# Relational Experience Replay: Continual Learning by Adaptively Tuning Task-wise Relationship

Quanziang Wang, Renzhen Wang, Yuexiang Li, Dong Wei, Hong Wang, Kai Ma, Yefeng Zheng, *Fellow, IEEE*, Deyu Meng, *Member, IEEE*

***Abstract*—Continual learning is a promising machine learning paradigm to learn new tasks while retaining previously learned knowledge over streaming training data. Till now, *rehearsal-based* methods, keeping a small part of data from old tasks as a memory buffer, have shown good performance in mitigating catastrophic forgetting for previously learned knowledge. However, most of these methods typically treat each new task equally, which may not adequately consider the relationship or similarity between old and new tasks. Furthermore, these methods commonly neglect sample importance in the continual training process and result in sub-optimal performance on certain tasks. To address this challenging problem, we propose Relational Experience Replay (RER), a bi-level learning framework, to adaptively tune task-wise relationships and sample importance within each task to achieve a better 'stability' and 'plasticity' trade-off. As such, the proposed method is capable of accumulating new knowledge while consolidating previously learned old knowledge during continual learning. Extensive experiments conducted on three benchmark image datasets (CIFAR-10, CIFAR-100, and Tiny ImageNet) and two text datasets (20News and DBpedia) show that the proposed method can consistently improve the performance of all baselines and surpass current state-of-the-art methods.**

***Index Terms*—Continual learning, stability-plasticity dilemma, bi-level optimization.**

## I. Introduction

DEEP neural networks have demonstrated remarkable performance across a wide range of machine learning tasks, including classification [1]–[3], semantic segmentation [4]–[6], and object detection [7]–[9]. Nonetheless, these models often face challenges in continual learning (CL) scenarios, where knowledge accumulates incrementally from data generated by a non-stationary distribution. Specifically, the models tend to overwrite previously acquired knowledge whenever new tasks come in, resulting in catastrophic forgetting [10], which poses a significant challenge in continual learning.

Manuscript received May 22, 2023; revised Jan 10, 2024; accepted April 20, 2024. This work was supported the National Key R&D Program of China (2022YFA1004100), and the China NSFC projects under contract 62306233, 12226004, 62272375. The Associate Editor coordinating the review of this manuscript and approving it for publication was Professor Sanghoon Lee. *(Corresponding author: Renzhen Wang.)*

Quanziang Wang and Renzhen Wang are with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, P.R. China (e-mail: quanziangwang@gmail.com, rzwang@xjtu.edu.cn).

Yuexiang Li, Dong Wei, Hong Wang, Kai Ma, and Yefeng Zheng are with Tencent Jarvis Lab, Shenzhen 518052, P.R. China (e-mail: vicyxli@tencent.com, donwei@tencent.com, kylekma@tencent.com, hazelhwang@tencent.com yefengzheng@tencent.com).

Deyu Meng is with the School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaanxi, China, and Macao Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macao (e-mail: dymeng@mail.xjtu.edu.cn.

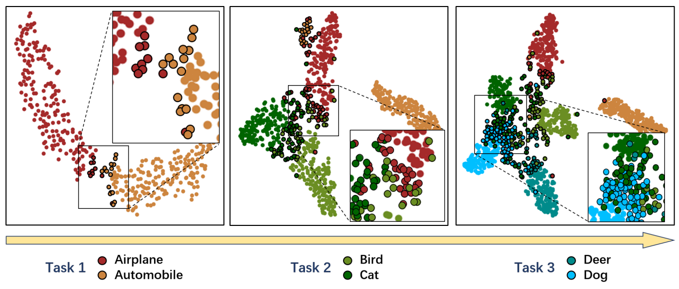


Fig. 1. Illustration of two main factors affecting stability and plasticity in continual learning: (i) The relationship between the new and old tasks varies dynamically. For example, the class 'bird' of task 2 is semantic-related to the 'airplane' of task 1 due to the closer distance between these two clusters in the latent space. (ii) The importance of data points within each class is different for model training. The samples with blue borders, located around the classification boundary, are more difficult/beneficial for the classification than other samples.

Recently, various CL methods have been proposed to alleviate the catastrophic forgetting problem. Existing algorithms for CL can be roughly divided into three categories [11]: 1) *Model-based* approaches [12]–[16], which dynamically modify the architecture of the model to handle the new task and use specific model structures for different tasks in the testing phase; 2) *Regularization-based* approaches [17]–[21], which commonly impose constraints to maintain the model parameters that are relatively important for previously learned tasks; 3) *Rehearsal-based* approaches [22]–[30], which store a small subset of examples from previously learned tasks in a memory buffer and replay them during the learning of new tasks. In this study, we mainly focus on exploring the potential of *rehearsal-based* methods, as they have shown relatively stable and effective performance in addressing the catastrophic forgetting in continual learning.

*Rehearsal-based* methods aim to strike an appropriate balance between learning new information and retaining previous knowledge, which is referred to as the 'stability-plasticity' dilemma [31]. Many *rehearsal-based* methods, such as Experience Replay (ER) [22], [23] and DER++ [26], primarily concentrate on leveraging stored information from previous tasks to mitigate model forgetting during the training of new tasks. However, these approaches have paid limited attention to modeling the intricate relationships between tasks. For example, the loss function of ER is the sum of the losses of incoming new task samples and old task ones stored in the memory buffer: $\mathcal{L}_{new} + \lambda\mathcal{L}_{old}$, where $\lambda$ is a hyperparameter and it is often tuned manually by grid search at the beginning of the training process to balance the trade-off between stability and plasticity. This trade-off hyperparameter generally necessitates careful pre-specification to ensure a stable training tendency throughout the entire continual learning process.

Actually, as shown in Fig. 1, under the settings of CL, the data streaming of new tasks inclines to induce the continual variation of data distribution. Therefore, the relationship between old and new tasks dynamically varies throughout the learning process. Taking image recognition as an example, the knowledge extracted from previous tasks (*e.g.*, cats) may be heavily influenced by semantically similar new tasks (*e.g.*, dogs), but less affected by unrelated ones (*e.g.*, automobiles). Obviously, the continual model should be capable of adjusting its focus in real-time between prioritizing stability in old tasks and accommodating plasticity to adapt to new tasks as their relationship evolves. However, it is difficult for existing methods to model such a complicated task-wise relationship by simply tuning and fixing hyperparameters at the beginning of network training. Moreover, previous studies do not extensively consider another intrinsic factor, *i.e.,* sample importance within the same class for feature representation learning. Concretely, easy-to-learn samples contain more representative knowledge for each class, while the 'hard' ones, located around decision boundaries for classification, can contribute to the learning of the classifier [32] (refer to Fig. 1). From this perspective, it is essential to model the relationship between new and old tasks and sample importance within the same class.

To this end, we propose *Relational Experience Replay* (RER), a bi-level learning framework to couple the aforementioned complicated task relationship and sample importance. Concretely, the inner-loop optimization problems aim to address the 'plasticity' by leveraging the examples from the newly coming task, while the outer-loop optimization problem of RER aims to address 'stability' by minimizing the empirical risk of a batch of balanced data (including samples from the memory buffer and new tasks). To fully leverage training samples by considering the inter-task relationship and sample importance, we design an explicit weighting function parameterized by a lightweight neural network (dubbed Relation Replay Net, or RRN). RRN maps pairwise abstract information of samples from new and old tasks to their corresponding loss weights. We theoretically prove that the RRN is updated based on the similarity of the average gradient between classes in the outer-loop optimization, indicating that our proposed RER can implicitly model task-wise relationships. We conduct extensive experiments on different benchmark datasets, and the results consistently demonstrate improvements over various baselines for *rehearsal-based* continual learning methods. In summary, our contributions are mainly four-fold:

1) The proposed method takes two factors, *i.e.*, task relationship across the whole continual learning process and sample importance within each class, into account for sample weight assignment. Such a design facilitates the model to deal with the 'stability-plasticity' dilemma that plagues the continual learning paradigm.

2) We theoretically prove that the proposed method can implicitly model the task relationship. Specifically, the updating formulation of the Relation Replay Net depends on the similarity between the gradient of each training sample and the averaged gradient of each class stored in the memory buffer.

3) As far as we know, in the continual learning problem, we are the first to propose an automated sample weighting strategy to adaptively assign a reasonable weight to each sample from the new and old tasks, which is more flexible than existing approaches based on manual tuning.

4) The proposed method can be applied to various *rehearsal-based* continual learning baselines and consistently improves their performance under various settings.

This paper is organized as follows. Section II provides a review of some related works and Section III briefly introduces the setting and necessary notations for continual learning. Section IV presents the proposed RER method in detail. Section V then provides the experiments and analysis of our method. The paper is finally concluded in Section VI.

## II. Related Work

### A. Continual Learning

**Rehearsal-based Methods.** The primary mechanism underlying *rehearsal-based* methods [22]–[29], [33] is using the information of data from old tasks to prevent forgetting while training new tasks. Specifically, these methods typically involve saving a portion of data samples from old tasks as a memory buffer, which is subsequently used alongside new incoming data samples to train the model. For example, GEM [25] formulates a quadratic programming problem to enforce orthogonality between the optimization direction for a new task and those previously stored in the memory buffer during training. A-GEM [34] relaxes the constraints in GEM by only restricting the dot product of the new and old sample gradients to be non-negative, so as to improve the computational efficiency of the algorithm. iCaRL [24] stores the most representative samples, *i.e.*, located around the class center in the latent space as the memory buffer, and uses the Nearest Class Mean (NCM) classifier to mitigate the impact of feature representation changes. Rainbow Memory [28] constructs the memory buffer by sampling more representative samples, determined based on the prediction confidence of the data after applying various augmentations. In addition to the ground truth labels, DER++ [26] also saves data probabilities yielded by the model from previous epochs in the memory buffer as soft labels, which are used for distillation to further prevent forgetting. On the flip side, *rehearsal-based* methods commonly lead to class imbalance problems, as they rely on a small memory buffer to store a limited number of examples from old tasks. To address this problem, LUCIR [33] normalizes the predicted logits and imposes constraints on the features to correct the imbalance problem between new and old samples. ER-ACE [29] calculates the loss function of the new and old tasks independently to alleviate the mutual interference between them.

**Other Continual Learning Methods.** *Model-based* approaches [12]–[16], [35] aim to enhance the model's adaptability to new tasks by automatically modifying its architecture. Despite their impressive performance in simulation experiments, these methods are known to pose challenges in training and optimization, often requiring significant computational resources. For example, PNN [12] saves all networks of previous tasks to avoid forgetting, which often occupies a large memory buffer and needs to train another network for the new task. *Regularization-based* approaches [17]–[21], [36] design different constraints to prevent changing important parameters of previous tasks during training new tasks. Typically, EWC [17] limits the updating of important parameters, which are selected by the Fisher matrix from a Bayesian perspective. SI [19] determines important parameters by evaluating the impact of parameter changes on the loss function. LwF [18] saves previous model predictions of new task samples as soft labels to distill the extracted knowledge. Despite their simplicity and effectiveness, these approaches also face challenges such as hyper-parameter tuning and sensitivity to the choice of regularization parameters. Besides, continual learning is also applicable to various realistic problems, such as semantic segmentation [37]–[39], few-shot learning [40]–[42], and emotion detection [43], [44], etc. For more details, we recommend referring to the literature provided in [11], [45].

### B. Sample Weighting Strategy

In terms of sample weighting, our method is closely related to L2RW [46] and Meta Weight Net (MW-Net) [47]. These approaches involve training a classification network on a noisy label dataset and being optimized through a meta-learning strategy to mitigate the impact of noisy labels. This meta-learning process is guided by a small clean dataset, referred to as the meta set. However, the challenges posed by CL are different from those of the noisy label problem. Specifically, the unstable data flow and severe catastrophic forgetting make it challenging to directly apply these sample weighting methods to CL. To mitigate this problem, we propose a novel pairwise sample weighting strategy to model task relationships and it does not require any additional high-quality data as the meta set like L2RW or MW-Net due to the limitation of continual learning settings. To our knowledge, this should be the first work to use the automatic sample weighting strategy for continual learning.

## III. Preliminaries

In this section, we briefly introduce the settings of the continual learning problem and two main *rehearsal-based* baselines. In continual learning, the model $f$ is required to learn a stream of tasks $\mathcal{T} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots\}$ and for each task $\mathcal{D}_t$, only a few samples can be stored in a memory buffer $\mathcal{M}$, where the buffer size is denoted as $M$. Typically, the model $f : \mathcal{X} \rightarrow \mathbb{R}^{C_t}$ is a classification neural network parameterized by $\theta$, where $C_t$ denotes the total number of classes across tasks 1 to $t$. In this study, we mainly focus on the *Class Incremental* (Class-IL) and *Task Incremental* (Task-IL) settings of continual learning. Specifically, in Task-IL, the task boundary is clear, *i.e.,* the task ID $t$ is known during both training and testing phases. Therefore, the model can feed the input data to the corresponding task-specific classifier using masks [26] based on its task ID. Conversely, Class-IL is more challenging as the task ID remains inaccessible.

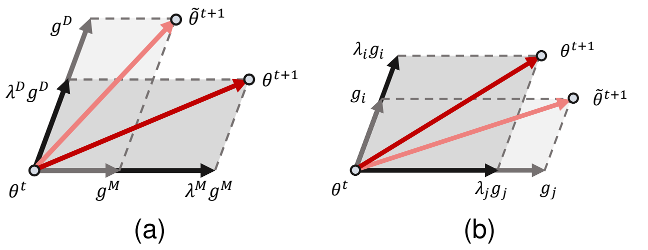


Fig. 2. Illustration of sample weights modeling (a) task relationships and (b) sample importance during training. Here, $g^D$ and $g^M$ represent gradients of new and old tasks, with $\lambda^D$ and $\lambda^M$ denoting their corresponding weights. Similarly, $g_i$ and $g_j$ are gradients of $x_i$ and $x_j$, with $\lambda_i$ and $\lambda_j$ denoting their corresponding weights.

### A. Experience Replay (ER)

ER is a fundamental *rehearsal-based* approach that aims to mitigate the forgetting of previously learned tasks while learning new ones by replying to samples stored in the memory buffer. Specifically, during each task training, it samples a batch of data $\mathcal{B}^D = \{(x_i^D, y_i^D)\}_{i=1}^B$ from the current $t$-th task $\mathcal{D}_t$ and another batch of data $\mathcal{B}^M = \{(x_i^M, y_i^M)\}_{i=1}^B$ from the memory buffer $\mathcal{M}$ with the same batch size $B$, where $x_i$ and $y_i$ represent an example and its corresponding label, respectively. The loss function of ER can be formulated as:

$$\mathcal{L}^{tr}(\theta) = \frac{1}{B}\sum_{i=1}^{B} \lambda_i^D L_{CE}(x_i^D; \theta) + \lambda_i^M L_{CE}(x_i^M; \theta), \tag{1}$$

where $L_{CE}$ denotes the cross-entropy (CE) loss, and $\lambda_i^D$ and $\lambda_i^M$ represent the sample weights. A prevalent convention is to set $\lambda_i^D = 1$ and $\lambda_i^M = \lambda$. The value of $\lambda$ is a crucial hyperparameter that necessitates manual specification before training in many previous works.

### B. Sample Weights in Continual Learning

Considering the stochastic gradient descent (SGD) of Eq. (1), we can get the updating formulation as

$$\theta^{t+1} = \theta^t - \eta \frac{1}{B}\sum_{i=1}^{B} \lambda_i^D g_i^D + \lambda_i^M g_i^M, \tag{2}$$

where $\eta$ is the learning rate, and $g_i^D$ and $g_i^M$ are gradients of $x_i^D$ and $x_i^M$, respectively. From this equation, we can find that the weights $\lambda_i^D$ and $\lambda_i^M$ control the direction of gradient descent from two aspects. Firstly, as shown in Fig. 2(a), when the new task weights surpass those of the memory buffer, the total gradient will tend to descend more along the direction of old tasks, and vice versa. Moreover, for any different training samples $x_i$ and $x_j$, their sample weights also adjust the gradient direction as illustrated in Fig. 2(b). Consequently, proper sample weights can enhance the model training by effectively capturing the relationship between new and old tasks, as well as the sample importance within each class. Therefore, the fixed weights in prior studies might yield sub-optimal results and need further improvement.

## IV. Relational Experience Replay

Most existing works alleviate the model forgetting problem by storing more information about old tasks, potentially

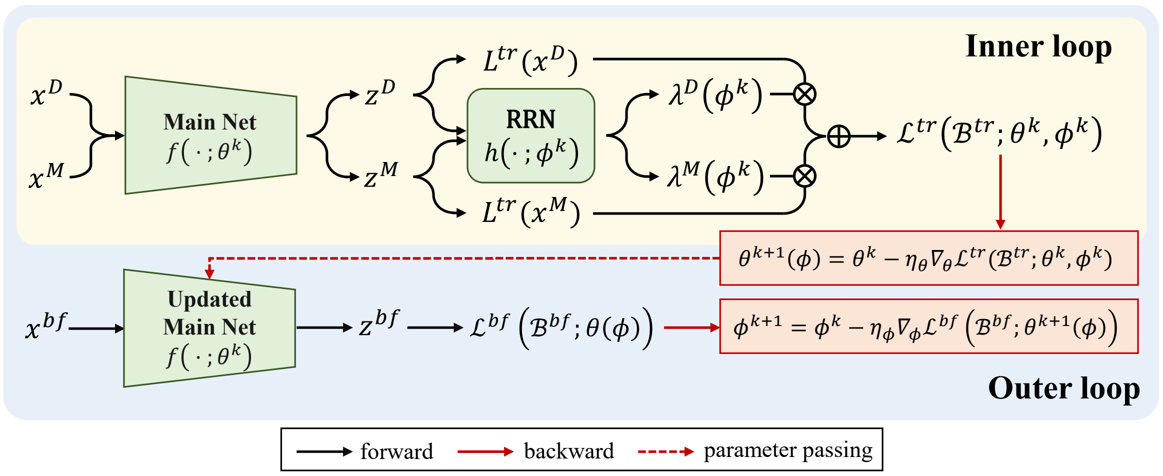

Inner loop
$x^D$
$x^M$
Main Net $f(\cdot;\theta^k)$
$z^D$
$z^M$
$L^{tr}(x^D)$
$L^{tr}(x^M)$
RRN $h(\cdot;\phi^k)$
$\lambda^D(\phi^k)$
$\lambda^M(\phi^k)$
$\mathcal{L}^{tr}(\mathcal{B}^{tr};\theta^k,\phi^k)$
$\theta^{k+1}(\phi)=\theta^k-\eta_\theta\nabla_\theta\mathcal{L}^{tr}(\mathcal{B}^{tr};\theta^k,\phi^k)$
$x^{bf}$
Updated Main Net $f(\cdot;\theta^k)$
$z^{bf}$
$\mathcal{L}^{bf}(\mathcal{B}^{bf};\theta(\phi))$
$\phi^{k+1}=\phi^k-\eta_\phi\nabla_\phi\mathcal{L}^{bf}(\mathcal{B}^{bf};\theta^{k+1}(\phi))$
Outer loop
forward
backward
parameter passing

Fig. 3. The main illustration of the proposed Relational Experience Replay at $k$-th iteration. In the inner loop, the Main Net is trained on the batch of paired data $\mathcal{B}^{tr}$. In the inner loop, the Relation Replay Net (RRN) generates the weights of the paired batch of training samples $\mathcal{B}^{tr}$, and the Main Net is updated by SGD based on the weighted training loss. In the outer loop, given the updated parameters of the Main Net, RRN is trained with the batch of data sampled from the memory buffer, which is a relatively balanced dataset.

leading to increased storage requirements. In this study, we propose a novel method to enhance continual learning by focusing on an alternative perspective—assigning proper weights to training samples based on two primary factors: 1) task relationship between new and old tasks, and 2) sample importance within each task for network training. To account for the first factor, we consider that a sample from the memory buffer containing similar or semantically relevant information to the new task samples is more prone to suffer from disturbance, potentially leading to more forgetting. In this situation, the model should probably pay more attention to the memory buffer sample by assigning larger weights to this old sample or smaller weights to new task samples to alleviate the forgetting issue. Conversely, the model should appropriately assign larger weights to the new task samples to effectively learn new knowledge without disturbing old tasks too much. As for the second factor, the data points from the same class also tend to make different contributions to model training. The easy-to-learn samples deliver more representative feature information of their classes, while the 'hard' samples are more conducive to refining the classification boundaries.

As mentioned above, the loss weights $\Lambda$, which control the balance between 'stability' and 'plasticity' during training across different continual learning scenarios, should be dynamically assigned to different samples instead of being manually tuned and fixed [26]. To this end, we propose a Relation Replay Net (RRN) that extracts the interaction knowledge of the new and old samples and generates their corresponding weights $\Lambda$ dynamically. It facilitates the main classification network (dubbed Main Net) to achieve a better trade-off between 'stability' and 'plasticity'. Intuitively, RRN depends on the training state of the Main Net, and the sample weights generated by RRN in turn affect the training of the Main Net. Therefore, instead of the naive end-to-end training, we adopt a bi-level learning framework to jointly optimize both RRN and the Main Net. The effectiveness of this bi-level optimization will be demonstrated in Sec. V-D. For simplicity, we initially apply the proposed approach to ER (termed Relational Experience Replay, or RER), and the implementation details based on more baselines can be found in Appendix B-1.

## A. Overview

The RER framework, as depicted in Fig. 3, consists of two key components: Main Net $f(\cdot;\theta)$ responsible for continual learning, which can be any commonly used backbone architecture, and RRN $h(\cdot;\phi)$ utilized for assigning sample weights.

On the one hand, a CL model must be able to adapt to different new tasks based on the knowledge learned from the old tasks, *i.e.*, task relationship would affect the 'plasticity' as aforementioned. Furthermore, each new task sample has a varying impact on the model's plasticity, highlighting the importance of sample weights in fine-tuning the model's attention. To model this dynamic 'plasticity', we assign weights for pairwise training samples from new and old tasks to train the Main Net, where the weights are generated by the RRN based on their relationships. Concretely, we combine the new task batch $\mathcal{B}^D$ and the memory batch $\mathcal{B}^M$ to pairwisely construct training sample batch $\mathcal{B}^{tr} = \{(x_i^D, x_i^M)\}_{i=1}^B$, where we omit the labels for notation convenience. Then, the weighted loss function for the Main Net $f$ formulates the inner loop optimization problem of the bi-level learning framework, that is:

$$\begin{aligned}\theta^*(\phi) &= \arg\min_\theta \mathcal{L}^{tr}(\mathcal{B}^{tr};\theta,\phi)\\ &\triangleq \frac{1}{B}\sum_{i=1}^{B}\lambda_i^D(\phi)L^{tr}(x_i^D;\theta)+\lambda_i^M(\phi)L^{tr}(x_i^M;\theta),\end{aligned} \tag{3}$$

where $L^{tr}$ is CE loss based on ER, and the sample weights $\Lambda = \{\lambda_i^D, \lambda_i^M\}_{i=1}^B$ of these data pairs are generated by the RRN automatically.

On the other hand, the model should possibly alleviate the forgetting issue while learning new tasks. The proposed RRN should pay more attention to 'stability' to prevent the Main Net from focusing too much on new tasks. We thus formulate the outer loop optimization problem over memory buffer data, making the Main Net returned by optimizing the inner loop one in Eq. (3) acts as a stable consolidation of knowledge from the learned tasks [1], *i.e.*,

$$\begin{aligned}\phi^* &= \arg\min_{\phi}\ \mathcal{L}^{bf}(\mathcal{B}^{bf};\theta^*(\phi)) \\ &\triangleq \frac{1}{B}\sum_{i=1}^{B} L^{bf}(x_i;\theta^*(\phi)),\end{aligned} \tag{4}$$

where the $L^{bf}(x;\theta)$ can be simply adopted as CE loss, and $\mathcal{B}^{bf}$ is another batch of samples from the memory buffer $\mathcal{M}$, which is different from the inner-loop training data $\mathcal{B}^M$. By dynamically generating specific weights for different samples, the RRN can help the Main Net to better balance the trade-off between new and old tasks, thereby facilitating the continual learning process.

Furthermore, the design of the RRN should consider the following two factors: 1) the RRN should take the **interaction information** between the new and old tasks into account; 2) the input of the RRN should be '**information abundant**' to ensure that the RRN could extract useful knowledge to generate meaningful weights. With this aim, we propose to pair each new task sample with a corresponding sample from the memory buffer and use their respective abstract information as inputs to the RRN. Specifically, for each sample pair $x_i^D$ from the new task and $x_i^M$ from the memory buffer, we take their losses $L^{D,M} = \left[L^{tr}(x_i^D), L^{tr}(x_i^M)\right]$, and logit norm $||z^{D,M}|| = \left[\|z_i^D\|_2, \|z_i^M\|_2\right]$ as the inputs of the RRN, where $z = f(x;\theta)$ is the predicted logits. By taking into account both label and feature information, the RRN can capture the semantic and distributional similarities between the paired samples and generate appropriate weights for the Main Net to balance the importance of new and old tasks during training. Thus, the paired sample weights generated by the RRN can be formulated as:

$$\left[\lambda_i^D(\phi), \lambda_i^M(\phi)\right] = h\left(L^{D,M}, \|z^{D,M}\|_2;\phi\right). \tag{5}$$

In summary, the proposed RER forms a bi-level learning framework to simultaneously model the 'plasticity' and 'stability' in the inner and outer loop optimization problems, respectively. The RRN, guided by a relatively balanced set $\mathcal{M}$, automatically generates sample weights to refine the optimization direction of the Main Net, thereby facilitating enhanced Main Net optimization and achieving a more favorable balance between the new and old tasks. In the inference stage, we can directly predict the testing images by the Main Net without involving the RRN. Note that our method can be easily adapted to other *rehearsal-based* baselines, such as ER-ACE [29] and DER++ [26] by some simple modification (please refer to Section V-C and Appendix B-1).

## B. Optimization Procedure

Since it is difficult to find closed-form solutions, the optimization of $\theta$ and $\phi$ as shown in Eq. (3) and Eq. (4) depends on two nested loops, which is computationally expensive. Considering computational efficiency and the large scale of data to be processed, we adopt an alternative online gradient-based optimization strategy to solve the proposed bi-level learning framework.

[1]Note that the parameter $\phi$ of the Relation Replay Net is regarded as a hyper-parameter of that of the Main Net.

**Algorithm 1** Relational Experience Replay training algorithm

**Input**: new task data $\mathcal{D}_t$, memory buffer $\mathcal{M}$

**Output**: Main Net and Relation Replay Net (RRN) parameters $\{\theta,\phi\}$

```
1:  while $\mathcal{D}_t \neq \emptyset$ do
2:    while $k < Iter_{max}$ do
3:      Sample a new task batch $\mathcal{B}^D \in \mathcal{D}_t$ and a buffer batch $\mathcal{B}^M \in \mathcal{M}$
4:      Construct a paired training batch $\mathcal{B}^{tr} \leftarrow \mathcal{B}^D$ and $\mathcal{B}^M$
5:      Calculate the inner-loop loss by Eq. (3)
6:      Update $\theta^k$ by Eq. (6)
7:      Sample another buffer batch $\mathcal{B}^{bf} \in \mathcal{M}$
8:      Calculate the outer-loop loss by Eq. (4)
9:      Update $\phi^k$ by Eq. (7)
10:     $k++$.
11:   end while
12: end while
```

**Updating $\theta$:** Referring to Eq (3), given the parameter $\phi^k$ of RRN at iteration step $k$, we optimize the parameter $\theta$ of Main Net by one-step gradient descent:

$$\theta^{k+1}(\phi) = \theta^k - \eta_\theta \nabla_\theta \mathcal{L}^{tr}(\mathcal{B}^{tr};\theta^k,\phi^k), \tag{6}$$

where $\eta_\theta$ is the inner-loop learning rate. Note that the updated parameter $\theta^{k+1}(\phi)$ is actually a function of $\phi$.

**Updating $\phi$:** With Main Net parameter $\theta$, we can optimize RRN parameter $\phi$ by Eq. (4) given $\theta^{k+1}(\phi)$ by the following formulation:

$$\phi^{k+1} = \phi^k - \eta_\phi \nabla_\phi \mathcal{L}^{bf}(\mathcal{B}^{bf};\theta^{k+1}(\phi^k)), \tag{7}$$

where $\eta_\phi$ is the outer-loop learning rate. More details of the gradient calculation can be found in Appendix A.

## C. Theoretical Analysis

According to Eqs. (6) and (7), we have the following proposition to further reveal how the proposed method models the task-wise relationship.

**Proposition 1.** *Let $g^{bf}(x) = \frac{\partial L^{bf}(x;\theta)}{\partial\theta}\Big|_{\theta^k}$ and $g^{tr}(x) = \frac{\partial L^{tr}(x;\theta)}{\partial\theta}\Big|_{\theta^k}$ denote the gradients of the buffer sample and the training sample with respect to the parameter $\theta$, respectively. Then the updating formulation of $\phi$ presented in Eq. (7) can be reformulated as*

$$\phi^{k+1} = \phi^k + \frac{\eta_\theta\eta_\phi}{B}\sum_{j=1}^{B} G(j)\cdot \frac{\partial h_j(\phi)}{\partial\phi}\Big|_{\phi^k}, \tag{8}$$

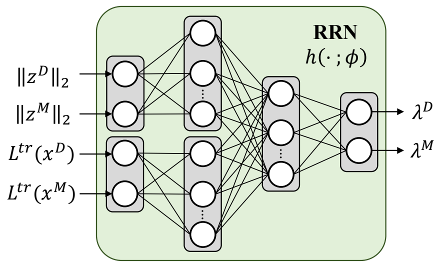


Fig. 4. The architecture of the Relation Replay Net (RRN).

*where the gradient* $\frac{\partial h_j(\phi)}{\partial \phi} = \left[\frac{\partial \lambda_j^D(\phi)}{\partial \phi}, \frac{\partial \lambda_j^M(\phi)}{\partial \phi}\right]^T$*, and the coefficient* $G(j)$ *is*

$$G(j) = \frac{1}{B}\sum_{c=1}^{C_t}\left(\sum_{i=1}^{B_c} g^{bf}(x_i)\right)\left[g^{tr}(x_j^D), \quad g^{tr}(x_j^M)\right]. \quad (9)$$

In Proposition 1, we demonstrate that the task-wise relationship in RRN can be implicitly modeled through the SGD updating process. This is achieved by computing the coefficient $G(j)$, which represents the mean dot product similarity between the gradient of buffer samples and that of training samples within each class. This measure effectively captures the relationship between new tasks and previously seen classes from a gradient perspective. The proof can be found in Appendix A.

### D. Implementation Details

**The Architecture of the Relation Replay Net.** We propose a two-hidden-layer neural network as the RRN. Intuitively, the first layer comprises two distinct linear layers designed to extract interaction information from paired training samples: one focuses on loss values, while the other attends to logit norms. Subsequently, the second layer is also a linear layer that merges the extracted information to automatically generate weights for the sample pairs. Fig. 4 illustrates the architecture of RRN where we set the number of hidden units as 16 for computation efficiency.

**Training Details.** When training a new task, the RRN aims to generate reasonable sample weights for both new task samples and buffer samples to improve the trade-off between 'stability' and 'plasticity', which needs a warm-up stage to explore the relationship between the new and old tasks. Specifically, in the warm-up stage (stage length denoted as $Iter_{warm}$), we update the RRN by Eq. (7) and update the Main Net via preset fixed weights like the previous methods. After the warm-up stage, we turn to use the sample weights generated by the RRN to guide the training of Main Net rather than prefixed weights. In addition, to reduce the calculation burden, we only update the RRN once while updating the Main Net for multiple steps (step number denoted as $Interval$). The specific choices of $Iter_{warm}$ and $Interval$ are presented in Section V-E.

## V. Experimental Results

To validate the effectiveness of our method, we conduct extensive experiments on various publicly available benchmark datasets for both image and text recognition tasks, including three image datasets, *i.e.*, **CIFAR-10**, **CIFAR-100** [48] and **Tiny ImageNet** [49], and two text datasets, *i.e.*, **20News** [50] and **DBpedia** [51]. Additionally, we present a thorough ablation analysis in this section to gain insight into our method.

### A. Experimental Settings

For the three image datasets, we divide them into task sequences of different lengths. Specifically, CIFAR-10 is divided into five tasks with each task constituting a binary classification problem. Both CIFAR-100 and Tiny ImageNet datasets are divided into 10 tasks, where each task is a 1-of-10 and 1-of-20 classification problem, respectively. To ensure robust and reliable evaluation, the framework was trained for 50 epochs on each task across these three datasets following DER++ [26].

For the two text datasets, we follow the basic settings in [52]. Then we split 20News into 5 tasks, with each task consists of 4 classes. Utilizing a two-hidden-layer MLP as the classification model, we train each task for 200 epochs. Additionally, for DBpedia, we partition it into 7 tasks and each of them is a binary classification problem. We leverage a fixed pretrained BERT model to extract text embeddings, followed by training a linear classifier on these embeddings, and we train each task for 50 epochs.

As aforementioned, this study mainly focuses on the configurations of Class-IL and Task-IL. The detailed experimental settings are consistent with DER++ [26] for a fair comparison. In our experiments, we evaluate the model by two common-used metrics, *i.e.*, Average Accuracy (ACC) and Backward Transfer (BWT) [25], [26]. We repeat each experiment five times to reduce the randomness of network training. Similar to [26], the total size of the memory buffer remains constant throughout the entire training process.

The proposed method is implemented with PyTorch [53]. The backbone of Main Net is the widely-used ResNet-18 [2]. In the inner loop, the Main Net is optimized by SGD with an initial learning rate of 0.03 for all datasets, and in the outer loop, Adam [54] is adopted to optimize RRN, where the initial learning rate is set as 0.001 with a weight decay of $10^{-4}$.

### B. Comparison with State-of-the-art Methods on Image Classification

As previously mentioned, our proposed method aims to enhance general *rehearsal-based* baselines by taking into account the 'stability-plasticity' dilemma, which involves the relationship between new and old tasks, and the specific importance of different samples. We integrate this approach into three representative baselines: ER, ER-ACE [29], and DER++ [26], which are termed Relational-ER (RER), Relational-ER-ACE (RER-ACE), and Relational-DER (RDER), respectively. Implementation details can be found in Appendix B-1.

Table I presents the comparison results between our proposed algorithm applied on the three baselines and various state-of-the-art methods over ACC and BWT metrics. These comparison methods include three *regularization-based* methods: oEWC [20], SI [19], and LwF [18], and four *rehearsal-based* methods: GEM [25], A-GEM [34], iCaRL [24], and

TABLE I
COMPARISON WITH THREE DIFFERENT BASELINES AND OTHER STATE-OF-THE-ART METHODS ON THE CIFAR-10 AND TINY-IMAGENET. THE RESULTS OF OUR METHOD ARE SHOWN IN GRAY CELLS AND THE BETTER RESULTS ARE PRESENTED IN **BOLD**.

| Buffer Size | Method | CIFAR-10 | | | | Tiny-ImageNet | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Class-IL | | Task-IL | | Class-IL | | Task-IL | |
| | | ACC | BWT | ACC | BWT | ACC | BWT | ACC | BWT |
| - | Upper bound | 92.20 ±0.15 | - | 98.31 ±0.12 | - | 59.99 ±0.19 | - | 82.04 ±0.10 | - |
| - | Lower bound | 19.62 ±0.05 | -96.39 ±0.12 | 61.02 ±3.33 | -46.24 ±2.12 | 7.92 ±0.26 | -76.73 ±0.08 | 18.31 ±0.68 | -64.97 ±1.70 |
| - | oEWC | 19.49 ±0.12 | -91.64 ±3.07 | 68.29 ±3.92 | -29.13 ±4.11 | 7.58 ±0.10 | -73.91 ±0.79 | 19.20 ±0.31 | -59.86 ±0.42 |
| - | SI | 19.48 ±0.17 | -95.78 ±0.64 | 68.05 ±5.91 | -38.76 ±0.89 | 6.58 ±0.31 | -67.91 ±0.96 | 36.32 ±0.13 | -53.26 ±0.75 |
| - | LwF | 19.61 ±0.05 | -96.69 ±0.25 | 63.29 ±2.35 | -32.56 ±0.56 | 8.46 ±0.22 | -76.74 ±0.44 | 15.85 ±0.58 | -67.79 ±0.23 |
| 200 | GEM | 25.54 ±0.76 | -82.61 ±1.60 | 90.44 ±0.94 | - 9.27 ±2.07 | - | - | - | - |
| | A-GEM | 20.04 ±0.34 | -95.73 ±0.20 | 83.88 ±1.49 | -16.39 ±0.80 | 8.07 ±0.08 | -77.02 ±0.22 | 22.77 ±0.03 | -56.61 ±0.32 |
| | iCaRL | 49.02 ±3.20 | -28.72 ±0.49 | 88.99 ±2.13 | - 1.01 ±4.15 | 7.53 ±0.79 | -22.70 ±0.44 | 28.19 ±1.47 | -10.36 ±0.31 |
| | GSS | 39.07 ±5.59 | -75.25 ±4.07 | 88.80 ±2.89 | - 8.56 ±1.78 | - | - | - | - |
| | ER | 55.84 ±0.71 | -47.77 ±1.39 | 92.41 ±0.59 | - 5.73 ±0.38 | 8.67 ±0.25 | -77.29 ±0.26 | 39.28 ±0.83 | -42.05 ±0.29 |
| | RER | **58.59 ±0.74** | **-44.50 ±0.80** | **92.85 ±0.36** | **- 5.40 ±0.65** | **9.35 ±0.21** | **-76.67 ±0.38** | **40.83 ±0.58** | **-41.19 ±0.55** |
| | ER-ACE | 63.02 ±1.29 | -20.35 ±1.76 | 92.59 ±0.36 | **- 5.33 ±0.41** | 11.67 ±0.29 | -49.03 ±1.61 | 42.08 ±0.35 | -37.71 ±1.10 |
| | RER-ACE | **63.52 ±0.71** | **-20.11 ±5.66** | **92.63 ±0.58** | - 5.43 ±0.67 | **12.18 ±0.41** | **-48.91 ±2.59** | **44.11 ±0.62** | **-36.62 ±1.17** |
| | DER++ | 62.30 ±1.07 | -35.83 ±1.34 | 90.74 ±1.01 | - 7.45 ±1.07 | 12.26 ±0.31 | -68.37 ±1.38 | 40.47 ±1.53 | -40.41 ±1.29 |
| | RDER | **65.38 ±0.42** | **-34.16 ±1.90** | **91.67 ±0.80** | **- 6.81 ±1.19** | **13.96 ±0.64** | **-67.02 ±1.24** | **40.87 ±0.92** | **-39.87 ±1.31** |
| 500 | GEM | 26.20 ±1.26 | -74.31 ±4.62 | 92.16 ±0.69 | - 9.12 ±0.21 | - | - | - | - |
| | A-GEM | 22.67 ±0.57 | -94.01 ±1.16 | 89.48 ±1.45 | -14.26 ±4.18 | 8.06 ±0.04 | -77.06 ±0.41 | 25.33 ±0.49 | -55.68 ±1.01 |
| | iCaRL | 47.55 ±3.95 | -25.71 ±1.10 | 88.22 ±2.62 | - 1.06 ±4.21 | 9.38 ±1.53 | -20.89 ±0.23 | 31.55 ±3.27 | - 7.30 ±0.79 |
| | GSS | 49.73 ±4.78 | -62.88 ±2.67 | 91.02 ±1.57 | - 7.73 ±3.99 | - | - | - | - |
| | ER | 69.01 ±0.37 | -33.02 ±2.62 | 94.28 ±0.27 | **- 3.09 ±1.61** | 10.40 ±0.16 | -74.36 ±0.58 | 48.82 ±0.34 | -31.06 ±1.53 |
| | RER | **69.22 ±1.96** | **-29.79 ±2.87** | **94.50 ±0.41** | - 3.40 ±0.33 | **11.50 ±0.47** | **-74.13 ±0.72** | **51.28 ±0.93** | **-30.29 ±1.24** |
| | ER-ACE | 71.26 ±0.66 | -13.37 ±1.06 | **94.31 ±0.23** | - 3.19 ±0.39 | 19.59 ±0.13 | -47.56 ±0.68 | 50.99 ±0.45 | -29.32 ±0.46 |
| | RER-ACE | **71.29 ±1.15** | **-12.53 ±2.41** | 94.25 ±0.23 | **- 3.16 ±0.80** | **20.41 ±0.66** | **-42.22 ±1.09** | **54.62 ±0.87** | **-25.15 ±0.98** |
| | DER++ | 72.11 ±1.41 | -23.40 ±1.32 | **94.21 ±0.32** | - 3.98 ±0.60 | 19.29 ±1.14 | -60.58 ±0.46 | 51.39 ±0.91 | -26.90 ±0.52 |
| | RDER | **73.99 ±1.03** | **-22.86 ±1.76** | 94.04 ±0.43 | **- 3.82 ±0.59** | **20.06 ±1.18** | **-56.16 ±1.38** | **52.56 ±0.69** | **-25.02 ±0.24** |
| 5120 | GEM | 25.26 ±3.46 | -75.27 ±4.41 | 95.55 ±0.02 | - 6.91 ±2.33 | - | - | - | - |
| | A-GEM | 21.99 ±2.29 | -84.49 ±3.08 | 90.10 ±2.09 | - 9.89 ±0.40 | 7.96 ±0.13 | -76.01 ±0.52 | 26.22 ±0.65 | -55.61 ±0.84 |
| | iCaRL | 55.07 ±1.55 | -24.94 ±0.14 | 92.23 ±0.84 | - 0.99 ±1.41 | 14.08 ±1.92 | -16.00 ±0.28 | 40.83 ±3.11 | - 2.60 ±0.35 |
| | GSS | 67.27 ±4.27 | -58.11 ±9.12 | 94.19 ±1.15 | - 6.38 ±1.71 | - | - | - | - |
| | ER | 83.30 ±0.50 | -13.79 ±1.40 | 96.95 ±0.15 | - 0.98 ±0.36 | 28.52 ±0.37 | -52.54 ±1.45 | 68.46 ±0.40 | **-10.13 ±0.20** |
| | RER | **83.53 ±0.55** | **-12.13 ±1.29** | **96.98 ±0.17** | **- 0.79 ±0.24** | **33.86 ±0.64** | **-45.56 ±2.13** | **69.31 ±0.41** | -10.68 ±0.25 |
| | ER-ACE | 82.98 ±0.38 | **- 3.99 ±0.61** | 96.76 ±0.08 | - 0.63 ±0.30 | **37.02 ±0.17** | **-33.29 ±1.03** | 68.69 ±0.19 | - 9.88 ±0.42 |
| | RER-ACE | **83.74 ±0.79** | - 4.05 ±1.81 | **96.80 ±0.20** | **- 0.57 ±0.27** | 36.97 ±0.94 | -33.79 ±3.07 | **69.05 ±0.73** | **- 9.51 ±0.79** |
| | DER++ | 84.50 ±0.63 | - 9.79 ±0.34 | 95.91 ±0.57 | - 1.57 ±0.25 | 37.88 ±0.37 | -30.62 ±1.78 | 68.05 ±0.53 | - 8.80 ±0.24 |
| | RDER | **85.56 ±0.38** | **- 8.81 ±0.71** | **96.21 ±0.22** | **- 1.42 ±0.09** | **39.67 ±0.96** | **-29.37 ±1.53** | **68.82 ±0.54** | **- 8.03 ±0.46** |

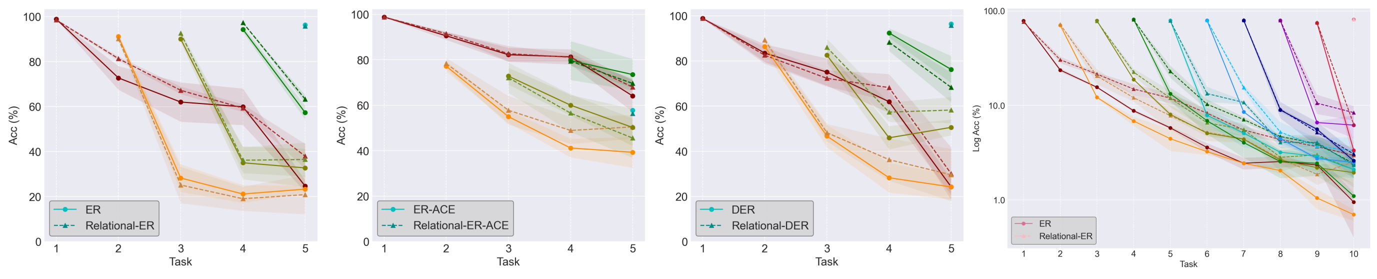
TABLE II
COMPARISON OF ACC WITH THREE DIFFERENT BASELINES ON THE CIFAR-100. THE RESULTS OF OUR METHOD ARE SHOWN IN GRAY CELLS AND THE BETTER RESULTS ARE PRESENTED IN **BOLD**.

| Settings | Buffer Size | ER | RER | ER-ACE | RER-ACE | DER++ | RDER |
|---|---|---|---|---|---|---|---|
| Class-IL | 100 | 11.31 ±0.21 | **13.94 ±0.89** | 18.59 ±1.08 | **20.20 ±0.86** | 14.98 ±0.65 | **20.79 ±1.05** |
| | 200 | 14.78 ±0.40 | **16.40 ±0.53** | 25.14 ±1.83 | **26.64 ±0.29** | 24.17 ±1.37 | **30.65 ±0.76** |
| | 500 | 23.10 ±0.32 | **26.97 ±0.75** | 36.02 ±0.84 | **36.06 ±1.14** | 35.19 ±1.30 | **39.50 ±1.54** |
| | 5120 | 51.43 ±1.01 | **54.08 ±0.63** | 53.93 ±2.04 | **54.38 ±1.08** | 55.58 ±1.86 | **60.07 ±0.23** |
| Task-IL | 100 | 58.64 ±1.31 | **59.77 ±0.03** | 59.91 ±1.01 | **61.10 ±0.93** | 58.32 ±1.27 | **59.07 ±0.73** |
| | 200 | 66.31 ±0.76 | **66.83 ±0.97** | 64.81 ±3.14 | **67.42 ±0.60** | 66.47 ±0.57 | **68.60 ±0.51** |
| | 500 | 73.10 ±0.99 | **73.99 ±0.51** | 74.13 ±0.84 | **74.54 ±1.37** | 74.10 ±1.62 | **75.59 ±0.85** |
| | 5120 | **86.16 ±0.47** | 85.35 ±0.34 | 84.69 ±1.32 | **85.15 ±0.41** | 86.23 ±2.18 | **86.54 ±0.31** |

GSS [55]. Besides, we also provide an upper-bound method and a lower-bound method for better reference, where the former trained on all data from old and new tasks together and the latter directly trained on the new task without any strategies to prevent model forgetting.

For CIFAR-10, it can be observed that our proposed approach can be adapted to different rehearsal-based baselines and achieve consistent performance improvement. For example, the proposed RDER achieves as much as 3.08% absolute performance gain compared to the baseline DER++ with a memory buffer of 200 under Class-IL. Besides, our method achieves significantly higher classification accuracy than all comparison state-of-the-art methods under different settings. On the other hand, our method can also significantly reduce the BWT of baselines, indicating that the proposed method can effectively reduce model forgetting while improving classification accuracy, *i.e.*, achieving a better balance of 'stability' and 'plasticity'.

For Tiny-ImageNet dataset, our method also consistently achieves the best results under almost all settings. Although iCaRL achieves sound BWT metric, it falls short in terms of ACC, revealing that a model that pays much attention to avoid forgetting may negatively impact classification performance. In contrast, our method improves both ACC and BWT compared to the corresponding baselines, highlighting that our method can achieve a better trade-off between new and old tasks. Notably, the results of GEM and GSS are not reported in Table I since the excessive computational overhead



(a) CIFAR-10 ER/RER (b) CIFAR-10 ER-ACE/RER-ACE (c) CIFAR-10 DER++/RDER (d) Tiny-ImageNet ER/RER

Fig. 5. Classification accuracy (%) of each task during the whole training process. The dot-solid lines represent the comparison methods ER, ER-ACE, or DER++ and the triangle-dashed lines represent our methods RER, RER-ACE, or RDER.

TABLE III
Class-IL results on two text classification datasets under different buffer sizes.

| Dataset | Buffer size | ER | | RER | | ER-ACE | | RER-ACE | | DER++ | | RDER | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ACC | BWT | ACC | BWT | ACC | BWT | ACC | BWT | ACC | BWT | ACC | BWT |
| 20News | 200 | 22.52 | -59.47 | **23.74** | **-56.95** | 33.37 | -32.83 | **34.11** | **-23.52** | 33.67 | -48.13 | **40.49** | **-33.38** |
| | 500 | 30.88 | -48.07 | **33.60** | **-42.99** | 37.68 | -24.88 | **38.31** | **-17.65** | 41.87 | -35.82 | **46.08** | **-21.57** |
| | 5120 | 45.70 | -24.22 | **46.68** | **-22.04** | 45.53 | -13.35 | **46.03** | **-5.75** | 49.35 | -23.94 | **51.37** | **-9.83** |
| Dbpedia | 50 | 66.74 | -38.20 | **68.22** | **-36.48** | 79.21 | -23.53 | **80.51** | **-22.12** | 88.75 | -12.20 | **89.50** | **-11.51** |
| | 100 | 84.23 | -11.79 | **86.31** | **-7.74** | **91.06** | **-6.08** | 90.99 | -7.56 | 93.55 | -3.87 | **93.77** | **-3.81** |
| | 200 | 79.81 | -22.91 | **81.33** | **-21.12** | 86.29 | -15.31 | **87.80** | **-13.50** | 94.81 | -5.17 | **96.10** | **-2.78** |

is unacceptable.

Moreover, we validate the effectiveness of our method on CIFAR-100 in Table II. Obviously, our method can achieve a significant improvement for all these three baselines across different buffer sizes. For instance, in the Class-IL setting with a buffer size of 100, our method applied to ER, ER-ACE, and DER can improve their ACC by 2.63%, 1.61%, and 5.81%, respectively. These results further demonstrate the strong adaptability of our method to multiple datasets and diverse settings.

## C. Comparison on Text Classification

To investigate the scalability of our proposed method across various data modalities, we conduct experiments in the domain of text classification on 20News and DBpedia following [52]. The results are shown in Table III. It can be observed that our method consistently improves the corresponding baseline under all settings. For 20News, the forgetting is more serious but our method can largely improve the baselines. Especially for DER++, our RDER can improve ACC by about 6.8 and increase BWT by about 14.8 with 200 buffer samples. Additionally, for another dataset DBpedia, our method can also significantly improve the performance of all three baselines for both ACC and BWT with various buffer sizes. These results indicate that our method can handle diverse data types and consistently improve the corresponding baseline by achieving better stability and plasticity trade-offs.

## D. Discussion

To further explore the proposed method, we conduct more experiments and analyses in this section.

**Does Our Method Mitigate the Model Forgetting?** In order to better analyze the forgetting problem for old tasks, we visualize the accuracy change for each task during the continual learning process in Fig. 5 and Appendix B-3. The visualization results show that our method better mitigates forgetting for previous tasks, *i.e.*, consolidating knowledge for old tasks better than corresponding baselines. Note that the performance of RDER is slightly lower than DER++ for some new tasks (shown in Fig. 5c). This is because our method aims to improve the generalization of all tasks, rather than just paying attention to new tasks with more data.

TABLE IV
ACC of ER and RER under **Setup 1** (semantic-relevant task) and **Setup 2** (semantic-irrelevant task).

| | Task 1 | Task 2 |
|---|---|---|
| **Setup 1** | ship/truck | airplane/automobile |
| ER | 97.56 | 81.38 |
| RER | 97.71 | 84.32 |
| **Setup 2** | ship/truck | cat/horse |
| ER | 97.56 | 88.29 |
| RER | 97.71 | 89.41 |

**How Task Similarity Affects Continual Learning Models?** In this section, we aim to explore how task similarity affects model training and how our RRN captures task relationships to assign sample weights. To this end, we design two distinct setups using CIFAR-10 to simulate scenarios involving similar and dissimilar tasks in continual learning, as summarized in Table. IV. Specifically, in **Setup 1** and **Setup 2**, task 1 contains the same classes ('ship' and 'truck'), while task 2 varies between the two setups: in **Setup 1**, it comprises semantically relevant classes ('airplane' and 'automobile') against task 1, whereas in **Setup 2**, it consists of semantically irrelevant classes ('cat' and 'horse'). We then compare the baseline ER with our RER on these two setups and present the average accuracy after training each task in Table. IV. We can observe that: 1) After training task 1, the performance of our method

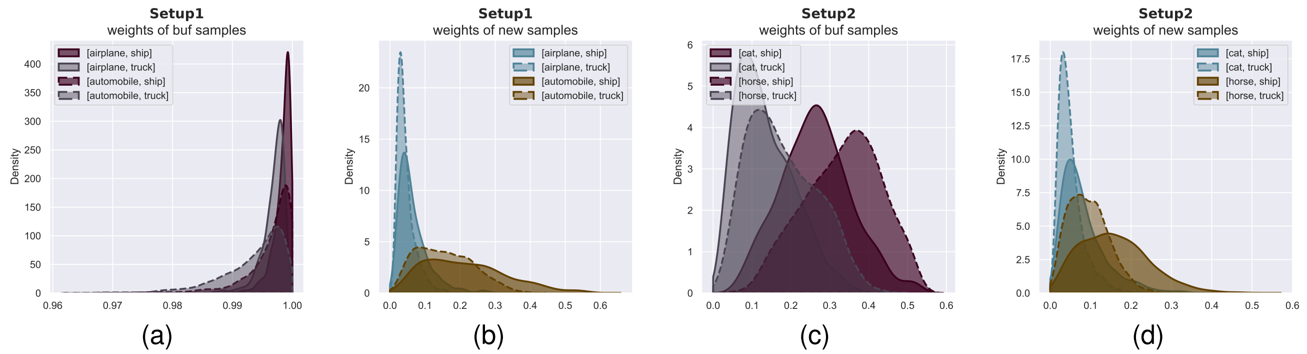


Fig. 6. Weight distributions at epoch 45 during training for (a) buffer samples and (b) new task samples in Setup 1, and (c) buffer samples and (d) new task samples in Setup 2.

is comparable to that of the baseline method, as there is no need to consolidate old knowledge during this training stage. 2) After training task 2, the performance of the baseline model ER in **Setup 1** is significantly lower than that in **Setup 2**. This is because the objects of the two tasks in **Setup 1** are relatively easy to confuse, leading to more severe forgetting. Conversely, in **Setup 2**, the objects of task 2 differ from task 1, which may not largely affect the previously learned knowledge. 3) After training task 2, our method RER consistently improves the performance of ER across both two setups. Specifically, RER largely reduces the accuracy gap between the two setups from 6.91 in ER to 5.09. This indicates that RER effectively captures the relationship between new and old tasks, resulting in improved performance.

To investigate the impact of sample weights generated by our RRN, we further analyze the weight distribution within **Setup 1** and **Setup 2** in Fig. 6. Since our method model pair-wisely assigned sample weights for new and old tasks, we thus output the weight distribution for all pairs of samples[2]. As shown in Fig. 6(a) and Fig. 6(b), the weights assigned to buffer samples in **Setup 1** are close to 1, significantly larger than those of new task samples. This observation can be attributed to the inherent similarity of objects between task 1 and task 2, which makes them prone to confusion or misclassification, thereby increasing the risk of forgetting. Consequently, RRN prioritizes retaining memory buffer samples to alleviate forgetting and assigns them larger weights accordingly. Conversely, in **Setup 2** as shown in Fig. 6(c) and Fig. 6(d), the sample weights of buffer samples are mainly distributed in the range of [0, 0.5], notably smaller than those in **Setup 1**. Meanwhile, the weights assigned to new samples are relatively larger than those in **Setup 1**. This is probably because RRN responds to the fact that new knowledge may not significantly affect the model of previously learned knowledge in subsequent epochs.

In summary, task relationships may significantly impact model training in continual learning, and our proposed RRN holds great potential to effectively capture such relationships by generating appropriate sample weights.

**Why We Need the Bi-level Optimization?** To evaluate the effectiveness of the bi-level optimization paradigm, we conduct an ablation study by end-to-end training of the Main Net and the RRN together based on DER++ (referred to as Vanilla). Specifically, we employ the same RRN to generate the paired sample weights and update the Main Net and the RRN by a single backward step through $\mathcal{L}^{tr} + \mathcal{L}^{bf}$. In Table V, we observe that the sample weights generated by the Vanilla method appear to be uninformative, and even impair the performance of the baseline approach (DER++), indicating that the end-to-end training paradigm cannot produce meaningful weights to enhance the generalization capability of the Main Net. In contrast, our RDER approach consistently outperforms both DER++ and the Vanilla method, thereby demonstrating the effectiveness of the proposed bi-level optimization framework to achieve a better trade-off between 'stability' and 'plasticity'.

TABLE V
ACC ON THE CIFAR-10 DATASET IN SOME DIFFERENT SETTINGS TO VERIFY THE EFFECTIVENESS OF OUR METHOD DESIGN.

| Memory Size | Method | Class-IL | Task-IL |
|---|---|---|---|
| 200 | DER++ [26] | 62.30 ± 1.07 | 90.74 ± 1.01 |
| | Vanilla | 62.85 ± 2.90 | 91.20 ± 1.88 |
| | Split RDER | 62.53 ± 0.66 | 91.36 ± 0.77 |
| | RDER (ours) | **65.38** ± 0.42 | **91.67** ± 0.80 |
| 500 | DER++ [26] | 72.11 ± 1.41 | 94.21 ± 0.32 |
| | Vanilla | 72.28 ± 0.93 | 93.42 ± 0.75 |
| | Split RDER | 72.57 ± 0.73 | 93.77 ± 0.33 |
| | RDER (ours) | **73.99** ± 1.03 | **94.04** ± 0.43 |
| 5120 | DER++ [26] | 84.50 ± 0.63 | 95.91 ± 0.57 |
| | Vanilla | 82.26 ± 2.45 | 95.3 ± 0.72 |
| | Split RDER | 85.35 ± 0.24 | 96.19 ± 0.46 |
| | RDER (ours) | **85.56** ± 0.38 | **96.21** ± 0.22 |

**Why Use the Memory Buffer to Train Relation Replay Net in the Outer Loop?** In Eq. (4) of the outer-loop optimization, we utilize the memory buffer $\mathcal{M}$ to train the RRN. However, there are concerns regarding the risk of overfitting when using the memory buffer to guide the RRN. To address this issue, we investigate two approaches for constructing the outer-loop training set: 1) Splitting the memory buffer $\mathcal{M}$ into two sets, which are utilized for training the Main Net in the inner loop and the RRN in the outer loop, respectively [56]. 2) Alternatively, training the RRN in the outer loop using the entire memory buffer $\mathcal{M}$, as we propose.

We present a comparison between the two approaches based on DER++ and report the results in Table V, where the two methods are denoted as 'Split RDER' and 'RDER',

[2]For example, in **Setup 1**, the sample weight pair for new and old tasks are [ship, airplane], [ship, automobile], [truck, airplane], and [truck, automobile] as shown in Fig. 6(a) and Fig. 6(b).

TABLE VI
COMPARISON OF ACC ON THE CIFAR-10 WITH SMALL BUFFER SIZES. THE RESULTS OF OUR METHOD ARE SHOWN IN GRAY CELLS AND THE BETTER RESULTS ARE PRESENTED IN **BOLD**.

| Settings | Buffer Size | ER | RER | ER-ACE | RER-ACE | DER++ | RDER |
|---|---|---|---|---|---|---|---|
| Class-IL | 50 | 36.47 ±2.92 | **37.43 ±1.27** | **42.16 ±1.78** | 41.95 ±1.07 | 46.70 ±4.14 | **49.75 ±1.92** |
| | 100 | 46.75 ±1.58 | **50.85 ±1.14** | 56.96 ±2.39 | **57.98 ±3.02** | 53.82 ±1.21 | **56.36 ±1.54** |
| Task-IL | 50 | **88.13 ±1.16** | 88.07 ±0.80 | **87.09 ±1.05** | 86.45 ±1.05 | 84.00 ±0.84 | **85.81 ±1.99** |
| | 100 | 90.27 ±1.05 | **90.56 ±0.41** | 90.28 ±0.38 | **91.01 ±0.04** | 87.48 ±1.43 | **89.16 ±1.37** |

respectively. Split RDER divides the memory buffer into two sets for outer- and inner-loop training with a ratio of $20\%-80\%$, following [56], whereas the RDER approach does not involve such a split. The results in Table V indicate that Split RDER exhibits a slight improvement over the baseline DER++, but its performance is still lower than our proposed RDER. These findings suggest that the first approach may lead to better generalization of the RRN while reducing the number of memory samples used to train the Main Net. However, this reduction in training samples may create a more serious imbalance between the new and old tasks, thereby reducing the overall performance, particularly for limited buffer size.

## E. Ablation Study

In this section, we first validate the effect of the small buffer sizes on our method based on the three baselines. And then we conduct a detailed ablation study based on RDER to evaluate the influence caused by two critical hyperparameters $Iter_{warm}$ and $Interval$ on CIFAR-10 with different buffer sizes.

**Effect of Small Buffer Size.** To further investigate the impact of small buffer sizes on our method, we evaluate our method applied to the three baselines on CIFAR-10 in Table VI. Specifically, RDER enhances DER++ by about +3.05% with a buffer size of 50 under the Class-IL setting, demonstrating that our method can further explore the information in the memory buffer to enhance the Main Net overall performance.

**Impact of the Warm-up Stage ($Iter_{warm}$).** Table VII presents the evaluation results for the various $Iter_{warm}$ under three different settings $\left[\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right] \times Iter_{max}$. The performance of RDER deteriorates significantly when a small value of $Iter_{warm}$ (*i.e.*, 17) is used, suggesting that the RRN requires an adequate number of warm-up steps to generate meaningful sample weights to accurately capture the task-wise relationship and sample importance within each task. On the other hand, it can be observed a slight drop in performance under $Iter_{warm} = 33$, since the preset weights may mislead the training in the warm-up stage. An excessively long warm-up stage also leads to insufficient iterations for the Main Net training guided by the generated weights. Hence, we recommend setting $Iter_{warm}$ to be half of the number of iteration epochs for each task, which performs the best in Table VII.

**Impact of the Relation Replay Net Updating Interval ($Interval$).** Here we vary the value of $Interval$ to investigate its impact on our framework. As shown in Table VIII, the setting $Interval = 5$ yields the highest classification accuracy across a range of memory buffer sizes. However,

TABLE VII
ACC ON CIFAR-10 WITH DIFFERENT LENGTH OF WARM-UP STAGE ($Iter_{warm}$).

| Memory Size | $Iter_{warm}$ | Class-IL | Task-IL |
|---|---|---|---|
| 50 | 17 | 48.02 ±0.70 | 85.13 ±1.27 |
| | 25 | **49.75 ±1.92** | **85.81 ±1.99** |
| | 33 | 47.41 ±0.91 | 85.12 ±1.35 |
| 200 | 17 | 64.53 ±1.13 | 91.84 ±0.29 |
| | 25 | **65.38 ±0.42** | 91.67 ±0.80 |
| | 33 | 64.84 ±1.08 | **92.88 ±0.42** |
| 500 | 17 | 72.17 ±1.11 | 93.57 ±0.17 |
| | 25 | **73.99 ±1.03** | **94.04 ±0.43** |
| | 33 | 73.03 ±1.49 | 93.85 ±0.44 |

TABLE VIII
ACC ON CIFAR-10 WITH DIFFERENT VALUES OF RELATION REPLAY NET UPDATING INTERVAL ($Interval$).

| Memory Size | $Interval$ | Class-IL | Task-IL |
|---|---|---|---|
| 50 | 1 | 48.12 ±0.51 | 84.44 ±1.48 |
| | 5 | **49.75 ±1.92** | **85.81 ±1.99** |
| | 10 | 46.21 ±1.00 | 85.18 ±1.65 |
| 200 | 1 | 64.27 ±1.46 | 91.57 ±0.59 |
| | 5 | **65.38 ±0.42** | 91.67 ±0.80 |
| | 10 | 64.66 ±0.29 | **91.71 ±0.62** |
| 500 | 1 | 71.23 ±0.99 | 93.15 ±0.68 |
| | 5 | **73.99 ±1.03** | **94.04 ±0.43** |
| | 10 | 72.11 ±0.46 | 93.62 ±0.23 |

small $Interval$, such as $Interval = 1$, often results in decreased performance due to the frequent alternation between updating the Main Net and the RRN, leading to oscillation during training. Additionally, frequent updates of the RRN are computationally expensive, which can impede the convergence of the overall framework. Conversely, large $Interval$, such as $Interval = 10$, can lead to faster computation, but we observed a significant drop in performance due to inadequate training of the RRN, which can result in the generation of suboptimal sample weights. To strike a balance between accuracy and computational efficiency, we propose an empirical formulation $Interval = \#epoch/10$, where $\#epoch$ represents the number of iteration epochs for each task.

**Inputs of the Relation Replay Net.** To further validate the effectiveness of the paired data and input terms of RRN, we conduct an ablation study based on RDER on CIFAR-10 with different memory buffer sizes. The results (final average accuracy) are listed in Table IX. We can observe that: 1) RDER-1, the model using only the loss of samples as input for RRN without incorporating the paired data structure, consistently improves baseline performance across diverse buffer sizes; 2) RDER-2, the model augmenting the RRN inputs of RDER-1 with the norm of predicted logits, further improves the performance over both ACC and BWT, indicating the efficacy

TABLE IX
ABLATION RESULTS OF RRN INPUTS.

| method | Paired | RRN Inputs | | M=200 | M=500 | M=5120 |
|---|---|---|---|---|---|---|
| | | loss | norm | | | |
| DER++ | / | / | / | 62.30 | 72.11 | 84.50 |
| RDER-1 | | ✓ | | 64.96 | 73.06 | 85.12 |
| RDER-2 | | ✓ | ✓ | 65.34 | 73.27 | 85.35 |
| RDER (ours) | ✓ | ✓ | ✓ | **65.38** | **73.99** | **85.56** |

of our proposed RRN input design; 3) RDER (ours), the model proposed in the main text, achieves the best performance and shows that the proposed paired data structure plays an essential role in the proposed RRN.

## VI. CONCLUSION

In this paper, we focus on the 'stability-plasticity' dilemma in continual learning and strive to adaptively tune the relationship across different tasks and samples. To this end, we propose a novel continual learning framework, Relational Experience Replay, which pairwisely adjusts the sample weights of samples from new tasks and the memory buffer. The sample weights generated by the Relation Replay Net can facilitate the optimization of the Main Net to achieve a better trade-off between 'stability' and 'plasticity'. The proposed method can be easily integrated with multiple *rehearsal-based* methods to achieve significant improvements. We theoretically and experimentally verify that the generated sample weights can extract the relationship between new and old tasks to automatically adjust the Main Net training and enhance the overall performance. We expect that our method can provide more insights into the 'stability-plasticity' dilemma and promote the development of the field of continual learning.

## APPENDIX A
## CALCULATION DETAILS ABOUT THE RELATION REPLAY NET UPDATING

In this section, we provide a detailed calculation of the derivatives of the RRN. Referring back to Section IV-B, the gradient descent of the RRN parameters is given in Eq. (7) and repeated here as

$$\phi^{k+1} = \phi^k - \eta_\phi \nabla_\phi \mathcal{L}^{bf}(\mathcal{B}^{bf}; \theta(\phi)), \tag{10}$$

where the $\theta(\phi)$ is the one-step updated parameters generated by Eq. (6).

Here we can use the chain rule to calculate the derivative of $\phi$ as follows.

$$\begin{aligned}
&\nabla_\phi \mathcal{L}^{bf}(\mathcal{B}^{bf}; \theta^k(\phi)) \\
=&\frac{1}{B}\sum_{i=1}^{B} \nabla_\phi L^{bf}(x_i; \theta^k(\phi)) \\
=&\frac{1}{B}\sum_{i=1}^{B} \frac{\partial L^{bf}(x_i;\theta)}{\partial \theta}\Big|_{\theta^k} \frac{\partial \theta(\phi)}{\partial \phi}\Big|_{\phi^k},
\end{aligned} \tag{11}$$

where the second term can be represented as

$$\begin{aligned}
\frac{\partial \theta(\phi)}{\partial \phi}\Big|_{\phi^k} = -\frac{\eta_\theta}{B}\sum_{j=1}^{B}\Bigg(&\nabla_\theta L^{tr}(x_j^D;\theta^k)\cdot \frac{\partial \lambda_j^D(\phi)}{\partial \phi}\Big|_{\phi^k} \\
&+ \nabla_\theta L^{tr}(x_j^M;\theta^k)\cdot \frac{\partial \lambda_j^M(\phi)}{\partial \phi}\Big|_{\phi^k}\Bigg).
\end{aligned} \tag{12}$$

Then substituting Eq. (12) into Eq. (11) and exchanging the order of the two summations, we can get

$$\begin{aligned}
&\nabla_\phi \mathcal{L}^{bf}(\mathcal{B}^{bf}; \theta^k(\phi)) \\
=&-\frac{\eta_\theta}{B}\sum_{j=1}^{B}\Bigg(\frac{1}{B}\sum_{i=1}^{B}\frac{\partial L^{bf}(x_i;\theta)}{\partial\theta}\Big|_{\theta^k}\cdot\frac{\partial L^{tr}(x_j^D;\theta)}{\partial\theta}\Big|_{\theta^k}\cdot\frac{\partial\lambda_j^D(\phi)}{\partial\phi}\Big|_{\phi^k} \\
&+\frac{1}{B}\sum_{i=1}^{B}\frac{\partial L^{bf}(x_i;\theta)}{\partial\theta}\Big|_{\theta^k}\cdot\frac{\partial L^{tr}(x_j^M;\theta)}{\partial\theta}\Big|_{\theta^k}\cdot\frac{\partial\lambda_j^M(\phi)}{\partial\phi}\Big|_{\phi^k}\Bigg) \\
=&-\frac{\eta_\theta}{B}\sum_{j=1}^{B}\left(G^D(j)\cdot\frac{\partial\lambda_j^D(\phi)}{\partial\phi}\Big|_{\phi^k}+G^M(j)\cdot\frac{\partial\lambda_j^M(\phi)}{\partial\phi}\Big|_{\phi^k}\right) \\
=&-\frac{\eta_\theta}{B}\sum_{j=1}^{B}G(j)\,\frac{\partial h(\phi)}{\partial\phi}\Big|_{\phi^k},
\end{aligned} \tag{13}$$

where the last term in Eq. (13) is the derivative of the RRN $h_j(\phi)$ outputs of the $j$-th training sample pair with respect to the network parameters $\phi$, that is

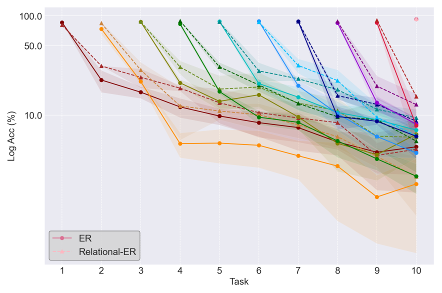
$$\frac{\partial h(\phi)}{\partial \phi} = \begin{bmatrix} \frac{\partial \lambda_j^D(\phi)}{\partial \phi}\Big|_{\phi^k} \\ \frac{\partial \lambda_j^M(\phi)}{\partial \phi}\Big|_{\phi^k} \end{bmatrix}, \tag{14}$$

and the coefficients $G(j) = \begin{bmatrix} G^D(j) & G^M(j) \end{bmatrix}$, where

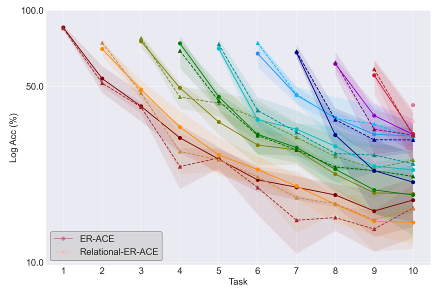
$$\begin{aligned}
G^D(j) &= \frac{1}{B}\sum_{i=1}^{B}\frac{\partial L^{bf}(x_i;\theta)}{\partial\theta}\Big|_{\theta^k}\cdot\frac{\partial L^{tr}(x_j^D;\theta)}{\partial\theta}\Big|_{\theta^k} \\
&\triangleq \frac{1}{B}\sum_{i=1}^{B} g^{bf}(x_i)\cdot g^{tr}(x_j^D), \\
G^M(j) &= \frac{1}{B}\sum_{i=1}^{B}\frac{\partial L^{bf}(x_i;\theta)}{\partial\theta}\Big|_{\theta^k}\cdot\frac{\partial L^{tr}(x_j^M;\theta)}{\partial\theta}\Big|_{\theta^k} \\
&\triangleq \frac{1}{B}\sum_{i=1}^{B} g^{bf}(x_i)\cdot g^{tr}(x_j^M),
\end{aligned} \tag{15}$$

respectively. Denote the gradient of the meta loss of the $i$-th sample of $\mathcal{D}^{bf}$ as $g^{bf}(x_i) = \frac{\partial L^{bf}(x_i;\theta)}{\partial\theta}\Big|_{\theta^k}$, and the gradient of training loss on the $j$-th sample pair of $\mathcal{D}^{tr}$ as $g^{tr}(x_j^D) = \frac{\partial L^{tr}(x_j^D;\theta)}{\partial\theta}\Big|_{\theta^k}$ and $g^{tr}(x_j^M) = \frac{\partial L^{tr}(x_j^M;\theta)}{\partial\theta}\Big|_{\theta^k}$. Obviously, the coefficient $G(j)$ represents the similarity between the gradient of training loss $\begin{bmatrix} g^{tr}(x_j^M) & g^{tr}(x_j^D) \end{bmatrix}$ and the average of the gradient of meta loss $g^{bf}(x_i)$. Furthermore, we can reformulate the average gradient of the meta loss by class. The coefficient $G(j)$ can be represented as:

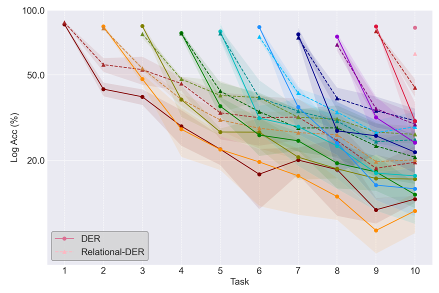
$$\begin{aligned}
G^D(j) &= \frac{1}{B}\sum_{c=1}^{C_t}\left(\sum_{i=1}^{B_c} g^{bf}(x_i)\right) g^{tr}(x_j^D), \\
G^M(j) &= \frac{1}{B}\sum_{c=1}^{C_t}\left(\sum_{i=1}^{B_c} g^{bf}(x_i)\right) g^{tr}(x_j^M),
\end{aligned} \tag{16}$$



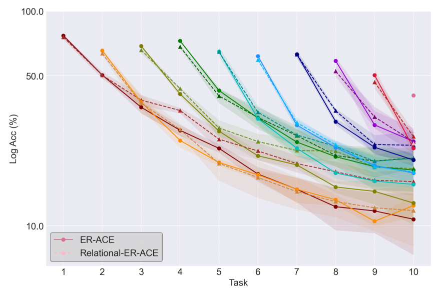
(a) CIFAR-100 ER/RER



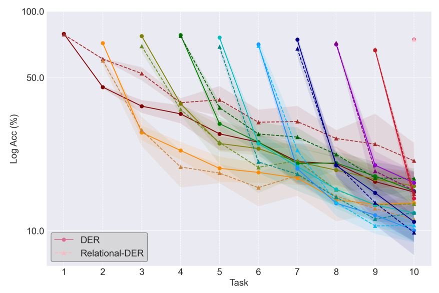
(b) CIFAR-100 ER-ACE/RER-ACE



(c) CIFAR-100 DER++/RDER



(d) Tiny-ImageNet ER-ACE/RER-ACE



(e) Tiny-ImageNet DER++/RDER

Fig. 7. Classification accuracy (%) of each task in the whole training process. The dot-solid lines represent the comparison methods ER, ER-ACE, or DER++ and the triangle-dashed lines represent our methods RER, RER-ACE, or RDER.

where $C_t$ is the number of all seen classes, $B_c$ is the sample number of each class in a batch, and $\sum_{c=1}^{C_t} B_c = B$. This formulation shows that the coefficients $G(j)$ implicitly model the relationship between the knowledge extracted from each training sample of the new task and that from the average meta samples. Then the first derivative term in Eq. (13) can be represented as:

$$\nabla_{\phi}\mathcal{L}^{bf}(\mathcal{D}^{bf};\theta(\phi)) = -\frac{\eta_\theta}{B}\sum_{i=1}^{B} G(j)\cdot \frac{\partial h(\phi)}{\partial \phi}\bigg|_{\phi^k}. \tag{17}$$

Therefore, the gradient of the RRN parameters $\phi$ can be calculated by Eq. (8), which can be easily done in PyTorch [53] with the automatic differentiation.

## Appendix B
## More Experiment Details

In this section, we first illustrate some experimental details and then present some additional results to demonstrate the effectiveness of our method further.

### 1) How to Apply Our Method to Other Baselines:

In the main text of the paper, we take ER as an example to illuminate how our proposed approach helps *rehearsal-based* continual learning models deal with the 'stability-plasticity' dilemma. Here we present how to apply our proposed method to other baselines, *i.e.*, ER-ACE [29], and DER++ [26].

*a) Relational-ER-ACE (RER-ACE):* ER-ACE [29], which represents 'Experience Replay with Asymmetric Cross-Entropy', combines the losses of the new task samples and the memory buffer samples as

$$\begin{aligned}\mathcal{L}^{tr}(\theta) = \frac{1}{B}\sum_{i=1}^{B}&\lambda^D L_{CE}(x_i^D;\theta, C_{curr})\\ &+\lambda^M L_{CE}(x_i^M;\theta, C_{curr}\cup C_{curr}),\end{aligned} \tag{18}$$

where the hyperparameters $\Lambda_D$ and $\Lambda_M$ are preset to 1 in [29]. The $\mathcal{L}_{CE}(\mathcal{D};C)$ is defined as:

$$L_{CE}(x;\theta,C) = -\log\frac{z_c(x)}{\sum_{c'\in C} z_{c'}(x)}, \tag{19}$$

where the sample $x$ belongs to the $c$-th class and $z_c(x)$ is the $c$-th element of the main classification network output $f(x;\theta)$.

Obviously, it is straightforward to apply our proposed method to ER-ACE, where the RRN still generates the weights $[\Lambda_D, \Lambda_M]$ for each training sample pair. The outer-loop optimization is the same as Eq. (4) and the generated sample weights are applied to the inner-loop optimization Eq. (18).

*b) Relational-DER (RDER):* The loss function of DER++ is shown in the following

$$\begin{aligned}\mathcal{L}^{tr}(\theta) = \frac{1}{B}\sum_{i=1}^{B}&\lambda^D L_{CE}(x_i^D;\theta) + \lambda^M L_{CE}(x_i^M;\theta)\\ &+\lambda^{KD} L_{KD}(x_i^M;\theta),\end{aligned} \tag{20}$$

which involves three hyperparameters $[\lambda^D, \lambda^M, \lambda^{KD}]$ and $L_{KD}$ represents the knowledge distillation loss to minimize the distance between predict logits and the saved previous logits. Intuitively, the outer-loop loss function for the RRN in Eq. (4) can be reformulated as:

$$L^{bf}(x;\theta) = L^{bf}_{CE}(x;\theta) + L^{bf}_{KD}(x;\theta), \tag{21}$$

which is the sum of the CE loss and the distillation loss.

### 2) Other Hyperparameters:

In the warm-up stage (*i.e.* the epochs before $Iter_{warm}$), we use preset weights in the inner loop optimization like previous methods. Specifically, for RER and RER-ACE, the weight for the CE loss of new and old task samples is preset as 1 and 0.5, respectively. And for RDER, the preset weights are 1, 0.5, and 0.2 for the CE loss of new task samples $\mathcal{L}_{CE}(\mathcal{D}_t)$, the CE

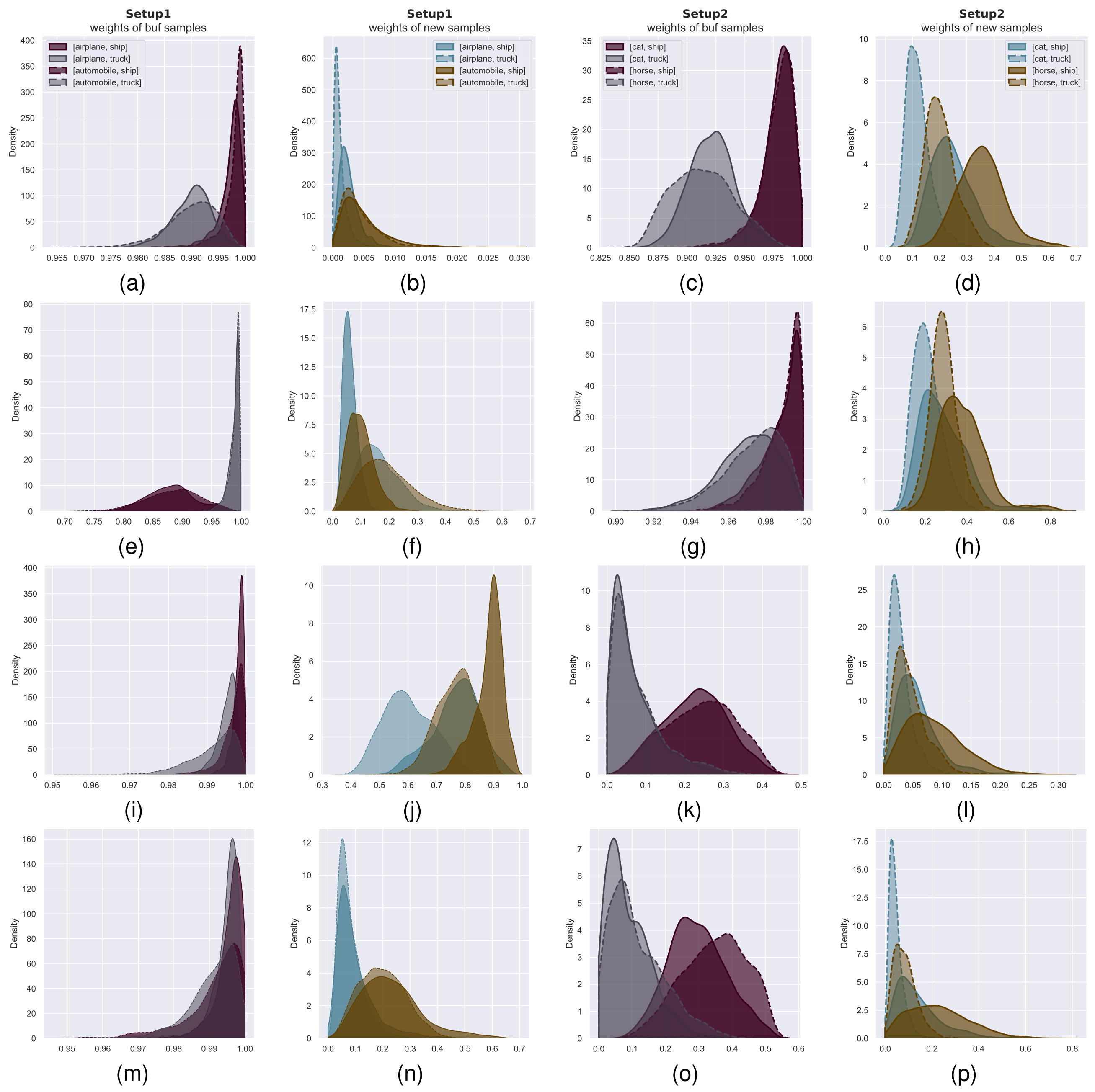


Fig. 8. Weight distributions at multiple epochs during training. The first two columns are weight distributions for buffer samples and new task samples in Setup 1, while the last two columns are those for buffer samples and new task samples in Setup 2. Each row represents the weight distribution of epochs 5, 15, 25, and 35 of the whole training process of task 2, and the weight distributions of epoch 45 are shown in Fig. 6.

loss of memory buffer samples $\mathcal{L}_{CE}(\mathcal{M}_t)$, and the distillation loss of memory buffer samples $\mathcal{L}_{KD}(\mathcal{M}_t)$, respectively.

### 3) Other Visualization of Each Task Accuracy:

Here we show the classification accuracy of each task of ER-ACE/RER-ACE and DER++/RDER on CIFAR-100 in Fig. 7(a) and Fig. 7(c), and on Tiny ImageNet in Fig. 7(b) and Fig. 7(d). Similar to Fig. 7, our method obviously improves the accuracy of the previous tasks. Besides, to balance the 'stability' and 'plasticity', the RDER achieves an overall higher performance even though may not outperform DER++ on some new tasks in Fig. 7(c). All of these results further demonstrate the effectiveness of our method, which can easily be adapted to multiple *rehearsal-based* methods.

### 4) Additional Visualization of Weight Distribution:

Here we show more weight distribution of different stages in the model training. As shown in Fig. 8, in different stages of training, weight distributions are dynamically changing, which also demonstrates that the sample weight should not be prefixed. In addition, buffer sample weights are usually larger at the beginning of training. This is because the model has not yet adapted to new samples, resulting in a greater impact on old knowledge, so larger weights may be needed to avoid forgetting. As training proceeds, the impact of task correlation on weight distribution gradually appears, reflecting the rules

TABLE X
COMPUTATION RESOURCE BASED ON NVIDIA GEFORCE RTX 2080TI.

| Method | Memory (Mb) | Average Training Time per Epoch (s) | Average Training Time per Iteration (s) | ACC on CIFAR-10 with M=200 |
|---|---|---|---|---|
| ER | 1584 | 21.82 | 0.0547 | 55.84 |
| RER | 2080 | 40.95 | 0.1156 | 58.59 |
| FastRER | 1712 | 29.05 | 0.0805 | 58.18 |

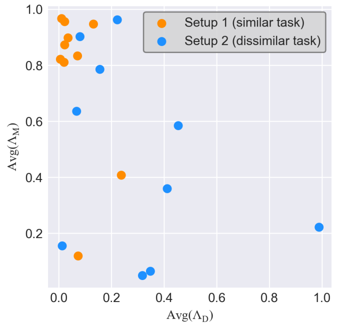


Fig. 9. Scatter plot of the average sample weights under **Setup 1** and **Setup 2**. The horizontal and vertical axes represent the sample weight magnitude of the new and old tasks, respectively.

we analyzed previously in Sec. V-D.

Furthermore, to explore how RRN generates weights for unseen data samples, we also display the average weight distribution on the test dataset under **Setup 1** and **Setup 2**. We use the test samples of task 2 and task 1 to construct the paired data as the inputs of RRN, and then we calculate the average of all generated weights. To avoid stochastic factors, we repeat our experiment 10 times and show their average weights in Fig. 9. As we can see, the average weights of **Setup 1** mainly concentrate on the left upper corner of Fig. 9. That is, for similar tasks, buffer sample weights are relatively larger than new weights to prevent forgetting, the same as the trend of training sample weights in Fig. 8. On the other hand, the average weights of **Setup 2** are relatively separated. These results further demonstrate that our RRN has extracted task relationships for generating weights and also works on unseen test data, rather than simply overfitting on the training data.

*5) The Computation Cost and Acceleration Strategy:*

The proposed method involves a bi-level optimization that introduces an additional calculation burden primarily due to the second-order derivation in the outer loop optimization. To mitigate this, we employ $Interval$ to regulate the update frequency of RRN to strike a better trade-off between the model performance and computation cost, which is a common way to speed up the bi-level optimization [47]. As shown in Table X, the memory usage of RER with $Interval = 5$ is approximately 500MB higher than that of ER, and the training time of RER is roughly twice that of ER. Importantly, it's worth noting that the proposed RRN and bi-level optimization procedures are only involved in the training stage, meaning that no computational overhead is required during testing.

Fortunately, many previous works are trying to accelerate this bi-level optimization framework, such as some first-order approximation [57]. We find that reducing the parameters involved in the second-order derivation of the outer loop optimization can greatly enhance the efficiency of bi-level optimization, with little impact on performance [58]–[60]. To speed up the optimization of RRN, we assume that the parameter $\phi$ of RRN only correlates to the final linear classification layer of the main net, rather than the entire network. With this assumption, the calculating of second-order derivation only involves a few parameters, which can significantly accelerate the optimization of our bi-level framework. For example, if we use ResNet-18 as the main net for the CIFAR-10 dataset, the final classifier comprises merely 5,120 parameters, whereas the entire network contains about 11.69M parameters. So the second-order derivation will be faster and more computation-friendly. As shown in Table X, we denote this accelerated method FastRER, which only increases memory usage by about 200Mb and training time by about 7.2 seconds. Furthermore, FastRER achieves comparable performance to RER.

## REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[3] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "DenseNet: Implementing efficient ConvNet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[4] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[6] A. Douillard, Y. Chen, A. Dapogny, and M. Cord, "PLOP: Learning without forgetting for continual semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4040–4050.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[8] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

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

[11] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[12] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[13] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "PathNet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.

[14] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, “Overcoming catastrophic forgetting with hard attention to the task,” in *International Conference on Machine Learning*, 2018, pp. 4548–4557.

[15] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. E. Bejnordi, “Conditional channel gated networks for task-aware continual learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3931–3940.

[16] S. Yan, J. Xie, and X. He, “DER: Dynamically expandable representation for class incremental learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3014–3023.

[17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, “Overcoming catastrophic forgetting in neural networks,” *National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[18] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[19] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” in *International Conference on Machine Learning*, 2017, pp. 3987–3995.

[20] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, “Progress & compress: A scalable framework for continual learning,” in *International Conference on Machine Learning*, 2018, pp. 4528–4537.

[21] H. Yin, P. Yang, and P. Li, “Mitigating forgetting in online continual learning with neuron calibration,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 260–10 272, 2021.

[22] R. Ratcliff, “Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions.” *Psychological Review*, vol. 97, no. 2, pp. 285–308, 1990.

[23] A. Robins, “Catastrophic forgetting, rehearsal and pseudo rehearsal,” *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.

[24] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “iCaRL: Incremental classifier and representation learning,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[25] D. Lopez-Paz and M. Ranzato, “Gradient episodic memory for continual learning,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 6467–6476.

[26] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, “Dark experience for general continual learning: a strong, simple baseline,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 15 920–15 930.

[27] A. Chaudhry, A. Gordo, P. Dokania, P. Torr, and D. Lopez-Paz, “Using hindsight to anchor past knowledge in continual learning,” in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 6993–7001.

[28] J. Bang, H. Kim, Y. Yoo, J.-W. Ha, and J. Choi, “Rainbow memory: Continual learning with a memory of diverse samples,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8218–8227.

[29] L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau, and E. Belilovsky, “New insights on reducing abrupt representation change in online continual learning,” in *International Conference on Learning Representations*, 2022.

[30] H. Ahn, J. Kwak, S. Lim, H. Bang, H. Kim, and T. Moon, “SS-IL: Separated softmax for incremental learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 844–853.

[31] G. A. Carpenter and S. Grossberg, “A massively parallel architecture for a self-organizing neural pattern recognition machine,” *Computer Vision, Graphics, and Image Processing*, vol. 37, no. 1, pp. 54–115, 1987.

[32] B. Zhou, Q. Cui, X.-S. Wei, and Z.-M. Chen, “BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[33] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, “Learning a unified classifier incrementally via rebalancing,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 831–839.

[34] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, “Efficient lifelong learning with A-GEM,” *arXiv preprint arXiv:1812.00420*, 2018.

[35] A. Mallya and S. Lazebnik, “PackNet: Adding multiple tasks to a single network by iterative pruning,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[36] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, “Variational continual learning,” in *International Conference on Learning Representations*, 2018.

[37] A. Maracani, U. Michieli, M. Toldo, and P. Zanuttigh, “Recall: Replay-based continual learning in semantic segmentation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7026–7035.

[38] C.-B. Zhang, J.-W. Xiao, X. Liu, Y.-C. Chen, and M.-M. Cheng, “Representation compensation networks for continual semantic segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7053–7064.

[39] G. Yang, E. Fini, D. Xu, P. Rota, M. Ding, T. Hao, X. Alameda-Pineda, and E. Ricci, “Continual attentive fusion for incremental learning in semantic segmentation,” *IEEE Transactions on Multimedia*, 2022.

[40] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei, and Y. Gong, “Few-shot class-incremental learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 183–12 192.

[41] D.-W. Zhou, F.-Y. Wang, H.-J. Ye, L. Ma, S. Pu, and D.-C. Zhan, “Forward compatible few-shot class-incremental learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9046–9056.

[42] Y. Cui, W. Deng, X. Xu, Z. Liu, Z. Liu, M. Pietikäinen, and L. Liu, “Uncertainty-guided semi-supervised few-shot class-incremental learning with knowledge distillation,” *IEEE Transactions on Multimedia*, 2022.

[43] S. Thuseethan, S. Rajasegarar, and J. Yearwood, “Deep continual learning for emerging emotion recognition,” *IEEE Transactions on Multimedia*, vol. 24, pp. 4367–4380, 2021.

[44] W. Nie, R. Chang, M. Ren, Y. Su, and A. Liu, “I-gcn: Incremental graph convolution network for conversation emotion detection,” *IEEE Transactions on Multimedia*, vol. 24, pp. 4471–4481, 2021.

[45] D.-W. Zhou, Q.-W. Wang, Z.-H. Qi, H.-J. Ye, D.-C. Zhan, and Z. Liu, “Deep class-incremental learning: A survey,” *arXiv preprint arXiv:2302.03648*, 2023.

[46] M. Ren, W. Zeng, B. Yang, and R. Urtasun, “Learning to reweight examples for robust deep learning,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 4334–4343.

[47] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, “Meta-weight net: Learning an explicit mapping for sample weighting,” *arXiv preprint arXiv:1902.07379*, 2019.

[48] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.

[49] Stanford, “Tiny imagenet challenge (CS231n),” http://tiny-imagenet.herokuapp.com/, 2015.

[50] 20 newsgroups. [Online]. Available: http://qwone.com/∼jason/20Newsgroups/

[51] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, “Dbpedia-a crystallization point for the web of data,” *Journal of web semantics*, vol. 7, no. 3, pp. 154–165, 2009.

[52] W. Hu, Q. Qin, M. Wang, J. Ma, and B. Liu, “Continual learning by using information of each class holistically,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 7797–7805.

[53] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[54] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[55] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, “Gradient based sample selection for online continual learning,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 11 816–11 825.

[56] Q. Pham, C. Liu, D. Sahoo, and H. Steven, “Contextual transformation networks for online continual learning,” in *International Conference on Learning Representations*, 2020.

[57] A. Nichol and J. Schulman, “Reptile: a scalable meta-learning algorithm,” *arXiv preprint arXiv:1803.02999*, vol. 2, no. 3, p. 4, 2018.

[58] R. Wang, X. Jia, Q. Wang, Y. Wu, and D. Meng, “Imbalanced semi-supervised learning with bias adaptive classifier,” in *11th International Conference on Learning Representations (ICLR 2023)*, 2023.

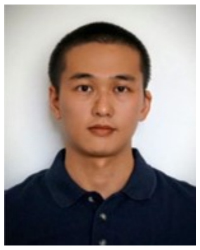
[59] X. Jia, R. Wang, D. Meng, and X. Feng, “Delving into the hierarchical structure for efficient large-scale bi-level learning,” 2022.

[60] Q. Wang, R. Wang, Y. Wu, X. Jia, and D. Meng, “Cba: Improving online continual learning via continual bias adaptor,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 082–19 092.

**Quanziang Wang** received the B.Sc degree from the Xi'an Jiaotong University, Xi'an, China, in 2019. He is currently working toward the Ph.D. degree with the School of Mathematics and Statistics, Xi'an Jiaotong University. His research mainly focus on continual learning and semi-supervised learning.

**Renzhen Wang** received the B.Sc. degree from Dalian University of Technology, Dalian, China, in 2016 and Ph.D. degree from Xi'an Jiaotong University, Xi'an, China, in 2022. He is currently an assistant professor in the School of Mathematics and Statistics, Xi'an Jiaotong University. His current research interests include semi-supervised learning, continual learning and medical image analysis.

**Yuexiang Li** received Ph.D. degree from the University of Nottingham, United Kingdom. He is the full Professor in Guangxi Key Laboratory for Genomic and Personalized Medicine, Guangxi Medical University (GXMU). He is the academic leader of the discipline of artificial intelligence, and leads the Medical AI ReSearch (MARS) group in the university. His research interest include intelligent analysis and processing of medical images (including microscopic images, pathological slices and multimodal medical images).

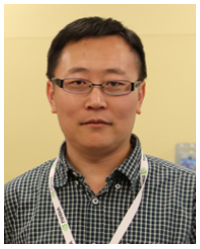


**Dong Wei** received the Ph.D. degree in Computer Engineering from the University of Singapore, Singapore, in 2013. Since 2018, he has been a Senior Researcher at the Tencent Jarvis Lab, Shenzhen, China. His research interests include medical image analysis, with a current focus on data and annotation efficient approaches.

**Hong Wang** received the Ph.D. degree from School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China, in 2021, under the supervision of Prof. Deyu Meng. She was a research intern with the Tencent Rhino Bird Elite Talent Program from 2020 to 2021. She is currently a senior researcher at Tencent, Shenzhen. Her current research interests focus on the design of model-driven and data-driven deep learning techniques for effective and interpretable image processing including natural image restoration and medical image analysis.

**Kai Ma** received the Ph.D. degree from the University of Illinois at Chicago, Chicago, IL, USA, in 2014. He was with Siemens Medical Solution, Princeton, NJ, USA, for more than five years. He is currently a Principal Researcher with the Jarvis Lab, Tencent, Shenzhen, China. His research interests include medical image analysis, deep learning, computer vision, and brain–computer interface.

**Yefeng Zheng** (Fellow, IEEE) received the B.E. and M.E. degrees from Tsinghua University, Beijing, in 1998 and 2001, respectively, and the Ph.D. degree from the University of Maryland, College Park, MD, USA, in 2005. After graduation, he joined Siemens Corporate Research, Princeton, NJ, USA. He is currently the Director and the Distinguished Scientist of Tencent Jarvis Lab, Shenzhen, China, leading the company's initiative on Medical AI. His research interests include medical image analysis, computer vision, natural language processing, and deep learning. Dr. Zheng is a fellow of the American Institute for Medical and Biological Engineering (AIMBE).



**Deyu Meng** (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 2001, 2004, and 2008, respectively. He was a Visiting Scholar with Carnegie Mellon University, Pittsburgh, PA, USA, from 2012 to 2014. He is currently a professor with School of Mathematics and Statistics, Xi'an Jiaotong University, and adjunct professor with Faculty of Information Technology, The Macau University of Science and Technology. His current research interests include model-based deep learning, variational networks, and meta-learning.