class incremental scenarios, the model is required to learn sequentially on these n tasks and good grasp of all the classes in $\{C_t\}_{t=1}^{n}$. Therefore, the key issue of the CIL problem is to alleviate catastrophic forgetting of the model on old classes.

Among all the approaches to dealing with this issue, the rehearsal-based approaches are the most widely-used due to their simplicity and effectiveness. In this sort of method, a memory buffer B of a fixed size $|B|$ is constructed to restore and replay samples of old tasks when the model learns on a new task. Concretely, in task $T_t (t \geq 1)$, buffer B is updated to include data of new classes in C_t. Meanwhile, in each new task $T_t (t \geq 2)$, buffer B is available to replay the samples of previous $(t-1)$ tasks for reviewing the old knowledge. Since B is updated in each task, we use $\{B_t\}_{t=1}^{n}$ to denote the updating memory buffer in n sequential tasks, where B_t denotes the memory

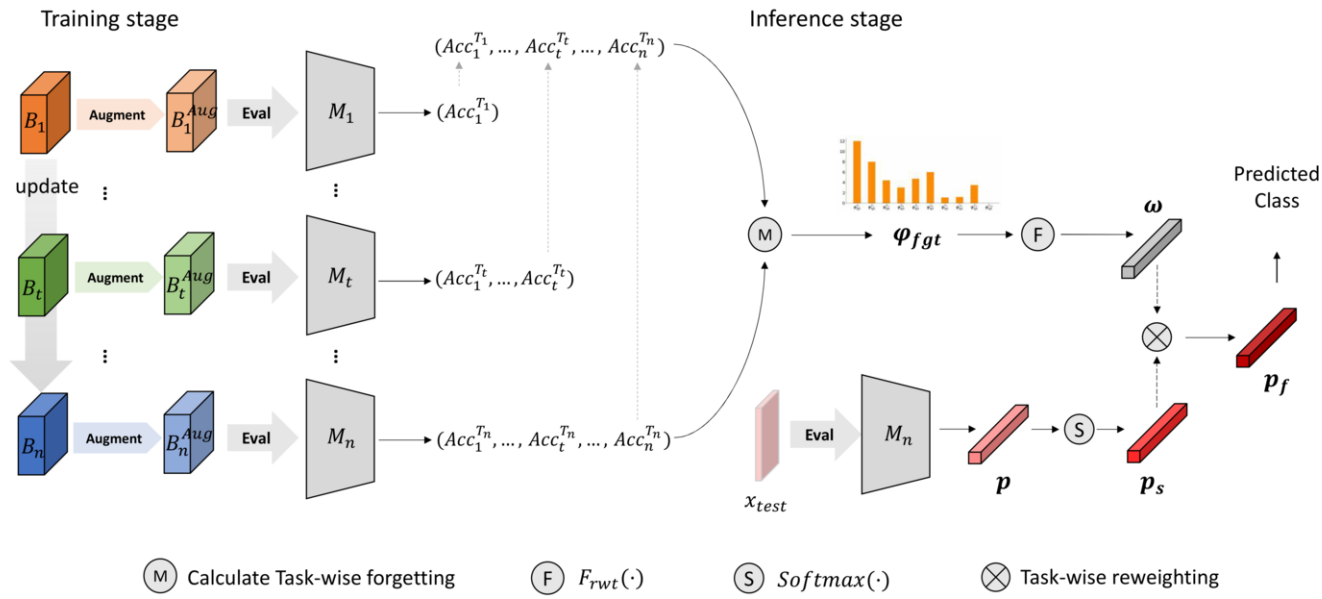

Figure 1. Overview of the proposed R-STAR method. In the training stage, the model is evaluated at the end of task T_t $(1 \leq t \leq n)$ on the augmented buffer B_t^{Aug} for each seen task to record their corresponding accuracy. The accuracy $Acc_t^{T_t}$ of task T_t is selected to form the initial accuracy $\boldsymbol{Acc^{ini}}$. In the reference stage, the difference between the initial accuracy $\boldsymbol{Acc^{ini}}$ and the final accuracy $\boldsymbol{Acc_n}$ on B_n^{Aug} is calculated to obtain the forgetting degree $\boldsymbol{\varphi_{fgt}}$. It is then activated by the designed function $F_{rwt}(\cdot)$ to calculate the task-wise reweighting ratio $\boldsymbol{\omega}$. Finally, it is used to reweight the softmax-normalized output score of the test sample x_{test} to reduce the scores bias for a better classification prediction.

buffer acquired at the end of task T_t.

Although recent rehearsal-based works do make progress in dealing with catastrophic forgetting, the scores bias between old and new classes, known as the main cause of the catastrophic forgetting for the rehearsal-based methods is still not fully addressed. Such scores bias is caused by unbalanced samples between those in the current buffer B_{t-1} and new task data D_t. Trained with such unbalanced data, the model tends to give higher prediction scores for classes in the new task and much lower scores for those classes in the previous tasks, which leads to limited overall performance.

3.2 Self-Taught Task-Wise Reweighting

In order to remedy the scores bias, we decide to introduce a robust self-taught paradigm for help. In this paradigm, the model is expected to be aware of the scores bias and reduced it before the model gives the final prediction scores. To achieve this goal, we first need to find a proper evaluation means to measure the bias degree for all tasks during the training stage so that it could be learned for later modification. Since the task that is more severely affected by scores bias tends to have a more severe performance degradation, we suppose it is reasonable to measure the scores bias degree for each task with the task-wise forgetting degree.

In order to evaluate the forgetting degree of the model on previous tasks, the direct way is to evaluate the model on part of the data of previous tasks. Given the fact that the memory buffer has been well-designed and saved in each rehearsal-based method, we decide to utilize the memory buffer to measure the forgetting degree of the model for each task. Since the memory buffer B_t $(1 \leq t \leq n)$ contains samples from all previous t tasks, it could be written in a task-wise split way, which is represented as:

$$B_t = \{B_t^{T_1}, B_t^{T_2}, ..., B_t^{T_t}\}, \tag{1}$$

where $B_t^{T_i}$ denotes the data saved from task T_i $(1 \leq i \leq t)$ in B_t. Once the performance of the model on $B_t^{T_i}$ has a decline compared with its previous performance on $B_i^{T_i}$ $(i < t)$, we could infer that the model also suffers a degradation on the corresponding task T_i. Therefore, we could measure the task-wise forgetting of the model with the accuracy evaluated on the buffer along the whole training.

As shown in Figure 1, assuming the model acquired at the end of task T_t as M_t, we evaluate M_t on each $B_t^{T_i}$ $(1 \leq i \leq t)$ after training on task T_t. We use $\boldsymbol{Acc_t}$ to denote the accuracy of model M_t on B_t, and it is written as:

$$\boldsymbol{Acc_t} = (Acc_t^{T_1}, Acc_t^{T_2}, ..., Acc_t^{T_t}), \tag{2}$$

where $Acc_t^{T_i}$ denotes the accuracy of model M_t on $B_t^{T_i}$ $(1 \leq i \leq t)$. Since the model M_t has just been trained on task T_t, $Acc_t^{T_t}$ actually indicates the initial accuracy of the model on task T_i. Therefore, we collect all the initial accuracy $Acc_t^{T_t}$ from task T_1 to task T_n and define the final initial accuracy over n tasks as:

$$\boldsymbol{Acc^{ini}} = (Acc_1^{T_1}, Acc_2^{T_2}, ..., Acc_n^{T_n}). \tag{3}$$

Given the initial accuracy $\boldsymbol{Acc^{ini}}$ on buffer for each task, we measure the forgetting degree of the model on each task by calculating the drop from Acc^{ini} to Acc_n. $\boldsymbol{\varphi_{fgt}}$ is defined to denote the task-wise forgetting degree and is calculated by:

$$\begin{aligned} \boldsymbol{\varphi_{fgt}} &= \max(\mathbf{0},\ (\boldsymbol{Acc^{ini}} - \boldsymbol{Acc_n})) \\ &= (\varphi_{fgt}^{T_1}, \varphi_{fgt}^{T_2}, ..., \varphi_{fgt}^{T_n}), \end{aligned} \tag{4}$$

where $\varphi_{fgt}^{T_i}$ $(1 \leq i \leq t)$ denotes the task-wise forgetting degree of the final model on task T_i. We set the maximum threshold as zero for $\varphi_{fgt}^{T_i}$ to avoid the effect of overfitting.

With the task-wise forgetting degree acquired, the next problem is how could the model learn from it to reduce the scores bias for each

task. Since the result of the final prediction is determined by the maximum classification score in the output logits, task-wisely reweighting the output logits could contribute to the reduction of score bias for all the tasks. Regarding this, we devise an activation function $F_{rwt}(\cdot)$ to calculate a reweighting ratio $\boldsymbol{\omega}$ with the task-wise forgetting result $\boldsymbol{\varphi_{fgt}}$. Formally, the activation function is written as:

$$\omega^{T_i} = F_{rwt}(\varphi_{fgt}^{T_i}) = \begin{cases} e^{(k+\varphi_{fgt}^{T_i})}, & \text{if } \varphi_{fgt}^{T_i} \neq 0 \\ 1, & \text{otherwise} \end{cases}, \tag{5}$$

where ω^{T_i} denotes the value of the reweighting ratio for task T_i, k is the hyperparameter to adapt the compensation ratios for different rehearsal-based methods. The selection range and the effect of k are discussed in ablation experiment section. Note that if $\varphi_{fgt}^{T_i}$ is zero, we keep the original scores of classes in task T_i by setting ω^{T_i} to 1. Formally, $\boldsymbol{\omega}$ is written in the form of:

$$\boldsymbol{\omega} = (\omega^{T_1}, \omega^{T_2}, ..., \omega^{T_n}), \tag{6}$$

With the task-wise reweighting ratio, the model could teach itself to modify the biased scores. Assuming output logits as $\boldsymbol{p} = (p^0, p^1, ..., p^{n|C|-1})$, since the logits scores in $\boldsymbol{p}$ may be negative values, directly reweight $\boldsymbol{p}$ with $\boldsymbol{\omega}$ could not correctly modify the scores of classes in each task. Therefore, a softmax layer is added after the output layer to obtain normalized output logits $\boldsymbol{p_s} = (p_s^0, p_s^1, ..., p_s^{n|C|-1})$ in which score p_s^i $(0 \leq i \leq n|C|-1)$ for class i is calculated by:

$$p_s^i = \frac{e^{p^i}}{\sum_{j=0}^{n|C|-1} e^{p^j}}. \tag{7}$$

Finally, the normalized output logits $\boldsymbol{p}_s$ is reweighted by $\boldsymbol{\omega}$ to get the final output logits $\boldsymbol{p}_f$:

$$\boldsymbol{p}_f = (\omega^{T_1} p_s^0, \omega^{T_1} p_s^1, ..., \omega^{T_n} p_s^{n|C|-1}). \tag{8}$$

In this way, task-wise reweighting is applied to the final prediction score $\boldsymbol{p_s}$ in the inference stage. Since our R-STAR only has one self-taught step in the inference stage, it could be directly applied to most of the rehearsal-based approaches without any change to their initial idea and effectively help them alleviate the scores bias problem.

3.3 Augmented Buffer for Reweighting

For some rehearsal-based methods like RPC[24], LUCIR [19], and Der [7], R-STAR could measure the task-wise forgetting degree $\boldsymbol{\varphi_{fgt}}$ of the model by directly evaluating M_n on B_n and calculating the task-wise drop from $\boldsymbol{Acc^{ini}}$ to $\boldsymbol{Acc_n}$. However, for ER [27], Derpp [7], and X-Der [6], the samples saved in the buffer from previous tasks are more fully utilized with extra regularization restrictions and retraining in new tasks. Therefore, the model may overfit the buffer data as training goes on, even if it suffers severe forgetting on previous tasks. This leads to an unexpected result as concerned in Eq.(4): most of the values in $\boldsymbol{\varphi}_{fgt}$ become zero, which hinders our task-wise forgetting measurement.

In order to alleviate this problem, we add an augmentation process to B_t and get B_t^{Aug}. Note that B_t^{Aug} is only used for the task-wise forgetting degree measurement and the original B_t is used for training. In order to choose an effective augmentation strategy, we try several proposed strategies (like AutoAugment [15], RandAugment [16], and the augmentation strategy in SimCLR [13]) and empirically adopt the augmentation strategy in SimCLR [13] as our augmentation strategy. We compare the results of $\boldsymbol{\varphi}_{fgt}$ between measuring with no-augmented buffer and augmented buffer. The visualized

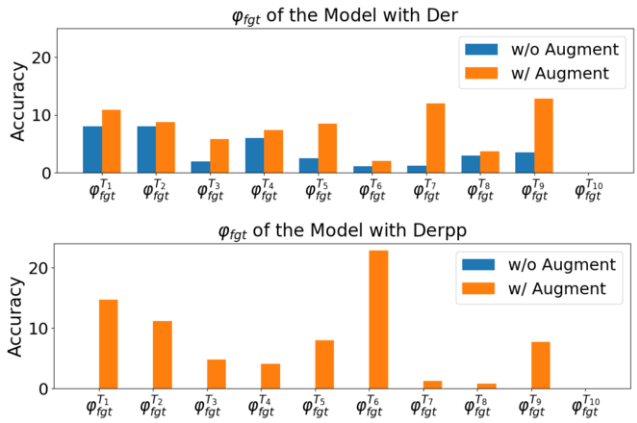

Figure 2. Comparative results of $\boldsymbol{\varphi_{fgt}}$ across 10 tasks on Split Cifar-100 between reweighting without augmentation ("w/o Augment") and reweighting with augmentation ("w/ Augment"). We adopt two typical rehearsal-based approaches Der (top) and Derpp (bottom) to show the effectiveness of the augmentation strategy to measure task-wise forgetting.

results in Figure 2 show that: (1) The augmentation strategy effectively alleviates the overfitting problem on the buffer data when measuring the $\boldsymbol{\varphi_{fgt}}$ for those approaches with full use of the buffer. (2) The augmentation strategy also makes improvements to the measurement of $\boldsymbol{\varphi_{fgt}}$ for those methods with light use of the buffer. These observations further prove that our augmentation strategy indeed enhances the robustness of our R-STAR and makes it a robust approach.

4 Experiments

4.1 Datasets and Evaluation Metrics

Following previous works [6, 7, 12], we evaluate our proposed method on four commonly-used benchmark datasets:

Split CIFAR-100 [33] contains totally 100 classes. It splits the CIFAR-100 dataset into 10 consecutive tasks and each task contains equally 10 classes. In each task, 500 and 100 images in the size of 32x32 are prepared for training and testing respectively.

Split miniImageNet [12] splits miniImageNet [29], a 100-class subset of the popular ImageNet dataset into 20 sequential tasks. Each task includes 5 different classes and all the images are in the size 84x84 with 3 channels.

Split tinyImageNet splits Tiny ImageNet [3], a 200-class subset of the popular ImageNet dataset, into 10 classification tasks with 20 classes in each task. Each image is in the size 64x64 with 3 channels.

Split CUB-200 derives from the Caltech-UCSD Birds-200-2011 [30] as a 200 classes subset. It is organized in the sequence of 10 tasks, each including 20 classes. Due to the larger image size of 224x224, it is a more challenging benchmark compared with the others and is adopted in many well-known works [12, 5, 34].

Metrics. We first report the performance of Final Average Accuracy (FAA) across all tasks which is also used in previous works [6, 5]. Assuming a_i^t as the top-1 accuracy of model M_t on the i-th task after training on task T_t, the FAA is formally written as:

$$\text{FAA} = \frac{1}{n}\sum_{i=1}^{n} a_i^n, \tag{9}$$

where n denotes the total number of tasks. FAA reflects the overall performance of the model on each whole dataset.

Then, we also follow previous work [6] to adopt Final Forgetting rate (FF) as another metric which indicates the forgetting on previous

Table 1. Comparative results on Final Average Accuracy (FAA) and Final Forgetting (FF) by combining R-STAR with various rehearsal-based methods. We show the results in the form of "basic performance + improvement" and also give the average improvement ("Avg. Improv.") across three benchmark datasets. The positive improvement for FAA is shown in red, while the decrease in FF is shown in blue.

Methods	Split Cifar100		Split miniImagenet		Split tinyImagenet		Split CUB-200		Avg. Improv.
Upper bound	70.44		53.55		59.99		78.54		-
Lower bound	9.43		4.51		7.92		8.56		-
Buffer size	500	2,000	2,000	5,000	500	2,000	500	2,000	-
	Final Average Accuracy								-
ER [27]	21.70 +2.75	38.32 +3.04	14.57 +2.03	24.92 +2.00	9.99 +3.05	17.82 +3.55	45.01 +3.89	61.10 +2.53	+2.86
BIC [31]	36.12 +1.78	47.48 +3.53	12.96 +1.73	14.49 +1.96	12.28 +2.37	14.81 +1.90	49.41 +3.46	55.80 +3.09	+2.48
RPC [24]	23.08 +2.70	39.79 +2.79	15.60 +1.51	26.13 +1.25	10.30 +1.02	18.11 +2.37	50.30 +4.00	65.21 +1.39	+2.13
LUCIR [19]	39.19 +2.37	48.61 +2.33	14.97 +1.08	17.06 +1.18	29.55 +1.87	31.40 +0.63	51.43 +1.52	65.01 +1.09	+1.51
Der [7]	36.60 +1.66	49.88 +2.07	22.96 +1.53	30.12 +1.57	17.75 +1.83	29.68 +1.73	51.61 +1.65	64.12 +2.03	+1.76
Derpp [7]	37.85 +1.17	51.69 +2.23	23.44 +1.99	30.83 +2.04	19.78 +1.88	30.97 +1.76	52.48 +1.68	66.05 +0.93	+1.71
X-Der [6]	48.08 +0.39	57.58 +0.02	28.19 +0.68	32.32 +0.67	29.77 +1.03	40.76 +0.85	59.23 +0.71	68.35 +0.67	+ 0.63
	Final Forgetting								-
ER [27]	83.70 -3.57	62.07 -3.83	86.48 -2.50	70.14 -3.00	97.36 -1.49	86.02 -3.43	51.71 -5.31	27.70 -3.54	-3.33
BIC [31]	59.16 -4.63	47.68 -4.36	87.37 -2.45	84.59 -3.30	90.62 -3.17	87.60 -3.19	44.50 -5.80	34.56 -4.62	-3.94
RPC [24]	82.42 -3.17	60.42 -3.68	83.92 -2.65	66.46 -2.32	96.64 -1.53	84.10 -4.08	42.94 -5.60	15.69 -2.18	-3.15
LUCIR [19]	64.59 -5.34	43.79 -3.94	75.91 -2.65	53.42 -1.92	49.67 -3.81	33.49 -0.67	29.42 -1.93	12.07 -0.78	-2.63
Der [7]	65.05 -4.80	41.70 -3.57	70.79 -3.76	59.61 -1.98	88.27 -3.02	64.93 -4.05	40.17 -2.41	17.59 -0.92	-3.06
Derpp [7]	61.39 -3.56	41.01 -4.25	69.54 -2.81	58.66 -2.70	84.57 -4.18	60.37 -3.94	39.96 -2.35	15.59 -1.02	-3.11
X-Der [6]	33.31 -1.69	10.54-0.32	44.20 -1.73	26.02 -1.50	62.83 -1.41	35.80 -0.51	10.34 -1.36	5.57 -0.98	-1.19

$(n-1)$ tasks of the final model. The FF is formally written as:

$$\text{FF} = \frac{1}{n-1}\sum_{j=1}^{n-1} f_j, \quad \text{s.t.} \quad f_j = \max_{l\in\{1,\ldots,n-1\}} a_j^l - a_j^{n-1}. \quad (10)$$

FF measures the average degradation for all tasks, where the degradation refers to the gap between the best performance and the final performance of the model during the whole training for each task.

4.2 Implementation Details

In order to evaluate the real merits of our proposed method when it is combined with existing CIL methods, we follow the settings in previous well-known works [7, 6] to provide reasonable basic performances of those CIL methods. For network architecture, we adopt ResNet18 [18] for Split CIFAR-100 and Split tinyImageNet, EfficientNet-B2 for Split miniImageNet, and Resnet50 [18] for Split CUB-200. Most the models are trained from scratch except the one for Split CUB-200 which is pretrained on Imagenet like previous works [12, 5] does. For memory buffer size, we also obey the initial settings in [6] for Split CIFAR-100 and Split miniImageNet, and set similar buffer sizes (i.e. 500 and 2000) for Split tinyImageNet and Split CUB-200. For training hyperparameters, we reproduce almost all the methods on Split Cifar100, Split miniImagenet, and Split tiny Imagenet with the reported hyperparameters in [7, 6] except SSIL [1] which has not been reproduced under these datasets and training protocols. The training hyperparameters for all the methods on Split CUB-200 are searched by ourselves and will be released with the code. Note that we avoid taking results directly from previous works [7, 6] and instead run all experiments 10 times with the released code for each method and adopt the average results as the final results. All the experiments are conducted on TITAN RTX 3090 GPUs. We will make our code public soon.

4.3 Main Results

For clear comparisons, we first provide the best and the poorest results achieved by training jointly on all data (**Upper-bound**) and sequentially on each task with no remedy to catastrophic forgetting (**Lower-bound**), respectively. Moreover, we adopt several well-known and state-of-the-art rehearsal-based approaches to combine or compare with our proposed R-STAR, including ER [27], LUCIR [19], BIC [31], RPC [24], Der [7], Derpp [7], SSIL [1], ER-ACE [8], and X-Der [6]. All these approaches, depending on whether they adopt strategies for score bias reduction and the bias reduction results they achieve, could be divided into the following three groups.

Approaches with no extra strategy for scores bias reduction: The approaches ER [27], RPC [24], Der [7], Derpp [7] focus on proposing an effective way to construct and utilize memory buffer to preserve the old knowledge. Since no extra strategy for the scores bias reduction is adopted by these methods, the improvements made by our R-STAR are remarkable when it performs as a component of these methods. It can be observed from Table 1 that R-STAR improves the FAA performance of all these methods by at least 1.71% higher accuracy and reduce the FF by at least 3.11% on average across the four datasets. For simple methods like ER [27], R-STAR even makes over 3% improvements for it on the challenging Split tinyImagenet dataset and reduce FF for it by more than 5% on the challenging Split CUB-200 dataset. All these observations provide evidence that R-STAR plays an important role in score bias reduction and greatly helps these methods reach better performance (i.e. higher FAA and lower FF).

Approaches with weak strategies for scores bias reduction: We also combine our R-STAR with BIC [31], LUCIR [19] and X-Der [6], which adopt relatively weak strategies. For these three methods, the scores bias could still be observed under most of the settings of four benchmark datasets and we find R-STAR could make further improvements to their performance. From the results in Table 1, we find that our R-STAR improves the FAA performance of BIC, LUCIR and X-Der by 2.48%, 1.51% and 0.62% on average across four datasets, respectively. Meanwhile, our R-STAR also helps to reduce the FF performance of BIC, LUCIR and X-Der by 3.94%, 2.63% and 1.19% on average across four datasets, respectively. For BIC and LUCIR, such improvement is as remarkable as that of the approaches with no bias correction strategy, which proves the advantages of the task-wise reweighting strategy over its original strategy. For X-Der,

Table 2. Comparative results (FAA) among various scores bias correction methods. For fair comparison, we choose to compare the improvements obtained by combining Derpp with different scores bias correction strategies.

Methods	Split Cifar100		Split miniImagenet		Split tinyImagenet		Split CUB-200		Avg. Improv
Buffer size	500	2000	2000	5000	500	2000	500	2000	-
SSIL [1]	38.21	49.07	23.18	30.89	20.86	31.33	51.12	66.03	-
ER-ACE [8]	38.75	49.93	22.60	29.93	20.22	31.26	50.04	66.21	-
Derpp [7]	37.85	51.69	23.44	30.83	19.78	30.97	52.48	66.05	-
Derpp + SSIL	38.31	52.05	24.17	31.37	21.11	31.83	52.77	66.07	+0.57
Derpp + ER-ACE	38.43	51.82	24.03	31.32	21.02	31.68	53.63	66.18	+0.62
Derpp + R-STAR (ours)	**39.02**	**53.92**	**25.43**	**32.87**	**21.66**	**32.73**	**54.16**	**66.98**	+1.71

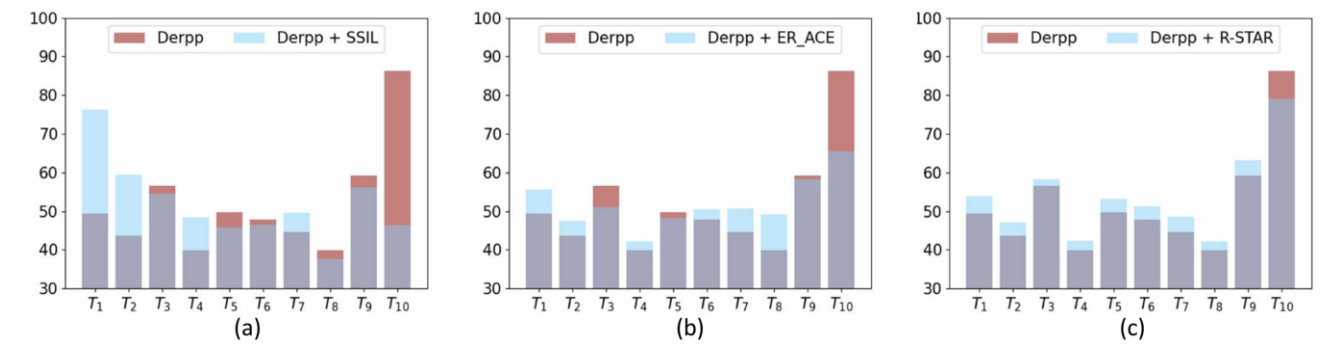

Figure 3. Visualized results among tasks $T_1 - T_{10}$ with three scores bias correction strategies on Split Cifar100 (buffer size = 2000): (a) Derpp + SSIL vs. Derpp, (b) Derpp + ER-ACE vs. Derpp, (c) Derpp + R-STAR vs. Derpp.

the scores bias is smaller due to its constraint on past and future classifier heads. Note that on Split Cifar100, X-Der has a balanced final performance among all tasks when adopting a large buffer size and suffers almost no score bias. Thus, R-STAR could only show minor advantages of the scores bias reduction under this setting. However, on the more challenging Split miniImagenet (more tasks), Split tinyImagenet (more classes per task), and Split CUB-200 (larger image size), where the X-Der still suffers from the scores bias problem, we find that R-STAR continues to help X-Der achieve better performances on these three challenging benchmark datasets by reaching around 1% FAA improvement and around 1.5% FF reduction. Overall, these results make a more convincing proof of the effectiveness of our R-STAR for it further improves the performance of these approaches with weak strategies for score bias reduction.

Approaches with strong strategies for scores bias reduction: SSIL [1] and ER-ACE [8] are recent works specifically proposed to avoid scores bias. The core idea of these approaches is to split the backward gradients among old and new tasks. To illustrate the advantages of our R-STAR over these two methods, we make comparisons from two perspectives. First, we want to illustrate that the R-STAR could help the methods with no bias correction (like Derpp) reach higher performance than the two strong bias correction methods. We thus provide their performance in Table 2, from which we could find that the performance of these two methods is higher than the original performance (shown in Table 1) of the best "no-bias-correction" method Derpp on all three datasets. However, by combining with our proposed R-STAR, Derpp could easily outperform these two methods on all the datasets by over 1% on average. That is to say, R-STAR provides a new effective way to reduce the scores bias for reaching higher performance. Second, we make a fairer comparison to prove the superiority of our R-STAR over these two methods and analyze the reason behind this. Specifically, We conduct two extra experiments with the two combinations of their proposed strategies and the Derpp to check if they could make higher improvements than R-STAR does. The two combinations are named "**Derpp+SSIL**" and "**Derpp+ER-ACE**" respectively. Concretely, we set Derpp+SSIL by replacing the original softmax with their proposed "Separate Softmax" and modifying logits distillation with their "Task-wise Knowledge Distillation". For Derpp+ER-ACE, we adopt "Separated Cross Entropy" for novel classifier heads when training the data of novel classes. Similarly, we name the combination with R-STAR as "**Derpp+R-STAR**". It can be observed from Table 2 that all three combinations outperform the original Derpp but Derpp+R-STAR reaches a higher average improvement (1.71%) than that (0.57% and 0.62%) of the other two combinations. For further analysis, we visualize the final task-wise accuracy of Derpp+SSIL, Derpp+ER-ACE, and Derpp+R-STAR among all tasks on Split tinyImagenet in Figure 3. Although Derpp+SSIL and Derpp+ER-ACE achieve more balanced performance among tasks and make further improvements to earlier tasks, they both have huge sacrifices in the current task performance. Rethinking their practice of only classifying the new samples within the new classes, we deduce that the relatively lower overall improvement owes to their worse performance on new tasks compared to the baseline Derpp during training on all sequential tasks. On the contrary, our R-STAR steadily improves the previous task performance with minor sacrifice on that of the latest task, which finally leads to the highest overall improvements. Overall, from these two perspectives, our R-STAR actually proves its advantages over these existing best bias correction methods.

4.4 Ablation Study

Our proposed R-STAR is composed of a task-wise reweighting strategy and an augmentation strategy. We ablate the two strategies combined with three other methods to show their influence on the final performance across three datasets. As can be observed from Table 3, we find that: (1) Adopting task-wise reweighting with no-augmented buffer could improve the performance of methods like LUCIR [19] and Der [7] which makes light use of memory buffer while having no effect on X-Der which makes full use of memory buffer.

Table 3. Ablation results (FAA) of different strategies in R-STAR on four benchmark datasets. "Reweight" indicates adopting the task-wise reweighting strategy, and "Augment" indicates applying an augmented buffer for reweighting.

Methods	Reweight	Augment	Split Cifar100		Split miniImagenet		Split tinyImagenet		Split CUB-200	
			500	2000	2000	5000	500	2000	500	2000
LUCIR [19]			39.19	48.61	14.97	17.06	29.55	31.40	51.43	65.01
	✓		39.19	48.61	15.74	17.77	31.32	31.85	52.07	65.68
	✓	✓	**41.56**	**50.94**	**16.05**	**18.24**	**31.42**	**32.03**	**52.95**	**66.10**
Der [7]			36.60	49.88	22.96	30.12	17.75	29.68	50.36	64.12
	✓		37.87	50.92	24.30	31.27	18.89	30.87	52.01	65.62
	✓	✓	**38.26**	**51.95**	**24.49**	**31.69**	**19.58**	**31.41**	**52.86**	**66.15**
X-Der [6]			48.08	**57.58**	28.19	32.32	29.77	40.76	59.23	68.35
	✓		48.08	**57.58**	28.19	32.32	29.77	40.76	59.23	68.35
	✓	✓	**48.47**	57.54	**28.87**	**32.99**	**30.80**	**41.61**	**59.94**	**69.02**

Table 4. Comparative results (FAA) of the variations of R-STAR on Split tinyImagenet. We compare our R-STAR with R-STAR-Avg and R-STAR-FB to demonstrate the effectiveness and rationality of our design.

Methods	Reweighting Strategy	Split tinyImagenet	
		500	2000
ER [27]	**Original**	9.99	17.82
	+ R-STAR-Avg	10.35	18.72
	+ R-STAR-FB	12.63	19.85
	+ R-STAR	**13.04**	**21.37**
Derpp [7]	**Original**	19.38	30.97
	+ R-STAR-Avg	19.58	31.07
	+ R-STAR-FB	20.88	31.76
	+ R-STAR	**21.66**	**32.73**
X-Der [6]	**Original**	29.77	40.76
	+ R-STAR-Avg	29.35	40.11
	+ R-STAR-FB	30.04	40.89
	+ R-STAR	**30.80**	**41.61**

That is to say, task-wise reweighting with no-augmented buffer is an effective strategy but it still needs further modifications for better generalization ability. (2) The augmentation strategy helps the task-wise reweighting strategy make further improvements for the SOTA method X-Der [6]. Since the main challenge for reweighting strategy is the overfitting problem on buffer data (as discussed in Sec. 3.3), these improvements prove that the augmentation strategy indeed helps the task-wise reweighting strategy alleviate the overfitting problem and enhance its robustness. (3) Moreover, the augmentation strategy could consistently further improve the performance of LUCIR [19] and Der [7], which indicates that the augmented buffer steadily optimizes the task-wise forgetting measurements. Overall, all these results prove the reasonability and effectiveness of each strategy in our R-STAR.

For further discussion, we adopt three rehearsal-based methods as basic methods and compare R-STAR with two variations on the Split tinyImagenet to ablate its design details. First, we want to highlight the difference between task-wise reweighting and trivial reweighting (reweighting with the average forgetting degree value) of the output scores. We thus set all $\varphi_{fgt}^{T_i}$ $(1 \leq i \leq n)$ to a constant average value as $\frac{1}{n}\sum_{i=1}^{n}\varphi_{fgt}^{T_i}$ and name this variation "**R-STAR-Avg**". It can be observed from Table 4 that R-STAR makes 3% improvements while the R-STAR-Avg only makes around 1% improvement for ER and Derpp and even has a bad effect for X-Der. Therefore, task-wise reweighting is proved to be an effective and reasonable design. Second, we want to illustrate that the changing size of $B_t^{T_i}$ across t tasks makes no bad effect on measuring φ_{fgt}. To ablate this, we sample a subset $B_t^{fix} = \{B_t^{FT_i}\}_{i=1}^{t}$ from B_t, where $B_t^{FT_i}$ is sampled from $B_t^{T_i}$ and is in the fixed size of $\frac{1}{n}|B|$. We then evaluate on augmented B_t^{fix} to obtain Acc_t and name such variation "**R-STAR-FB**". It can be observed from Table 4 that R-STAR reaches higher improvements than R-STAR-FB, which proves that our evaluation on augmented $B_t^{T_i}$ without fixed size is a better choice. Overall, the comparison results further prove the details of our R-STAR design are reasonable.

Table 5. Effect of k on the performance of our R-STAR when combined with different methods. "Ori." denotes the basic performance of the rehearsal-based methods. We show the final results (FAA) of the model on Split tinyImagenet and Split CUB-200 both with buffer size of 2000.

Datasets	Methods	Ori.	k				
			0	0.5	1	1.5	2
Split tinyImagenet	**ER + R-STAR**	17.82	18.17	18.94	20.91	**21.37**	20.98
	Derpp + R-STAR	30.97	31.02	31.86	32.52	**32.73**	32.00
	X-Der + R-STAR	40.76	40.87	**41.61**	41.37	40.99	40.79
Split CUB-200	**ER + R-STAR**	61.10	61.26	62.22	63.5	**63.63**	62.90
	Derpp + R-STAR	66.05	66.14	66.56	**66.98**	66.51	66.09
	X-Der + R-STAR	68.35	68.42	**69.02**	68.78	68.54	68.41

Moreover, we show extra experiment results in Table 5 with three typical rehearsal-based methods (i.e. ER, Derpp, and X-Der) on two challenging benchmark datasets (i.e. Split tinyImagenet and Split CUB-200) to ablate the effect of the hyperparameter k. As we introduced in Sec. 3.2, k is set to help R-STAR adapt to different methods on different benchmark datasets. In our experiments, we search for the value of hyperparameter k for most of the methods on four benchmark datasets in the range of [0, 2]. From the results in Table 5, we can easily find that: (1) For the rehearsal-based methods suffering from more severe scores bias problem (i.e. ER and Derpp), larger value of k (i.e. 1.5) could better assist R-STAR to make higher improvement. In reverse, for the rehearsal-based methods with slight scores bias problem, smaller value of k (i.e. 0.5) makes better help. (2) In the whole selection range of k, R-STAR could consistently improve the performance for these rehearsal-based methods, and the improvements keeps remarkable in the range of [0.5, 1.5]. Overall, all these observations provide a clear understanding on the effect of k and a clear definition of the auxiliary role of k for our R-STAR.

5 Conclusion

In this paper, we propose a novel Robust Self-Taught Task-Wise Reweighting (R-STAR) method. It introduces a robust self-taught learning paradigm which makes the model aware of the scores bias and reduces the bias in a self-taught way. Acting as a flexible and effective component for most rehearsal-based approaches, R-STAR takes almost no extra training time and excessive performance sacrifice on the new task while making consistent improvements for these approaches. Additionally, we also prove its superiority over existing bias correction methods and ablate its design details and the effect of key hyperparameters. Extensive results on the commonly-used CIL benchmarks demonstrate the effectiveness of our R-STAR.

Acknowledgements

This work was supported by National Natural Science Foundation of China (61976220). Zhiwu Lu is the corresponding author.

References

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

[2] 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).

[3] Arijit Banerjee and Vignesh Iyer. Cs231n project report-tiny imagenet challenge, 2015.

[4] Eden Belouadah and Adrian Popescu, 'Il2m: Class incremental learning with dual memory', in *IEEE/CVF international conference on computer vision*, pp. 583–592, (2019).

[5] Lorenzo Bonicelli, Matteo Boschini, Angelo Porrello, Concetto Spampinato, and Simone Calderara, 'On the effectiveness of lipschitz-driven rehearsal in continual learning', *arXiv preprint arXiv:2210.06443*, (2022).

[6] Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara, 'Class-incremental continual learning into the extended der-verse', *arXiv preprint arXiv:2201.00766*, (2022).

[7] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara, 'Dark experience for general continual learning: a strong, simple baseline', *Advances in neural information processing systems*, **33**, 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:2203.03798*, (2022).

[9] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari, 'End-to-end incremental learning', in *European conference on computer vision (ECCV)*, pp. 233–248, (2018).

[10] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr, 'Riemannian walk for incremental learning: Understanding forgetting and intransigence', in *European conference on computer vision (ECCV)*, pp. 532–547, (2018).

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

[12] 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 preprint arXiv:1902.10486*, (2019).

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton, 'A simple framework for contrastive learning of visual representations', *ArXiv*, **abs/2002.05709**, (2020).

[14] WU Chenshen, L HERRANZ, LIU Xialei, et al., 'Memory replay gans: Learning to generate images from new categories without forgetting [c]', in *The 32nd International Conference on Neural Information Processing Systems*, pp. 5966–5976, (2018).

[15] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le, 'Autoaugment: Learning augmentation policies from data', *arXiv preprint arXiv:1805.09501*, (2018).

[16] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le, 'Randaugment: Practical automated data augmentation with a reduced search space', in *IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, (2020).

[17] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan, 'Remind your neural network to prevent catastrophic forgetting', in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pp. 466–483. Springer, (2020).

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', in *IEEE conference on computer vision and pattern recognition*, pp. 770–778, (2016).

[19] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin, 'Learning a unified classifier incrementally via rebalancing', in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 831–839, (2019).

[20] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid, 'Memory-efficient incremental learning through feature adaptation', in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pp. 699–715. Springer, (2020).

[21] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer, 'Generative feature replay for class-incremental learning', in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 226–227, (2020).

[22] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun, 'Mnemonics training: Multi-class incremental learning without forgetting', in *IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 12245–12254, (2020).

[23] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi, 'Learning to remember: A synaptic plasticity driven framework for continual learning', in *IEEE/CVF conference on computer vision and pattern recognition*, pp. 11321–11329, (2019).

[24] Federico Pernici, Matteo Bruni, Claudio Baecchi, Francesco Turchini, and Alberto Del Bimbo, 'Class-incremental learning with pre-allocated fixed classifiers', in *International Conference on Pattern Recognition (ICPR)*, pp. 6259–6266. IEEE, (2021).

[25] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng, 'Self-taught learning: transfer learning from unlabeled data', in *the 24th international conference on Machine learning*, pp. 759–766, (2007).

[26] 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*, pp. 2001–2010, (2017).

[27] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro, 'Learning to learn without forgetting by maximizing transfer and minimizing interference', *arXiv preprint arXiv:1810.11910*, (2018).

[28] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim, 'Continual learning with deep generative replay', *Advances in neural information processing systems*, **30**, (2017).

[29] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al., 'Matching networks for one shot learning', *Advances in neural information processing systems*, **29**, (2016).

[30] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie, 'The caltech-ucsd birds-200-2011 dataset', (2011).

[31] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu, 'Large scale incremental learning', in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 374–382, (2019).

[32] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang, 'Incremental learning using conditional adversarial networks', in *IEEE/CVF International Conference on Computer Vision*, pp. 6619–6628, (2019).

[33] Friedemann Zenke, Ben Poole, and Surya Ganguli, 'Continual learning through synaptic intelligence', in *International Conference on Machine Learning*, pp. 3987–3995. PMLR, (2017).

[34] Bo Zhao, Shixiang Tang, Dapeng Chen, Hakan Bilen, and Rui Zhao, 'Continual representation learning for biometric identification', in *IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1198–1208, (2021).