# Appendix

## A Problem Formulation and RKD Methods

### A.1 Problem Formulation of CIL.

In the CIL setting, a dataset $\mathcal{D} = \{(x, y)|x \in \mathcal{X}, y \in \mathcal{Y}\}$ is split to $T$ subsets: $\mathcal{D} = \mathcal{D}^1 \cup \mathcal{D}^2 \cup \cdots \cup \mathcal{D}^t$, where $\mathcal{X}$ is a set of images with labels $\mathcal{Y}$ and the subsets have no overlapped classes, then a learning system is trained to learn each subset incrementally. At task $t$, we have a model $f^{t-1}_{\theta,W}$ which has incrementally learned the old classes $\tilde{\mathcal{C}}^{t-1} = \{\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^{t-1}\}$, where $\theta, W$ denote the parameters of the feature extractor and the linear layer of the network, respectively and $\mathcal{C}^i$ denotes the classes in $i$ subset (task $i$). Now, given the new subsets $\mathcal{D}^t$ with the new classes $\mathcal{C}^t$, the goal is to train a new model $f^t_{\theta,W}$ that can perform classification on all the classes $\tilde{\mathcal{C}}^t = \tilde{\mathcal{C}}^{t-1} \cup \mathcal{C}^t$. In rehearsal-knowledge-distillation-based (RKD) methods, they store a small number of image exemplars of the old classes after the completion of each incremental learning task for experience replay at the future tasks. We denote $E^t$ as the selected exemplars of the current task (the new classes) to be stored after the completion of task $t$. We denote $\tilde{E}^t = \tilde{E}^{t-1} \cup E^{t-1}$, $\tilde{\mathcal{D}}^t = \mathcal{D}^t \cup \tilde{E}^t$, $\tilde{\mathcal{X}}^t = \{x|(x, y) \in \tilde{\mathcal{D}}^t\}$, $\tilde{\mathcal{Y}}^t = \{y|(x, y) \in \tilde{\mathcal{D}}^t\}$ as all the stored exemplars of the old classes, all the observable dataset, all the available images and labels at task $t$, respectively.

### A.2 Training Strategy of RKD Method.

Most previous works [1, 2, 3, 4] of RKD methods have the common process that uses all the available data to train the new model by minimizing two losses: the classification cross-entropy (CE) loss and the knowledge distilation (KD) loss. The CE loss is used to learn new classes and The KD loss is used to encourage the new network $f^t_{\theta,W}$ to mimic the output of the previous task model $f^{t-1}_{\theta,W}$. The CE loss ($L_{CE}$) and the KD loss ($L_{kd}$) are typically computed as follows:

$$L_{CE} = \sum_{(x,y)\in\tilde{\mathcal{D}}^t} \sum_{i=1}^{m+n} -\delta_i(x) log[\sigma_i(f^t_{\theta,W}(x))] \tag{1}$$

$$L_{kd} = \sum_{x\in\tilde{\mathcal{X}}^t} \sum_{i=1}^{m} -\sigma_i(f^{t-1}_{\theta,W}(x)) log[\sigma_i(f^t_{\theta,W}(x))]. \tag{2}$$

where $\delta_i(x)$ is the label indicator function, $m, n$ are the number of learned and new classes respectively and $\sigma$ is either the $softmax$ or $sigmoid$ function. So the new model $f^t_{\theta,W}$ are trained by the overall loss:

$$L = L_{kd} + \lambda L_{CE} \tag{3}$$

where $\lambda$ is the hyper parameter. Note that $f^t_{\theta,W}$ is continually updated at task $t$, whereas the network $f^{t-1}_{\theta,W}$ is frozen and will not be stored after the completion of task $t$.

However, the dataset of the new classes ($\mathcal{D}^t$) in the new task are out-of distribution (OOD) with the original training data ($\tilde{\mathcal{D}}^{t-1}$) of the old model $f^{t-1}_{\theta,W}$, so the performances of KD suffer from huge degradation. Moreover, RKD methods suffer from the task-recency bias [5]. After training the new model, to tackle the task-recency bias, various RKD methods have different subsequent processing. For example, iCaRL [1] takes the nearest-mean-of-exemplars (NME) classification strategy to make inference, BiC [3] trains a bias-correction layer with a balanced dataset and EEIL [2] further fine-tunes the whole model by using the balanced dataset of stored exemplars.

## B EDBL Algorithm

The training process of our method (EDBL) is shown in **Algorithm 1**. At each incremental learning task, we first make data augmentation by re-sampling Mixup, then we train the new model with the mixed data in the same way as in the basic RKD method.At last, we fine-tunes the whole model by Eq. 16 in the balancing training phase.

**Algorithm 1** EDBL Algorithm for CIL

**Input:** Exemplars ($\tilde{E}^t, t > 1$) and data of new classes ($D^t$), $f^{t-1}$.
**Output:** Exemplars $\tilde{E}^{t+1} = \tilde{E}^t \cup E^t$, The New Model $f^t$
**Mixup with Re-sampling:**
Employ Mixup and Re-sample old classes to generate interpolated dataset $\hat{D}$.
**Phase 1: MKD Training**
**for** $i = 1$ **to** $T_1$ **do**
  sample a mini-batch $\hat{D}_m$ from $\hat{D}$
  $L_{ce}$ ←Eq. 2, $L_{kd}$ ← Eq. 3,
  $L(\Theta) = \frac{1}{m}\sum_{(\hat{x},\hat{y})\in\hat{D}_m} L_{ce}(\hat{x},\hat{y},\Theta) + \gamma_1 L_{kd}(\hat{x},\hat{y},\Theta)$
  Update $\Theta^{t+1} = \Theta^t - \eta_1 \nabla L(\Theta)$
**end for**
**Phase 2: Balancing Training**
**for** $i = 1$ **to** $T_2$ **do**
  sample a mini-batch $\hat{D}_m$ from $\hat{D}$
  $\mathcal{IIB}(\hat{x},\hat{y},\Theta)$ ← Eq. 15
  $L_{overall}(\Theta) = \frac{1}{m}\sum_{(\hat{x},\hat{y})\in\hat{D}_m} \lambda_k \frac{L_{ce}(\hat{x},\hat{y},\Theta)}{\mathcal{IIB}(\hat{x},\hat{y},\Theta)} + \gamma_2 L_{kd}(\hat{x},\hat{y},\Theta)$, $L_{ce}, L_{kd}$ is computed by Eq.2 and 3
  Update $\Theta^{t+1} = \Theta^t - \eta_2 \nabla L_{overall}(\Theta)$
**end for**
**Exemplar Management:** Utilize the strategy in [1] to make exemplar management to select $E^t$ (maybe also remove some samples from $\tilde{E}^t$) to generate $\tilde{E}^{t+1}$ .

# C Implement Detail

## C.1 Typical Training Hyper-parameters Selection

We draw $\lambda$ in Eq. 1 randomly from the Beta function B = (1, 1) for all the experiments. In re-ampling Mixup, we heuristically make sure the number of samples from old classes in a batch isn't less than 32. We semi-heuristically set the typical training hyper-parameters, e.g. epoch, learning rate, batch-size, etc. and tune a few of parameters on CIFAR-100 and Tiny-Imagenet experiments with 5 incremental learning phases, then we use the same setting of these parameters for other experiments on CIFAR-10/100 and Tiny-Imagenet, respectively. All experiments use the same batch size, weight decay, momentum: 128, 0.0002, 0.9, respectively. Other training details of the two stages are as below:

**MKD Training.** In Mixup-based Knowledge distillation (MKD) We train the new model with different hyper parameters on different datasets. For CIFAR-10/100, we train the network for 150 epochs at each task. The learning rate is set to 0.1, and reduced by a factor of 10 at 60, 100, 130 epochs. As for Tiny-Imagenet, the number of training epochs is 250 at each task. The learning rate is set to 0.1, and reduced by a factor of 10 at epochs 75, 125, 175 and 225.

**Balancing Training.** For CIFAR-10/100, the training epoch is 100, the learning rate is set to 0.01 and reduced by a factor of 10 at 30, 60, 80 epochs. For Tiny-Imagenet, the number of training epochs is 150 for each task. The learning rate is set to 0.01, and reduced by a factor of 10 at epochs 60, 100 and 130.

## C.2 Tuning on $\lambda_k$ and $\alpha$

We mainly make tuning on $\lambda_k$ in Eq. 14 and $\alpha$ in Eq. 16. $\lambda_k$ origins from the IB method and it is given by $\lambda_k = \gamma n_k^{-1} / \sum_{i=1}^{K} n_i^{-1}$, where $k$ is the label, $n_i$ is the number of samples in the $k$-th class and $\gamma$ is the hyper-parameter. We tune $\gamma$ and $\alpha$ by grid searching and adopt different values on different dataset and different experiments, which are given in Table 6.

Table 6

| Dataset | Experiments | $\gamma$ | $\alpha$ |
|---|---|---|---|
| CIFAR-10 | Base-0-2 Phases | 10 | 1e-6 |
| | Base-0-5 Phases | 100 | 5e-6 |
| CIFAR-100 | Base-0-2 Phases | 100 | 5e-6 |
| | Base-0-5 Phases | 300 | 5e-6 |
| | Base-0-10 Phases | 100 | 5e-6 |
| | Base-half-5 Phases | 300 | 5e-6 |
| | Base-half-10 Phases | 100 | 5e-6 |
| Tiny-Imagenet | Base-half-5 Phases | 10 | 1e-6 |
| | Base-half-10 Phases | 10 | 5e-6 |

# D Comparison Results of Average Accuracy

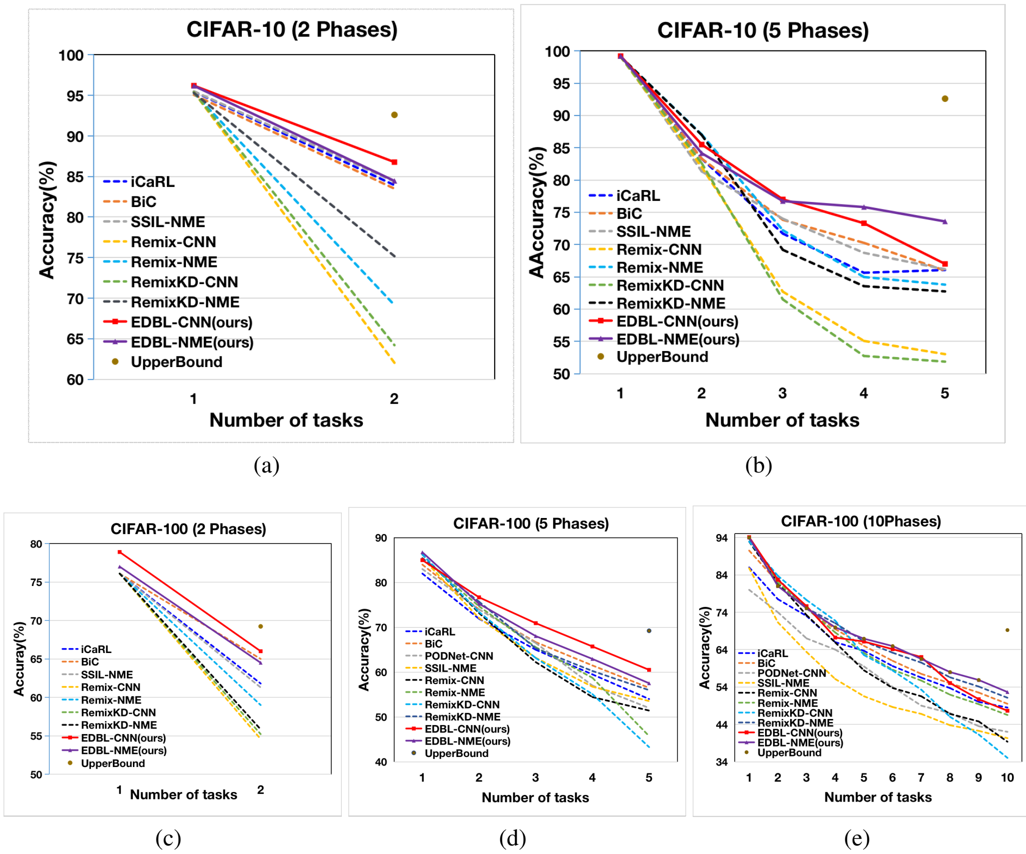


Figure 4: Comparison results of average accuracy of Base-0 experiments on CIFAR-10 with 2, 5 phases and CIFAR-100 with 2, 5, 10 pahses.

We conducted experiments on CIFAR-10/100 following Base-0 protocol to evaluate our method. The memory budgets on CIFAR-10 and CIFAR-100 are fixed to 200 and 2000, respectively. CIFAR-10 is split into 2 and 5 phases while CIFAR-100 is split into 2, 5 and 10 phases. Remix[6] adapted Mixup to generate more effective interpolated data to tackle the long-tail learning. Our method employs Mixup technique, so we use Remix as a compared method and directly employ Remix to train the new model to learn new classes incrementally with the optimal hyper-parameters given in [6]. In this supplement, to compare Remix with our method fairly, we further combine Remix with knowledge distillation (KD) to conduct experiments and report the results of CNN output and the nearest-mean-of-exemplar strategy (NME) (denoted as RemixKD-CNN and RemixKD-NME, respectively).

The comparison of the results are shown in Fig. 4. From Fig. 4, we can find that our methods (EDBL-CNN and EDBL-NME) outperform all the baselines nearly on every incremental task except EDBL-CNN loses to some baselines at the experiment on CIFAR-100 with 10 phases. Especially, our methods surpass Remix-CNN, Remix-NME and RemixKD-CNN, RemixKD-NME significantly by large margins about [1.1%, 22%] at the last incremental phase. We re-conducted the experiment of our methods on CIFAR-10 with 5 phases and we got better results, compared with the results given in Table 2. The incremental average accuracies of EDBL-CNN and EDBL-NME on CIFAR-10 with 5 phases are 80.4% and 81.894%, respectively, which surpass all the baselines significantly.

# References


[1] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on*

*Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[2] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018.

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

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

[5] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[6] Hsin-Ping Chou, Shih-Chieh Chang, Jia-Yu Pan, Wei Wei, and Da-Cheng Juan. Remix: rebalanced mixup. In *European Conference on Computer Vision*, pages 95–110. Springer, 2020.