International Journal of Computer Vision
https://doi.org/10.1007/s11263-024-02138-z

Incremental Model Enhancement via Memory-based Contrastive Learning

Shiyu Xuan[1] · **Ming Yang**[2] · **Shiliang Zhang**[1,3]

Received: 6 July 2023 / Accepted: 31 May 2024

Abstract
Training data of many vision tasks may be sequentially arrived in practice, e.g., the vision tasks in autonomous driving or video surveillance applications. This raises a fundamental challenge that, how to keep improving the performance on a specific task by learning from sequentially available training splits. This paper investigates this task as Incremental Model Enhancement (IME). IME is distinct from the conventional Incremental Learning (IL), where each training split typically corresponds to a set of independent classes, domains, or tasks. In IME, each training split may only cover part of the entire data distribution for the target vision task. Consequently, the IME model should be optimized towards the joint distribution of all available training splits, instead of optimizing towards each newly arrived one like IL methods. To deal with above issues, our method stores feature vectors of previously observed training data in the memory bank, which preserves compressed knowledge of the previous training data. We hence adopt the memorized features and each newly arrived training split for training via Memory-based Contrastive Learning (MCL). A new Contrastive Relation Preserving (CRP) scheme updates the memory bank to prevent obsoleteness of the preserved features and works with MCL simultaneously to boost the model performance. Experiments on several large-scale image classification benchmarks demonstrate the effectiveness of our method. Our method also works well on semantic segmentation, showing strong generalization ability on diverse vision tasks.

Keywords Incremental learning · Contrastive loss · Memory bank · Knowledge distillation

1 Introduction

Training data of many vision tasks may not be acquired at once. Instead, they may be collected gradually and become available sequentially in practice. For instance, new annotations can be iteratively collected by Active Learning models (Kovashka et al., 2016; Vijayanarasimhan and Grauman, 2011). More importantly, most of vision tasks in serious real-word applications, such as autonomous driving or video surveillance, require a humongous amount of training data collected in diverse geo-locations and weather conditions. These data may be collected in years and available for training over time. This raises a fundamental challenge that how to improve the performance on a vision task by learning from sequentially arrived training data. A straightforward solution is training from scratch repeatedly, i.e., combining all data and training the model whenever the new training data is available. However, this strategy suffers from the high computation and storage complexity. Relying on all existing raw data also leads to other limitations, e.g., some previous data might be not accessible due to privacy protection. Efficient algorithms that can gradually improve the performance with the new incoming training data are highly preferred. We formalize this task as Incremental Model Enhancement (IME).

The setup of IME is quite common in real applications but has not been fully explored. Conventional Incremental Learning (IL), also known as lifelong learning or continual learning, includes class incremental learning (CIL) (Rebuffi

Communicated by Nicu Sebe.

✉ Shiliang Zhang
slzhang.jdl@pku.edu.cn

Shiyu Xuan
shiyu_xuan@stu.pku.edu.cn

Ming Yang
m-yang4@u.northwestern.edu

[1] State Key Laboratory of Multimedia Information Processing, School of Computer Science, Peking University, Beijing 100871, China

[2] Multimodality Cognition, Ant Group, Seattle, WA, USA

[3] Peng Cheng Laboratory, Shenzhen 518055, China

Published online: 20 July 2024

et al., 2017; Yu et al., 2020; Tao et al., 2020; Buzzega et al., 2020; Belouadah and Popescu, 2019), domain incremental learning (DIL) (Pu et al., 2021; Bobu et al., 2018; Mancini et al., 2019), and task incremental learning (TIL) (Dhar et al., 2019; Li and Hoiem, 2017; Lopez-Paz and Ranzato, 2017). Those IL methods aim to continuously learn new classes, domains, or tasks without forgetting the knowledge of previously learned classes, domains, or tasks. As shown in Fig. 1a, each newly arrived data of IL corresponds to a set of new classes, domains or tasks, with its own training and testing sets. The training and testing sets have the identical label space and similar feature distributions. Therefore, each training split can be treated as an independent task (Kim et al., 2023). An unbiased model for these new classes, domains or tasks can be trained directly with the corresponding training set. Based on the theory proposed in Kim et al. (2023), an IL model can be learned by minimizing the loss on the new training set, meanwhile avoiding forgetting the previously learned tasks.

As illustrated in Fig. 1b, the training data for a given vision task can be available gradually in IME. Each training split of IME shares the same label space of the target task, but may just cover part of the data distribution for the target vision task because of its limited data scale. As more training splits become available, the distribution of the combined training set can be better aligned with the corresponding vision task as shown in Fig. 1b. As training splits are related to each other, IME needs to consider the joint distribution of the entire training set. The challenge of IME lies in how to learn from each biased new training split without accessing all previous training data.

To seek an efficient solution to IME, this paper stores features of previous training samples in a memory bank, and incrementally updates the model with new training *data* and memorized *features*. In other words, we introduce a mem-

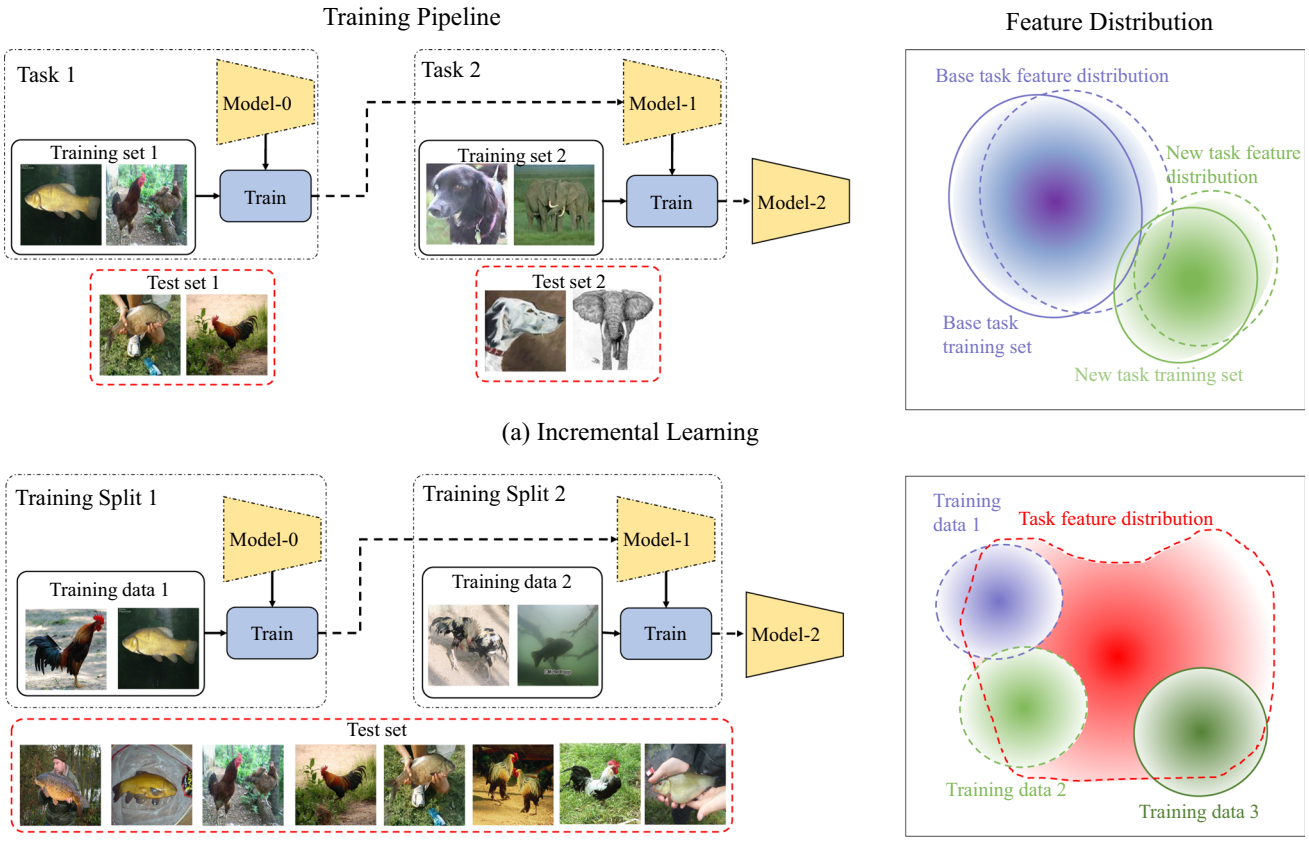

Fig. 1 Illustration of the task setups and training pipelines of Incremental Learning (IL) and Incremental Model Enhancement (IME), respectively, using animal recognition as an example. Both IL and IME are trained with sequentially arrived data. After learning to recognize "fish" and "chicken", IL aims to optimize on a new task, e.g., recognizing "dog" and "elephant". Differently, IME aims to gradually improve the performance on a specific vision task, e.g., recognizing "chicken" and "fish". Each training split of IME shares the same label space but may only covers part of the task feature distribution, making either direct fine-tuning or existing IL methods cannot obtain satisfying results (best viewed in color pdf)

ory bank to approximate the optimization on the combined training set. Different from most IL methods (Belouadah and Popescu, 2019; Buzzega et al., 2020; Rebuffi et al., 2017; Prabhu and Torr, 2020; Bang et al., 2021) that store raw samples in the memory bank, our method stores sample features to achieve better memory efficiency. Note that, features in the memory bank cannot propagate gradients to the backbone due to the missing of original input images. They thus cannot be directly used to update the model. Moreover, after the model is updated, there is a mismatch between memorized features extracted by the previous model and those freshly extracted by the updated model, leading to the obsoleteness issue of the features in the memory bank. Since the raw images of the previous training data are not stored, it is impossible to re-extract those features using the updated model. Dedicated approaches are required to optimize the backbone based on the memory bank and to address the memory obsoleteness issue.

We propose a memory-based contrastive learning framework, where a new Memory-based Contrastive Learning (MCL) optimizes the model with a memory bank. Meanwhile, we employ Contrastive Relation Preserving (CRP) to deal with the feature obsoleteness issue in the memory bank. MCL spots samples with the same label in both new training split and previous training splits, then computes contrastive loss on them, i.e., to pull samples with the same label together and push apart those with different labels. As images in previous training set are not available, MCL adopts feature vectors stored in the memory bank as their representations. MCL works with memory bank to approximate the optimization on the set combining previous and new training data.

CRP is proposed to mitigate the feature obsoleteness issue in the memory bank with two intuitions, i.e., regularizing the model update to constrain the mismatch of memorized features against the updated model, and updating the memorized features accordingly, on the fly. The regularization is achieved by preserving the similarity relationship between training samples and memorized features in the previous training stage. A feature adaptation method is proposed to estimate the changes of memorized features, which facilitates updating the obsolete features.

We conduct experiments on TinyImageNet (Stanford, 2015), miniImageNet (Vinyals et al., 2016), ImageNet1K (Russakovsky et al., 2015), and CUB (Wah et al., 2011). Experimental results show that existing IL algorithms do not work well on this IME setup. For instance, the recent RM (Bang et al., 2021) achieves marginal improvements over the directly fine-tuning baseline, e.g., it even decreases the baseline accuracy from 55.49 to 55.12% on ImageNet1K. Our method achieves the best performance among competitors and improves the baseline accuracy from 55.49 to 61.05% on ImageNet1K. Our method also shows promising performance on semantic segmentation task on Cityscapes (Cordts et al., 2016). It consistently outperforms the fine-tuning baseline and recent methods like LwF (Li and Hoiem, 2017) and DER (Buzzega et al., 2020).

IME is practical and meaningful for many open-world vision applications due to its capability of continuously boosting the model performance with sequentially arrived training data. As an under-explored task, it differs from existing IL in task setup and optimization objective as illustrated in Fig. 1. To the best of our knowledge, this is an original work clarifying the formulation of IME and defining its testing protocols. The proposed method enjoys superior performance, low computational complexity, and the advantage of privacy protection because it does not access previous raw training samples. It has the potential to benefit CV tasks that obtain training samples sequentially. Code of our method will be released.

2 Related Work

This work is closely related to Incremental Learning and Contrastive Learning. This section briefly reviews those two lines of research and discusses our differences with them.

2.1 Incremental Learning

Existing IL algorithms can be summarized into four categories, i.e., regularization-based, knowledge distillation-based, parameter isolation-based, and memory-based methods, respectively.

Regularization-based methods employ certain regularization to restrict the update of model parameters, hyper-parameters or the direction of gradients (Kirkpatrick et al., 2017; Lee et al., 2017; Farajtabar et al., 2020; Saha et al., 2021; Wang et al., 2021). EWC (Kirkpatrick et al., 2017) adds an extra quadratic penalty term into the loss function to penalize the change of weights that are important for the previously learned tasks. OGD (Farajtabar et al., 2020), GPM (Saha et al., 2021), and Adam-NSCL (Wang et al., 2021) project the gradient steps in the orthogonal direction to the gradient subspaces that are deemed critical for the previous tasks.

Knowledge distillation-based methods integrate knowledge distillation (Hinton et al., 2015) to preserve the knowledge of previously learned tasks (Li and Hoiem, 2017; Buzzega et al., 2020). LwF (Li and Hoiem, 2017) prevents from forgetting the learned knowledge using outputs of previous model to distill the updated model. Inspired by the LwF, PODNet (Douillard et al., 2020) distills not only final outputs but also the outputs of intermediate layers through Pooled Outputs Distillation.

Parameter isolation-based methods allocate different model parameters or parts to different tasks (Mallya and Lazebnik, 2018; Yan et al., 2021). PackNet (Mallya and Lazebnik,

2018) leverages network pruning to add multiple tasks to a single deep neural network. DEN (Yoon et al., 2017) proposes a dynamically expandable network, that can dynamically expand network capacity. Those methods commonly train large networks or require task-ID during inference, thus present limited flexibility and cannot deal with class or domain incremental learning. Additional treatments like network pruning (He et al., 2017) could be leveraged to speed up the inference.

Memory-based methods use a fixed-size buffer to record cues of previous samples to avoid catastrophic forgetting (Belouadah and Popescu, 2019; Buzzega et al., 2020; Rebuffi et al., 2017; Prabhu and Torr, 2020; Bang et al., 2021). These methods can also be called as rehearsal-based methods. Most of those methods directly preserve a subset of training data. Many methods are proposed to select the most representative training data. iCaRL (Rebuffi et al., 2017) selects samples closest to the mean feature of each class. GSS (Aljundi et al., 2019) proposes a gradient based sample selection method to select examples in the memory buffer. RM (Bang et al., 2021) increases the diversity of samples in the memory buffer by considering per-sample classification uncertainty and data augmentation. Another line of works focuses on how to train the model with the preserved training samples. RM (Bang et al., 2021) involves previous samples and current data in training batches. iCaRL (Rebuffi et al., 2017) and DER (Buzzega et al., 2020) mix rehearsal with knowledge distillation and regularization.

Although memory-based methods achieve superior performance, directly preserving previous training samples breaks the rule of privacy protection and may have a prohibitive memory requirement in practice. One possible solution is to preserve features of the previous training samples in the memory bank. However, the preserved features cannot directly provide gradient to update the backbone and suffer the obsoleteness issue. Those issues have not been well addressed in previous methods, e.g., MEIL (Iscen et al., 2020) only uses memorized features to train the classifier layer, and does not adopt them to update other layers of the backbone. SDC (Yu et al., 2020) proposes a feature adaptation method to update the prototype of each category after the training of each task. This method only considers the semantic drift of the prototypes. The preserved features are not used to train the model. Moreover, the feature adaptation method cannot be performed on-the-fly after each training iteration. This issue limits its applicability in addressing the obsoleteness issue of the preserved features when using them to supervise the model training. REMIND (Hayes et al., 2020) extracts the preserved features using a backbone with several frozen layers. These layers are frozen to prevent the obsoleteness issue of the preserved features. However, this design limits the ability to update the entire model, which in turn restricts the performance in IME task.

2.2 Contrastive Learning

Contrastive Learning is commonly used to learn an embedding space. With the powerful contrastive loss (Hadsell et al., 2006), self-supervised representation learning has achieved remarkable progress (He et al., 2020; Chen et al., 2020a; Chen and He, 2021). Those methods follow a similar intuition to pull together an anchor and its data augmentation in the embedding space, and push apart it from negative samples. Some methods focus on how to involve more negative samples into the training, e.g., MoCo (He et al., 2020) introduces a memory queue to store more negative samples. SimCLR (Chen et al., 2020a) directly leverages a large training batch. The construction of the positive and negative samples is also critical for contrastive learning. VFT (Zhu et al., 2021) proposes Positive Extrapolation and Negative Interpolation to generate more difficult positive and negative samples, respectively. MoCHi (Kalantidis et al., 2020) proposes hard negative mixing at the feature level to introduce harder negative samples into the learning. Besides self-supervised learning, SEED (Fang et al., 2021) integrates contrastive learning into knowledge distillation. SupCon (Khosla et al., 2020) extends contrastive learning into the fully-supervised setup.

Due to its powerful capability in learning representations, contrastive loss is also used in many other tasks. PCL (Yao et al., 2022) uses contrastive loss to mine rich semantic relationship from different domains and learn domain-invariant features for the domain generalization task. Based on the contrastive loss, KCL (Kang et al., 2020) designs balanced k-positive sampling strategy to prevent from learning the biased model from the long-tailed training data.

2.3 Differences with Other Methods

This paper studies a new IME setup that aims to use sequentially arrived data to enhance the performance on a specific vision task. It works with a small training split at each training step, and has a different training objective compared against IL. As shown in Fig. 1, each task can be treated as an independent problem in IL (Kim et al., 2023). As an unbiased model for new classes, domains or tasks can be trained by optimizing towards each new training set. IL can be achieved by learning new task while avoiding forgetting the previously learned tasks. Differently, the joint distribution of a combined training set should be adopted for the optimization of IME model. Experiments in Sect. 5 show that IL methods do not perform well in this IME setting.

Similar to IME, the label space in DIL is also the same. Each newly arrived domain of DIL has its own training and testing sets, which follow a similar feature distribution. Therefore, an unbiased model to each new domain can be trained by optimizing towards the newly arrived training

data. Most of IL tasks follow the above assumption, making many CIL methods can deal with DIL (Buzzega et al., 2020; Li and Hoiem, 2017). Besides the standard DIL setup, some other settings have been proposed. General Incremental Learning (Xie et al., 2022) considers the case where both the class distribution and class-specific domain distribution change over time. CoTTA (Wang et al., 2022b) continually adapts the model to a target domain with several unlabeled data during inference time. The domain of each newly arrived data in these settings remains independent with others. The difference between IME and DIL lies in that, each training split of IME is a subset of the training set, and shows biased feature distribution *w.r.t.* the testing set. The IME model thus should be trained by optimizing towards the joint distribution of all available training splits.

In online learning (Hoi et al., 2021), training samples are arrived sequentially and each training sample can only be used once. With only one training sample at each training iteration, existing optimization methods, e.g., gradient descending, suffer from convergence problems. Recent online learning methods mainly focus on the effective model optimization in an online manner, thus face different challenges with IME. Online incremental learning trains the model with an online stream of data and tasks, where tasks may include new classes (class incremental setting) or new domains (domain incremental setting). It presents a similar challenge of IL. Most IL methods can also deal with online incremental learning, e.g., LwF, DER.

Our method and IL algorithms also present different ways of using the memory bank. IL methods often preserve the raw training data in the memory bank, and use those samples to perform rehearsal to prevent forgetting learned knowledge in previous tasks. Our method preserves features in the memory bank, thus presents better efficiency and alleviates the limitation of accessing the previous training data.

The proposed MCL works with memory bank and effectively tackles the issue that features cannot provide gradients to update the backbone. Some methods like MoCo (He et al., 2020) also use memory bank to perform the contrastive learning. The aim of their memory bank is to integrate more negative samples into training. Different from them, we introduce the memory bank to approximate the optimization on the combined training set. A new CRP is further proposed to address the feature obsoleteness issue. Extensive experiments also demonstrate the superior performance of our method over previous works.

3 Overview

We denote the entire training set $\mathcal{D}$ as T splits that arrive sequentially, i.e., $\mathcal{D} = \left\{D^t\right\}_{t=1}^{T}$. The t-th training split is $D^t = \{x_i, y_i\}_{i=1}^{n}$, where x_i, y_i denote the i-th image and its label. Let $\mathcal{T}$ denotes the test set that shares the same label space and similar data distribution with $\mathcal{D}$. IME aims to improve the performance on $\mathcal{T}$ through sequentially training on new splits from D^1 to D^T. When being trained on D^t, IME assumes that previous training splits $\{D^i\}_{i<t}$ are no longer accessible.

Each D^t can be regarded as a sampled subset of $\mathcal{D}$, and it may only cover part of the data distribution for the target vision task as illustrated in Fig. 1b. To improve the model performance, the model should be trained on all available training splits $\sum_{i=1}^{t} D^i$. The training objective at the t-th training split can be conceptually denoted as,

$$\theta^{t*}, \omega^{t*} = \arg\min_{\theta^t, \omega^t} \mathcal{L}\left(D^t; \sum_{i=1}^{t-1} D^i; \theta^t; \omega^t\right), \tag{1}$$

where $\mathcal{L}$ is the loss function, θ^t and ω^t are parameters learned at the t-th training split. θ^t denotes parameters of feature extraction backbone $\mathrm{h}(\theta, \cdot)$ and ω^t are parameters of the classifier $\mathrm{g}(\omega, \cdot)$. The model maps an image x to K classification scores,

$$\mathbf{f} = \mathrm{h}(\theta, x), \mathrm{g}(\omega, \mathbf{f}) \rightarrow \mathbb{R}^K, \tag{2}$$

where $\mathbf{f}$ is the feature of x.

Equation (1) is supposed to train on the entire dataset. An approximation can be preserving old samples in the memory bank like IL methods Buzzega et al. (2020), and training on the memory bank and each new training split. To pursue a more efficient approximation, we introduce a memory bank $\mathcal{M}$ to preserve feature vectors and labels of previously observed training samples. $\mathcal{M}$ is constructed by randomly selecting $\rho\%$ samples of each training split, then adding their features and labels to $\mathcal{M}$. The parameter ρ controls the size of $\mathcal{M}$. Using $\mathcal{M}$ as an alternative to $\sum_{i=1}^{t-1} D^i$, the optimization of Eq. (1) can be denoted as,

$$\theta^{t*}, \omega^{t*} = \arg\min_{\theta^t, \omega^t} \left[\mathcal{L}(D^t; \mathcal{M}; \theta^t; \omega^t) + \mathrm{diff}\left(\mathcal{M}, \sum_{i=1}^{t-1} D^i\right)\right], \tag{3}$$

where $\mathrm{diff}(\mathcal{M}, \sum_{i=1}^{t-1} D^i)$ regularizes the difference between memorized features in $\mathcal{M}$ and freshly extracted features from $\sum_{i=1}^{t-1} D^i$. $\mathrm{diff}(\cdot)$ is minimized to ensure the features in $\mathcal{M}$ are up-to-date.

Equation (3) can be optimized by decreasing the training loss on D^t and $\mathcal{M}$, as well as maintaining an updated $\mathcal{M}$. We propose Memory-based Contrastive Learning (MCL) to pull samples in the same category together, and push samples from different categories apart. MCL computes a constructive loss $\mathcal{L}_{MCL}$ on D^t and $\mathcal{M}$ to approximate the optimization

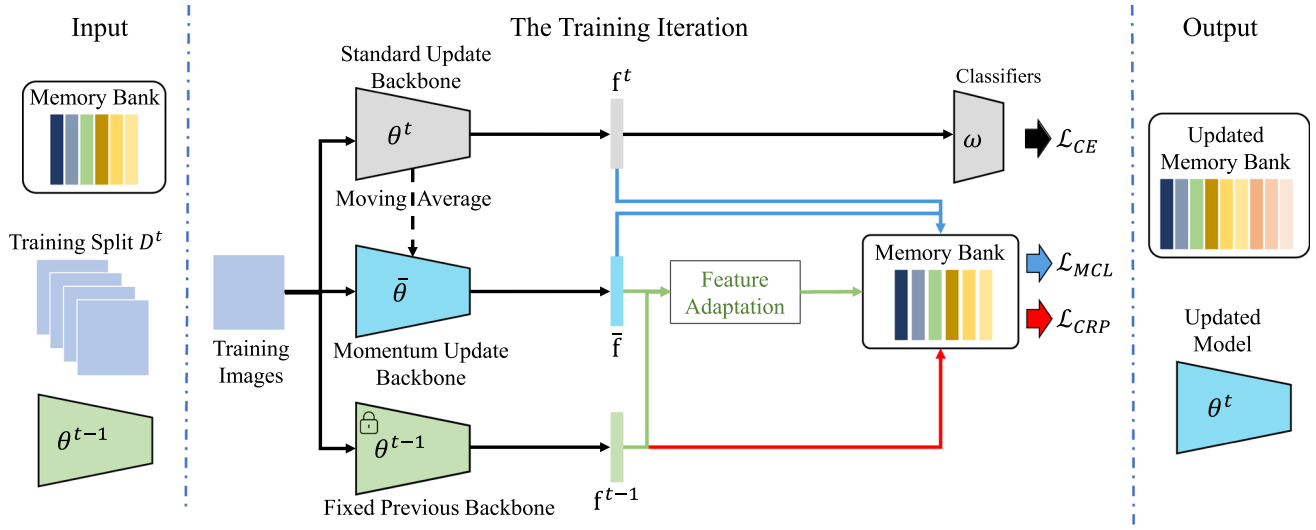

Fig. 2 Overview of the proposed IME framework. The training step starts as a new training split D^t arrives, and finally outputs the updated memory bank and model parameters θ^t. At each training iteration, training images are fed into the standard update backbone, momentum update backbone and fixed backbone from previous iteration to generate features $\mathbf{f}^t$, $\bar{\mathbf{f}}$, and $\mathbf{f}^{t-1}$, respectively. Together with features in the memory bank, $\mathbf{f}^t$ and $\mathbf{f}^{t-1}$ are used to calculate $\mathcal{L}_{CRP}$. $\mathbf{f}^t$ and $\bar{\mathbf{f}}$ are used to calculate $\mathcal{L}_{MCL}$. The cross entropy loss $\mathcal{L}_{CE}$ is computed with feature $\mathbf{f}^t$ in Eq. (7). Details of $\mathcal{L}_{MCL}$ and $\mathcal{L}_{CRP}$ are presented in Sects. 4.2 and 4.1, respectively

on $\sum_{i=1}^{t} D^i$. It is defined as,

$$\mathcal{L}_{MCL} = \sum_{x_i \in D^t} \text{loss}(x_i, y_i, \mathcal{M}). \tag{4}$$

$\mathcal{M}$ stores the features extracted by previous models. As the model is being updated on the new training splits, the features in $\mathcal{M}$ have a mismatch with the current model. In another word, $\mathcal{M}$ becomes outdated over time, which is harmful to $\mathcal{L}_{MCL}$ computation. We propose the Contrastive Relation Preserving (CRP) to decrease $\text{diff}(\cdot)$ through estimating the changes of memorized features, and regularizing the model update, respectively. The similarity relationship between training images and $\mathcal{M}$ represents the structure of the learned embedding space. We preserve this relationship during the model training to constrain the mismatch of memorized features against the updated model.

For training split D^t, we use $\psi(\mathbf{f}_i^t, \mathcal{M})$ to compute the relationship between a feature vector $\mathbf{f}_i^t$ of image x_i extracted with parameters θ^t and features in $\mathcal{M}$. CRP maintains a similarity relationship between two adjacent training splits. A regularization loss $\mathcal{L}_{CRP}$ can be represented as,

$$\mathcal{L}_{CRP} = \sum_{i=1}^{n} \text{dis}(\psi(\mathbf{f}_i^t, \mathcal{M}), \psi(\mathbf{f}_i^{t-1}, \mathcal{M})), \tag{5}$$

where n is the size of current training split, $\mathbf{f}_i^{t-1}$ is the feature vector extracted with parameters θ^{t-1} trained at the previous training split, i.e., $\mathbf{f}_i^{t-1} = \text{h}(\theta^{t-1}, x_i)$, and $\text{dis}(\cdot)$ measures the differences between two relationships.

Our overall training loss for the t-th training split consists of $\mathcal{L}_{CRP}$, $\mathcal{L}_{MCL}$, as well as the cross entropy loss $\mathcal{L}_{CE}$ computed on D^t, respectively. It is summarized as,

$$\mathcal{L}_{overall} = \mathcal{L}_{CE} + \lambda_1 \mathcal{L}_{CRP} + \lambda_2 \mathcal{L}_{MCL}, \tag{6}$$

where λ_1 and λ_2 denote the loss weights. $\mathcal{L}_{CE}$ is computed at each training split D^t by comparing ground-truth labels and the labels predicted by classifiers ω^t with features $\mathbf{f}^t$. We also classify $\mathbf{f}^t$ using the classifiers learned in previous training splits, and compute cross entropy loss on those labels. As discussed in a previous work (Jung et al., 2018), this helps to maintain the previously learned decision boundaries and preserve the prototypes of corresponding classes in the embedding space. The $\mathcal{L}_{CE}$ can be denoted as,

$$\mathcal{L}_{CE} = \sum_{i}^{n} \left[\ell(\text{g}(\omega^t, \mathbf{f}_i^t), y_i) + \frac{1}{t-1} \sum_{j=1}^{t-1} \ell(\text{g}(\omega^j, \mathbf{f}_i^t), y_i) \right], \tag{7}$$

where $\mathbf{f}_i^t = \text{h}(\theta^t, x_i)$ and $\ell(\cdot)$ computes the cross-entropy loss. ω^j denotes classifier parameters learned in the j-th training split. Note that, the first and the second items of Eq. (7) both use standard cross-entropy loss. The difference of them lies in that the second item uses the logits predicted by classifiers learned at previous training splits. Equation (7) updates θ^t and ω^t. $\omega^{j=1:t-1}$ is used as a fixed parameter. As shown in Sect. 5.4, enforcing the loss computed with ω^j boosts the performance.

Figure 2 illustrates the framework of our method, where each training step outputs an updated model and an updated

memory bank. The following parts proceed to present the details of CRP and MCL.

4 Proposed Method

4.1 Contrastive Relation Preserving

CPR incorporates Feature Adaptation (FA) and $\mathcal{L}_{CRP}$, which work jointly from different aspects to handle the feature obsoleteness issue. $\mathcal{L}_{CRP}$ regularizes the model change to constrain the mismatch between memorized features against those freshly extracted ones. The FA method is used to estimate the differences between memorized features and freshly extracted ones. It hence uses those estimated differences to update obsolete features, making them get similar to freshly extracted ones.

Feature adaptation in CRP updates features in $\mathcal{M}$. As the images of previous training splits are not accessible, it is not feasible to extract their features with the updated model. We hence denote the change of the k-th feature $\mathbf{m}_k \in \mathcal{M}$ as $\Delta(\mathbf{m}_k)$ and compute the updated feature as $\hat{\mathbf{m}}_k = \mathbf{m}_k + \Delta(\mathbf{m}_k)$. We use the changes of sample features $\Delta(\mathbf{f}_i)$ in the current training split to estimate $\Delta(\mathbf{m}_k)$, if $\mathbf{f}_i$ and $\mathbf{m}_i$ are similar with each other. With $x_i \in D^t$, $\Delta(\mathbf{f}_i)$ can be computed by extracting its features with new and previous models. $\Delta(\mathbf{m}_k)$ is estimated by the weighted sum of $\Delta(\mathbf{f}_i)$, i.e.,

$$\Delta(\mathbf{m}_k) = \sum_{i=1}^{n} w_{ik} \Delta(\mathbf{f}_i), \tag{8}$$

where the w_{ik} is the weight. Because $\mathbf{m}_k$ is updated on the training split D^{t-1}, we use the similarity between $\mathbf{m}_k$ and $\mathbf{f}_i^{t-1}$ to calculate w_{ik}, i.e.,

$$\begin{aligned} w_{ik} &= \frac{\mathbb{1}(\bar{y}_k = y_i)\, \mathrm{es}(\mathbf{m}_k, \mathbf{f}_i^{t-1})}{\sum_{j=1}^{n} \mathbb{1}(\bar{y}_k = y_j)\, \mathrm{es}(\mathbf{m}_k, \mathbf{f}_j^{t-1})}, \\ \mathrm{es}(\mathbf{m}_k, \mathbf{f}_i^{t-1}) &= \exp(\mathrm{sim}(\mathbf{m}_k, \mathbf{f}_i^{t-1})/\tau), \end{aligned} \tag{9}$$

where $\mathbf{f}_i^{t-1}$ is extracted by the model trained on D^{t-1}, $\bar{y}_k$ and y_i denote the labels of $\mathbf{m}_k$ and x_i, respectively. $\mathbb{1}(\cdot)$ is the indicator function, $\mathrm{sim}(\cdot,\cdot)$ computes the cosine similarity, and τ is a predefined temperature parameter. w_{ik} can be precalculated at the beginning of each training split for better efficiency.

$\Delta(\mathbf{f}_i)$ can be efficiently computed by comparing $\mathbf{f}_i^t$ with $\mathbf{f}_i^{t-1}$, if x_i is in the current training batch. For the images outside the training batch, repetitively extracting their features after each training iteration is time consuming. Simply ignoring them leads to inconsistency of $\Delta(\mathbf{f}_i)$ because the model is continuously updated. We leverage momentum update to slow down the model change to mitigate the inconsistency of $\Delta(\mathbf{f}_i)$, i.e.,

$$\begin{aligned} \bar{\theta} &= \alpha\bar{\theta} + (1-\alpha)\theta^t, \\ \bar{\omega} &= \alpha\bar{\omega} + (1-\alpha)\omega^t, \end{aligned} \tag{10}$$

where $\bar{\theta}$ and $\bar{\omega}$ are parameters of the momentum update model, θ^t and ω^t are parameters updated at each training iteration. $\alpha \in [0, 1)$ is the momentum coefficient, which slows down the changes to $th\bar{e}ta$ and $om\bar{e}ga$. Based on the momentum update model, only when x_i is in the training batch, $\Delta(\mathbf{f}_i)$ is updated using,

$$\Delta(\mathbf{f}_i) = \mathrm{h}(\bar{\theta}, x_i) - \mathrm{h}(\theta^{t-1}, x_i). \tag{11}$$

The regularization loss $\mathcal{L}_{CRP}$ is computed to constrain the mismatch of memorized features against the updated model on each training split. Knowledge distillation methods are widely used to constrain the model update (Romero et al., 2014; Hinton et al., 2015; Zagoruyko and Komodakis, 2016; Gou et al., 2021). The relationship between samples in each training split and features in $\mathcal{M}$ provides the structure of the learned embedding space. Therefore, the increase of $\mathrm{diff}(\cdot)$ can be regularized by preserving this relationship. Due to the limitation of GPU memory, we randomly sample z feature vectors and their labels from $\mathcal{M}$ to construct the memory training batch $\mathcal{B}$ at each iteration. The relationship between $x_i \in D^t$ and features in $\mathcal{B}$ can be represented as a similarity distribution, i.e.,

$$\psi(\mathbf{f}_i^t, \mathcal{M}) = [p_{1i}^t, p_{2i}^t, \ldots, p_{zi}^t], \tag{12}$$

where p_{ki}^t is the normalized similarity between $\mathbf{f}_i^t$ and $\hat{\mathbf{m}}_k$ in $\mathcal{B}$. It is computed as

$$p_{ki}^t = \frac{\mathrm{es}(\mathbf{f}_i^t, \hat{\mathbf{m}}_k)}{\sum_{j=1}^{z} \mathrm{es}(\mathbf{f}_i^t, \hat{\mathbf{m}}_j)}. \tag{13}$$

Similarly, we compute the relationship at the previous training split D^{t-1} as,

$$\psi(\mathbf{f}_i^{t-1}, \mathcal{M}) = [p_{1i}^{t-1}, p_{2i}^{t-1}, \ldots, p_{zi}^{t-1}], \tag{14}$$

where p_{ki}^{t-1} is computed by comparing $\mathbf{f}_i^{t-1}$ with $\hat{\mathbf{m}}_k$ in similar way of Eq. (13),

$$p_{ki}^{t-1} = \frac{\mathrm{es}(\mathbf{f}_i^{t-1}, \hat{\mathbf{m}}_k)}{\sum_{j=1}^{z} \mathrm{es}(\mathbf{f}_i^{t-1}, \hat{\mathbf{m}}_j)}. \tag{15}$$

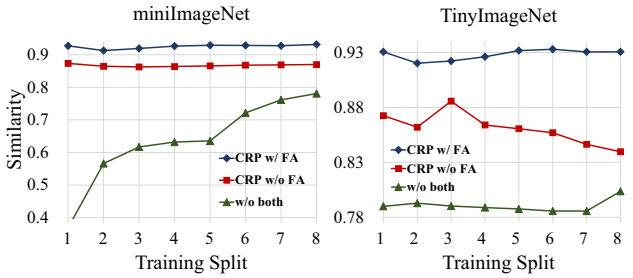

Fig. 3 Visualization of cosine similarity between memorized features and freshly extracted features on TinyImageNet and miniImageNet

The regularization loss $\mathcal{L}_{CRP}$ is hence computed as the KL-divergence between two similarity distributions, i.e.,

$$\mathcal{L}_{CRP} = \sum_{i=1}^{n}\sum_{k=1}^{z} -p_{ki}^{t-1}\log(p_{ki}^{t}). \tag{16}$$

Our regularization loss uses feature relationship to perform distillation, which is different from the previous methods that leverage soft targets (Hinton et al., 2015), feature maps (Romero et al., 2014) or attention maps (Zagoruyko and Komodakis, 2016). These methods only consider applying regularization on individual training samples. Regularization on feature relationship is more effective in leveraging useful cues encoded in $\mathcal{M}$.

Visualization of CRP is shown in Fig. 3. It illustrates the similarity between memorized features and those freshly extracted by the updated model after each training split. Note that, similarity close to 1 means memorized features are similar to those freshly extracted ones, hence shows less mismatch between two features. The figure shows that both $\mathcal{L}_{CRP}$ and Feature Adaption (FA) are effective. Combining $\mathcal{L}_{CRP}$ and FA ensures up-to-date features in the memory, hence keeps a low diff$(\cdot)$ in Eq. (3). More extensive experiments will be presented in Sect. 5.4.

4.2 Memory-based Contrastive Learning

MCL is expected to optimize the first term in Eq. (3), through pulling together feature vectors in the same category, while pushing apart vectors from different categories. Suppose the sampled memory training batch is $\mathcal{B}$. For an image and its label $\{x_i, y_i\} \in D^t$, we divide its positive and negative samples in $\mathcal{B}$ into two sets $P_i = \{\hat{\mathbf{m}}_j\}_{j=1}^{\|P_i\|}$ and $N_i = \{\hat{\mathbf{m}}_a\}_{a=1}^{\|N_i\|}$ according to the label. The MCL is conducted to optimize the distance between $\mathbf{f}_i^t$ and P_i, N_i, respectively.

Besides optimizing the distance between $\mathbf{f}_i^t$ and $\mathcal{B}$, we pull $\mathbf{f}_i^t$ and $\bar{\mathbf{f}}_i$ together to ensure the effectiveness of the momentum update. Note that, $\bar{\mathbf{f}}_i$ is extracted by the momentum update backbone, i.e., $\bar{\mathbf{f}}_i = \mathrm{h}(\bar{\theta}, x_i)$. This is equivalent to adding $\bar{\mathbf{f}}_i$ into the positive feature set, i.e., $P_i^* = [P_i, \bar{\mathbf{f}}_i]$, and computing the $\mathcal{L}_{MCL}$ on P_i^* and N_i as,

$$\mathcal{L}_{MCL} = \sum_{i=1}^{n} \frac{-1}{\|P_i^*\|} \sum_{j\in P_i^*} \log \frac{\mathrm{es}(\mathbf{f}_i^t, \hat{\mathbf{m}}_j)}{\mathrm{es}(\mathbf{f}_i^t, \hat{\mathbf{m}}_j) + \sum_{a\in N_i} \mathrm{es}(\mathbf{f}_i^t, \hat{\mathbf{m}}_a)}, \tag{17}$$

where $\|P_i^*\|$ is the cardinality of P_i^*.

Visualization to the features learned by the MCL and baseline is shown in Fig. 4a–e, where t-SNE (Maaten and Hinton, 2008) is used to visualize the feature distribution optimized by different methods in training split 1 (D^1), training split 2 (D^2), the last training split (D^9), and the test set, respectively. Different colors indicate different categories.

In Fig. 4a, the base model is trained on D^1. It learns a good representation for D^1, but shows poor discriminative power for samples in D^2, i.e., overlapped distributions among different categories. In Fig. 4b, directly fine-tuning the model on D^2 leads to a good embedding space for D^2, but degrades the discriminative ability on D^1. As shown in Fig. 4c, our method learns a good embedding space for both D^1 and D^2.

After training the model on all the training splits, our method learns a good embedding space as shown in Fig. 4e. It is clear that, the learned model preserves a good discriminative ability on the oldest training split D^1, showing the effectiveness of optimizing Eq. (3). On the contrary, although is trained on all training splits, FT baseline shows limited discriminative power on the target task. The visualization shows that our MCL and CRP can achieve the goal of incrementally boosting the performance on a fixed target vision task with sequentially arrived training data.

Our method can be flexibly applied to dense prediction tasks like semantic segmentation. We illustrate the semantic segmentation results in Fig. 5. Similar to image classification, FT baseline cannot gradually improve the result with sequentially arrived training splits, e.g., as marked in yellow rectangles, the unclear boundary and incorrect segmentation have not been corrected. Our method can improve segmentation boundaries and correct some pixels with wrong classification labels with sequentially arrived training splits, validating the effectiveness of our method on other vision tasks.

5 Experiments

5.1 Experimental Setup

Datasets We use TinyImageNet (Stanford, 2015), miniImageNet (Vinyals et al., 2016), ImageNet1K (Russakovsky et al., 2015), and CUB (Wah et al., 2011) to perform image classification, and Cityscapes (Cordts et al., 2016) to perform semantic segmentation.

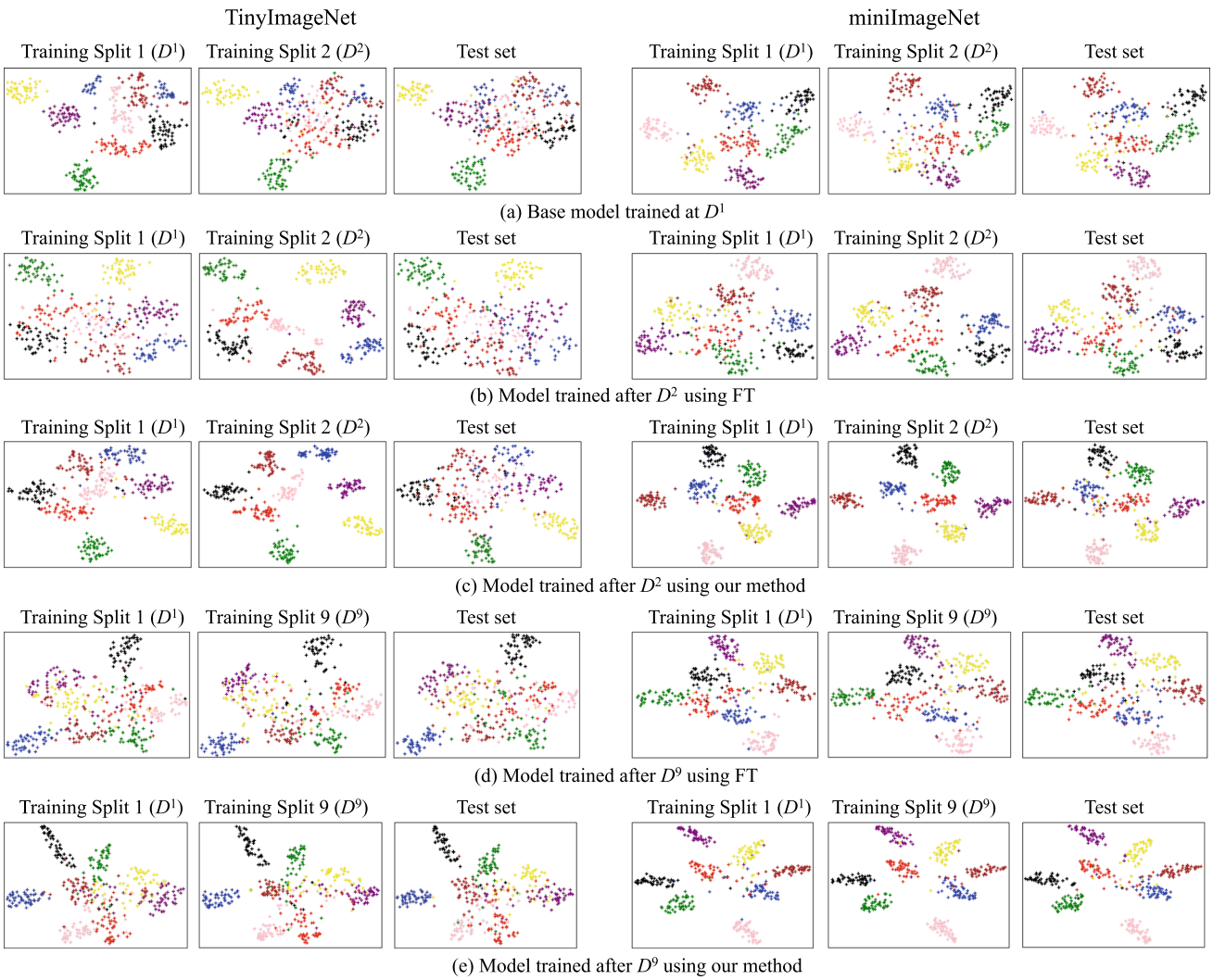

Fig. 4 T-SNE (Maaten and Hinton, 2008) visualization of learned features on randomly sampled categories from TinyImageNet and miniImageNet, respectively. "FT" denotes directly fine-tuning the previous model on the new training split

TinyImageNet contains 200 classes with 500 images per class in the training set and 100 images per class in the test set. miniImageNet contains 100 classes with 600 images in each class. We randomly select 100 images from each class to construct the test set. For ImageNet1K and CUB, we use the original training/testing split, which have 1281,167/5,994 images for training and 50,000/5,794 for testing, respectively. Cityscapes is used for urban scene understanding. We follow the original data splits which divided the 5000 high-quality pixel-level annotated images into 2975/500/1525 images for training, validation and testing.

To simulate the sequentially arrived data, we randomly divide the training set into 10 independent subsets with equal size. Each subset has the same label space. We randomly select 2 subsets to construct D^1 and select one subset each time to simulate the remaining D^2–D^9. As the training sets of CUB are small, we randomly select 4 subsets to construct D^1 and select 2 subsets each time to construct training splits D^2–D^4. The size of the first training split is larger than the other training splits to simulate the situation that the base model is first trained with a larger dataset. In addition, we further validate the influence of the base model on the performance in Sect. 5.4. Similar to CUB, for the Cityscapes dataset, we randomly select 4 subsets to construct D^1 and select 2 subsets each time to construct training splits D^2–D^4. The image number of D^1 for TinyImageNet, miniImageNet, ImageNet1K, CUB, and Cityscapes is 20,000, 10,000, 256,233, 2398 and 1190, respectively.

Evaluation Metrics Classification accuracy (Acc^t) is adopted to evaluate our methods, where the superscript t denotes the performance after the training on the t-th training split. Since the model is sequentially trained, we also report mean classification accuracy mAcc. We compute mAcc by averaging the performance after training splits D^2–D^9 (D^2–

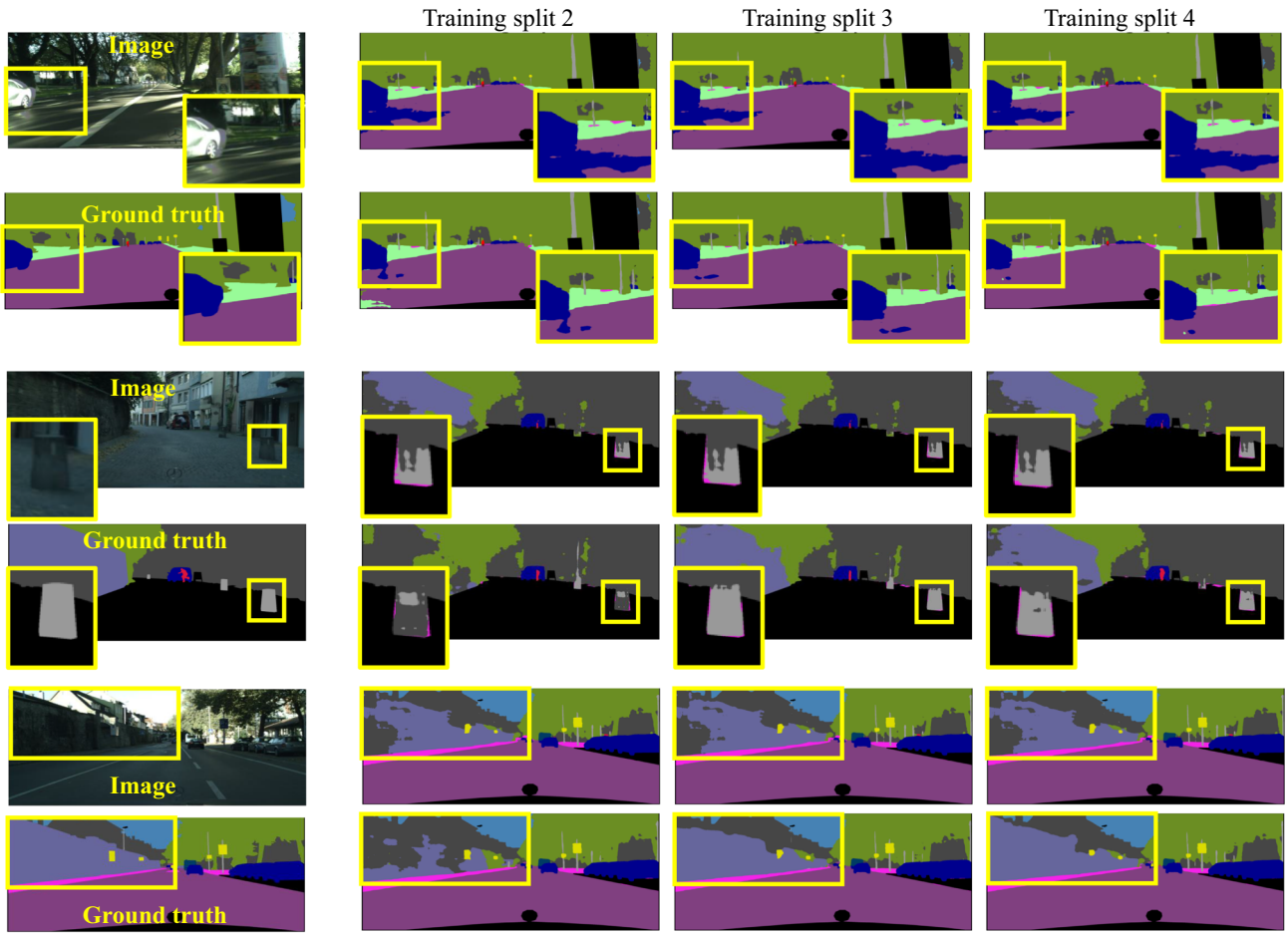

Fig. 5 Visualization of semantic segmentation results after each training split. DeepLabV3 is used as the segmentation head. In each example, the first and second row shows the result of fine-tunning baseline and our method, respectively

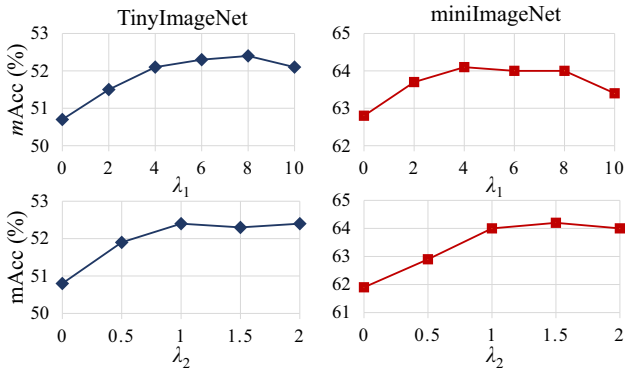

Fig. 6 Effects of the loss weights λ_1 and λ_2 in Eq. (6)

D^4 for CUB). We apply the same base model trained at D^1 to different algorithms for a fair comparison. The mIoU which measures the Intersection-over-Union (IOU) for each foreground class and averages over all the classes is used to evaluate our methods on semantic segmentation. Similar to image classification, the average mIoU on D^2-D^4 is reported.

5.2 Implementation Details

All models are implemented with PyTorch (Paszke et al., 2019). SGD is used for the optimization. ResNet-12 (Sun et al., 2019), modified ResNet-18, ResNet-18 (He et al., 2016), and ResNet-50 (He et al., 2016) are used as the backbone on miniImageNet, TinyImageNet, ImageNet1K and CUB respectively. Since the TinyImageNet has a much lower image resolution (64×64), we modify the first convolutional layer to 3×3 with stride 1 and remove the first Maxpooling layer. The training images are resized to 84×84, 64×64 on miniImageNet and TinyImageNet, and 224×244 on other datasets. Randomly cropping and flipping are used as data augmentation. We train the model for 100 epochs at each training split with a mini-batch size of 64 (expect for ImageNet1K where the mini-batch size is set as 256). Model is trained with cross-entropy loss at the first training split D^1.

The learning rate starts at 0.05 and decreases by $1/10$ after every 30 epochs. The parameter z, i.e., the number of feature vectors in the memory training batch is set as 2048. λ_1 and

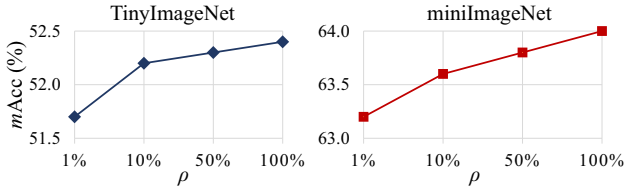

Fig. 7 Effects of the size of memory bank controlled by ρ

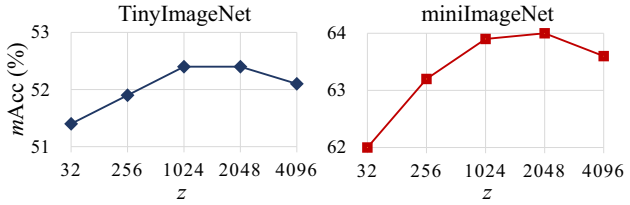

Fig. 8 Effects of the batch size z sampled from the memory bank at each training iteration

λ_2 in Eq. (6) are set as 8 and 1, respectively. The sample rate ρ ranges between 1 and 100%. The temperature parameter τ in Eq. (9) is set as 0.07. The momentum α in Eq. (10) is set as 0.9997. Those parameters will be studied in Sect. 5.4.

For the semantic segmentation, two well-known methods DeepLabV3 (Chen et al., 2017) and OCRNet (Yuan et al., 2020) are used with ResNet-50 and ResNet101 as backbone, respectively. The learning rate starts at 0.1 and decreases with polynomial learning rate decay with factor as 0.9. The training images are randomly cropped from the original images with size 769. We train the model for 140 epochs at each training split with a mini-batch size of 8.

5.3 Parameter Analysis

The Impact of the Loss Weights is first studied in Fig. 6. To test loss weights λ_1 and λ_2, we fix λ_2 to 1 when λ_1 is being tested, hence fix λ_1 to 8 when λ_2 is being tested. It can be observed that, setting either of them to 0 leads to a substantial performance drop, indicating the impact of $\mathcal{L}_{MCL}$ and $\mathcal{L}_{CRP}$. Setting too large λ_1 and λ_2 degrades the weight of $\mathcal{L}_{CE}$, hence degrades the performance.

The Impact of the Size of Memory Bank is studied in Fig. 7. Larger ρ stores more sample features, hence is beneficial for the performance enhancement. Smaller memory size also achieves reasonably good performance. ρ = 10% leads to a compact memory bank size and only degrades the mAcc by about 0.5%. ρ hence can be flexibly adjusted according to memory capacity.

The Impact of the Batch Size $\mathbf{z}$ sampled from the memory bank at each training iteration is also tested. Larger memory training batch size z samples more features from $\mathcal{M}$, thus involves more positive and negative samples for each training image. As shown in Fig. 8, larger z boosts the performance.

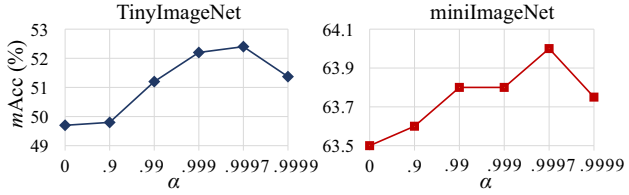

Fig. 9 Effects of the parameter α in Eq. (10)

However, too large z, e.g., 4096, leads to a degradation of performance. One possible explanation may be that introducing too many positive and negative samples posts more strict constraints on the optimization, making the convergence difficult.

The parameter α in Eq. (10) determines the update speed of the momentum update model. Setting $\alpha = 1$ stops the momentum update and setting $\alpha = 0$ removes the momentum update model. Figure 9 shows that, a reasonably large α enhances the performance. This result indicates that momentum update model is beneficial to the IME task.

These tuned parameters are applied on other datasets and tasks, where they generalize well as shown in following experiments.

5.4 Ablation Study

The Impact of Individual Component is studied to show the effectiveness of each component in our method. Experimental results are summarized in Table 1. It can be observed that, only applying $\mathcal{L}_{MCL}$ does not improve the direct fine-tuning baseline on two datasets. This is because only applying $\mathcal{L}_{MCL}$ leads to outdated features in $\mathcal{M}$, which degrade the effectiveness of $\mathcal{L}_{MCL}$. Adding either the regularization loss $\mathcal{L}_{CRP}$ or feature adaptation to $\mathcal{L}_{MCL}$ substantially boosts the performance on both datasets. This shows the effectiveness of CRP, which combines $\mathcal{L}_{CRP}$ and feature adaptation to maintain the memorized features in $\mathcal{M}$. The combination of $\mathcal{L}_{MCL}$, $\mathcal{L}_{CRP}$, and feature adaption achieves the best performance.

$\mathcal{L}_{MCL}$ is the key to update the model with the supervision of the features preserved in the memory bank. To validate its importance, we further conduct an experiment by removing $\mathcal{L}_{MCL}$ and only using CE, $\mathcal{L}_{CRP}$ and FA. This setting degrades the performance from 52.38% to 50.84% on TinyImageNet. Table 1 also indicates the importance of momentum update model, e.g., removing it degrades the mAcc from 52.38 to 49.69% on TinyImageNet. The momentum update model is also a strong technique to improve the performance of a model. To validate that the performance gains are not brought by the momentum update model, we further conduct the experiment. Although the momentum update model improves the mAcc of FT baseline from 44.87

Table 1 Ablation study on individual component

Dataset Settings	Tiny mAcc	Mini mAcc
FT	44.87	56.67
$\mathcal{L}_{MCL}$	43.97	57.90
$\mathcal{L}_{MCL}$ + FA	50.71	62.89
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$	49.76	63.48
$\mathcal{L}_{CRP}$ + FA	50.84	61.92
$\mathcal{L}_{MCL}$ + KD + FA	51.30	63.66
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA†	52.24	63.00
FT + EMA	46.47	60.14
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA*	49.69	63.52
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA	**52.38**	**63.97**

Bold value indicates the best performance result
"Tiny" and "mini" denote the TinyImageNet and miniImageNet, respectively. "FT" denotes fine-tuning on new data. $\mathcal{L}_{MCL}$ and $\mathcal{L}_{CRP}$ are losses in Eq. (17) and Eq. (16), respectively. "FA" denotes feature adaptation. "KD" denotes the standard knowledge distillation loss in (Hinton et al., 2015). "EMA" denotes the momentum update model
* denotes setting α as 0 to remove the momentum update model
† removes the loss computed with ω^j from $\mathcal{L}_{CE}$ in Eq. (7)

Table 2 Comparison with other methods that preserve features

Dataset Settings	Tiny mAcc	Mini mAcc
$\mathcal{L}_{MCL}$ + FA	50.71	62.89
$\mathcal{L}_{MCL}$ + SDC	46.76	59.43
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + SDC	50.21	62.36
REMIND	47.23	59.87
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA	**52.38**	**63.97**

Bold value indicates the best performance result
"SDC" denotes the feature adaptation method proposed in SDC (Yu et al., 2020). "REMIND" denotes that using method proposed in REMIND (Hayes et al., 2020) trains the model

to 46.47% on TinyImageNet, this result still falls behind our method. The further performance improvement is mainly from our proposed components. Replacing $\mathcal{L}_{CRP}$ with the knowledge distillation loss (Hinton et al., 2015) also leads to a worse performance showing that preserving relationship between samples and memorized features is a more reasonable choice as the regularization loss. Table 1 also indicates that removing the loss computed with ω^j in $\mathcal{L}_{CE}$ degrades the performance. We hence could conclude that each component in our method is effective in boosting the model performance for IME.

Comparison with other methods preserving features: some methods also preserve features in the memory bank (Yu et al., 2020; Hayes et al., 2020). To validate that our method is more effectiveness for IME, we compare with these methods in Table 2. SDC (Yu et al., 2020) proposes a feature adaptation method to update the prototype of each category. It can also be used to update the preserved features. Replacing CRP with SDC decreases the mAcc from 52.38 to 46.76% on TinyImageNet. Combining SDC with our $\mathcal{L}_{CRP}$, also degrades the performance of our method. REMIND (Hayes et al., 2020) preserves the features extracted by a backbone with several frozen shallow layers. These preserved features are used to update the other deep layers. Following its setting, we freeze the first 3 blocks of the backbone after the first training split to extract preserved features, and tune the last block. The performance of REMIND is also inferior to the one of our method. REMIND limits the ability to update the entire model, which hence restricts its performance in IME task. These results can further validate the effectiveness of our CRP.

Effects of the initial training set size: We divide the training set into 10 subsets. In the above experiments, we randomly select 2 subsets to construct D^1. Here we further evaluate the impact of a stronger base model trained by a larger initial training split containing 4 subsets. Results are summarized in Table 3. Similar to the results in Table 1, each proposed component works well with a stronger base model. $\mathcal{L}_{MCL}$, $\mathcal{L}_{CRP}$, and feature adaption are beneficial to the final performance. Removing any of them leads to a clear degradation of the performance. The impact of initial training split size is shown in Fig. 10. The performance is boosted as we increase the size of the first training split. Including more training data in the first training split can improve the performance of the base model. A better base model provides a stronger initialization and benefits subsequent optimizations in each training split, finally leading to a better overall mACC as shown in Table 3 and Fig. 10.

Discussions on the challenge of IME: Deep SLDA (Hayes and Kanan, 2020) emphasizes the importance of learning classifiers in CIL. By freezing the backbone after the first task and solely updating the classifier, substantial performance improvements can be achieved. To validate whether the classifier plays an equally critical role in tackle the challenge of IME as it does in CIL, we test the performance of three settings: (a) linear Probing, which freezes the backbone after the first training split and only updates the classifier with newly arrived training splits; (b) deep SLDA, which utilizes a covariance matrix to update the classifier; (c) classifier Upper-Bound, which freezes the backbone after the first training split and trains the classifier with all training splits merged together. Setting (a) serves as the baseline. Setting (c) learns an unbiased classifier since it is trained using merged training splits. If the classifier performance is the bottleneck of IME, substantial performance gains should be observed between settings (a) and (c). The results are shown in Table 4, where Linear Probing and Classifier Upper-Bound achieve similar performance. Training the classifier with merged training splits also fail to enhance performance.

Table 3 Effectiveness of each proposed component with a stronger base model

Dataset Settings	Tiny *m*Acc	Mini *m*Acc
FT	51.71	62.33
$\mathcal{L}_{MCL}$	52.99	63.00
$\mathcal{L}_{MCL}$ + FA	56.12	67.46
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$	54.21	66.91
$\mathcal{L}_{CRP}$ + FA	57.06	67.45
$\mathcal{L}_{MCL}$ + KD + FA	56.46	67.68
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA†	57.46	68.16
FT + EMA	53.35	64.79
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA*	54.21	67.10
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA	**57.56**	**68.30**

Bold value indicates the best performance result
The base model is trained with 40% training data. Notations are consistent with those in Table 1
* denotes setting α as 0 to remove the momentum update model
† removes the loss computed with ω^j from $\mathcal{L}_{CE}$ in Eq. (7)

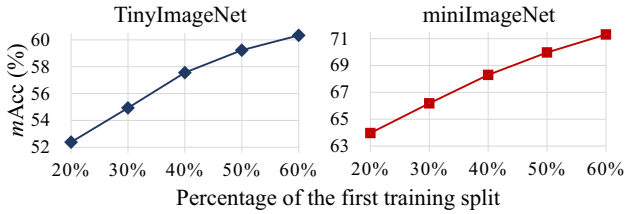

Fig. 10 Effects of the initial training split size

It indicates that, only tuning classifiers with newly arrived training splits cannot bring substantially performance gains. These results demonstrate that the classifier is not the bottleneck in the performance of IME, which poses different challenges with CIL.

Kim et al. (2023) demonstrate that IL can be tackled through a combination of (a) a method capable of learning each sub-task in IL without forgetting them, and (b) an out-of-distribution detection method to assign each test instance to the correct sub-task model. To assess the suitability of this theory for IME, we implement different methods in Table 5 to satisfy the criteria (a): "Independent" denotes different models are trained independently for each training split; "PackNet" denotes the models are trained by performing iterative pruning and network re-training as in Mallya and Lazebnik (2018). "Expand" denotes the previously learned parameters are frozen and some additional feature dimensions are added for each new training split as in Yan et al. (2021). To satisfy criteria (b) during inference, each sample is simultaneously passed to multiple models trained by different training splits. We consider the final prediction as correct, if any of those model could provide a correct prediction. Results are summarized in Table 5, where those methods

Table 4 Ablation study on different training settings for the classifier

Dataset Settings	Tiny *m*Acc	Mini *m*Acc
Linear Probing	39.82	56.93
Deep SLDA	39.77	57.21
Classifier Upper-Bound	40.21	57.38
Ours	**52.38**	**63.97**

Bold value indicates the best performance result

Table 5 The performance of the IL theory proposed by Kim et al. (2023) on the IME task

Dataset Settings	Tiny *m*Acc	Mini *m*Acc
Independent	42.47	54.61
PackNet	45.05	56.32
Expand	47.89	58.37
Ours	**52.38**	**63.97**

Bold value indicates the best performance result

get lower performance than our proposed method. It indicates that, the theory presented by Kim et al. (2023) is not suitable for IME. This is because each training split in IME cannot be regarded as an independent task like TIL, CIL or DIL. To improve the IME performance, it is crucial to consider the joint distribution of all training splits. This new challenge makes current IL methods do not work well on IME.

Performance on semantic segmentation: to further validate our method, we conduct experiments on semantic segmentation. The well-known method DeepLabV3 (Chen et al., 2017) is employed as the segmentation head. The results are shown in Table 6. Similar to the result in the image classification, the proposed components are effective in semantic segmentation. For example, without $\mathcal{L}_{CRP}$ or feature adaptation, the performance improvement of $\mathcal{L}_{MCL}$ is limited. The combination of $\mathcal{L}_{MCL}$, $\mathcal{L}_{CRP}$ and feature adaptation achieves the best performance, validating the generalization ability of our method on other task. To verify our approach is applicable to different semantic segmentation methods, the experiments with OCRNet (Yuan et al., 2020) are also conducted. Our method still outperforms the fine-tuning baseline with a clear margin.

5.5 Comparison with Other Methods

Image classification: As IL is closely related to the IME, we compare our method with several recent IL methods on TinyImageNet, miniImageNet, ImageNet1K and CUB respectively. Those IL methods follow similar intuition to avoid forgetting of learned knowledge and learn new tasks or domains from new data. Table 7 and Fig. 11 summarize the results.

Table 6 Ablation studies of each component on Cityscapes

Settings	Avg. mIoU
FT	73.18
$\mathcal{L}_{MCL}$	73.37
$\mathcal{L}_{MCL}$ + FA	73.70
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$	74.78
$\mathcal{L}_{CRP}$ + FA	74.62
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA†	74.72
FT + EMA	73.47
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA$*$	73.86
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA	**75.06**
FT ‡	74.39
$\mathcal{L}_{MCL}$ + $\mathcal{L}_{CRP}$ + FA‡	**76.26**

Bold value indicates the best performance result
"FT" denotes fine-tuning on new data. $\mathcal{L}_{MCL}$ and $\mathcal{L}_{CRP}$ are losses in Eq. (17) and Eq. (16), respectively. "KD" denotes the standard knowledge distillation loss in (Hinton et al., 2015). "EMA" denotes the momentum update model. Superscript $*$ denotes setting α as 0 to remove the momentum update model, and † removes the loss computed with ω^j from $\mathcal{L}_{CE}$ in Eq. (7). ‡ denotes using OCRNet as segmentation head

As shown in Table 7, the fine-tuning baseline marginally improves the initial model trained on D^1. The distillation-based method LwF (Li and Hoiem, 2017) improves the baseline performance, showing that memorizing previously learned cues is beneficial for the model enhancement. We find that, the regularization-based method EWC (Kirkpatrick et al., 2017) does not improve the baseline. Perhaps EWC penalizes the change of parameters that are useful for the previous training set. This setup is reasonable for IL, because it needs to maintain the performance of previous tasks when learning a new task. EWC reasonably preserves critical parameters to ensure the performance of previous tasks. For IME, each training split only covers part of the feature distribution for the target vision task. Applying too strict regularization on previous parameters in EWC limits the optimization on available training data.

Memory-based methods GEM (Lopez-Paz and Ranzato, 2017), DER (Buzzega et al., 2020), RM (Bang et al., 2021), Co2L (Cha et al., 2021), FOSTER (Wang et al., 2022a), and CSCCT (Ashok et al., 2022) perform better than the baseline by preserving 10% training images in the memory. However, reducing the percentage of preserved images from 10 to 2% substantially degrades their performance to a level similar to the baseline. It is clear that these methods are more sensitive

Table 7 Performance comparison with other IL methods on 4 datasets

Dataset	TinyImageNet			miniImageNet			ImageNet1K			CUB		
Methods	*m*Acc	Acc*	O	*m*Acc	Acc*	O	*m*Acc	Acc*	O	*m*Acc	Acc*	O
FT	44.87	48.93	0	56.67	59.91	0	55.49	56.69	0	71.70	73.30	0
Upper	56.50	63.25	1180	68.36	75.09	1010	66.27	69.60	183,980	76.55	78.58	860
LwF	49.17	52.99	0	60.18	64.46	0	60.03	61.51	0	72.85	75.16	0
EWC	45.03	48.73	0	56.59	58.89	0	57.48	58.85	0	72.41	73.97	0
GEM 10%	45.58	49.9	118	59.62	63.88	101	56.76	58.5	18,398	69.45	70.94	86
GEM 2%	44.74	48.54	24	57.23	60.03	20	–	–	-	69.25	70.79	17
DER 10%	49.76	53.82	118	62.35	66.32	101	58.10	59.47	18,398	71.92	74.09	86
DER 2%	47.47	50.66	24	60.83	64.59	20	–	–	–	71.32	73.87	17
RM 10%	46.69	50.34	118	59.76	63.81	101	55.12	56.99	18,398	72.97	74.92	86
RM 2%	45.02	48.02	24	57.53	60.26	20	–	–	–	72.04	74.02	17
Co2L 10%	49.81	53.78	118	61.95	65.94	101	58.18	59.64	18,398	71.67	73.78	86
Co2L 2%	45.26	48.14	24	60.06	63.26	20	–	–	–	70.87	72.91	17
FOSTER 10%	49.98	54.03	118	62.04	66.03	101	57.50	58.85	18,398	70.60	73.67	86
FOSTER 2%	46.24	49.28	24	61.28	63.88	20	–	–	–	69.97	72.45	17
CSCCT 10%	48.82	53.11	118	61.67	65.89	101	57.61	58.56	18,398	71.42	74.51	86
CSCCT 2%	45.43	47.96	24	60.12	63.38	20	–	–	–	70.25	72.67	17
Ours 100%	**52.38**	**56.15**	195	**63.97**	**67.59**	98	**61.05**	**62.32**	2,498	**75.11**	**76.72**	46
Ours 10%	52.21	55.92	20	63.66	67.55	10	60.83	62.02	250	74.63	76.63	5
Ours 1%	51.70	55.09	2	63.20	67.14	1	60.66	61.94	25	74.07	76.15	0.5

Bold value indicates the best performance result
The left column shows the percentage of preserved training data of many methods. "FT" denotes fine-tuning on the newly added data. "Upper" denotes the performance upper bound that trains the model from scratch on the fused training set. "*" denotes the accuracy of final training split. "O" denotes the memory overhead (in MB) caused by preserved images or features

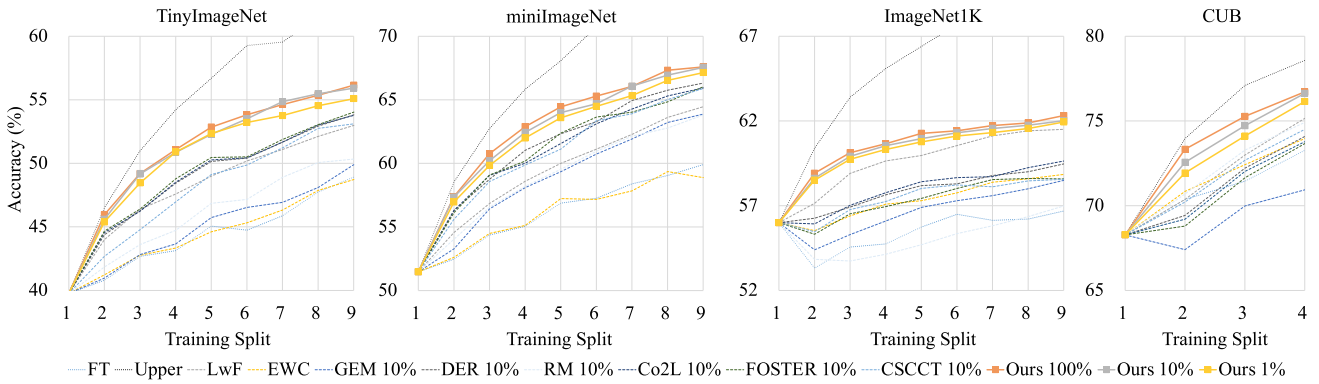

Fig. 11 Performance comparison at each training split on TinyImageNet, miniImageNet, ImageNet1K, and CUB, respectively. The proposed method substantially outperforms other IL methods

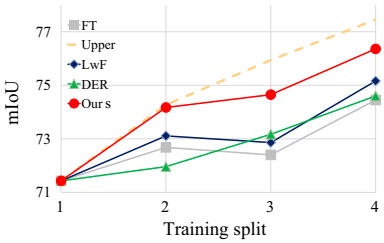

Fig. 12 Performance comparison at each training split on Cityscapes. DeepLabV3 is used as the segmentation head. The "Upper" denotes the performance upper bound of training from scratch repeatedly

to the number of preserved images, and need to consume a large memory overhead to ensure a good performance. Co2L leverages contrastive loss and introduces Instance-wise Relation Distillation (IRD) to regulate the changes in the feature relationship between batch samples. It only focuses on the relationship between samples within the current training task. In contrast, $\mathcal{L}_{CRP}$ regulates the relationship between samples in each training split and preserved features in $\mathcal{M}$. Our method achieves better performance compared with Co2L. FOSTER dynamically expands new modules to learn new categories in CIL. However, in IME, the joint distribution of all training splits must be considered. Freezing most parameters and introducing only new tunable modules with limited tunable parameters do not perform well. Compared with those works, our method performs substantially better. Our work is also less sensitive to the size of memory bank, e.g., similar performance is achieved by preserving 10 and 1% features, respectively. Our work hence mitigates the memory overhead and achieves significantly better accuracy. Storing features instead of original images is also friendly to privacy protection.

To show the scalability of our method, we further test it on the large-scale ImageNet1K, where the training set contains about 1 M images. On this large-scale dataset, our method achieves the mAcc of 61.05%, also outperforms the competing methods. Upon all these experiments, we could conclude that our method achieves promising performance on the IME task.

Image classification on other network architectures and pre-training methods: Transformer (Dosovitskiy et al., 2020) shows better capability to capture long-range dependencies than CNN, and is getting more popular in vision tasks. In addition, recent self-supervised methods (He et al., 2020; Li et al., 2020; Chen et al., 2020b) also achieve impressive representation learning performance. We further validate the effectiveness of our method on Transformers, as well as backbones trained with self-supervised methods. The experiments are conducted on CUB, Cars196 (Krause et al., 2013), and OfficeHome (Venkateswara et al., 2017).

Cars196 is a widely used dataset for fine-grained image recognition to recognize different cars. OfficeHome is constructed for object recognition in office and home environments with four domains, which are Art, Clipart, Product, and Real World. We follow the same training setting as CUB. We randomly divide the training sets of Cars196 and OfficeHome into 10 independent subsets with equal size. D^1 is constructed with 4 subsets. Each of remaining training splits D^2-D^4 is constructed with 2 subsets. Results are shown in Table 8, where our method also exhibits the best performance when using ViT-small or ResNet-50 trained with MoCoV2 as the backbone. We also compare our method with prompt learning-based methods (Wang et al., 2022, d) that learn several prompt tokens based on a powerful frozen Transformer. These methods do not perform well on IME because learning prompt tokens is not sufficient to enhance the representation ability of the backbone for a fixed vision task. These results highlight the effectiveness of our method across different types of backbones.

Scalability of our method: To demonstrate the scalability of our method on large-scale datasets and a stronger backbone, we conduct experiments on ImageNet-1K and iNaturalist2018 (Van Horn et al., 2018) using ResNet-50

Table 8 Performance comparison on CUB, Cars196, and OfficeHome with ViT-small (Dosovitskiy et al., 2020) and ResNet-50 trained by MoCoV2 (Chen et al., 2020b) as the backbone

Datasets	CUB				Cars196				OfficeHome			
Backbone	ViT-small		MoCoV2		ViT-small		MoCoV2		ViT-small		MoCoV2	
Methods	*m*Acc	Acc*	*m*Acc	Acc*	*m*Acc	Acc*	*m*Acc	Acc*	*m*Acc	Acc*	*m*Acc	Acc*
FT	77.53	77.71	60.78	63.65	81.04	83.25	76.72	79.47	78.60	78.99	73.15	74.46
Upper	82.86	83.91	69.80	73.85	87.77	90.23	82.89	85.97	81.72	82.79	78.22	79.71
LwF	78.22	80.43	62.87	66.67	81.16	84.44	76.22	79.34	78.77	79.24	72.89	74.34
EWC	76.89	77.65	61.17	63.55	81.62	83.48	77.13	79.77	78.16	78.50	73.24	74.40
GEM 10%	75.98	78.23	60.15	64.26	80.99	84.12	67.42	71.67	70.19	72.34	71.20	72.46
DER 10%	78.24	79.65	61.97	66.22	82.72	85.95	79.27	82.28	79.21	79.78	73.87	75.63
RM 10%	78.73	80.52	62.52	65.81	82.39	84.82	77.60	80.28	78.82	79.83	73.95	75.54
Co2L 10%	77.89	79.47	61.82	65.88	81.79	84.03	78.26	80.64	79.01	80.29	73.92	75.27
FOSTER 10%	77.62	79.28	61.28	65.03	81.46	83.26	77.43	79.48	78.79	79.11	73.09	74.79
CSCCT 10%	78.02	79.32	61.79	66.25	81.50	83.45	77.07	79.25	78.46	78.87	73.79	75.03
S-Prompt	76.67	77.10	-	-	81.24	83.24	-	-	79.35	80.47	-	-
L2P	76.53	77.27	-	-	81.31	83.46	-	-	79.67	81.24	-	-
Ours 100%	**82.29**	**83.64**	**67.27**	**72.57**	**86.79**	**89.36**	**81.99**	**85.06**	**81.48**	**81.94**	**77.62**	**78.50**
Ours 10%	81.67	83.28	66.82	72.19	86.36	88.92	81.87	84.79	81.14	81.66	77.20	77.81
Ours 1%	81.03	82.83	66.36	71.92	85.84	88.48	81.13	84.34	80.91	81.35	76.89	77.53

Bold value indicates the best performance result
"Upper" denotes the performance upper bound that trains the model from scratch on the fused training set. "*" denotes the accuracy of final training split

Table 9 Performance comparison on ImageNet-1K and iNaturalist2018 with ResNet-50 as the backbone

Datasets	ImageNet1K		iNaturalist2018	
Methods	*m*Acc	Acc*	*m*Acc	Acc*
FT	60.38	61.23	61.42	66.03
Upper	72.58	76.01	76.89	84.33
LwF	64.79	66.53	62.91	68.13
EWC	62.04	64.07	61.17	65.73
GEM 10%	61.63	63.78	60.82	63.23
DER 10%	63.28	64.89	64.61	69.63
RM 10%	61.27	61.92	61.33	68.70
Co2L 10%	63.80	64.42	64.74	70.28
FOSTER 10%	62.93	63.18	63.55	69.23
CSCCT 10%	62.53	62.77	63.06	68.27
Ours 100%	**67.24**	**68.28**	**68.56**	**73.73**
Ours 10%	66.91	68.04	67.78	73.03
Ours 1%	66.27	67.48	67.02	72.43

Bold value indicates the best performance result
"Upper" denotes the performance upper bound that trains the model from scratch on the fused training set. "*" denotes the accuracy of final training split

as the backbone. Results are presented in Table 9. We follow a similar setting in Petit et al. (2023) to get a subset of iNaturalist2018 with 1000 classes. When leveraging the more powerful ResNet-50 architecture, our method achieves a *m*Acc of 67.24% on ImageNet1K. Notably, this presents an appreciable performance increase of approximately 6.31% towards the theoretical upper bound compared with the results in Table 7. Those compared methods show less performance improvements, e.g., LwF only improves *m*Acc from 60.03 to 64.79%. In iNaturalist2018, our method outperforms other methods by a clear margin, e.g., surpassing LwF by about 5.65%. The substantial performance enhancement of our method indicates its impressive scalability and efficacy when integrated with a powerful backbone. We conclude that our method presents reasonably good scalability, and also works well with strong backbones.

Domain incremental setting: The domain incremental learning aims to learn from different domains sequentially while retaining the performance on learned domains. It thus can be regarded as a special case of IME by splitting each training split into different domains. To validate the effectiveness of the proposed method under this setting, we conduct experiments on OfficeHome under the domain incremental setting. The training set is divided into four training splits, each containing one domain. The order of the domains is Art, Clipart, Product, and Real World, respectively. The results are summarized in Table 10, where our method achieves 80.80% *m*Acc, outperforming all compared incremental learning methods. This validates that our method can also deal with the domain shift between different training splits.

Table 10 Performance comparison on OfficeHome under domain incremental setting with ResNet-50 as the backbone. "Upper" denotes the performance upper bound that trains the model from scratch on the fused training set. "*" denotes the accuracy of final training split

Datasets	OfficeHome	
methods	*m*Acc	Acc*
FT	71.67	73.43
Upper	81.79	84.06
LwF	77.02	79.45
EWC	76.28	78.50
GEM 10%	72.67	74.70
DER 10%	78.75	80.07
RM 10%	77.17	78.32
Co2L 10%	78.94	80.85
FOSTER 10%	78.43	79.12
CSCCT 10%	78.86	79.98
Ours 100%	**80.80**	**82.73**
Ours 10%	80.57	82.64
Ours 1%	80.13	82.06

Bold value indicates the best performance result

Semantic segmentation: In addition to visualizations in Fig. 5, more results on dense prediction task semantic segmentation are shown in Fig. 12. Compared with direct fine-tuning, our method consistently improves the performance with the sequentially arrived data. After three new training splits, our method improves the mIOU by about 5%. The last training split brings about 1.9% gains. With 10% samples in the memory bank, our method also outperforms the incremental learning methods LwF (Li and Hoiem, 2017) and DER (Buzzega et al., 2020) by 1.2 and 1.8% mIOU at the last training split, respectively. In addition, storing features in the memory bank also leads to less memory requirement of our method. These results can further validate the generalization ability of our method on different vision tasks.

6 Conclusion

This paper studies a new task coined as Incremental Model Enhancement (IME). It aims to enhance the model performance on a specific vision task using sequentially arrived training data. Our method adopts a memory bank to store features of previous training data, which serves to preserve previously learned knowledge, and then learn from new training data via Contrastive Relation Preserving (CRP) and the Memory-based Contrastive Learning (MCL), respectively. CRP and MCL work iteratively to accumulate meaningful cues in previous data and evolve with new data. Experiments on several large-scale image classification and semantic segmentation benchmarks demonstrate the effectiveness of CRP and MCL. Comparisons with recent IL methods show the promising performance of our method in aspects of accuracy and memory efficiency. Table 7 and Fig. 11 show a substantial gap between current method and the performance upper bound. Therefore, more efforts are still needed to explore this challenging task.

Acknowledgements This work is supported in part by the Natural Science Foundation of China under Grant No. U20B2052, 61936011, in part by the Okawa Foundation Research Award.

Data Availability This paper uses public datasets to conduct experiments. Those datasets are available in following URLs. TinyImageNet (Stanford, 2015): http://tiny-imagenet.herokuapp.com/. miniImageNet (Vinyals et al., 2016): https://goo.gl/e3orz6/. ImageNet1K (Russakovsky et al., 2015): https://www.image-net.org/. CUB (Wah et al., 2011): http://www.vision.caltech.edu/datasets/cub_200_2011/. Cityscapes (Cordts et al., 2016): https://www.cityscapes-dataset.com/. OfficeHome (Krause et al., 2013): https://www.hemanthdv.org/officeHomeDataset.html/. iNaturalist2018 (Van Horn et al., 2018): https://github.com/visipedia/inat_comp/tree/master/2018.

References

Aljundi, R., Lin, M., Goujaud, B., & Bengio, Y. (2019). Gradient based sample selection for online continual learning. arXiv preprint arXiv:1903.08671.

Ashok, A., Joseph, K., Balasubramanian, V. N. (2022). Class-incremental learning with cross-space clustering and controlled transfer. In *ECCV* (pp. 105–122). Springer.

Bang, J., Kim, H., Yoo, Y., Ha, J. W., & Choi, J. (2021). Rainbow memory: Continual learning with a memory of diverse samples. In *CVPR* (pp. 8218–8227).

Belouadah, E., & Popescu, A. (2019). Il2m: Class incremental learning with dual memory. In *ICCV* (pp. 583–592).

Bobu, A., Tzeng, E., Hoffman, J., & Darrell, T. (2018). Adapting to continuously shifting domains. In *ICLR workshop*.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., & Calderara, S. (2020). Dark experience for general continual learning: A strong, simple baseline. arXiv preprint arXiv:2004.07211.

Cha, H., Lee, J., & Shin, J. (2021). Co2l: Contrastive continual learning. In *ICCV* (pp. 9516–9525).

Chen, LC., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv:1706.05587.

Chen, X., & He, K. (2021). Exploring simple siamese representation learning. In *CVPR* (pp. 15750–15758).

Chen, X., Fan, H., Girshick, R., & He, K. (2020b). Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *ICML, PMLR* (pp. 1597–1607).

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *CVPR*.

Dhar, P., Singh, R. V., Peng, K. C., Wu, Z., & Chellappa, R. (2019). Learning without memorizing. In *CVPR* (pp. 5138–5146).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., & Uszkoreit, J. (2020). An image is worth 16 × 16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Douillard, A., Cord, M., Ollion, C., Robert, T., & Valle, E. (2020). Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV* (pp. 86–102). Springer.

Fang, Z., Wang, J., Wang, L., Zhang, L., Yang, Y., & Liu, Z. (2021). Seed: Self-supervised distillation for visual representation. arXiv preprint arXiv:2101.04731.

Farajtabar, M., Azizan, N., Mott, A., & Li, A. (2020). Orthogonal gradient descent for continual learning. In *International conference on artificial intelligence and statistics* (pp. 3762–3773). PMLR.

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *IJCV, 129*, 1789–1819.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)* (pp. 1735–1742). IEEE.

Hayes, TL., Kafle, K., Shrestha, R., Acharya, M., & Kanan, C. (2020). Remind your neural network to prevent catastrophic forgetting. In *ECCV* (pp. 466–483). Springer.

Hayes, T. L., & Kanan, C. (2020). Lifelong machine learning with deep streaming linear discriminant analysis. In *CVPRW* (pp. 220–221).

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *CVPR* (pp. 9729–9738).

He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *ICCV* (pp. 1389–1397).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770–778).

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Hoi, S. C., Sahoo, D., Lu, J., & Zhao, P. (2021). Online learning: A comprehensive survey. *Neurocomputing, 459*, 249–289.

Iscen, A., Zhang, J., Lazebnik, S., & Schmid, C. (2020). Memory-efficient incremental learning through feature adaptation. In *ECCV* (pp. 699–715). Springer.

Jung, H., Ju, J., Jung, M., & Kim, J. (2018). Less-forgetful learning for domain expansion in deep neural networks. In *AAAI*.

Kalantidis, Y., Sariyildiz, M. B., Pion, N., Weinzaepfel, P., & Larlus, D. (2020). Hard negative mixing for contrastive learning. *NeuIPS, 33*, 21798–21809.

Kang, B., Li, Y., Xie, S., Yuan, Z., & Feng, J. (2020). Exploring balanced feature spaces for representation learning. In *ICLR*.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., & Krishnan, D. (2020). Supervised contrastive learning. arXiv preprint arXiv:2004.11362.

Kim, G., Xiao, C., Konishi, T., & Liu, B. (2023) Learnability and algorithm for continual learning. arXiv preprint arXiv:2306.12646.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., & Hassabis, D. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences, 114*(13), 3521–3526.

Kovashka, A., Russakovsky, O., Fei-Fei, L., & Grauman, K. (2016). Crowdsourcing in computer vision. *Foundations and Trends ® in Computer Graphics and Vision, 10*(3), 177–243.

Krause, J., Stark, M., Deng, J., & Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *ICCVW* (pp. 554–561).

Lee, S. W., Kim, J. H., Jun, J., Ha, J. W., & Zhang, B. T. (2017). Overcoming catastrophic forgetting by incremental moment matching. arXiv preprint arXiv:1703.08475.

Li ,J., Zhou, P., Xiong, C., & Hoi, S. C. (2020). Prototypical contrastive learning of unsupervised representations. arXiv preprint arXiv:2005.04966.

Li, Z., & Hoiem, D. (2017). Learning without forgetting. *TPAMI, 40*(12), 2935–2947.

Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learning. *NeurIPS, 30*, 6467–6476.

Mallya, A., & Lazebnik, S .(2018). Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR* (pp. 7765–7773).

Mancini, M., Bulo, S. R., Caputo, B., & Ricci, E. (2019). Adagraph: Unifying predictive and continuous domain adaptation through graphs. In *CVPR* (pp. 6568–6577).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & Desmaison, A. (2019). Pytorch: An imperative style, high-performance deep learning library. *NeurIPS, 32*, 8026–8037.

Petit, G., Popescu, A., Belouadah, E., Picard, D., & Delezoide, B. (2023). Plastil: Plastic and stable exemplar-free class-incremental learning. In *Conference on lifelong learning agents* (pp. 399–414). PMLR.

Prabhu, A., Torr, P. H., & Dokania, P. K. (2020). Gdumb: A simple approach that questions our progress in continual learning. In *ECCV* (pp. 524–540). Springer.

Pu, N., Chen, W., Liu, Y., Bakker, E. M., & Lew, M. S. (2021). Lifelong person re-identification via adaptive knowledge accumulation. In *CVPR* (pp. 7901–7910).

Rebuffi, S. A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *CVPR* (pp. 2001–2010).

Romero, A., Ballas, N., & Kahou, S. E.(2014). Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. *IJCV, 115*(3), 211–252.

Saha, G., Garg, I., & Roy, K. (2021). Gradient projection memory for continual learning. arXiv preprint arXiv:2103.09762.

Stanford. (2015). Tiny ImageNet Challenge (CS231n). http://tiny-imagenet.herokuapp.com/

Sun, Q., Liu, Y., Chua, T. S., & Schiele, B. (2019). Meta-transfer learning for few-shot learning. In *CVPR* (pp. 403–412).

Tao, X., Chang, X., Hong, X., Wei, X., & Gong, Y. (2020). Topology-preserving class-incremental learning. In *ECCV* (pp. 254–270). Springer.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*, 2579–2605.

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., & Belongie, S. (2018). The inaturalist species classification and detection dataset. In *CVPR* (pp. 8769–8778).

Venkateswara, H., Eusebio, J., Chakraborty, S., & Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. In *CVPR* (pp. 5018–5027).

Vijayanarasimhan, S., & Grauman, K. (2011). Cost-sensitive active visual category learning. *IJCV, 91*, 24–44.

Vinyals, O., Blundell, C., Lillicrap, T., & Wierstra, D. (2016). Matching networks for one shot learning. *NeurIPS, 29*, 3630–3638.

Wah, C., Branson, S., Welinder, P., Perona, P., & Belongie, S. (2011). *The caltech-ucsd birds-200-2011 dataset*. California Institute of Technology.

Wang, Q., Fink, O., Van Gool, L., & Dai, D. (2022b). Continual test-time domain adaptation. In *CVPR* (pp. 7201–7211).

Wang, S., Li, X., Sun, J., & Xu, Z. (2021). Training networks in null space of feature covariance for continual learning. In *CVPR* (pp. 184–193).

Wang, Z., Zhang, Z., Lee, C. Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., & Pfister, T. (2022d). Learning to prompt for continual learning. In *CVPR* (pp. 139–149).

Wang, F. Y., Zhou, D. W., Ye, H. J., & Zhan, D. C. (2022a). Foster: Feature boosting and compression for class-incremental learning. In *ECCV* (pp. 398–414). Springer.

Wang, Y., Huang, Z., & Hong, X. (2022). S-prompts learning with pre-trained transformers: An Occam's razor for domain incremental learning. *NeurIPS, 35*, 5682–5695.

Xie, J., Yan, S., & He, X. (2022). General incremental learning with domain-aware categorical representations. In *CVPR* (pp. 14351–14360).

Yan, S., Xie, J., & He, X. (2021). Der: Dynamically expandable representation for class incremental learning. In *CVPR* (pp. 3014–3023).

Yao, X., Bai, Y., Zhang, X., Zhang, Y., Sun, Q., Chen, R., Li, R. & Yu, B. (2022). Pcl: Proxy-based contrastive learning for domain generalization. In *CVPR* (pp. 7097–7107).

Yoon, J., Yang, E., Lee, J., & Hwang, S. J. (2017). Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547.

Yu, L., Twardowski, B., Liu, X., Herranz, L., Wang, K., Cheng, Y., Jui, S., & Weijer, J. V. D. (2020). Semantic drift compensation for class-incremental learning. In *CVPR* (pp. 6982–6991).

Yuan, Y., Chen, X., & Wang, J. (2020). Object-contextual representations for semantic segmentation. In *ECCV* (pp. 173–190). Springer.

Zagoruyko, S., & Komodakis, N. (2016). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928.

Zhu, R., Zhao, B., Liu, J., Sun, Z., & Chen, C. W. (2021). Improving contrastive learning by visualizing feature transformation. In *ICCV* (pp. 10306–10315).