Recurrent Network Expansion for Class Incremental Learning

Kai Jiang, Xueru Bai, *Senior Member, IEEE*, and Feng Zhou, *Member, IEEE*

***Abstract*—Class incremental learning (CIL) is the key to achieving adaptive vision intelligence, and one of the main streams for CIL is network expansion (NE). However, state-of-the-art (SOTA) methods usually suffer from feature diffusion, growing parameters, feature confusion, and classifier bias. In view of this, a novel dynamic structure dubbed as recurrent NE (RNE) is proposed by establishing connections among task experts. Specifically, the previous task experts transfer features sequentially through a shared module and the new task expert makes adjustments based on received features rather than reextracted ones, thereby focusing more on the key area and avoiding feature diffusion. Furthermore, the RNE is compressed by replacing additional task experts with lightened ones, in order to significantly reduce the number of parameters while keeping the performance almost unaltered. In addition, feature confusion is alleviated by a decoupled classifier and classifier bias is corrected by pseudo-feature generation. Extensive experiments on four widely adopted benchmark datasets, i.e., CIFAR-100, ImageNet-100, Food-101, and ImageNet-1K, have demonstrated that RNE achieves SOTA performance in both ordinary and challenging CIL settings.**

***Index Terms*—Bias correction, class incremental learning (CIL), decoupled classifier, recurrent structure.**

I. Introduction

UNLIKE human beings that can learn new concepts consistently without forgetting, existing AI systems lack continual learning ability [1], [2], [3], i.e., they always overfit on new tasks and forget previous ones when learning multiple tasks in stages, known as catastrophic forgetting [4], [5], [6]. To address this issue, class incremental learning (CIL) is proposed, which learns different tasks with multiple disjoint categories sequentially and attempts to perform well on all tasks.

There have been much efforts to improve the performance of CIL [7], [8], [9], [10], [11], [12], [13]. Among them, an effective and simple way is rehearsal [14], [15], which constructs an exemplar set to store a limited number of samples from previous tasks for future training. Due to limited capacity, however, saving only a subset of the training data still encounters severe catastrophic forgetting.

Received 11 October 2024; revised 16 April 2025; accepted 19 August 2025. This work was supported by the National Natural Science Foundation of China under Grant 62425113 and Grant 62131020. *(Corresponding author: Xueru Bai.)*

Kai Jiang and Xueru Bai are with the National Key Laboratory of Radar Signal Processing, Xidian University, Xi'an 710071, China (e-mail: xrbai@xidian.edu.cn).

Feng Zhou is with the School of Aerospace Science and Technology, Xidian University, Xi'an 710071, China (e-mail: fzhou@mail.xidian.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2025.3601373

In view of this, *distillation* [11], [12], [13], [16], [17], [18] and *parameter regularization* [7], [8], [9], [10] maintain the classification capability of previous tasks by constraining the output logits, the intermediate features, or part of the crucial parameters. In a nutshell, they attempt to adapt all tasks with a single branch, which can express new concepts in the same feature space without affecting previous tasks. However, such a strategy suffers from the stability–plasticity dilemma [19], [20], i.e., maintaining the stability of the original feature space hinders the learning of new concepts. As the number of tasks grows, the model will eventually fail to accommodate new tasks due to insufficient capacity.

Dynamic structure [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31] freezes part of the parameters of old tasks and introduces new training parameters to solve new tasks. Among them, network expansion (NE) [22], [27], [29] improves the classification performance significantly by adding a complete network per task. As illustrated on the left side of Fig. 1, a network $\mathcal{F}_1$ is learned for task 1, and a new network $\mathcal{F}_t$, dubbed as the task expert, is added for each subsequent task t. It is worth noting that since no connections exist among task experts, they may output various representations of the same input when the training data change. Accordingly, accurate feature representations of the original categories will be replaced by multiple distorted representations as t continuously increases. As shown by the Grad-CAM visualization [32] of a conventional NE method, i.e., DER [22], on the right side of Fig. 1, the model gradually focuses on the invalid area during incremental learning, i.e., feature diffusion occurs. In this scenario, extracted features of old categories gradually expand into irrelevant regions, which is induced by the progressive accumulation of imprecise semantic representations of old categories extracted by subsequent task experts. By feature confusion, we mean that the features of different categories are misclassified by the classifier, which is induced by catastrophic forgetting during cross-task learning. In this scenario, semantic features of the old categories retain precise, but the classifier fails to assign the corresponding labels correctly.

The core challenge in addressing feature diffusion lies in enabling new task experts to maintain accurate representations of old categories. Since the corresponding task expert of each old category consistently extracts accurate representations, establishing connections between the new and the old task experts emerges as an intuitive approach. On this basis, we establish interconnections among task experts through a set

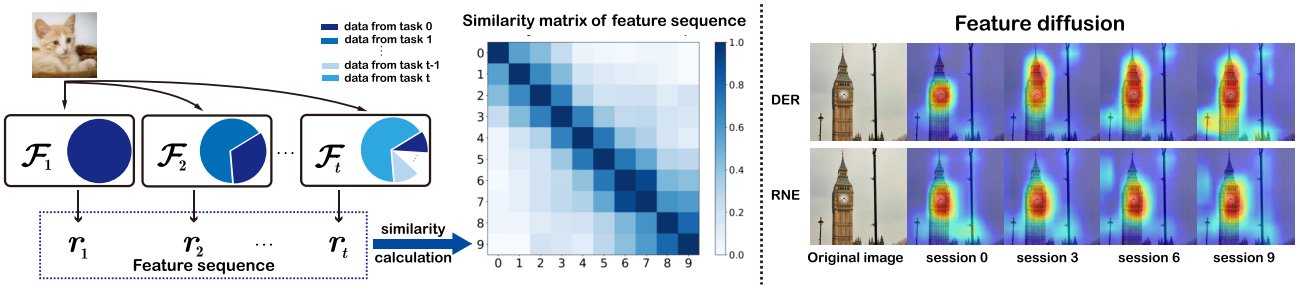

Fig. 1. Illustration of feature sequence and feature diffusion.

of parameter-shared modules, which can enable subsequent task experts to acquire intermediate features from previous task experts without redundant reconstruction and facilitate adaptive refinement of feature representations. Originated from task incremental learning [33], [34], [35], [36], [37], NE with cross connections (NEwCs) [26], [33], [38] builds dense connections (DCs) among task experts, which can be served as a solution for feature diffusion. It transfers knowledge from previous task experts to the new one following the scheme of transfer learning [39], [40]. However, unlike task incremental learning with available task labels, the task label is unavailable for NEwC and all the task experts should participate in category inference, thereby inducing a large computational burden.

To tackle the above issues, this article proposes recurrent NE (RNE), which reduces feature diffusion in CIL by constructing connections among task experts with affordable computing cost. Then, in order to achieve sustainable NE, a lite version dubbed as RNE-compress is designed, which expands the feature space with only a few parameters while maintaining competitive performance. On this basis, the classifier is decoupled and a bias correction strategy is designed. Specifically, the classifier is decoupled into multiple task-level classifiers, which are modified into a more causal form to reduce feature confusion among tasks. After that, features of the new task samples are exploited to regenerate pseudo features of the previous task, and a balanced feature set is constructed with these pseudo features and new task features. Finally, the classifier is fine-tuned to achieve balanced classification. Overall, the main contributions include the following.

1) The feature diffusion phenomenon is uncovered and RNE is designed to alleviate it. In particular, RNE allows task experts to share intermediate feature maps sequentially to improve the efficiency of feature extraction.
2) The RNE is compressed to drastically reduce the number of parameters, which can achieve superior CIL performance with small increment in the computational cost for a new task.
3) The classifier is decoupled into a causal form, allowing the new subclassifier to utilize all the feature sequences while keeping previous subclassifiers insusceptible to new task features.
4) Pseudo features of old categories are generated with new category samples to obtain a balanced feature set, which is then adopted for classifier retraining to achieve balanced classification.

II. Related Work

The CIL aims to design models that can continuously learn new tasks with disjoint categories without forgetting. In this section, we will give a brief discussion of the current CIL methods.

A. Regularization

Such methods assume that the classification knowledge is stored within model parameters and add constraints on the direction of parameter updating to maintain the representation of previous tasks. For instance, Chaudhry et al. [41], Yang et al. [42], and Zenke et al. [10] penalize the parameter drift to avoid catastrophic forgetting. Kirkpatrick et al. [8] measure the importance of each parameter through the Fisher matrix and update it accordingly. In addition, some approaches [43], [44] reckon that avoiding the forgetting of previous tasks could be achieved by simply making the corresponding gradients orthogonal to those of the new task. Wang et al. [45] propose a reserver loss as a new regularization technique in the pretraining stage for few-shot CIL. Liu et al. [46] propose a reserver loss as a new regularization technique in the pretraining stage for few-shot CIL. Knowledge distillation is also widely used as a function regularization, which targets the intermediate [16], [17], [18] or final [11] output of prediction function. Recently, Ji et al. [47] propose a decoupled knowledge distillation to mitigate the sample imbalance between old tasks and the new one. These methods generally inherit previous knowledge from a single teacher and have limited performance. To this end, MTD [48] proposes multiteacher distillation to find multiple diverse teachers for CIL.

B. Rehearsal

Such methods construct an exemplar set, which stores limited samples of previous tasks for future training [11]. Due to the strict memory budgets, work [13] does not select samples uniformly from previous tasks but adjusts the number of samples of each category dynamically. Luo et al. [49] keep more compressed exemplars by downsampling their nondiscriminative pixels. Kim et al. [50] incorporate a feature augmentation technique motivated by adversarial attacks to alleviate the collapse of the decision boundaries caused by

sample deficiency for the previous tasks. Ho et al. [51] propose a dynamic prototype-guided memory replay to guide sample selection for memory replay.

C. Dynamic Structure

Such methods improve model plasticity by introducing new training parameters and maintain model stability by freezing parameters of previous tasks. For instance, Aljundi et al. [21] adopt a task-level selector to choose the best suitable task expert for each sample to be classified. Yan et al. [22] introduce NE by adding a complete network per task and enhance the plasticity with an auxiliary loss. Wang et al. [23] also expand a complete network per task but distill the new and the old networks into a single one for next expansion, in order to maintain a constant number of parameters. Douillard et al. [25] and Hu et al. [26] replace the backbone with transformer [52] and then design the corresponding NE structures. Zhou et al. [28] only expand the shallow layers while sharing the deep layers to reduce network parameters. Wang et al. [29] train independent modules in a decoupled manner and achieve bidirectional compatibility among modules through two additionally allocated prototypes. It is worth noting that most of these methods do not interact among task experts, thereby having limited information interaction and CIL performance. Rusu et al. [33] first introduce DCs to CIL, which needs an accurate task ID to choose the corresponding branch. On this basis, two progressive networks are proposed [26] and [38]. Although previous knowledge can be transferred to the new task expert successfully, more parameters are added and higher computing cost is induced.

D. Bias Correction

Such methods adopt postprocessing after incremental training to deal with classifier bias caused by imbalanced training data. Fine-tuning the classifier with a small resampled balanced dataset is widely adopted in [17], [22], and [25]. However, its performance is not satisfying due to the limited number of samples. BiC [53] preserves a few samples in advance and then utilizes them to train a set of additional parameters after incremental training, in order to adjust the output of the classifier. However, the reduction of training samples due to sample preservation may lead to even worse performance. WA [12] directly adjusts the classifier weights to address classifier bias, whereas improper choice of the adjustment factor can degrade the performance heavily.

III. Methodology

In this section, we will introduce the proposed RNE in detail, which attempts to address the issues of feature diffusion, parameter redundancy, feature confusion, and classifier bias in CIL. First, the definition of CIL is given in Section III-A. Then, the overall structure is introduced in Section III-B. The recurrent structure and the compressed recurrent structure are introduced in Section III-C. The decoupled classifier and the bias correction strategy are presented in Sections III-D and III-E, respectively.

A. Problem Setup

CIL aims to learn a unified classification model from a data stream containing different categories. The entire training process is divided into several sessions sequentially, with each session learns one task containing multiple disjoint categories. At the tth session, the model receives the training data $\mathcal{D}_t = \{(x_i^t, y_i^t)\}$, where $x_i^t \in \mathcal{X}_t$ is an input sample; $y_i^t \in \mathcal{Y}_t$ is the corresponding label; and $\mathcal{X}_t$ and $\mathcal{Y}_t$ denote the training set and the label set, respectively. Based on the rehearsal strategy, only a small number of samples from previous tasks are stored in an exemplar set $\mathcal{V}_t \subseteq \cup_{j=1}^{t-1}\mathcal{D}_j$ with fixed capacity. Then, the model is trained on $\mathcal{D}_t \cup \mathcal{V}_t$ and evaluated on the test set of all known categories. Without loss of generality, we will only focus on details of the tth session in the following discussions. In particular, we denote the label space of old categories and new categories as $\mathcal{Y}_o = \cup_{j=1}^{t-1}\mathcal{Y}_j$ and $\mathcal{Y}_n = \mathcal{Y}_t$, respectively, with $\mathcal{Y}_o \cap \mathcal{Y}_t = \emptyset$. In addition, $|\mathcal{Y}_n| = K$ represents the number of new categories and $|\mathcal{Y}_o| = M$ represents the number of old categories.

B. Overall Structure

The overall structure of the proposed method is shown in Fig. 2, which consists of a feature extractor and a decoupled classifier. The feature extractor consists of several task experts and the decoupled classifier consists of several subclassifiers. The number of task experts and subclassifiers is the same as the number of tasks. When the tth task arrives, a new task expert is added to the feature extractor and a new subclassifier is added to the decoupled classifier. Then, network training is performed with previous task experts frozen.

In the process of model inference, the input image is conveyed to each task expert, which transmutes the image into multiscale intermediate feature maps and shares the feature maps with the next task expert. Then, the last feature maps of all the task experts are combined into a feature sequence, which is fed into the decoupled classifier and transmuted into logits. Finally, the softmax activation transfers them into a classification probability.

In the first training stage, i.e., normal training, the model transmutes the image into logits, compares it with the ground truth to compute the loss, and updates the parameters. In the second training stage, i.e., bias correction, a category from the current task is selected and features of all the corresponding training samples are extracted. Then, pseudo features are generated from these features, based on which the classifier is retrained for bias correction.

C. Recurrent Structure

As illustrated in Fig. 1, the model of an NE method consists of multiple task experts, with each task expert $\mathcal{F}$ consisting of a series of layers, i.e., $\mathcal{F} = \{f_1, \ldots, f_L\}$. Then, intermediate features of the tth expert at the lth layer can be represented as

$$r_0^t = x \tag{1}$$

$$r_l^t = f_l^t\left(r_{l-1}^t; \varphi_l^t\right) \tag{2}$$

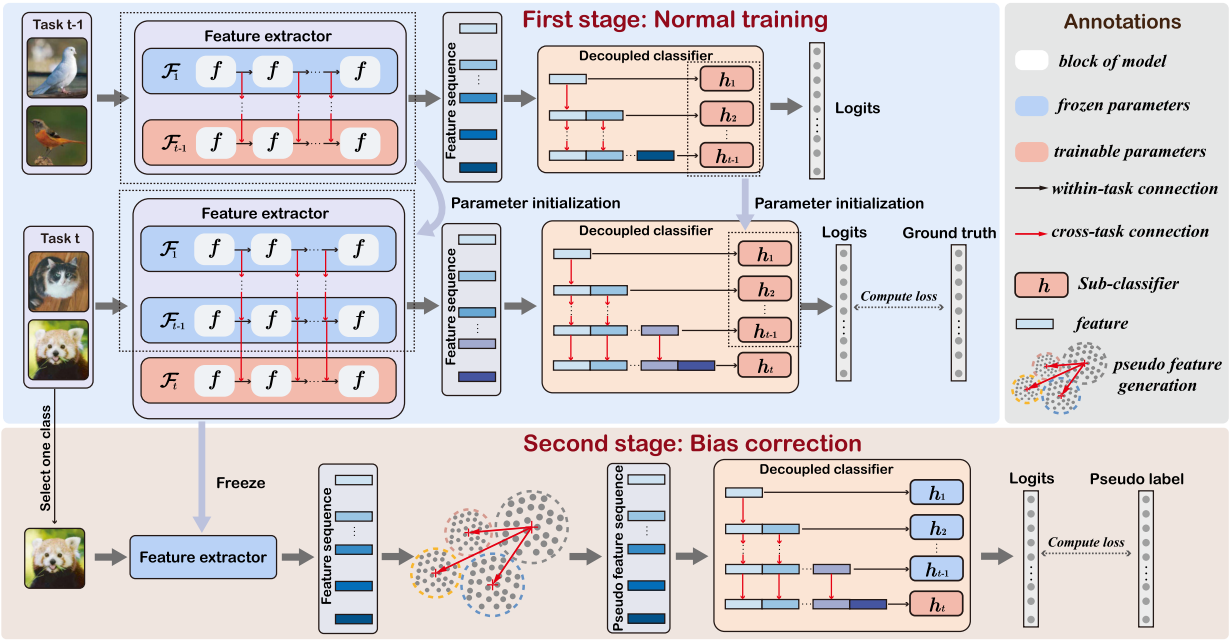

Fig. 2. Overall structure of the proposed method.

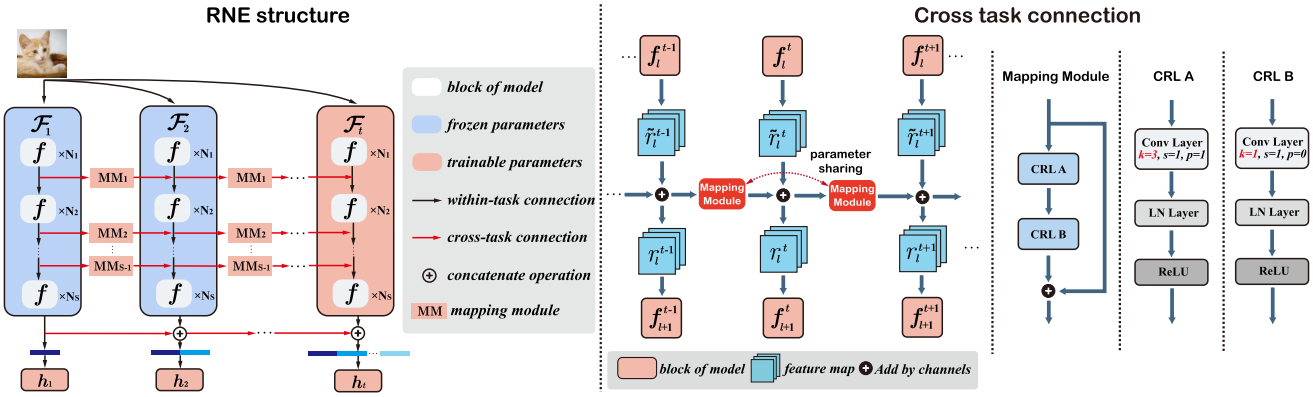

Fig. 3. Details of the recurrent structure and cross task connection, where $N_1 \sim N_s$ represents the number of stacked layers and s denotes the times that the size of feature map changes during forward propagation of a task expert.

where x denotes the input tensor, r_l^t denotes the feature tensor at the output of the lth block, and φ_l^t is the parameter set of the lth block.

The intermediate features serve as explicit representations of the knowledge embedded within the feature extractor. For a dynamic structure, different task experts should exhibit a consistent mapping relationship when processing the same image. However, variations of the input distribution can alter such relationships. To be specific, due to the rehearsal strategy, the feature sequence output by all the task experts within the dynamic structure contains temporal correlation between the training data distribution of each task, as demonstrated by the feature correlation matrix in Fig. 1. For adjacent task experts, they usually exhibit similar mappings, i.e., high correlation. For distant experts, however, the temporal correlation decreases rapidly and large disparity exists in the output features. In this scenario, multiple representations of the same category are given by different task experts, which blur distinctions between categories and cause catastrophic forgetting in late sessions of incremental learning.

To tackle this issue, we strengthen interconnections among task experts and design the recurrent structure, as shown in Fig. 3. Inspired by sequence models [54], [55], we treat the expansion process of the dynamic structure as an extension of the original static classification framework in the temporal dimension. Then, we construct connections among task experts and obtain a feature sequence, whose temporal correlation reflects the distribution variation of training samples during incremental learning.

Specifically, a feature sharing structure is designed and inserted between adjacent task experts

$$\tilde{r}_l^t = f_l^t\left(r_{l-1}^t\right) \tag{3}$$

$$r_l^t = \tilde{r}_l^t + \mathrm{MM}\left(r_l^{t-1}\right) \tag{4}$$

where MM is a mapping module consisting of a simple residual structure with two basic CRL blocks and is shared

at the same depth as the task experts to learn the generality within intermediate features.

It is worth noting that in a typical dynamic structure, information flows in a top-to-bottom manner during feature extraction, with each task expert operating autonomously and interconnected solely via the classifier. By integrating the mapping module, however, the information is not only passed down through the task experts but also conveyed to the neighboring task experts, mimicking the information flow of a multilayer RNN [54] and facilitating the discerning of temporal correlations within the feature sequence. In addition, similar to the parameter sharing strategy of RNN, the mapping module is shared among the same layers of task experts, which can reduce the number of parameters and enhance connections between the new and previous tasks. By this means, the frozen task experts can be optimized mildly during the training of a new task, thereby improving the model plasticity.

In RNE, we do not construct connections at each layer of task experts because: 1) DCs adopted by NEwC methods [26], [33], [38] are inefficient and can reduce the plasticity of the new task expert [33] and 2) excessive connections can introduce noise to extracted features of the new task expert [23] and hinder new task learning. Therefore, we only build connections at key layers where the size of feature maps changes, e.g., the layer that doubles the channel dimension of the feature map and halves its width and height.

Since the representation of new tasks with the previous task experts is enhanced by the recurrent structure, we further reduce the parameter redundancy of RNE by reducing the parameters of task experts. The compressed model, i.e., RNE-compress, is illustrated in Fig. 4, where a complete network is adopted as the feature extraction backbone and is trained together with the first task expert in the first task. Then, a simplified version of the general network, i.e., the compressed network, serves as the task expert for each stage. In particular, it shares a similar structure with the feature extraction backbone, with the dimensionality of layers reduced to 1/4 and the number of parameters reduced to 6% of the original by the feature reduction module (which consists of convolutional layers). Specifically, the feature extraction backbone is frozen, while the mapping module and the feature reduction module are trained with a lower learning rate to improve the feature extraction ability.

D. Decoupled Classifier

As shown in Fig. 5(a), a general classifier transmutes features to logits directly and brings feature confusion among tasks. Although some methods decoupled the classifier completely, as shown in Fig. 5(b), the new classifier loses information of previous task experts. To tackle this issue, we propose the decoupled classifier shown in Fig. 5(c), where previous task classifiers are decoupled with the new task expert while feeding features from all task experts to the new task classifier.

The classifier of previous tasks is trained on a closed set composed of old categories, making it inherently biased to produce higher responses to certain categories within the task. For instance, when inputting an image from a new category

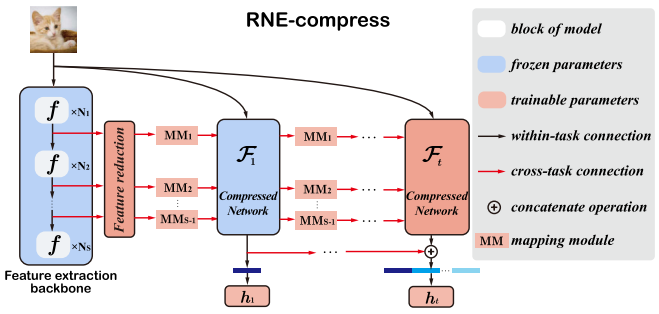

Fig. 4. Structure of RNE-compress.

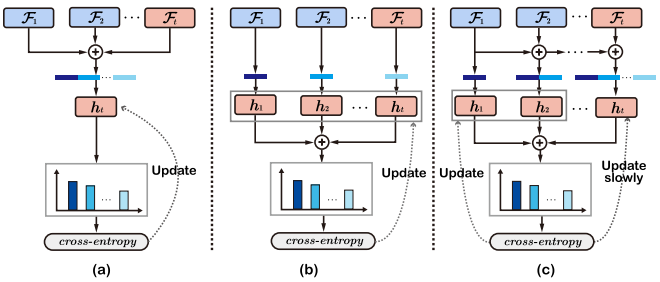

Fig. 5. Comparisons of different classifiers. (a) General classifier. (b) Decoupled completely. (c) Our classifier.

into a subclassifier of a previous task, it is still likely to be classified as one of the old categories with high confidence. Therefore, fine-tuning the classifier for the new task is indispensable for all CIL methods. The proposed decoupled classifier specifically addresses this factor. Since the model has not been trained on the new task but has completed training on old ones, the subclassifier corresponding to new tasks should receive larger gradient updates. Conversely, for subclassifiers associated with old tasks, only minor adjustments are needed to adapt their responses to new task categories, thus requiring smaller gradient updates. To tackle this issue, we constrain the update of classifier by the following equation:

$$\begin{cases} \hat{\phi}_{1,\ldots,t-1} = \phi_{1,\ldots,t-1} + \gamma \cdot \lambda \cdot \dfrac{\partial \mathcal{L}}{\partial \phi_{1,\ldots,t-1}} \\ \hat{\phi}_t = \phi_t + \lambda \cdot \dfrac{\partial \mathcal{L}}{\partial \phi_t} \end{cases} \tag{5}$$

where $\phi_{1,\ldots,t-1}$ is the parameter set of classifier $\{h_1, \ldots, h_{t-1}\}$, ϕ_t is the parameter set of classifier h_t, λ is the learning rate, and λ is a factor to slow the update of previous task classifiers.

The cross-entropy loss for the new and old category samples is calculated by the following equations:

$$\mathcal{L}_{\text{ce}} = \alpha \cdot \mathcal{L}_{\text{cen}} + \mathcal{L}_{\text{ceo}} \tag{6}$$

$$\alpha = \frac{\mathcal{B}_{\text{old}}}{\mathcal{B}_{\text{new}}} \cdot \beta \tag{7}$$

where $\mathcal{L}_{\text{ce}}$ is the cross-entropy loss comprising of the cross-entropy loss $\mathcal{L}_{\text{cen}}$ for new categories and the cross-entropy loss $\mathcal{L}_{\text{ceo}}$ for old categories, $\mathcal{B}_{\text{old}}$ is the number of old categories of a batch, $\mathcal{B}_{\text{new}}$ is the number of new categories of a batch, and β is a dynamic factor defined by the following equation:

$$\beta = 0.1 + 0.9 \frac{\text{epoch}_c}{\text{epoch}_{\text{all}}} \tag{8}$$

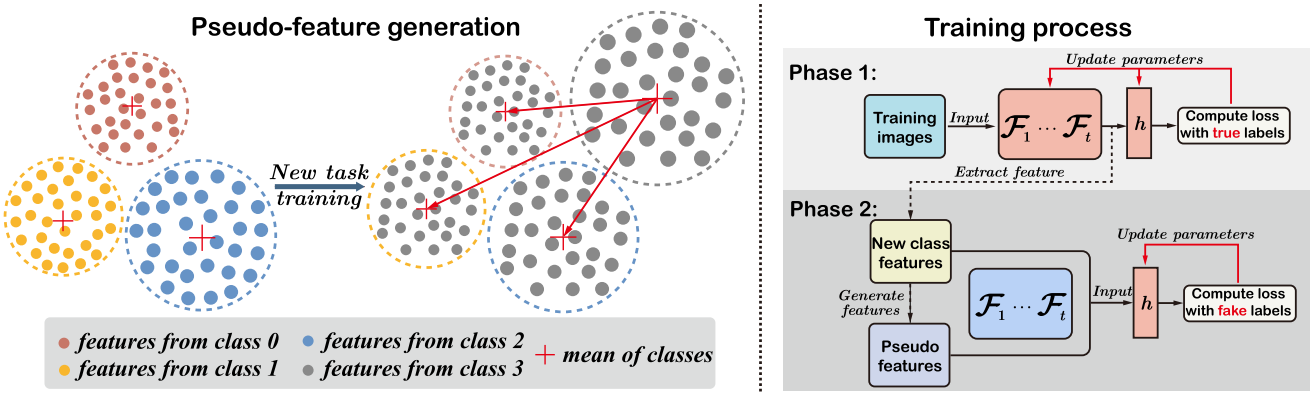

Fig. 6. Illustration of pseudo-feature generation.

where epoch_c denotes the current number of iterations and $\text{epoch}_{\text{all}}$ denotes the total number of iterations. By this means, the model is able to learn the deep representation of new task instead of embezzling features learned from previous tasks.

E. Bias Correction Using Pseudo Features

Current bias correction strategies with postprocessing focus more on samples [11], [17], [29], [53]. However, since the classifier inputs are features rather than samples, a large feature set rather than a large sample set is required for bias correction. In addition, due to the frozen task experts of previous tasks, the NE methods are free from feature drift [11], [17], [29], [53], and the mean and variance of features calculated in previous incremental sessions maintain representativeness. In view of this, we design a pseudo-feature generation strategy to reconstruct old category features from new task samples, in order to obtain a large and balanced feature set. As illustrated in Fig. 6, it trains the model normally like most CIL methods do at the first phase and retrains the classifier with pseudo feature vectors at the second phase.

Algorithm 1 Pseudo-Feature Generation ($t > 1$)

Input: New data $\mathcal{D}_t$, exemplar-set $\mathcal{V}_t$, task experts $\mathcal{F}_1 \sim \mathcal{F}_t$ after the first training phase, the set of feature mean ε_t ($\varepsilon_1 = \emptyset$), the set of feature variance η_t ($\eta_1 = \emptyset$).

Output: Pseudo-features r^m_{fake}.

for $k = 0$ **to** K **do**
  Extract $r^k_{new} | x^k_{new}, x^k_{new} \in \mathcal{D}_t$ by Eq. (9)
  Calculate μ^k_{new} and σ^k_{new} by Eq. (10), (11), (12)
  $\varepsilon_t \leftarrow \varepsilon_t \cup \{\mu^k_{new}\}$, $\eta_t \leftarrow \eta_t \cup \{\sigma^k_{new}\}$
end
for $m = 0$ **to** M **do**
  Extract $r^m_{old} | x^m_{old}, x^m_{old} \in \mathcal{V}_t$ by Eq. (13)
  Update $\mu^m_{old} \in \varepsilon_t$ and $\sigma^m_{old} \in \eta_t$ by Eq. (14), (15), (16)
end
Choose the k^*-th category from new task by Eq. (17)
Generate r^m_{fake} by Eq. (18) with the k^*-th category
return r^m_{fake}

The pseudocode for pseudo-feature generation is shown in Algorithm 1. Supposing that we obtain the classification model $g_t(x) = \{\mathcal{F}_1, \ldots, \mathcal{F}_t, h_1, \ldots, h_t\}$ at the first training phase, then, at the second training phase, we freeze all the task experts $\{\mathcal{F}_1, \ldots, \mathcal{F}_t\}$, extract features of the new task samples by (9), and calculate the mean μ^k_{new} and variance σ^k_{new} of features for the kth category in the new task by

$$r^k_{\text{new}} = \mathcal{F}_1\left(x^k_{\text{new}}\right) \oplus \cdots \oplus \mathcal{F}_t\left(x^k_{\text{new}}\right), \quad x^k_{\text{new}} \in \mathcal{D}_t \tag{9}$$

$$\bar{r}^k_{\text{new}} = \frac{1}{N_k} \sum r^k_{\text{new}} \tag{10}$$

$$\mu^k_{\text{new}} = \bar{r}^k_{\text{new}} \tag{11}$$

$$\sigma^k_{\text{new}} = \left(\frac{1}{N_k} \sum \left(r^k_{\text{new}} - \mu^k_{\text{new}}\right)^2\right)^{\frac{1}{2}} \tag{12}$$

where x^k_{new} denotes the samples of the kth new category in the new task and N_k denotes the sample number of the new category in the new task. All the operations are based on vectors.

Meanwhile, we extract features of the old categories from the exemplar set by (13) and then update the corresponding mean and variance of features by

$$r^m_{\text{old}} = \mathcal{F}_t\left(x^m_{\text{old}}\right), \quad x^m_{\text{old}} \in \mathcal{V}_t \tag{13}$$

$$\bar{r}^m_{\text{old}} = \frac{1}{N_m} \sum r^k_{\text{old}} \tag{14}$$

$$\mu^m_{\text{old}} = \tilde{\mu}^m_{\text{old}} \oplus \bar{r}^m_{\text{old}} \tag{15}$$

$$\sigma^m_{\text{old}} = \tilde{\sigma}^m_{\text{old}} \oplus \left(\frac{1}{N_m} \sum \left(r^m_{\text{old}} - \bar{r}^m_{\text{old}}\right)^2\right)^{\frac{1}{2}} \tag{16}$$

where x^m_{old} denotes the samples of the mth old category from the exemplar set, N_m denotes the sample number of the mth old category from the exemplar set, $\tilde{\mu}^m_{\text{old}}$ denotes the mean of the feature for the mth old category before updating, and $\tilde{\sigma}^m_{\text{old}}$ denotes the variance of the feature for the mth old category before updating. All the operations are based on vectors.

To minimize the impact of pseudo features on new categories, we only select one category from the new task for pseudo-feature generation. In detail, we calculate the average Euclidean distance between the mean vector of each new category and those of all the old categories and then choose

the one with the maximum Euclidean distance from the new task as the generator, as shown in the following equation:

$$k^* = \arg\max_k \frac{1}{M}\sum_{m=1}^{M}\left(\mu_{\text{new}}^k - \mu_{\text{old}}^m\right)^2. \tag{17}$$

Then, the pseudo feature is generated by

$$r_{\text{fake}}^m = \frac{r_{\text{new}}^{k^*} - \mu_{\text{old}}^m}{\sigma_{\text{new}}^{k^*}}\sigma_{\text{old}}^m + \mu_{\text{old}}^m \tag{18}$$

where $r_{\text{new}}^{k^*}$ denotes the sample feature of the chosen category in the new task and r_{fake}^m denotes the corresponding pseudo feature for the mth old category in previous tasks. Such a dynamic structure can prevent task experts from serious feature drift and make the means and variances of previous task features representative in subsequent tasks.

After pseudo-feature generation, a large-scale, balanced dataset with all known categories is constructed, which is then utilized to retrain the classifier $\{h_1, \ldots, h_t\}$ (see the following equation):

$$\hat{\phi}_{1,\ldots,t} = \arg\min_{\phi_{1,\ldots,t}} \mathcal{L}_{\text{ce}}\left(y_{\text{pre}}^i, y_{\text{fake}}^i\right) \tag{19}$$

where $\mathcal{L}_{\text{CE}}$ represents the cross-entropy loss, y_{pre}^i represents the predicted label, and y_{fake}^i represents the fake label corresponding to the generated features. The classifier is trained with only a few epochs and tested in a validation set constructed by the exemplar set and part of the new category samples.

IV. Experiments

In this section, extensive experiments are conducted to validate the effectiveness of the proposed method. Specifically, the proposed method is validated on CIFAR-100 [56], ImageNet-100 [57], and Food-101 [58] datasets with widely used benchmark protocols, and is compared with other CIL methods in both performance and computational cost. Meanwhile, results on ImageNet-1K [57] are reported. In addition, an ablation study is performed to verify the validity of each module. Furthermore, the memory budget is adjusted to more stringent conditions to demonstrate model robustness.

A. Experimental Setting

1) Datasets: The following experiments are conducted on four standard CIL benchmarks.

1) *CIFAR-100:* This consists of 32 × 32 colored images from 100 categories, with 50000 training samples (500 per category) and 10000 test samples (100 per category).
2) *ImageNet-100:* A subset of ImageNet-1000 [40] with 100 randomly selected categories from 1000 categories, containing about 130000 high-resolution colored images for training (approximately 1300 images for each category) and 5000 images for validation (with 50 per category).
3) *Food-101:* This consists of 101 food categories with 750 training samples and 250 test samples per category, and the maximum length for a single image is 512 pixels.
4) *ImageNet-1K:* This consists of 1000 categories with over 12.8 million training images (approximately 1300 high-resolution images per category for training and 50 images for validation). Details of the datasets are provided in Table I.

TABLE I
Details of the Three Datasets

Dataset	Classes	Training images	Test images	Avg. size
CIFAR-100	100	50,000	10,000	32×32
ImageNet-100	100	129,394	5,000	407×472
Food-101	101	75,750	25,250	475×496
ImageNet-1K	1,000	1,281,167	50,000	482×415

2) Protocols: A widely adopted protocol is selected from recent CIL works [11], [22], [27], [28], [29], [49]. The model is trained on half of categories (e.g., 50 categories for CIFAR-100) at the first session, and the remaining categories are learned evenly in the subsequent N sessions, where N can be 5, 10, and 25. After the training of each session, the model is validated on the test data of all known categories. Each experiment is conducted more than three times (5 for CIFAR-100 and 3 for others) and the average results are reported. For ImageNet-1K, the model is trained on 100 categories at each session following recent CIL works [11], [22], [27], [28], [29], [49].

3) Memory Budgets: We adopt the rehearsal strategy and construct the exemplar set, where a constant memory budget is allocated for each category. Specifically, 20 exemplars for each new category are added to the exemplar set after training at each session, and each exemplar is selected according to the herding algorithm [59].

4) Compared Methods: The proposed approach is compared with six single-branch approaches, i.e., iCaRL [11], BiC [53], UCIR [16], WA [12], PODNet [17], and CSCCT [18]; as well as five dynamic structure-based approaches, i.e., DER [22], FOSTER [23], TCIL [27], MEMO [28], and BEEF [29]. In addition, we provide two methods that can be used in conjunction with other methods for comparison, i.e., CCFA [50] and MTD [48]. They can be combined with PODNet as a single-branch method or DER as a dynamic structure-based approach. Following FOSTER, we apply AutoAugmentation [60] to all the methods in order to enhance the sample utilization efficiency. Each method adopts the same data augmentation technique for fairness. In addition, the method that only adopts the rehearsal strategy serves as the lower bound of CIL and is denoted as Replay. Following [23], [28], and [29], the method that utilizes all the samples for training serves as the upper bound of CIL and is denoted as Bound. The proposed method and all the baselines are implemented with Pytorch [61] in PyCIL [61].

5) Implementation Details: An 18-layer ResNet [62] is adopted as the backbone for all the methods. For CIFAR-100, the kernel size of the first layer is modified as 3×3 considering its low resolution, and the first max-pooling layer is removed.

1) For existing CIL methods, two sets of hyperparameters are adopted: the first set adopts the same hyperparameters as the original articles and the second set adopts the same hyperparameters as the proposed method. The better results are selected for comparison.

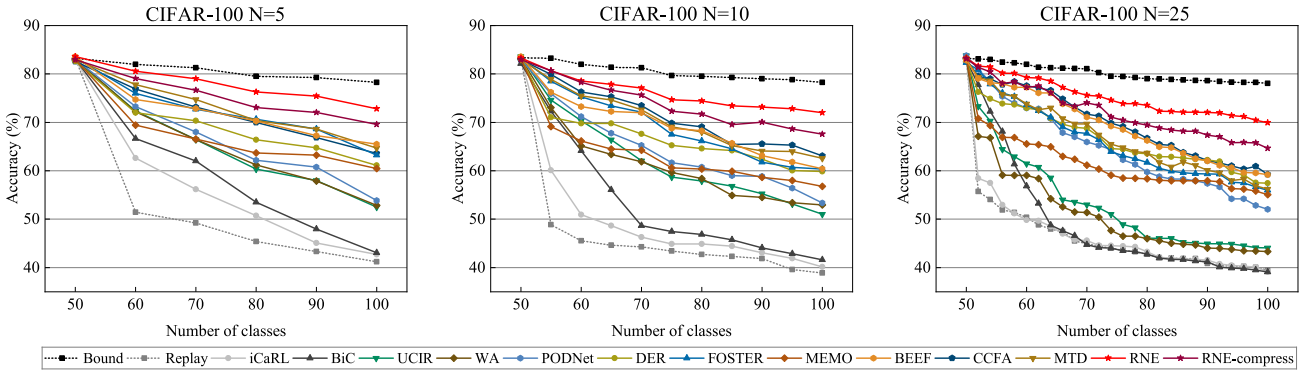

Fig. 7. Accuracy at each session on the CIFAR-100 benchmark.

TABLE II
RESULTS ON CIFAR-100 AND IMAGENET-1K BENCHMARK

Methods	CIFAR-100												ImageNet-1K		
	N=5				N=10				N=25				N=10		
	$\bar{A}\uparrow$	$D\downarrow$	$\bar{P}\downarrow$	$\bar{F}\downarrow$	$\bar{A}\uparrow$	$D\downarrow$	$\bar{P}\downarrow$	$\bar{F}\downarrow$	$\bar{A}\uparrow$	$D\downarrow$	$\bar{P}\downarrow$	$\bar{F}\downarrow$	$\bar{A}\uparrow$	$\bar{P}\downarrow$	$\bar{F}\downarrow$
Bound	80.59	0	11.20	0.56	80.52	0	11.20	0.56	80.19	0	11.20	0.56	79.89†	11.21	7.29
Replay	52.58	37.07	11.20	0.56	46.85	38.69	11.20	0.56	46.48	38.04	11.20	0.56	55.78†	11.21	7.29
iCaRL [11]	56.74	35.55	**11.20**	**0.56**	49.87	38.13	**11.20**	**0.56**	46.95	38.19	**11.20**	**0.56**	57.94†	**11.21**	**7.29**
BiC [53]	59.54	36.19	**11.20**	**0.56**	52.25	36.20	**11.20**	**0.56**	49.38	38.50	**11.20**	**0.56**	–	**11.21**	**7.29**
UCIR [16]	65.43	25.76	11.98	**0.56**	62.72	27.24	12.02	**0.56**	53.56	34.19	12.10	**0.56**	–	12.03	**7.29**
WA [12]	65.62	25.41	**11.20**	**0.56**	61.87	25.32	**11.20**	**0.56**	51.59	34.81	**11.20**	**0.56**	59.86†	**11.21**	**7.29**
PODNet [17]	66.86	24.43	11.98	**0.56**	64.85	24.91	12.02	**0.56**	58.76	31.24	12.10	**0.56**	–	12.03	**7.29**
CCFA [50]+PODNet	67.24†	–	11.98	**0.56**	65.50†	–	12.02	**0.56**	**62.91**†	–	12.10	**0.56**	**67.82**†	12.03	**7.29**
MTD [48]+PODNet	**67.64**†	–	11.98	**0.56**	**65.58**†	–	12.02	**0.56**	60.94†	–	12.10	**0.56**	67.21†	12.03	**7.29**
DER [22]	69.54	17.16	39.25	1.95	67.08	18.43	67.27	3.35	66.82	20.82	151.4	7.53	66.87†	61.69	40.1
FOSTER [23]	72.42	15.04	**20.54**	1.03	69.42	17.56	**21.47**	**1.07**	66.05	22.32	**22.10**	**1.10**	68.34†	**21.38**	**13.9**
MEMO [28]	67.78	17.83	32.32	**0.90**	63.77	21.49	53.40	1.23	61.75	23.22	116.7	2.24	68.09†	49.17	14.7
BEEF [29]	72.37	**12.75**	39.27	1.95	69.22	17.85	67.34	3.35	69.31	19.08	151.4	7.53	67.09†	61.69	40.1
CCFA [50]+DER	72.17	14.69	39.25	1.95	**71.47**	**15.20**	67.27	3.35	**69.86**	**19.00**	151.4	7.53	68.53	61.69	40.1
MTD [48]+DER	**73.20**	13.57	39.25	1.95	70.69	15.67	67.27	3.35	66.98	22.07	151.4	7.53	**69.12**	61.69	40.1
RNE	**77.96**	**5.45**	39.84	2.31	**76.18**	**6.25**	67.83	4.07	**75.37**	**8.31**	151.7	9.33	**71.45**	64.64	51.8
RNE-compress	75.56	8.67	**13.68**	**0.72**	73.99	10.70	**15.57**	**0.82**	72.22	13.63	**20.84**	**1.04**	69.72	**15.54**	**11.3**

† denotes that the result is directly cited from the corresponding paper.

2) For RNE, the model is trained for 200 epochs at each session. Following [11], [12], [16], [17], [23], [28], [29], and [63], the learning rate λ is initialized as 0.1 and decreased to zero with a cosine annealing scheduler [64]. The batch size is set to 128 for CIFAR-100 and 256 for other datasets. The SGD optimizer is deployed, where the momentum factor is set to 0.9 and the weight decay is set to 0.0005.

6) Evaluation Metrics: Assuming that there are N tasks and the classification accuracy after learning task t is A_t, then the performance is measured by $\bar{A} = (1/N)\sum_{t=1}^{N} A_t$. If the last accuracy of the upper bound method, i.e., Bound, is $A_{N,\text{Bound}}$, then the forgetting rate $D = A_{N,\text{Bound}} - A_{N,\text{model}}$ is calculated to measure the difference between the upper bound and CIL models. In addition, given the number of model parameters P_t and the floating-point operations (FLOPs) F_t for task t, the average number of parameters $\bar{P} = (1/N)\sum_{t=1}^{N} P_t$ and the average FLOPs $\bar{F} = (1/N)\sum_{t=1}^{N} F_t$ are adopted to measure the memory consumption and the computational cost, respectively.

B. Experimental Results and Analyses

1) CIFAR-100: Table II and Fig. 7 present the experimental results on CIFAR-100, with Rows 1 and 2 show the upper bound and the lower bound of CIL, respectively. Rows 3–9 show the results of single-branch methods, and Rows 10–13 show the results of NE methods. Obviously, NE methods outperform the single-branch methods. In addition, Row 14 shows the results of RNE, whose average accuracy is 3.56%, 3.31%, and 5.51% higher than state-of-the-art (SOTA) methods for N = 5, 10, and 25. As indicated by Row 15, the average accuracy of RNE-compress is 1.16%, 1.12%, and 1.96% higher than SOTA methods, while the average number of parameters is only 34%, 23%, and 14% of RNE for N = 5, 10, and 25. Meanwhile, the average FLOPs is only 31%, 20%, and 11% of RNE for N = 5, 10, and 25, which is the lowest among comparative NE methods. For N = 5, the parameter size and computational complexity of RNE-compress are comparable to single-branch methods. Furthermore, all the CIL methods are compared with the Bound. The forgetting rate D of RNE

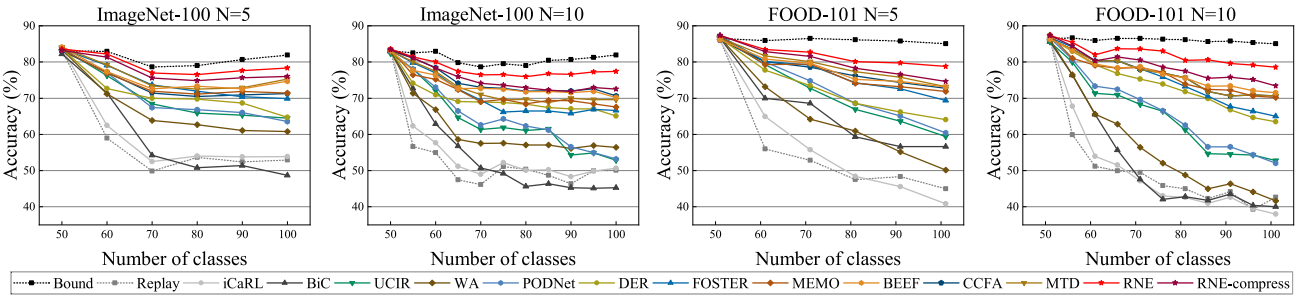

Fig. 8. Accuracy at each session on the ImageNet-100 and FOOD-101 benchmarks.

TABLE III
RESULTS ON THE IMAGENET-100 AND FOOD-101 BENCHMARKS

Methods	ImageNet-100								FOOD-101							
	N=5				N=10				N=5				N=10			
	$\bar{A}\uparrow$	$D\downarrow$	$\bar{P}\downarrow$	$\bar{F}\downarrow$	$\bar{A}\uparrow$	$D\downarrow$	$\bar{P}\downarrow$	$\bar{F}\downarrow$	$\bar{A}\uparrow$	$D\downarrow$	$\bar{P}\downarrow$	$\bar{F}\downarrow$	$\bar{A}\uparrow$	$D\downarrow$	$\bar{P}\downarrow$	$\bar{F}\downarrow$
Bound	81.05	0	11.21	7.29	80.89	0	11.21	7.29	85.96	0	11.21	7.29	86.01	0	11.21	7.29
Replay	58.52	28.98	11.21	7.29	53.18	31.84	11.21	7.29	55.97	40.09	11.21	7.29	50.56	42.41	11.21	7.29
iCaRL [11]	60.08	28.04	**11.21**	**7.29**	54.99	31.24	**11.21**	**7.29**	56.88	44.24	**11.21**	**7.29**	50.30	47.05	**11.21**	**7.29**
BiC [53]	59.80	33.22	**11.21**	**7.29**	54.82	36.62	**11.21**	**7.29**	66.30	28.43	**11.21**	**7.29**	52.85	45.02	**11.21**	**7.29**
UCIR [16]	66.96	21.13	11.99	**7.29**	62.10	29.04	12.03	**7.29**	71.47	25.64	11.99	**7.29**	65.44	32.30	12.03	7.29
WA [12]	67.01	22.12	**11.21**	**7.29**	60.77	29.46	**11.21**	**7.29**	65.06	34.86	**11.21**	**7.29**	56.85	43.43	**11.21**	**7.29**
PODNet [17]	67.69	20.32	11.99	**7.29**	63.23	28.72	12.03	**7.29**	72.53	24.62	11.99	**7.29**	66.48	33.00	12.03	**7.29**
DER [22]	71.57	12.16	39.28	25.5	70.02	16.85	67.30	43.8	72.78	20.97	39.28	25.5	73.77	21.53	67.30	43.8
FOSTER [23]	74.76	12.00	**20.63**	13.4	70.93	15.32	**21.48**	**13.9**	77.14	15.62	**20.63**	13.4	75.79	19.02	**21.48**	**13.9**
MEMO [28]	74.45	10.20	32.31	**11.4**	71.76	14.35	53.40	15.5	77.80	13.06	32.31	**11.4**	76.44	14.87	53.40	15.5
BEEF [29]	75.77	7.18	39.28	25.5	73.86	11.74	67.30	43.8	78.26	11.95	39.28	25.5	77.15	**13.51**	67.30	43.8
CCFA [50]+DER	74.07	10.60	39.28	25.5	**74.09**	**11.30**	67.30	43.8	77.99	12.29	39.28	25.5	76.81	14.45	67.30	43.8
MTD [48]+DER	**76.23**	**6.60**	39.28	25.5	72.90	12.30	67.30	43.8	**79.13**	**11.84**	39.28	25.5	**77.90**	13.59	67.30	43.8
RNE	**79.22**	**3.52**	42.06	32.0	**78.12**	**4.52**	70.26	56.7	**82.00**	**6.28**	42.06	32.0	**82.20**	**6.67**	70.26	56.7
RNE-compress	77.77	5.90	**14.13**	**9.92**	75.37	9.38	**15.89**	**11.6**	80.22	10.41	**14.13**	**9.9**	78.09	12.65	**15.89**	**11.6**

is only 5.45%, 6.25%, and 8.31% for N = 5, 10, and 25. However, the forgetting rate is 11.75%, 16.70%, and 18.62% for the best NE method and 23.13%, 24.91%, and 31.24% for the best single-branch method, indicating that RNE can alleviate catastrophic forgetting effectively. With N increasing from 5 to 25, the forgetting rate D of RNE only increases by 2.86%, whereas that of the SOTA methods increases at least by 6.33%. Therefore, RNE exhibits the best performance for incremental learning with more sessions, and RNE-compress also maintains better performance than existing CIL methods with the smallest number of parameters and FLOPs among NE methods.

2) ImageNet-1K: In order to evaluate the performance of RNE in large-scaled dataset, Table II reports the result on ImageNet-1K following recent CIL works [11], [22], [27], [28], [29], [49]. It is observed that RNE achieves an average accuracy of 71.45% across the ten incremental sessions, which is at least 3.11% higher than existing CIL methods. In addition, RNE adopts the same data augmentation as FOSTER [23], MEMO [28], and BEEF [29].

3) ImageNet-100: Table III and Fig. 8 present the experimental results on ImageNet-100. Comparisons between Rows 3–9 and Rows 10–13 show that the dynamic structure performs better than the single-branch method in terms of average accuracy and forgetting rate. Row 14 shows that the accuracy of RNE is 3.45% and 4.26% higher than methods with dynamic structure for N = 5 and 10, respectively; and its average accuracy is only 1.83% lower than the upper bound. In addition, the forgetting rates D of RNE are only 3.52% and 4.52% for N = 5 and 10, while those of the existing CIL methods are at least 7.18% and 11.74%. Row 15 shows the results of RNE-compress, which also outperforms the existing CIL methods with only 1/3 and 1/5 of the parameters and computational cost of RNE for N = 5 and 10. Therefore, RNE-compress also performs well on high-resolution datasets with less parameters and computational cost than RNE.

4) FOOD-101: Table III and Fig. 8 summarize the experimental results on CIFAR-100, where Rows 1 and 2 show the upper bound and lower bound of CIL respectively; Rows 3–9 show the results of single-branch methods; and Rows 10–13 show the results of NE methods. Similarly, NE methods outperform all the single-branch methods. Row 14 shows the results of RNE, whose average accuracy is 5.54%, 6.74%, and 6.06% higher than existing methods for N = 5, 10, and 25. Row 15 shows the results of RNE-compress, whose average accuracy is 3.14%, 4.57%, and 2.91% higher than the existing methods. Although the performance degrades slightly, the average number of parameters decreases by 34%, 23%, and 14% while the average FLOPs decrease by 31%, 20%, and 11% for N = 5, 10, and 25, respectively. Under N = 5,

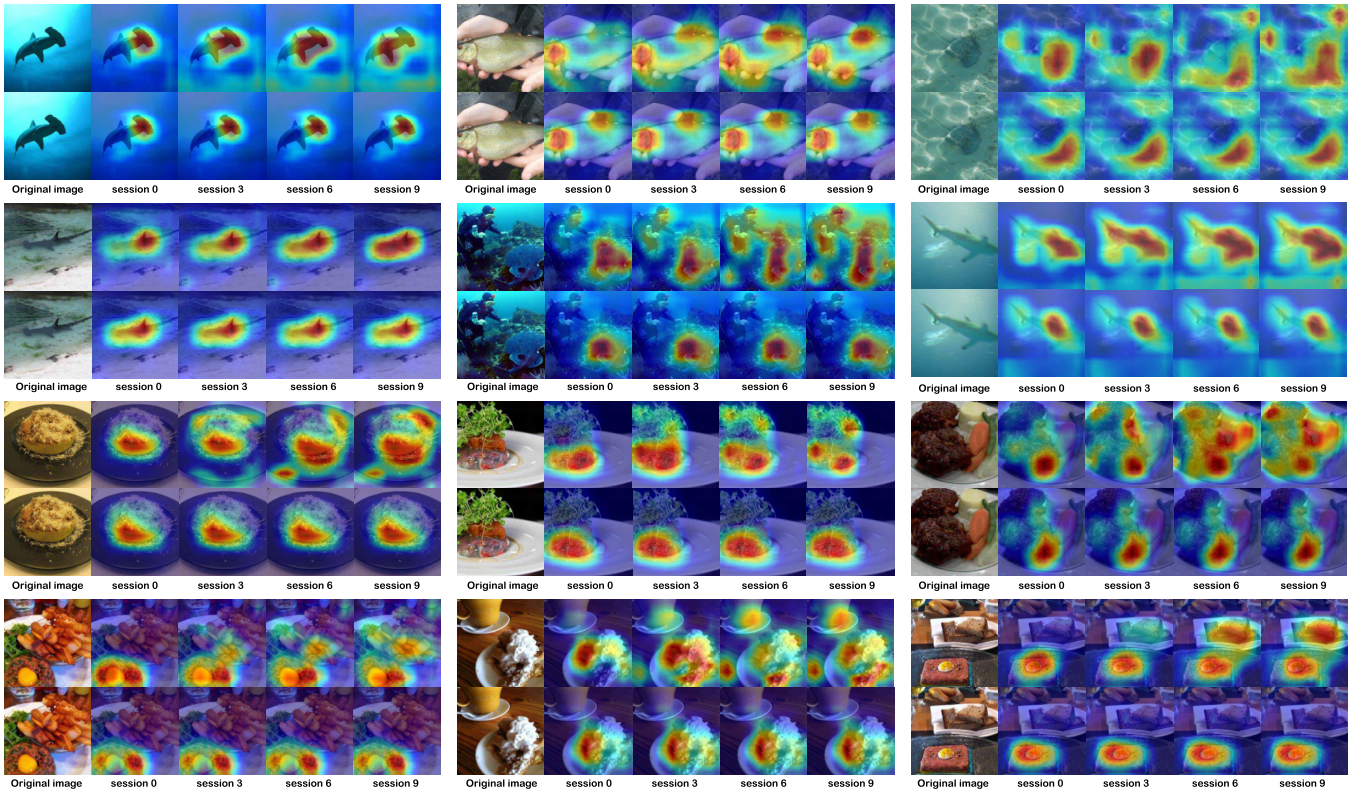

Fig. 9. Visualizations of feature diffusion. For each image sample, the first row provides the visualizations of DER and the second row provides the visualizations of RNE.

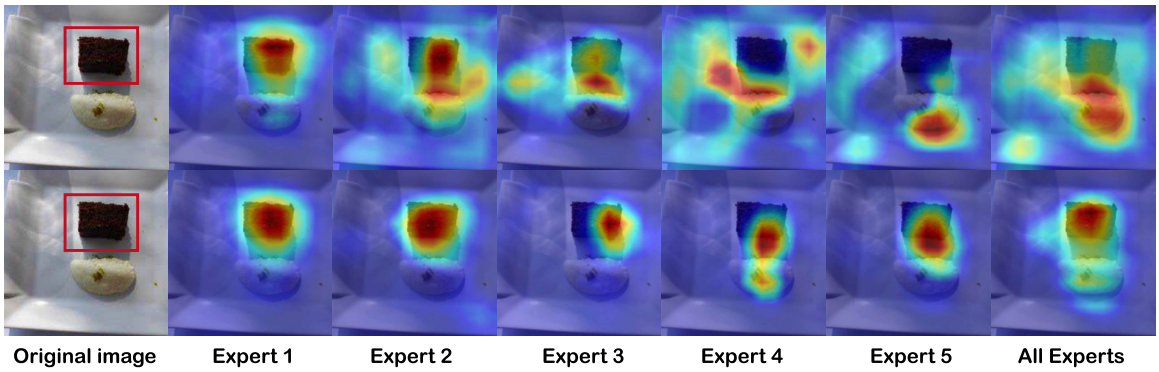

Fig. 10. Visualizations of different experts for a single input, where the ground truth is marked by the red box. The first row shows the regions of interest for each task expert using a conventional NE method without cross-task connection, and the second row shows the regions of interest for each task expert using the proposed RNE.

the parameter size and computational complexity are even comparable to the single-branch method. Furthermore, we compare all the CIL methods with the Bound. The forgetting rate D is only 5.45%, 6.25%, and 8.31% for $N = 5$, 10, and 25, respectively, for RNE; whereas it becomes 12.75%, 17.56%, and 19.08% for the best NE method and 23.13%, 24.91%, and 26.24% for the best single-branch method. Therefore, the performance of RNE approaches the upper bound. In addition, RNE-compress also maintains better performance than other CIL methods with the smallest number of parameters and FLOPs among NE methods.

5) Visualizations and Discussion: Fig. 9 provides more visualizations of feature diffusion, where the attention of a general NE method gradually diffuses to invalid areas, while the RNE is always focusing on the key area. In the following, we will discuss the inherent mechanism of RNE in tackling feature diffusion. The data in incremental learning can be divided into two types, i.e., old task data and new task data; and the task experts can also be split into two types, i.e., previous task experts and new task experts. Accordingly, there are totally four scenarios: 1) old task data and previous task expert; 2) new task data and previous task expert; 3) old task data and new task expert; and 4) new task data and new task expert. Only scenario 3) can cause feature diffusion, as the new task expert cannot learn from the old tasks and fails to obtain their proper feature representations.

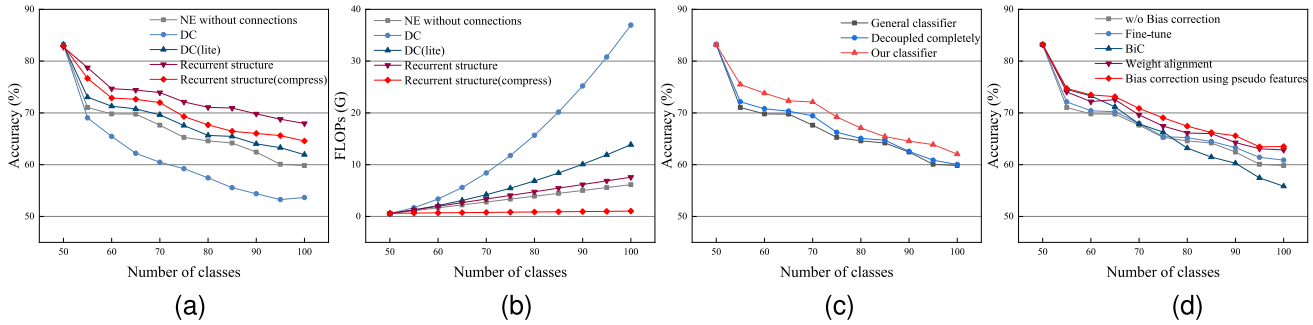

Fig. 11. Comparison of each module. (a) Recurrent structure. (b) FLOPs comparison. (c) Decoupled classifier. (d) Bias correction.

TABLE IV
CONTRIBUTION OF EACH COMPONENT

CIFAR-100 N=10				
Recurrent structure	Decoupled classifier	Bias correction	$\bar{A}\uparrow$	$D\downarrow$
Baseline			67.08	18.43
✔			73.20	10.31
✔	✔		75.13	7.73
✔		✔	74.58	7.57
✔	✔	✔	**76.18**	**6.25**

In the following, we perform a comprehensive analysis of the feature output by each task expert by feeding an image from the initial task into the incrementally trained model. The mechanism of feature diffusion is visualized by the Grad-CAM images in Fig. 10, where the ground truth is marked by the red box. Specifically, the first row provides the visualization results of each task expert in a conventional NE method without cross-task connection, and the second row provides the visualization results of each task expert in the proposed method. In addition, the last column presents the synthesized result of all task experts. For the first row, it is observed that only the first task expert focuses on the target region, while the others gradually become defocused. In addition, the last image shows severe feature diffusion, which is induced by progressive accumulation of erroneous feature representations extracted by subsequent task experts. On the contrary, the task experts in the second row focus on similar target regions, indicating that the proposed method alleviates feature diffusion by guiding each task expert to maintain a similar representation for the old category.

C. Ablation Study

To verify the effectiveness of each component in RNE, we conduct an ablation study on CIFAR-100 with $N = 10$. In Table IV, Row 1 shows the results of the baseline method, i.e., DER [22]; Row 2 shows the results after adding the recurrent structure (see Section III-C) to the baseline; Rows 3 and 4 show the results after adding the decoupled classifier (see Section III-D) and bias correction (see Section III-E), respectively, to the recurrent structure; and Row 5 shows the results after adding all the three modules.

We also demonstrate the validity of each module, i.e., recurrent structure, decoupled classifier, and bias correction,

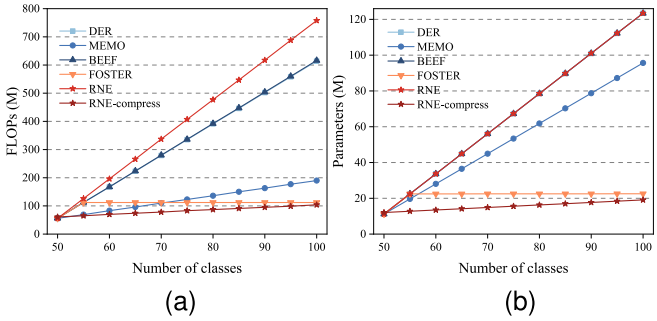

Fig. 12. Comparison of each method in FLOPs and parameters. The backbone network is ResNet-18 and the size of input images is 32×32 (in CIFAR-100 ten-step setting). (a) FLOPs. (b) Parameters.

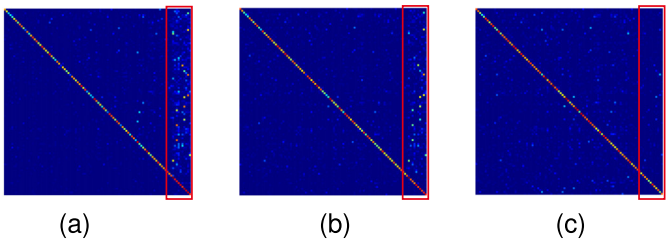

Fig. 13. Comparisons of the confusion matrices. (a) DER. (b) RNE without bias correction. (c) RNE with bias correction.

by applying them to the baseline independently. In Fig. 11(a), DC refers to a dense connection that exists in each layer between the new task expert and all previous ones, and DC (lite) refers to a dense connection that only exists in the key layers between the new task expert and all previous ones. Both of them originate from PNN [33]. It is observed that the recurrent structure and the compressed version outperform DC and DC (lite). Fig. 11(b) shows the FLOPs at each session, which indicates that the computational cost of DC and DC (lite) grows quadratically (since $((t-1)t/2)$ connections for task t are constructed at a layer), thereby resulting in nonsustainable NE. On the contrary, the FOLPs of the recurrent structure is similar to that of the NE method, and the FLOPs of the compressed version is much lower than that of the NE method. In Fig. 11(c), the proposed decoupled classifier is compared with a general classifier and a completely decoupled classifier [25]. Obviously, the proposed classifier performs better as it learns the causality of feature

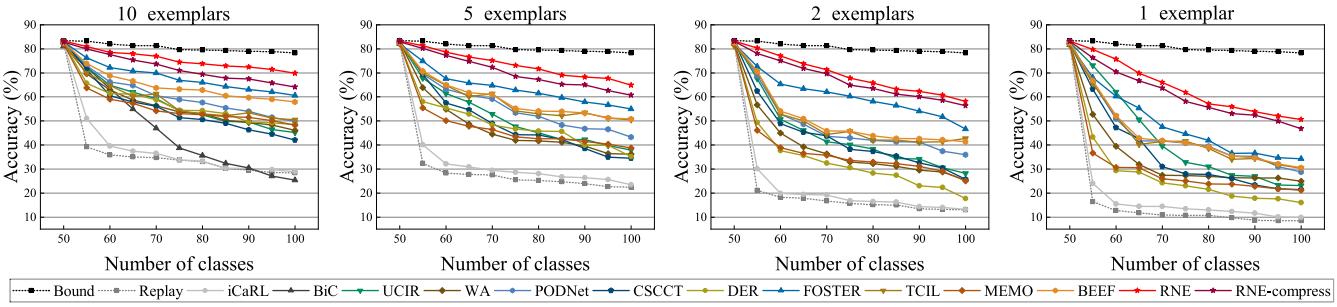

Fig. 14. Accuracy at each session in the robustness test on the CIFAR-100 dataset.

TABLE V
RESULTS OF ROBUSTNESS TEST

Methods	CIFAR-100 N=10									
	20 exemplars		10 exemplars		5 exemplars		2 exemplars		1 exemplar	
	$\bar{A}\uparrow$	$D\downarrow$	$\bar{A}\uparrow$	$D\downarrow$	$\bar{A}\uparrow$	$D\downarrow$	$\bar{A}\uparrow$	$D\downarrow$	$\bar{A}\uparrow$	$D\downarrow$
Replay	46.85	38.69	37.80	49.84	31.16	55.83	21.98	65.06	17.34	69.80
iCaRL [11]	49.87	38.13	39.66	49.37	33.97	54.85	23.83	64.26	20.07	68.26
BiC [53]	52.25	36.20	46.23	52.90	–	–	–	–	–	–
UCIR [16]	62.72	27.24	58.15	33.10	52.49	40.42	44.95	50.13	42.98	54.88
WA [12]	61.87	25.32	57.35	32.15	48.34	41.86	39.93	52.86	35.60	53.37
PODNet [17]	**64.85**	**24.91**	**61.38**	**28.46**	**56.86**	**35.01**	**48.90**	**42.31**	**44.92**	**49.56**
CSCCT [18]	60.03	29.91	55.75	36.24	50.23	43.74	43.94	52.57	37.68	56.45
DER [22]	67.08	18.43	58.07	29.81	49.98	38.40	35.14	60.50	29.34	60.65
FOSTER [23]	69.42	17.56	**68.66**	**17.67**	**64.91**	**23.28**	**62.81**	**31.63**	**49.78**	**44.01**
TCIL [27]	**72.87**	**16.70**	64.72	21.49	59.43	27.81	50.54	35.61	44.85	47.94
MEMO [28]	63.77	21.49	56.72	30.04	48.27	39.53	38.49	53.25	31.29	56.51
BEEF [29]	69.22	17.85	65.37	20.30	59.99	27.47	51.14	34.45	45.37	47.72
RNE	**76.18**	**6.25**	**75.69**	**8.45**	**73.60**	**13.43**	**69.42**	**20.16**	**64.20**	**27.63**
RNE-compress	73.99	10.70	72.27	14.14	70.64	17.50	67.50	21.91	61.47	31.51

sequence. In Fig. 11(d), the bias correction is compared with the existing postprocessing methods, i.e., Fine-tune [22], BiC [53], and weight alignment [12]. It shows that BiC only works well in early sessions and fine-tuning has limited performance, while the proposed bias correction strategy achieves the best performance.

Fig. 12 shows the variation of FLOPs (which indicates the computational cost) and the number of parameters (which indicates the memory usage) for the proposed and comparative methods. It is observed that the computational cost and the number of parameters for RNE are almost the same as those of DER and BEEF. In contrast, the RNE-compress demonstrates the smallest computational cost and number of parameters.

Fig. 13 provides the visualization of confusion matrices, where misclassification of old categories decreases significantly, indicating that the preference for new tasks can be suppressed effectively.

D. Robustness Test

In this section, we use the CIFAR-100 dataset to conduct robustness tests on RNE by reducing the capacity of the exemplar set. The original capacity is 2000 images with 20 images per category, accounting for 4% of the total training images. Then, the capacity is reduced to 50%, 25%, and 10% of the original, i.e., 10, 5, and 2 images per category, respectively. As shown in Table V, the average accuracy of the normal dynamic method DER decreases rapidly with the reduction of the exemplar set. In addition, most CIL methods exhibit strong dependence on the exemplar set. On the contrary, the RNE exhibits more robustness to the reduction of exemplar-set capacity. As illustrated in Fig. 14, the accuracy curves of RNE decrease relatively more slowly than other CIL methods as: 1) the recurrent structure receives information of previous tasks from previous task experts and becomes less dependent on old exemplars and

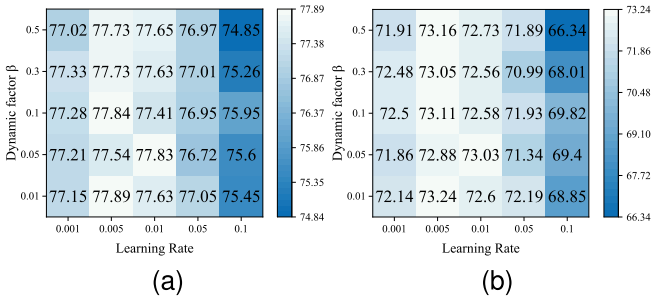

Fig. 15. Evaluation of the hyperparameter sensitivity. (a) Average accuracy. (b) Last accuracy.

2) the bias correction strategy can mitigate the classifier bias effectively.

In addition, we evaluate the sensitivity of hyperparameters, i.e., the learning rate λ and the initial value of dynamic factor β for the proposed RNE by conducting experiments on CIFAR-100 with $N = 5$. Specifically, λ is chosen from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ and β is chosen from $\{0.01, 0.05, 0.1, 0.3, 0.5\}$. The average accuracy is shown in Fig. 15(a) and the last accuracy is shown in Fig. 15(b). The results indicate that the proposed method represents robustness to hyperparameters.

V. Conclusion

This article proposed the RNE, i.e., RNE, for CIL by constructing connections among task experts elegantly. Then, the RNE was compressed to reduce the number of parameters and FLOPs while maintaining superior CIL performance. In addition, the classifier is decoupled in a more causal way to reduce feature confusion and avoid overfitting and was retrained with pseudo features to address the issue of classifier bias. Experiments have shown that RNE outperforms existing CIL methods with less complexity and exhibits robustness to restricted exemplar-set capacity.

Future work will be focused on CIL of image series and on designing models applicable to open environments, such as the emergence of unknown categories.

References

[1] M. De Lange et al., "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3366–3385, Jul. 2022.

[2] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online continual learning in image classification: An empirical survey," *Neurocomputing*, vol. 469, pp. 28–51, Jan. 2022.

[3] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: Survey and performance evaluation on image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5513–5533, May 2023.

[4] R. French, "Catastrophic forgetting in connectionist networks," *Trends Cognit. Sci.*, vol. 3, no. 4, pp. 128–135, Apr. 1999.

[5] L. Golab and M. T. Özsu, "Issues in data stream management," *ACM SIGMOD Rec.*, vol. 32, no. 2, pp. 5–14, Jun. 2003.

[6] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, vol. 24. Amsterdam, The Netherlands: Elsevier, 1989, pp. 109–165.

[7] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[8] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, 2017.

[9] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 139–154.

[10] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3987–3995.

[11] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2001–2010.

[12] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia, "Maintaining discrimination and fairness in class incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13208–13217.

[13] Y. Liu, B. Schiele, and Q. Sun, "RMM: Reinforced memory management for class-incremental learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 3478–3490.

[14] A. Robins, "Catastrophic forgetting in neural networks: The role of rehearsal mechanisms," in *Proc. 1st New Zealand Int. Two-Stream Conf. Artif. Neural Netw. Expert Syst.*, Nov. 1993, pp. 65–68.

[15] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Sci.*, vol. 7, no. 2, pp. 123–146, Jun. 1995.

[16] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 831–839.

[17] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle, "PODNet: Pooled outputs distillation for small-tasks incremental learning," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 86–102.

[18] A. Ashok, K. Joseph, and V. N. Balasubramanian, "Class-incremental learning with cross-space clustering and controlled transfer," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 105–122.

[19] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers Psychol.*, vol. 4, p. 504, Mar. 2013.

[20] S. Grossberg, "Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world," *Neural Netw.*, vol. 37, pp. 1–47, Jan. 2013.

[21] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3366–3375.

[22] S. Yan, J. Xie, and X. He, "DER: Dynamically expandable representation for class incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 3014–3023.

[23] F. Y. Wang, D. W. Zhou, H. J. Ye, and D.-C. Zha, "FOSTER: Feature boosting and compression for class-incremental learning," in *Proc. 17th Eur. Conf. Comput. Vision (ECCV)*, vol. 13685, Jun. 2022, pp. 398–414.

[24] T.-Y. Wu et al., "Class-incremental learning with strong pre-trained models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 9591–9600.

[25] A. Douillard, A. Ramé, G. Couairon, and M. Cord, "DyTox: Transformers for continual learning with dynamic token expansion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 9285–9295.

[26] Z. Hu, Y. Li, J. Lyu, D. Gao, and N. Vasconcelos, "Dense network expansion for class incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 11858–11867.

[27] B. Huang, Z. Chen, P. Zhou, J. Chen, and Z. Wu, "Resolving task confusion in dynamic expansion architectures for class incremental learning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 1, pp. 908–916.

[28] D.-W. Zhou, Q. Wang, H.-J. Ye, and D. Zhan, "A model or 603 exemplars: Towards memory-efficient class-incremental learning," in *Proc. 11th Int. Conf. Learn. Represent.*, May 2023, pp. 9689–9703.

[29] F.-Y. Wang et al., "Beef: Bi-compatible class-incremental learning via energy-based expansion and fusion," in *Proc. 11th Int. Conf. Learn. Represent.*, May 2023, pp. 12773–12787.

[30] Y. Liu, X. Hong, X. Tao, S. Dong, J. Shi, and Y. Gong, "Model behavior preserving for class-incremental learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7529–7540, Oct. 2023.

[31] Z. Zhang, Y. Chen, and C. Zhou, "Self-growing binary activation network: A novel deep learning model with dynamic architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 624–633, Jan. 2024.

[32] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.

[33] A. A. Rusu et al., "Progressive neural networks," 2016, *arXiv:1606.04671*.

[34] J. Schwarz et al., "Progress & compress: A scalable framework for continual learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4528–4537.

[35] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," 2017, *arXiv:1708.01547*.

[36] D.-W. Zhou, Y. Yang, and D.-C. Zhan, "Learning to classify with incremental new class," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2429–2443, Jun. 2022.

[37] J. Zhang, T. Wang, W. W. Y. Ng, and W. Pedrycz, "KNNENS: A k-nearest neighbor ensemble-based method for incremental learning under data stream with emerging new classes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 9520–9527, Nov. 2023.

[38] A. Razdaibiedina, Y. Mao, R. Hou, M. Khabsa, M. Lewis, and A. Almahairi, "Progressive prompts: Continual learning for language models," 2023, *arXiv:2301.12314*.

[39] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.

[40] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2066–2073.

[41] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 532–547.

[42] Y. Yang, D.-W. Zhou, D.-C. Zhan, H. Xiong, and Y. Jiang, "Adaptive deep models for incremental learning: Considering capacity scalability and sustainability," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 74–82.

[43] D. López-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Apr. 2017, pp. 1–11.

[44] S. Wang, X. Li, J. Sun, and Z. Xu, "Training networks in null space of feature covariance for continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 184–193.

[45] X. Wang, Z. Ji, Y. Yu, Y. Pang, and J. Han, "Model attention expansion for few-shot class-incremental learning," *IEEE Trans. Image Process.*, vol. 33, pp. 4419–4431, 2024.

[46] J. Liu, Z. Ji, Y. Pang, and Y. Yu, "NTK-guided few-shot class incremental learning," *IEEE Trans. Image Process.*, vol. 33, pp. 6029–6044, 2024.

[47] Z. Ji, Z. Jiao, Q. Wang, Y. Pang, and J. Han, "Imbalance mitigation for continual learning via knowledge decoupling and dual enhanced contrastive learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 2, pp. 3450–3463, Feb. 2025.

[48] H. Wen et al., "Class incremental learning with multi-teacher distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 28443–28452.

[49] Z. Luo, Y. Liu, B. Schiele, and Q. Sun, "Class-incremental exemplar compression for class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 11371–11380.

[50] T. Kim, J. Park, and B. Han, "Cross-class feature augmentation for class incremental learning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 13168–13176.

[51] S. Ho, M. Liu, L. Du, L. Gao, and Y. Xiang, "Prototype-guided memory replay for continual learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 8, pp. 10973–10983, Aug. 2024.

[52] A. Vaswani et al., "Attention is all you need," 2017, *arXiv:1706.03762*.

[53] Y. Wu et al., "Large scale incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 374–382.

[54] J. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Jun. 1990.

[55] A. Graves and A. Graves, "Long short-term memory," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Germany: Springer, 2012, pp. 37–45.

[56] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Tech. Rep., 2009. [Online]. Available: https://www.cs.utoronto.ca/$\sim$kriz/learning-features-2009-TR.pdf

[57] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[58] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101—Mining discriminative components with random forests," in *Proc. 13th Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2014, pp. 446–461.

[59] M. Welling, "Herding dynamical weights to learn," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, Jun. 2009, pp. 1121–1128.

[60] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation strategies from data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 113–123.

[61] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8026–8037.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[63] D.-W. Zhou, H.-J. Ye, and D.-C. Zhan, "Co-transport for class-incremental learning," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 1645–1654.

[64] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2016, *arXiv:1608.03983*.