Prospective Representation Learning for Non-Exemplar Class-Incremental Learning

Wuxuan Shi[1], **Mang Ye**[1,2*]
[1]School of Computer Science, Wuhan University, Wuhan, China
[2] Taikang Center for Life and Medical Sciences, Wuhan University, Wuhan, China
{wuxuanshi, yemang}@whu.edu.cn

NeurIPS 2024

Background

- Class-Incremental Learning:

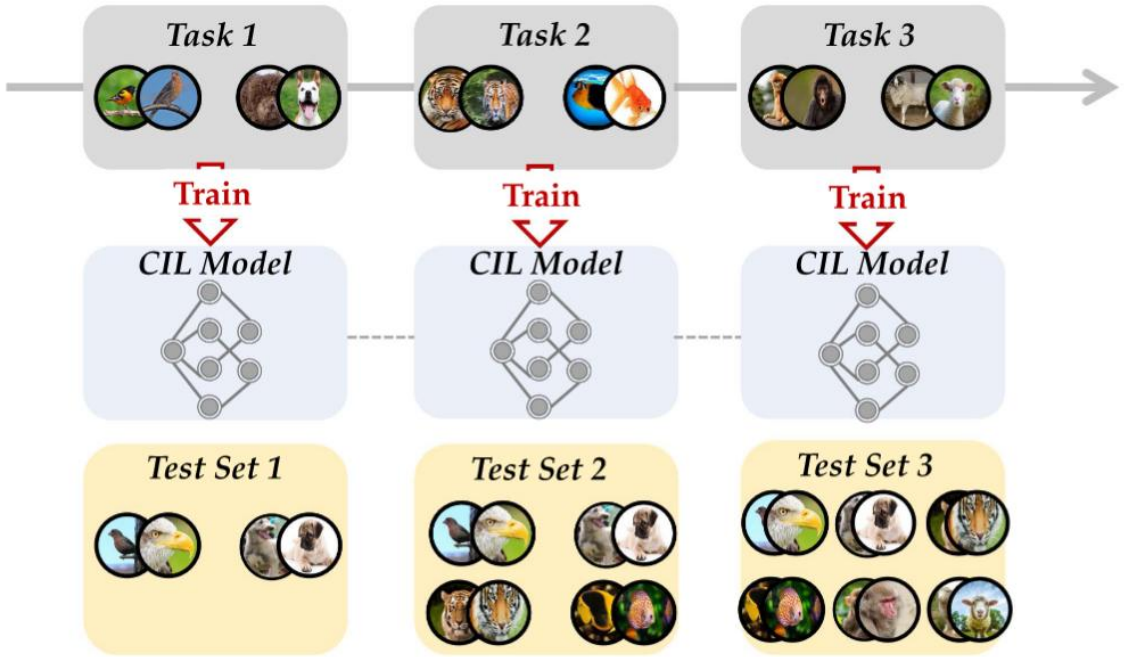


Fig. 1. The setting of CIL. Non-overlapping classes arrive sequentially, and the model needs to learn to classify all the classes incrementally. After learning each task, the model is evaluated among all seen classes. An ideal model should perform well in the newly learned classes and remember the former without forgetting.

Background

▶ Existing Methods:

Data Replay: the model can keep a relatively small set, namely exemplar set, to reserve the representative instances from former tasks.

Dynamic Networks: dynamic networks are designed to dynamically adjust the model's representation ability to fit the evolving data stream.

Parameter Regularization: estimating the importance of different parameters for past tasks and then limit the updating of these important parameters when learning new tasks.

Knowledge Distillation: assuming the old model is a good "teacher" for all the seen classes in previous tasks.

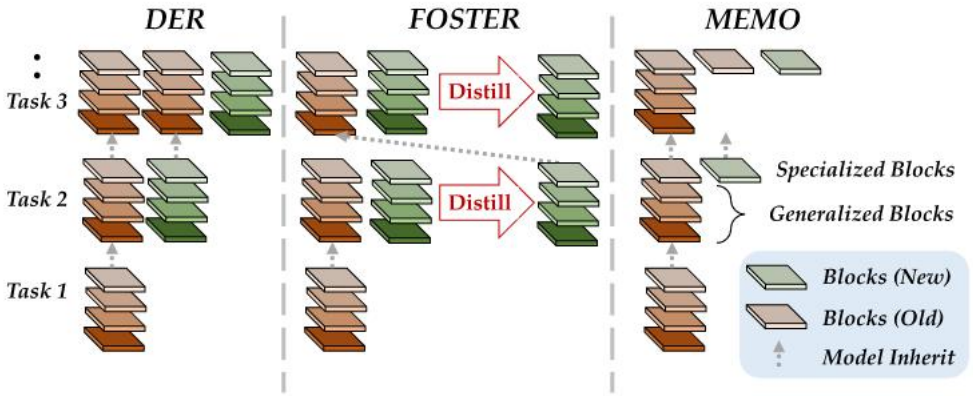


Fig. 4. Illustration of network structure evolving in backbone expansion. **Left:** DER expands a new backbone per incremental task. **Middle:** FOSTER adds an extra model compression stage, which maintains limited model storage. **Right:** MEMO decouples the network structure and only expands specialized blocks.

Motivation

In non-exemplar class-incremental learning (NECIL), a serious challenge is to discriminate between old and new classes without access to old data.

Most methods usually start considering handling conflicts between old and new classes **only when new tasks arrive**.

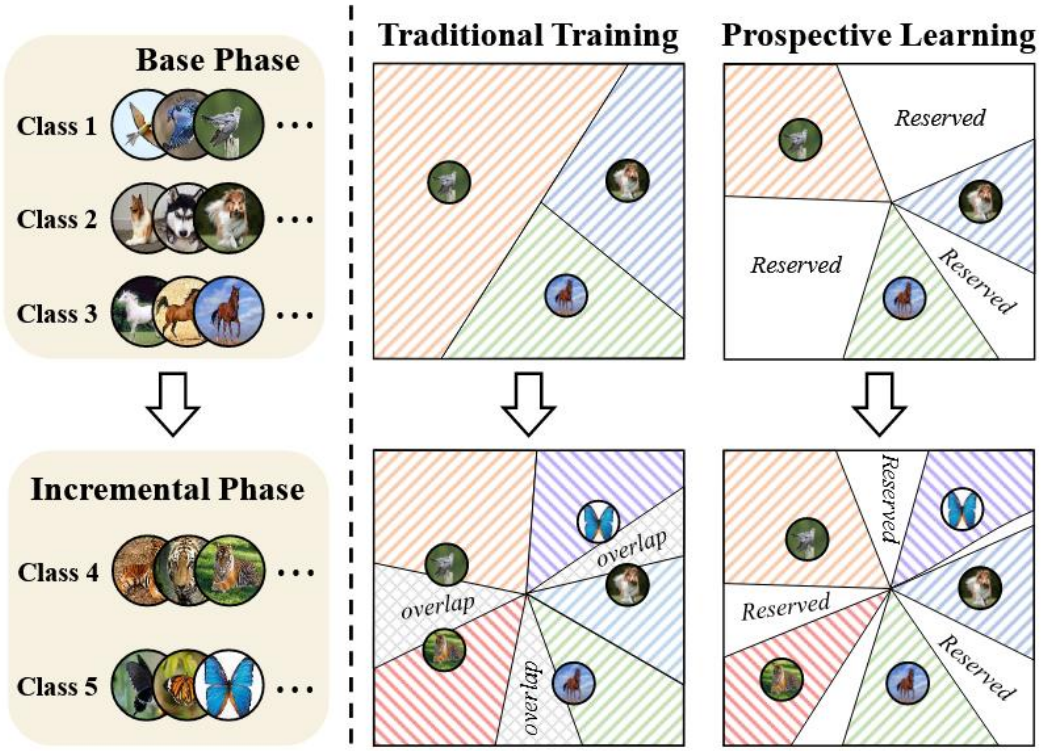


Figure 1: The traditional training paradigm in NECIL considers conflicts between old and new classes only when new classes arrive and is prone to overlap. We suggest prospective learning to reduce conflicts: (1) reserve space for unknown classes; (2) make the newly coming class embedded in the reserved space.

Baseline

▶Base phase

$$\underset{\theta_t,\varphi_t}{\operatorname{argmin}}\mathcal{L}_t = \mathcal{L}_{ce}(\theta_t, \varphi_t; D_t) = -\underset{(x,y)\sim D_t}{\mathbb{E}}\left[y \cdot \log\left(\mathcal{G}_{\varphi_t}(\mathcal{F}_{\theta_t}(x))\right)\right]$$

▶Incremental phase

$$\mathcal{L}_t = \mathcal{L}_{ce}(\theta_t, \varphi_t; D_{0:t-1}) + \mathcal{L}_{ce}(\theta_t, \varphi_t; D_t)$$

knowledge distillation + prototype rehearsal

$$\mathcal{L}_{kd}(\theta_t; \theta_{t-1}, D_t) = \sum_{x \in X_i} \|\mathcal{F}_{\theta_t}(x) - \mathcal{F}_{\theta_{t-1}}(x)\|_2$$

encourage the model to mimic the previous representation

$$\mathcal{L}_{pro}(\varphi_t; \tilde{\boldsymbol{P}}_{0:t-1}) = -\underset{(\tilde{\boldsymbol{p}}^c, c)\sim \tilde{\boldsymbol{P}}_{0:t-1}}{\mathbb{E}}\left[c \cdot \log\left(\mathcal{G}_{\varphi_t}(\tilde{\boldsymbol{p}}^c)\right)\right].$$

$$\boldsymbol{p}^c = \underset{(x,y)\sim D_{t-1}}{\mathbb{E}}\left[\mathcal{F}_{\theta_{t-1}}(x) \mid y = c\right]$$

$$\mathcal{L}_t = \mathcal{L}_{ce}(\theta_t, \varphi_t; D_t) + \alpha_1 \mathcal{L}_{kd}(\theta_t; \theta_{t-1}, D_t) + \alpha_2 \mathcal{L}_{pro}(\varphi_t; \tilde{\boldsymbol{P}}_{0:t-1})$$

Method

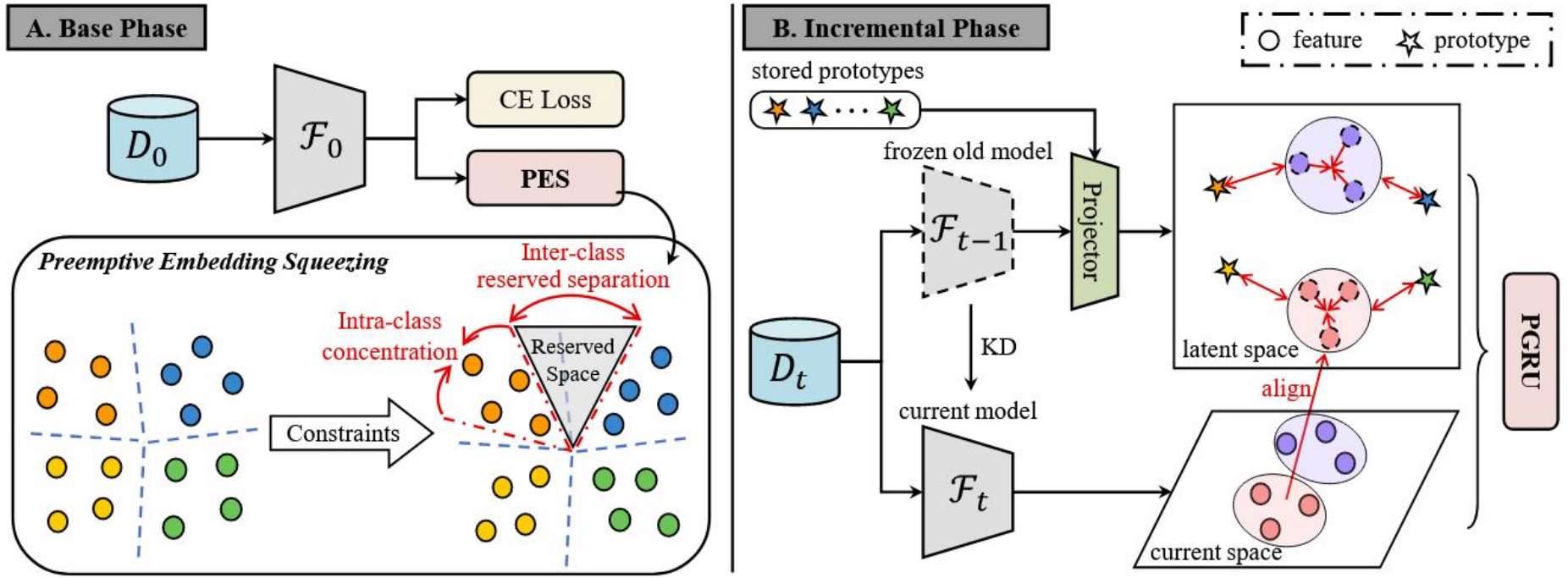


Figure 2: Overview of our Prospective Representation Learning (PRL) for NECIL. (A) During the base phase, we impose a preemptive embedding squeezing (PES) constraint to squeeze the space of the current class in preparation for accepting future new classes. (B) During the incremental phase, a prototype-guided representation update (PGRU) strategy is proposed to keep new class features away from old class prototypes in the latent space, which guides the update of the current model to mitigate the confusion of new classes with old classes.

Method

▶For a batch $B = \{x^i, y^i\}_{i=1}^n \in D_t$

$$s = \sum_{\substack{\forall x^i, x^j \in B \\ y_i = y_j}} \langle \mathcal{F}_{\theta_t}(x^i), \mathcal{F}_{\theta_t}(x^j) \rangle,$$

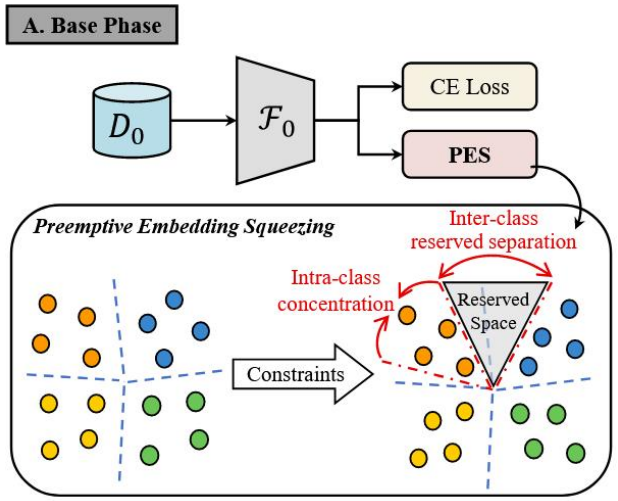
$$d = \sum_{\substack{\forall x^i, x^k \in B \\ y_i \neq y_k}} \langle \mathcal{F}_{\theta_t}(x^i), \mathcal{F}_{\theta_t}(x^k) \rangle,$$

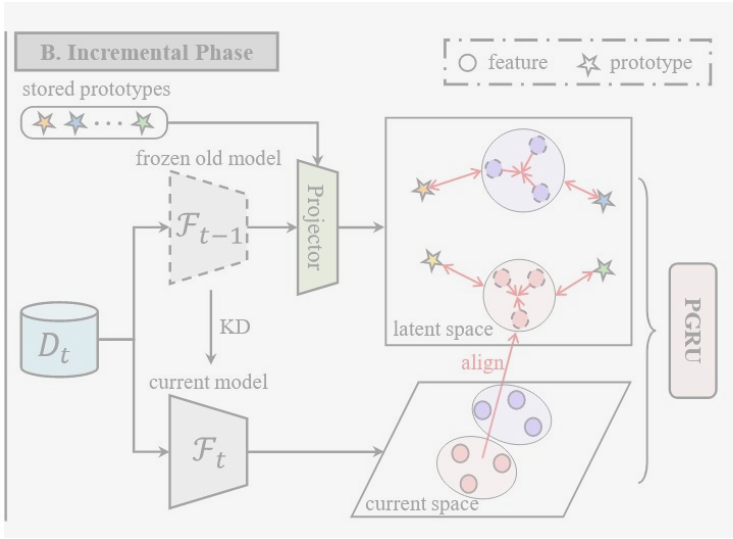
$$\mathcal{L}_{PES}(\theta_t; D_t) = (1 - s) + \lambda * (1 + d),$$

facilitate intra-class concentration

reinforce inter-class reserved separation

▶ $\mathcal{L}_t = \mathcal{L}_{ce}(\theta_t, \varphi_t; D_t) + \gamma * \mathcal{L}_{PES}(\theta_t; D_t).$



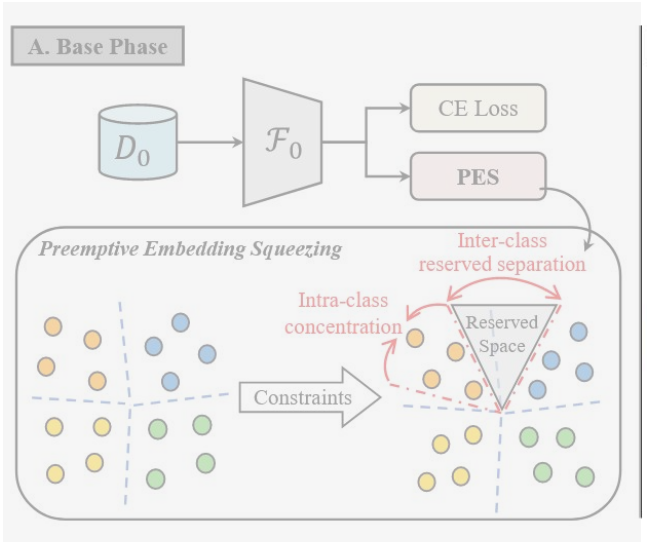


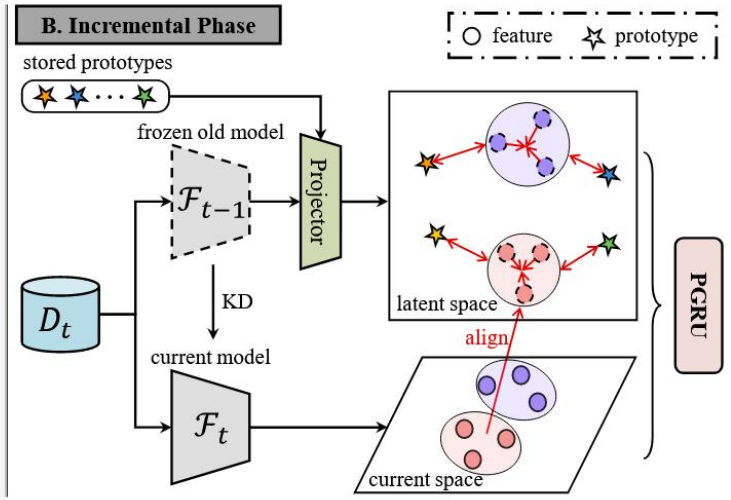
Method

▶Aim: embed the new class into the previously reserved space

The plain idea is to keep the new classes well clustered and distanced from the old ones.

However, it is not practical to establish a relationship directly between the saved prototypes and the new class features extracted by the current model due to the continual updating of the current embedding space.





$$\mathcal{L}_{ort} = \sum_{\substack{\forall x^i \in B \\ \forall \boldsymbol{p}^c \in \boldsymbol{P}_{0:t-1}}} |\langle \mathcal{P}_{\phi_t}(\mathcal{F}_{\theta_{t-1}}(x^i)), \mathcal{P}_{\phi_t}(\boldsymbol{p}^c) \rangle|$$

promote orthogonality between the new class features and the old class prototypes.

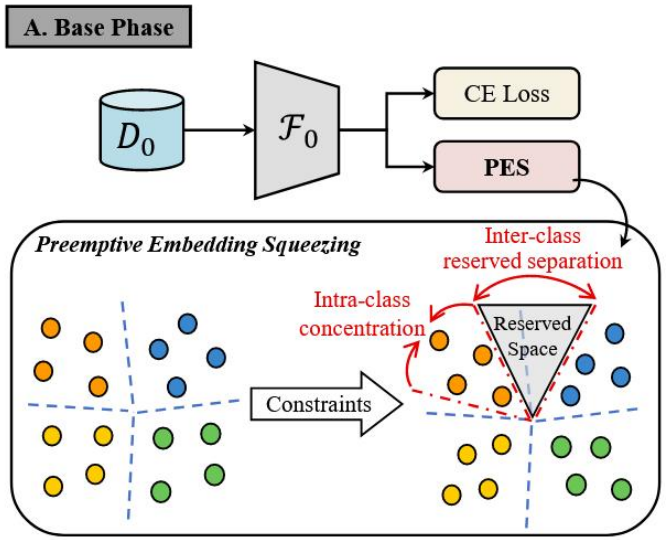
$$\mathcal{L}_{align} = \sum_{x \in X_i} \mathcal{L}_{MSE}(\mathcal{P}_{\phi_t}(\mathcal{F}_{\theta_{t-1}}(x^i)), \mathcal{F}_{\theta_t}(x^i))$$

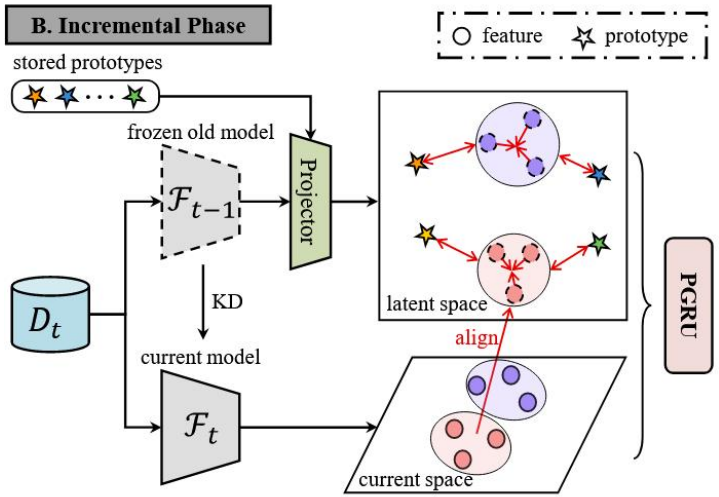
$$\mathcal{L}_{PGRU} = \mathcal{L}_{ort}(\phi_t; D_t, \boldsymbol{P}_{0:t-1}) + \mathcal{L}_{align}(\theta_t, \phi_t; D_t)$$

▶Final loss of incremental phase

$$\begin{aligned}\mathcal{L}_t = &\mathcal{L}_{ce}(\theta_t, \varphi_t; D_t) + \alpha_1 \mathcal{L}_{kd}(\theta_t; \theta_{t-1}, D_t) + \\ &\alpha_2 \mathcal{L}_{pro}(\varphi_t; \tilde{\boldsymbol{P}}_{0:t-1}) + \alpha_3 \mathcal{L}_{PGRU}(\theta_t, \phi_t; D_t, \boldsymbol{P}_{0:t-1}).\end{aligned}$$

Method





Algorithm 1 Proposed Method

input: Data streams D, Model $\{\mathcal{F}_\theta, \mathcal{G}_\varphi\}$, Factors λ and γ, Projector $\mathcal{P}_{\phi_t}$

for all phases $t \in \{0, 1, .., T\}$ **do**
  Get training set D_t
  for minibatch $B = \{x^i, y^i\}_{i=1}^n \in \mathcal{D}_t$ **do**
    if $t = 0$ **then**
      Compute $\mathcal{L}_t = \mathcal{L}_{ce} + \gamma * \mathcal{L}_{PES}$
      Update model $\{\mathcal{F}_{\theta_t}, \mathcal{G}_{\varphi_t}\}$
    else
      Get prototypes set $\boldsymbol{P}_{0:t-1}$
      Compute $\mathcal{L}_t = \mathcal{L}_{ce} + \alpha_1 \mathcal{L}_{kd} + \alpha_2 \mathcal{L}_{pro} + \alpha_3 \mathcal{L}_{PGRU}$
      Update model $\{\mathcal{F}_{\theta_t}, \mathcal{G}_{\varphi_t}\}$ and projector $\mathcal{P}_{\phi_t}$
    end if
  end for
  Compute $\boldsymbol{p}^c = \mathop{\mathbb{E}}_{(x,y)\sim D_t} [\mathcal{F}_{\theta_t}(x) \mid y = c]$
  Update prototypes set $\boldsymbol{P}_{0:t-1}$
end for
return Model $\{\mathcal{F}_{\theta_t}, \mathcal{G}_{\varphi_t}\}$

Experiments

Table 1: Quantitative comparisons of the average incremental accuracy (%) with other methods on CIFAR-100, TinyImageNet and ImageNet-Subset. *P* represents the number of incremental phases. The best performance is shown in **bold**, and the sub-optimal performance is underlined. The relative improvement compared to the SOTA NECIL methods is shown in red.

Methods	CIFAR-100			TinyImageNet			ImageNet-Subset		
	P=5	*P*=10	*P*=20	*P*=5	*P*=10	*P*=20	*P*=5	*P*=10	*P*=20
Fine-tuning	23.15	12.96	7.93	18.64	10.68	5.75	23.43	13.12	7.96
Joint	76.72	76.72	76.72	63.08	63.08	63.08	78.94	78.94	78.94
EWC [23]	24.48	21.20	15.89	18.80	15.77	12.39	—	20.40	—
LwF_MC [67]	45.93	27.43	20.07	29.12	23.10	17.43	—	31.18	—
MUC [68]	49.42	30.19	21.27	32.58	26.61	21.95	—	35.07	—
SDC [8]	56.77	57.00	58.90	—	—	—	—	61.12	—
PASS [6]	63.47	61.84	58.09	49.55	47.29	42.07	64.40	61.80	51.29
SSRE [7]	65.88	65.04	61.70	50.39	48.93	48.17	—	67.69	—
SOPE [11]	66.64	65.84	61.83	53.69	52.88	51.94	—	69.22	—
POLO [51]	68.95	68.02	65.71	54.90	53.38	49.93	70.81	69.11	—
PRAKA [10]	70.02	68.86	65.86	53.32	52.61	49.83	69.81	68.98	63.95
NAPA [52]	70.44	69.04	67.42	52.77	51.78	49.51	69.15	68.83	63.09
PRL (Ours)	**71.26**	**70.17**	**68.44**	**58.12**	**57.24**	**54.51**	**72.85**	**71.54**	**66.88**
Improvement	+0.82	+1.13	+1.02	+3.22	+3.86	+2.57	+2.04	+2.32	+2.93

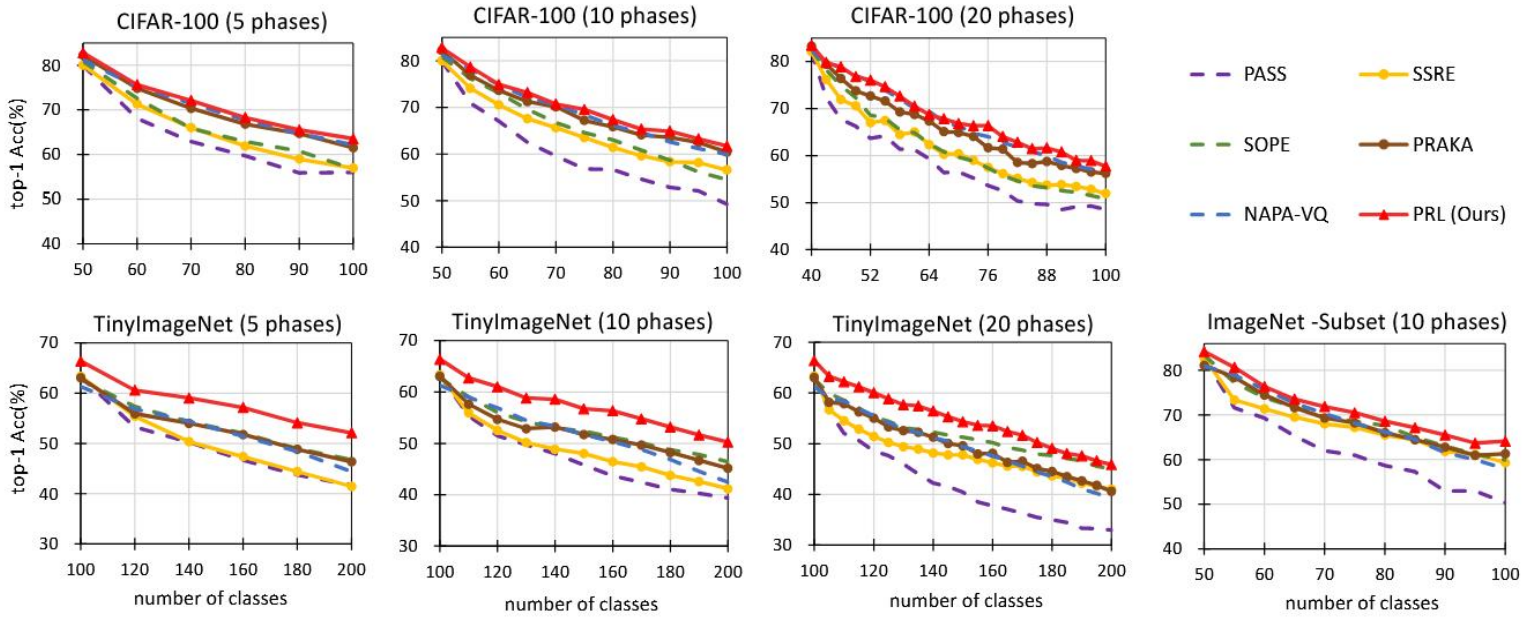


Figure 3: Detailed accuracy curves showing the top-1 accuracy of each incremental phase on CIFAR-100, TinyImageNet and ImageNet-Subset.

Experiments

Table 2: Ablation study (in average incremental accuracy) of our method on CIFAR-100 and TinyImageNet datasets.

Methods	CIFAR-100			TinyImageNet		
	P=5	*P*=10	*P*=20	*P*=5	*P*=10	*P*=20
baseline	69.25	68.52	65.93	55.04	54.15	51.65
baseline w/ PES	70.57	69.64	67.58	57.08	55.84	53.58
baseline w/ PGRU	70.36	69.23	67.17	56.79	56.05	53.16
PRL	**71.26**	**70.17**	**68.44**	**58.12**	**57.24**	**54.51**

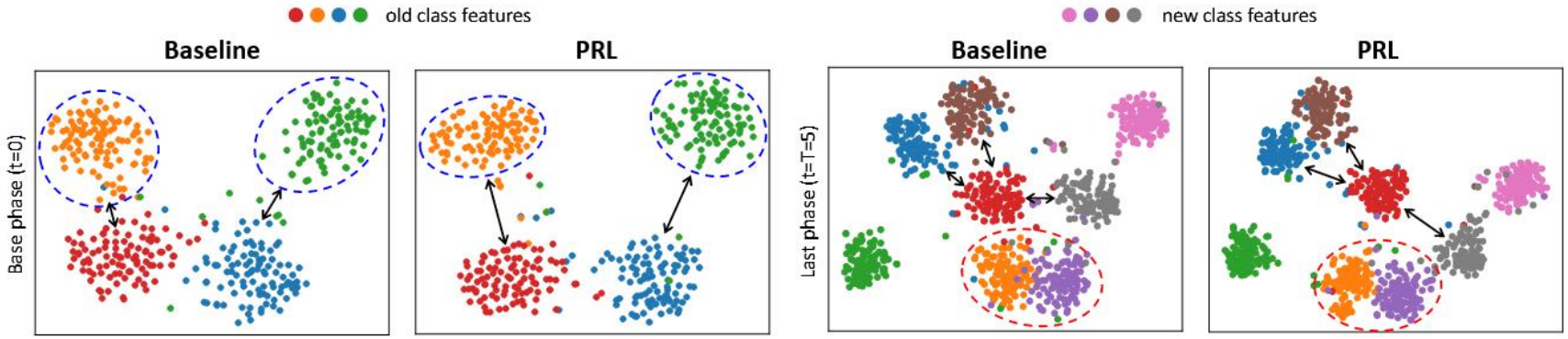


Figure 4: Visualization of the impact of PRL on the feature representations. Dashed circles and arrows highlight observable differences between baseline and PRL. PRL visually concentrates the distribution of features within classes, disperses the distribution of features between classes, and mitigates inter-class confusion.