This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

General Incremental Learning with Domain-aware Categorical Representations

Jiangwei Xie[1*] Shipeng Yan[1,3,4*] Xuming He[1,2]
[1]School of Information Science and Technology, ShanghaiTech University
[2]Shanghai Engineering Research Center of Intelligent Vision and Imaging
[3]Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
[4]University of Chinese Academy of Sciences
{xiejw,yanshp,hexm}@shanghaitech.edu.cn

Abstract

Continual learning is an important problem for achieving human-level intelligence in real-world applications as an agent must continuously accumulate knowledge in response to streaming data/tasks. In this work, we consider a general and yet under-explored incremental learning problem in which both the class distribution and class-specific domain distribution change over time. In addition to the typical challenges in class incremental learning, this setting also faces the intra-class stability-plasticity dilemma and intra-class domain imbalance problems. To address above issues, we develop a novel domain-aware continual learning method based on the EM framework. Specifically, we introduce a flexible class representation based on the von Mises-Fisher mixture model to capture the intra-class structure, using an expansion-and-reduction strategy to dynamically increase the number of components according to the class complexity. Moreover, we design a bi-level balanced memory to cope with data imbalances within and across classes, which combines with a distillation loss to achieve better inter- and intra-class stability-plasticity trade-off. We conduct exhaustive experiments on three benchmarks: iDigits, iDomainNet and iCIFAR-20. The results show that our approach consistently outperforms previous methods by a significant margin, demonstrating its superiority.

1. Introduction

In order to achieve human-level intelligence, it is indispensable for a learning system to continuously accumulate knowledge over time in an ever-changing environment, known as continual or incremental learning [5]. To cope with real-world scenarios, we consider a *general incremental learning* problem [3, 18], where both the class distribution and class-specific domain distributions of the incoming data continuously change across sequential learning sessions. This requires a model to incrementally learn not only novel class concepts but also new variants of previously-learned concepts.

Existing works on the general incremental learning typically focuse on the online class incremental learning setting [3, 18], which has to sacrifice the performance due to its strict computation/memory constraints. In this work, we instead aim to tackle the offline incremental learning setting, which has the potential to achieve significantly higher performance than the online counterpart. We note that, unlike the offline class incremental learning [7], this general incremental learning problem additionally faces an *intra-class stability-plasticity dilemma* which refers to the trade-off between adapting novel examples and preserving current knowledge of the class, and an *intra-class domain imbalance* problem, where the model is biased toward incoming domains due to a limited memory. The intra-class problem is particularly challenging since the domain labels are usually unknown in practice.

The majority of current research on the class incremental learning either focuses on improving the inter-class stability-plasticity trade-off and imbalance issue [7, 25] or mainly attempts to tackle the intra-class stability-plasticity dilemma [29, 30, 33]. Recent works on the online general class incremental learning [3, 18] typically ignore the intra-class structure of data distribution. In particular, those methods usually adopt the same feature representation for the data from both incoming and existing domains of a class, which makes it difficult to learn new domains without interference with previously-learned representation of that class. Such domain-invariant representations sacrifice the intra-class plasticity, often resulting in a poor intra-class trade-off between plasticity and stability.

In this work, we develop a novel domain-aware learning framework for the general incremental learning problem, which enables us to address both inter-class and intra-class

*Both authors contributed equally. This work was supported by Shanghai Science and Technology Program 21010502700.

challenges in a unified manner. To this end, we introduce a flexible class representation based on the von Mises-Fisher (vMF) mixture model to capture the intra-class structure and a bi-level balanced memory to cope with data imbalance within and across classes. In detail, we build a vMF mixture model on deep features of each class to learn a domain-aware representation and design an expansion-and-reduction strategy to dynamically increase the number of its components in new sessions. Combining with an inter- and intra-class forgetting resistence strategy like distillation, our design is capable of achieving better inter- and intra-class stability-plasticity trade-off. Moreover, based on the learned class representation, we propose a balanced memory at both inter- and intra-class level to mitigate bias toward new classes and new domains.

To learn our domain-aware representation, we devise an iterative training procedure for model update at each incremental session. Specifically, when new data comes, we first inherit the learned model from last session and allocate new components for the mixture model of each incoming category. We then adopt the Expectation-Maximization (EM) algorithm to jointly learn the backbone and mixture models, treating the component assignments of input data as latent variables. We incorporate strategies overcoming inter-class forgetting like [13, 27, 37] and adopt intra-class knowledge distillation for alleviating inter- and intra-class catastrophic forgetting, respectively. After each model update, we further perform a mixture reduction step based on hierarchical clustering to maintain a compact clustering result. During inference, we first extract input features via the backbone network and then infer its component assignment in class, followed by taking the class with the maximal component probability as prediction.

We validate our approach by extensive comparisons with prior incremental learning methods on three benchmarks: iDigits, iDomainNet and iCIFAR-20. For each benchmark, we conduct experiments on splits with varying class and domain distributions over time. The empirical results and ablation study show that our method consistently outperforms other approaches across all benchmarks.

In summary, the main contributions of our work are three-folds as follows:

- We formulate a new offline general incremental learning problem where both class distribution and intra-class domain distribution continuously change over time. This problem has the stability-plasticity dilemma and imbalance issue at both inter- and intra-class level.
- We propose a method based on vMF mixture models to learn a domain-aware representation for addressing the general stability-plasticity dilemma and develop a bi-level balanced memory strategy to mitigate both the inter- and intra-class data imbalance issue.
- Extensive experiments on three benchmarks show that our strategy consistently outperforms existing methods by a sizable margin.

2. Related Works

Existing literature in incremental learning can be summarized from three perspectives, including the problem settings, stability-plasticity dilemma and imbalance strategy.

Problem Settings Most previous works focus on either class incremental learning [4, 6, 9, 13, 23, 28, 30, 35, 37, 38] or domain incremental learning [29, 30, 33]. Only a few attempts [1, 3, 18] address the general incremental learning problem, but these methods mainly learn from a data stream in an online fashion. In contrast, we study the offline learning setting, which allows multiple passes of incoming data at each incremental session. This paradigm is important in many real world applications [14, 22] in which offline learning regularly outperforms the online version. To the best of our knowledge, we are the first to tackle the general incremental learning that allows offline training in sessions.

Several recent studies tackle the class-incremental domain adaptation problem [17, 36], which aims at adapting the model trained on a source domain to a target domain including novel classes. They mainly focus on the performance on the target domain and thus do not need to cope with the intra-class forgetting challenge. By contrast, our work requires the model to perform well on not only old and new classes, but also old and new domains of those classes.

It is also worth noting that while continual learning without data memory has attracted much attention in literature [32, 40, 42], these methods typically perform less effectively than those with a limited data memory for storing old examples [9, 37]. In this work, we allow methods to access a finite number of previously seen examples as in the majority of existing incremental learning approaches.

Stability-Plasticity Dilemma To alleviate the forgetting of learned representation, current continual learning methods can be largely grouped into three categories. The first are the regularization-based methods [6, 10, 15], which add regularization on the parameters directly to prevent dramatic changes of the important parameters. The second are the distillation-based methods [4, 9, 13, 28, 30, 35], which adopt knowledge distillation to preserve the representation by penalizing the difference between outputs of previous and current models. The third are the structure-based methods [24, 37], which allocate new parameters at each new session and prevent the change of the representation learned at previous sessions. However, the representations in these methods are usually domain invariant, which cannot provide enough intra-class plasticity without sacrificing the intra-class stability. By contrast, our method can achieve

better stability-plasticity dilemma by discovering and maintaining the intra-class structure.

Imbalance Strategy The imbalance issue is largely caused by the limited size of memory. To deal with this problem, most works [4, 13, 34, 41] adjust the classifier weights or prediction logits between classes to eliminate the bias after learning the representations. We note that these works mainly focus on solving the inter-class imbalance problem and can not cope with or easily be extended to solve the intra-class domain imbalance due to the missing domain labels. By contrast, our work can simultaneously achieve both inter- and intra-class balance with the help of domain label estimation in the EM framework.

3. Method

In this work, our goal is to address the general incremental learning problem in which both class and domain distribution change over time. To this end, we propose to learn a domain-aware representation capable of achieving a better stability-plasticity trade-off at both intra- and inter-class level. In particular, we develop a mixture model for each class to capture the intra-class structure and learn the mixture model via a new EM-based framework.

We first present the problem setup in Sec. 3.1, followed by the presentation of model architecture in Sec. 3.2. Then we introduce the adaptation of model at each session in Sec. 3.3 and memory selection strategy in Sec. 3.4, respectively. Finally, we describe the inference process in Sec. 3.5.

3.1. Problem Setup

Firstly, we introduce the problem setup of general incremental learning. Formally, at session t, the model observes incoming data $\mathbb{D}_t = \{(\boldsymbol{x}_t^i, y_t^i)\}_{i=1}^{N_t}$ where $\boldsymbol{x}_t^i \in \mathbb{X}$ denotes the i-th image, $y_t^i \in \mathbb{C}_t$ is its class label, and N_t is the number of new examples. Here we assume each class has multiple domains representing different variations in the class, e.g. background or style variation. We denote the underlying domain label as z_t^i for the data point $(\boldsymbol{x}_t^i, y_t^i)$ and $z_t^i \in \mathbb{Z}_t^c$ where $\mathbb{Z}_t^c$ is its domain label space. The class label space of the model is all observed classes $\tilde{\mathbb{C}}_t = \cup_{i=1:t}\mathbb{C}_i$, and the domain label space is $\tilde{\mathbb{Z}}_t^{\mathrm{y}} = \cup_{i=1:t}\mathbb{Z}_i^{\mathrm{y}}$ for class y. It is worthy noting that $P(\mathbb{C}_t \cap \tilde{\mathbb{C}}_{t-1} \neq \emptyset) > 0$ and $P(\mathbb{Z}_t^{\mathrm{y}} \cap \tilde{\mathbb{Z}}_{t-1}^{\mathrm{y}} \neq \emptyset) > 0$, which means previously observed categories or domains may repeatedly appear in the subsequent sessions. Given a loss function $\mathcal{L}(y, \hat{y})$ where y is ground truth and $\hat{y}$ is the label prediction, the risk associated with the model $\mathcal{M}$ is defined as follows

$$\mathbb{E}_{\mathrm{y}\sim P(\mathrm{y}|t)}[\mathbb{E}_{\mathrm{z}\sim P(\mathrm{z}|\mathrm{y},t)}[\mathbb{E}_{\mathrm{x}\sim p(\mathrm{x}|\mathrm{y},\mathrm{z},t)}[\mathcal{L}(y,\hat{y})]]] \tag{1}$$

where $P(\mathrm{y}|t)$ denotes the class distribution on $\mathrm{y} \in \tilde{\mathbb{C}}_t$, $P(\mathrm{z}|\mathrm{y},t)$ refers to the class-specific domain distribution on $\mathrm{z} \in \tilde{\mathbb{Z}}_t^{\mathrm{y}}$, and $p(\mathrm{x}|\mathrm{y},\mathrm{z},t)$ is the conditional data generation distribution given class y and domain z. For simplicity, we assume $p(\mathrm{x}|\mathrm{y},\mathrm{z},t)$ does not change over sessions in this work, which often holds in real-world scenarios. At session t, due to memory limitation, model can only keep a small subset of the dataset, which is denoted as memory $\mathbb{M}_{t+1}$. The data available for training at session t is the union of $\mathbb{D}_t$ and $\mathbb{M}_t$, denoted as $\tilde{\mathbb{D}}_t = \mathbb{D}_t \cup \mathbb{M}_t$. For notation clarity, we omit the subscript t in the following subsections.

3.2. Model Architecture

At session t, our model $\mathcal{M}$ consists of a backbone network $\mathcal{F}$ with parameters $\boldsymbol{\theta}$ and a mixture model with parameters $\boldsymbol{\phi}$. We denote the parameters of our entire model as $\Theta = \{\boldsymbol{\theta}, \boldsymbol{\phi}\}$. Concretely, given an image $\boldsymbol{x}$, we extract the feature $\boldsymbol{v} = \mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{x})$. We perform L^2 normalization on the feature $\boldsymbol{v}$ and obtain the unit length feature vector $\tilde{\boldsymbol{v}} = \boldsymbol{v}/\|\boldsymbol{v}\|$, in order to alleviate the imbalance issue (following the practices in [13]). For each class, we model the feature distribution over $\tilde{\mathrm{v}}$ with a mixture model as follows:

$$p(\tilde{\mathrm{v}}|\mathrm{y}) = \sum_{k=1}^{K_{\mathrm{y}}} P(\mathrm{z}=k|\mathrm{y})p(\tilde{\mathrm{v}}|\mathrm{z}=k,\mathrm{y}) \tag{2}$$

where K_{y} is the number of components in the mixture model of class y, and $P(\mathrm{z}|\mathrm{y})$ represents the component proportions, which follows a multinomial distribution. In practice, we set the distribution $P(\mathrm{z}=k|\mathrm{y}) = 1/K_{\mathrm{y}}$ as uniform distribution to mitigate the intra-class domain imbalance issue. Moreover, the probability density function $p(\tilde{\mathrm{v}}|\mathrm{z},\mathrm{y}) = C_d(\kappa)e^{\kappa\tilde{\boldsymbol{\mu}}_{\mathrm{y},\mathrm{z}}^\top\tilde{\mathrm{v}}}$ follows the von Mises-Fisher(vMF) distribution [2], which can be considered as multivariate normal distribution for directional features on the hyper-sphere. Concretely, The concentration parameter $\kappa \geq 0$, $d \geq 2$, and the normalization coefficient $C_d(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2}I_{d/2-1}(\kappa)}$ with $I_r(\cdot)$ represents the modified Bessel function of the first kind and order r. Note that we assume every component shares the same κ for convenience in this work.

3.3. Model Adaptation

We now introduce our incremental learning strategy (see Fig. 1 for an overview). Specifically, to update the model in a session, we first develop an expansion-and-reduction strategy to dynamically determine the number of components in the mixture model. This enables the model to better accommodate new distributions especially when the number of domains in a class changes. Given a model structure, we then introduce an EM-based framework to learn the mixture model by treating its component assignments as latent variables. Concretely, we expand the mixture models at the beginning of each session, and then jointly learn the backbone and mixture models using the EM framework. Finally,

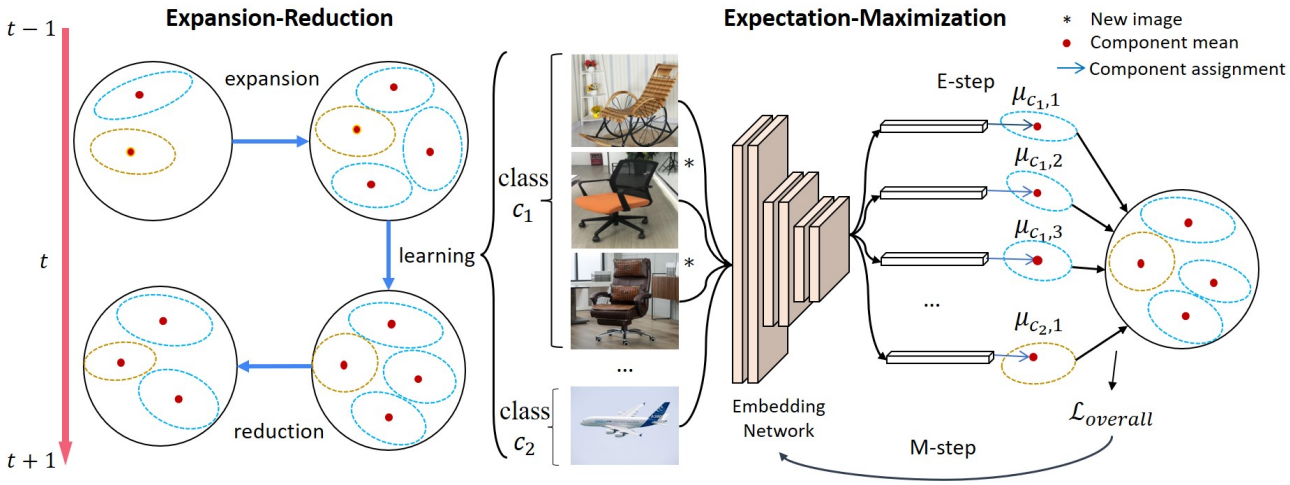


Figure 1. **Method Overview:** At session t, the model starts with the state at last session. It observes incoming images and map them into unit hyper-sphere in the feature space. For the classes in the incoming categories like c_1 in the figure, we first expand the mixture model for each incoming class, followed by the learning with an EM framework. In E-step, we perform a component assignment by choosing the component with the closest mean $\boldsymbol{\mu}$. In M-step, we update both the embedding network and mixture models with overall loss. After the learning, we perform a mixture model reduction to reduce the redundant components for each class.

we perform mixture model reduction to maintain a compact representation.

Expansion-and-Reduction As the number of new domains for each class is unknown, we first increase the number of components and then use a reduction step to obtain the final number of components after the model training. At the beginning of each session t, the model $\mathcal{M}$ is inherited from last session and then expanded with new components and/or mixture models. Concretely, for each class $y \in \mathbb{C}_t$, we add m components to the corresponding mixture model if y has been encountered previously (i.e., $y \in \mathbb{C}_{t-1}$), or create a new mixture model with m components if y is a new class, where the newly-added components are randomly initialized and m is a large number. After expansion, the feature distribution over $\tilde{v}$ becomes

$$p(\tilde{\mathbf{v}}|\mathrm{y}) = \sum_{k=1}^{K_{\mathrm{y}}^{t-1}+m} P(\mathrm{z}=k|\mathrm{y}) p(\tilde{\mathbf{v}}|\mathrm{z}=k,\mathrm{y}) \qquad (3)$$

where K_{y}^{t-1} is the final number of components of class y at session $t-1$ and $P(\mathrm{z}=k|\mathrm{y})$ remains a uniform distribution.

After the model learning (as described below), we perform a mixture model reduction step to avoid over-segmenting the feature space with redundant components or increasing the overall model complexity. Specifically, we group the vMF components and re-represent each group by a new single vMF density. We can view vMF component clustering as a standard data clustering with additional requirement that data points sharing the same original vMF component should end up in the same output component. In practice, we adopt a hierarchical clustering on the original components, which works reasonably well empirically. We treat every component as a distinct cluster and then recursively merge pairs of clusters with distance smaller than a predefined threshold δ within each class. To merge the paired clusters i and j, the mean $\tilde{\boldsymbol{\mu}}$ of the new component's vMF density is updated as follows

$$\boldsymbol{\mu} = \frac{1}{n_i+n_j}\Big(\sum_{l=1}^{n_i}\tilde{v}_l^i + \sum_{q=1}^{n_j}\tilde{v}_q^j\Big), \quad \tilde{\boldsymbol{\mu}} = \frac{\boldsymbol{\mu}}{||\boldsymbol{\mu}||} \qquad (4)$$

where n_i, n_j are the number of examples in the cluster i and j, respectively. $\tilde{v}_l^i$ is the normalized feature of l-th example in the i-th cluster.

Model Learning To learn the domain-aware categorical representation, we develop an EM learning algorithm to train the entire model $\mathcal{M}$ with a limited memory, in which the component assignments z of image $\boldsymbol{x}$ are treated as latent variables. The log-likelihood of a given data $\boldsymbol{x}$ can be written as

$$\log P(\mathrm{y}|\boldsymbol{x},\Theta) \geq \mathbb{E}_{Q(\mathrm{z})}\left[\log \frac{P(\mathrm{z},\mathrm{y}|\boldsymbol{x};\Theta)}{Q(\mathrm{z})}\right] \qquad (5)$$

where the right hand $\mathbb{E}_{Q(\mathrm{z})}\left[\log\left(P(\mathrm{z},\mathrm{y}|\boldsymbol{x};\Theta)/Q(\mathrm{z})\right)\right]$ is the evidence lower bound (ELBO).

E-step In the E-step, we compute a new estimate of the component assignments using the learned parameters Θ' from the last M-step, represented as $Q^*(\mathrm{z})$. Concretely, $Q^*(\mathrm{z}) = [\![\mathrm{z}=\hat{z}]\!]$ where $[\![\cdot]\!]$ is the indicator function and

$\hat{z}$ is defined as

$$\begin{aligned}\hat{z} &= \arg\max_k P(\mathrm{z}=k|\mathbf{x},\mathrm{y};\Theta') \\ &= \arg\max_k \frac{p(\boldsymbol{x}|\mathrm{z}=k,\mathrm{y};\Theta')P(\mathrm{z}=k|\mathrm{y})}{\sum_{l=1}^{K_\mathrm{y}} p(\boldsymbol{x}|\mathrm{z}=l,\mathrm{y};\Theta')P(\mathrm{z}=l|\mathrm{y})} \\ &= \arg\max_k \tilde{\boldsymbol{\mu}}_{\mathrm{y},k}^\top \tilde{\boldsymbol{v}}\end{aligned} \tag{6}$$

where $\tilde{\boldsymbol{\mu}}_{\mathrm{y},k}$ is the mean of the k-th component for class y. In other words, we update the component assignment of image $\boldsymbol{x}$ within class y by taking the component with the closest mean feature $\tilde{\boldsymbol{\mu}}$ on the hyper-sphere. In contrast to the standard E-step using the posterior $P(\mathrm{z}|\mathrm{y},\boldsymbol{x};\Theta')$ as $Q(\mathrm{z})$, our method can be viewed as a hard-EM approximation.

M-step In the M-step, we maximize the ELBO by mini-batch SGD based on the component assignments obtained in the E-step. This optimization problem can be rewritten as follows

$$\begin{aligned}\min_\Theta \mathbb{E}_{(\mathrm{x},\mathrm{y})\sim p(\mathrm{x},\mathrm{y})}[\, KL\,[Q^*(\mathrm{z})||P(\mathrm{z}|\mathrm{y},\boldsymbol{x};\Theta)] \\ -\log P(\mathrm{y}|\boldsymbol{x};\Theta)],\end{aligned} \tag{7}$$

which enforces the model to learn the classification at both inter- and intra-class level. Moreover, given dataset $\tilde{\mathbb{D}}$, this expectation can be rewritten as follows

$$\mathcal{L}_{\text{clf}} = \mathcal{L}_{\text{clf}}^{\text{inter}} + \lambda \mathcal{L}_{\text{clf}}^{\text{intra}} \tag{8}$$

$$\begin{aligned}&= -\frac{1}{|\tilde{\mathbb{D}}|}\sum_{i=1}^{|\tilde{\mathbb{D}}|}\big(\log P(\mathrm{y}=y_i|\boldsymbol{x}_i;\Theta) \\ &+ \lambda \log P(\mathrm{z}=\hat{z}_i|\boldsymbol{x}_i,y_i;\Theta)\big)\end{aligned} \tag{9}$$

where $\mathcal{L}_{\text{clf}}^{\text{inter}}$ is the inter-class classification loss, $\mathcal{L}_{\text{clf}}^{\text{intra}}$ refers to the intra-class classification loss, λ is the hyper-parameter to balance these two losses, and the posterior of component assignment is computed as follows

$$P(\mathrm{z}=k|\mathrm{y},\boldsymbol{x};\Theta) = \frac{e^{\kappa\tilde{\boldsymbol{\mu}}_{\mathrm{y},k}^\top\tilde{\boldsymbol{v}}}}{\sum_{l=1}^{K_\mathrm{y}} e^{\kappa\tilde{\boldsymbol{\mu}}_{\mathrm{y},l}^\top\tilde{\boldsymbol{v}}}}. \tag{10}$$

It is worth noting that the component assignment z depends on the prediction of label y. Consequently, we design a schedule of λ which starts with zero and gradually grows over iterations as the quality of label prediction y improves. Moreover, to maintain inter-class class balance, we assume class distribution $P(\mathrm{y})$ follows uniform distribution, and then the prediction probability is given by

$$\begin{aligned}P(\mathrm{y}=c|\boldsymbol{x};\Theta) &= \frac{p(\boldsymbol{x}|\mathrm{y}=c)P(\mathrm{y}=c)}{\sum_{m=1}^{|\tilde{\mathbb{C}}_t|}\sum_{n=1}^{K_m} p(\boldsymbol{x}|\mathrm{z}=n,\mathrm{y}=m)} \\ &= \frac{\frac{1}{K_c}\sum_{i=1}^{K_c} e^{\kappa\tilde{\boldsymbol{\mu}}_{c,i}^\top\tilde{\boldsymbol{v}}}}{\sum_{m=1}^{|\tilde{\mathbb{C}}_t|}\sum_{n=1}^{K_m}\frac{1}{K_m}e^{\kappa\tilde{\boldsymbol{\mu}}_{m,n}^\top\tilde{\boldsymbol{v}}}}\end{aligned} \tag{11}$$

To prevent intra-class forgetting and preserve the learned intra-class structure, we employ a knowledge distillation loss within each class, which is defined as

$$\begin{aligned}\mathcal{L}_{\text{dis}} = \frac{1}{|\tilde{\mathbb{D}}|}\sum_{i=1}^{|\tilde{\mathbb{D}}|}\frac{1}{|\tilde{\mathbb{C}}|}\sum_{c=1}^{|\tilde{\mathbb{C}}|} KL(P(\mathrm{z}|\mathrm{y}=c,\boldsymbol{x}_i;\Theta)|| \\ P(\mathrm{z}|\mathrm{y}=c,\boldsymbol{x}_i;\Theta_{\text{old}}))\end{aligned} \tag{12}$$

where Θ_{old} is the parameters of learned model from previous session. Furthermore, we introduce a component regularization loss [26] on the mixture model of each class to learn a tight cluster, which is computed as below

$$\mathcal{L}_{\text{reg}} = -\frac{1}{|\tilde{\mathbb{C}}|}\sum_{y\in\tilde{\mathbb{C}}}\sum_{i=1}^{K_y}\sum_{j=i+1}^{K_y}\frac{1}{K_y*(K_y-1)}\tilde{\boldsymbol{\mu}}_{y,i}^\top\tilde{\boldsymbol{\mu}}_{y,j} \tag{13}$$

Since we uses many components at the beginning of each session to expand current model, this loss can prevent the model from over-segmenting the feature space.

Finally, the overall loss function in the M-step is a linear combination of those three losses, defined as follows

$$\mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{clf}} + \beta\mathcal{L}_{\text{dis}} + \eta\mathcal{L}_{\text{reg}} \tag{14}$$

where β, η are the loss weighting coefficients.

3.4. Memory Selection

We introduce a bi-level balanced strategy to build the data memory $\mathbb{M}_t$, which maintains a class-balanced and domain-balanced replay dataset in each session. Concretely, we first assign $m = \mathcal{B}/|\tilde{\mathbb{C}}|$ exemplars for each class to ensure the inter-class class balance, where $\mathcal{B}$ is the maximum number of exemplars that can be saved. Subsequently, given the mixture model for each class c, we select $m/|K_c|$ samples from each component of class c uniformly to achieve an intra-class domain balance. We note that such a strategy aims to achieve better average performance.

3.5. Model Inference

Given a new image $\mathbf{x}$, the model inference computes its normalized feature $\tilde{\boldsymbol{v}}$ and predicts the label $\hat{y}$ by taking the class of the closest component in the feature space, which can be written as

$$\hat{y} = \arg\max_c \max_k \tilde{\boldsymbol{\mu}}_{c,k}^\top\tilde{\boldsymbol{v}} \tag{15}$$

4. Experiments

We conduct a series of experiments to verify the effectiveness of our method. In this section, we first introduce the experiment setup including the benchmarks, types of distribution shifts and the comparison methods in Sec. 4.1, followed by the implementation details in Sec. 4.2. Then we show our experimental results in Sec. 4.3. In the end, we demonstrate the analysis of our method to provide more insights in Sec. 4.4.

Table 1. **Results:** Average incremental accuracy(%) over sessions on iCIFAR-20, iDomainNet, and iDigits with three representative splits. 's' represents the number of sessions in the split. E.g. 5s means this split has 5 sessions.

Methods	iCIFAR-20			iDomainNet			iDigits		
	NC (5s)	ND (5s)	NCD (10s)	NC (10s)	ND (6s)	NCD (10s)	NC (5s)	ND (4s)	NCD (10s)
Replay	74.22	72.47	67.56	45.73	47.40	42.74	83.22	92.75	80.24
iCaRL [33]	78.98	73.06	70.90	51.64	48.40	44.60	89.03	93.26	85.12
EE2L [13]	78.50	73.86	70.52	52.03	48.03	43.54	89.97	93.91	85.46
Meta-DR [33]	73.22	71.09	66.65	46.40	48.73	44.15	89.89	94.00	86.31
UCIR [13]	78.19	76.01	72.54	50.17	49.25	44.53	89.41	94.17	86.29
UCIR w/ ours	78.82	78.08	75.62	50.52	52.85	49.81	90.51	95.50	**90.42**
GeoDL [27]	78.92	76.43	72.96	51.64	49.33	45.12	89.72	93.98	86.24
GeoDL w/ ours	79.49	79.40	76.19	51.81	53.32	51.20	89.86	94.80	89.82
DER [37]	82.17	74.87	74.56	66.58	46.76	50.00	89.01	93.62	84.80
DER w/ ours	**82.52**	**84.03**	**82.11**	**66.85**	**61.05**	**57.07**	**91.32**	**97.07**	88.65

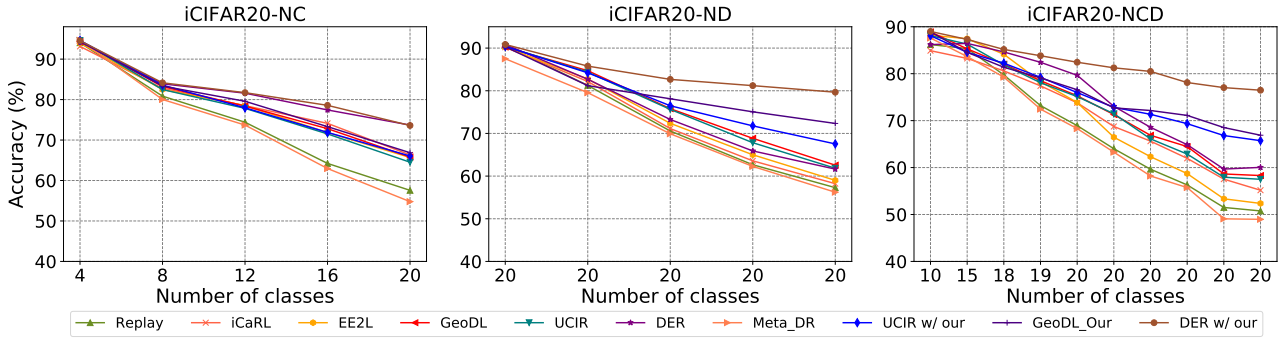


Figure 2. **Performances w.r.t sessions** on iCIFAR-20 benchmark with three splits

4.1. Experiment Setup

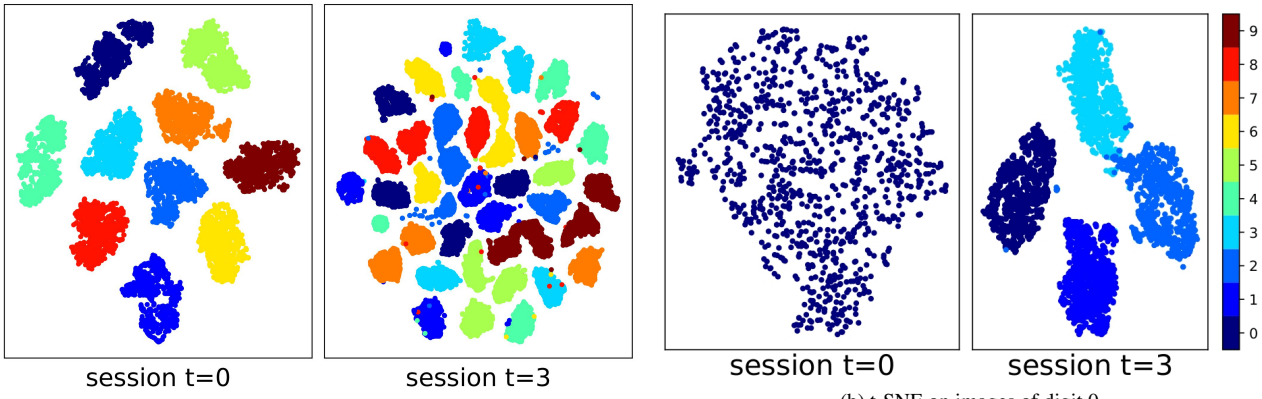
We conduct experiments on three benchmarks, including iDigits, iDomainNet and iCIFAR-20:

- *iDigits:* We follow [33] to construct a digit recognition benchmark, which includes four datasets: MNIST [39], SVHN [19], MNIST-M [11] and SYN [11]. Each dataset represents a distinct domain.
- *iDomainNet:* It is constructed from DomainNet [21], a well-known dataset for domain adaptation. It contains six domains, which are *Clipart*, *Infograph*, *Painting*, *Quickdraw*, *Real* and *Sketch*. Each domain contains 345 categories of common objects. Because some domains in these classes contains only few images($\sim$ 10), we select the top 100 classes with most images, which contains 132,673 training data in total. The smallest domain in these 100 classes has 52 images.
- *iCIFAR-20:* It is based on CIFAR-100 [16], which has 20 super classes and 5 subclasses for each super class. These subclasses are considered to be different domains of the same class and model needs to predict super class labels in the recognition task.

Each domain of the iCIFAR-20 has the same number of training images. By contrast, for iDomainNet and iDigits, each domain has a different number of training images. We evaluate these methods using three representative splits to simulate different scenarios for every benchmark, in which the distributions of classes and domains shift:

- *New Class(NC)*: Incoming data contains images from new categories only.
- *New Domain(ND)*: Incoming data contains images from new domains only.
- *New Class and Domain(NCD)*: Incoming data contains images from new categories or new domains.

For the NC split, we construct iCIFAR20-NC and iDigits-NC by splitting iCIFAR-20 and iDigits into 5 sessions with 4 and 2 classes per session, respectively. Moreover, the model is trained in batches of 60 classes with 10 sessions in total on iDomainNet-NC. For the ND split, every session has all the classes and each class has one incoming domain, where iCIFAR-20, iDigits and iDomainNet are split into 5, 4 and 6 sessions, respectively. For the NCD



(a) t-SNE on images across classes

(b) t-SNE on images of digit 0

Figure 3. **t-SNE Visualization** on all data so far seen of DER w/ ours across sessions for iDigits NC split. Different colors represent class label for the left and domain label for the right.

Table 2. **Ablation Study:** Contribution of each component evaluated on iCIFAR-20.

Components				NC		ND		NCD	
Mixture model	Expansion-Reduction	component regularization Loss	Bi-level Memory	Final(%)	Avg(%)	Final(%)	Avg(%)	Final(%)	Avg(%)
✗	✗	✗	✗	72.06	82.07	67.35	77.18	54.32	70.64
✓	✗	✗	✗	72.39	81.38	68.98	80.39	68.47	78.71
✓	✓	✗	✗	73.95	82.32	70.32	81.28	69.86	80.26
✓	✓	✓	✗	74.14	82.46	71.92	82.04	71.56	80.97
✓	✓	✓	✓	**74.49**	**82.77**	**80.13**	**84.11**	**73.70**	**82.17**

split, we divide all domains in the dataset into ten sessions for each of the three datasets. For more information about these splits, please refer to the appendix.

We adopt the Replay, iCaRL [25], EE2L [4], UCIR [13], GeoDL [27], DER [37] and Meta-DR [33] as the comparison methods. Here the Replay refers that the model is fine-tuned using both the memory and incoming data. It is noteworthy that iCaRL, EE2L, UCIR, GeoDL and DER are all designed to address the class incremental learning problem, whereas Meta-DR are proposed to resolve domain incremental learning problem. In contrast, our method mainly learns the intra-class structure and can be utilized by any class incremental learning method to address the stability-plasticity dilemma at the inter- and intra-class level. Consequently, we combine our method with three existing incremental learning approaches, UCIR, GeoDL and DER, in our experimental evaluation.

4.2. Implementation Details

All these methods are implemented with PyTorch [20]. We resize the images in iCIFAR-20 and iDigits to 32x32 and the images in iDomainNet to 112x112. For the iCIFAR-20 and iDomainNet benchmarks, we follow DER [37] and adopt the standard ResNet18 [12] architecture as the feature extractor. We use SGD optimizer to train the network with 200 epochs in total for each session. Learning rate starts with 0.1 and is reduced by 0.1 at 80 and 120 epoch. We set the fixed memory size for these two benchmarks as 2000 instances. For the iDigits benchmarks, we choose the modified 32-layer ResNet used in [13, 25] because it is a simple dataset and large networks can be easily overfitted. We train the methods with SGD optimizer for 70 epochs for each session, beginning with learning rate 0.1, which is reduced by 0.1 at 48 and 63 epochs. We set the fixed memory size as 500 for iDigits. In addition, the batch size is selected as 128 for iCIFAR-20 and iDigits, and 256 for iDomainNet. Weight decay is 0.0005 for all benchmarks. The distillation loss coefficient $\beta = 1$. The coefficient λ in Eq. (8) linearly increases from 0 to 0.1 for the first 10 epochs and then is fixed at 0.1. The coefficient η of the regularization loss is set as 0.1. For the expansion of mixture model, the number of components m to add for each class is set as 30 for all benchmarks. For the reduction of mixture model, the threshold δ is chosen as 0.7 for all benchmarks. We run the E-step to update the component assignments at the beginning of each epoch. Following [9], these hyper-parameters are tuned on a val set built from the original training data.

4.3. Experimental Results

Tab. 1 summarizes the average accuracy over sessions of different methods. We combine our methods with three different methods - UCIR, GeoDL and DER, since they performs better than other baselines on at least one split.

On all kinds of data distribution shifts in each benchmark, our method consistently improve the performance of these three methods, which demonstrates its effectiveness.

Table 3. **Sensitive Study** Influence of threshold δ in mixture model reduction on our method for iCIFAR-20 NCD split.

Threshold	$\delta = 0.5$		$\delta = 0.55$		$\delta = 0.6$		$\delta = 0.65$		$\delta = 0.7$		$\delta = 0.75$		$\delta = 0.8$	
	Final	Avg	Final	Avg	Final	Avg	Final	Avg	Final	Avg	Final	Avg	Final	Avg
DER w/ ours	74.08	80.70	75.24	81.19	75.47	81.21	75.74	81.26	76.5	82.17	77.44	82.05	76.49	82.01

Particularly, we can see that our method solves the performance bottleneck of DER on the ND split and DER w/ ours consistently achieves the highest average accuracy in the majority of cases, e.g. 84.11% on iCIFAR20-ND. Besides, UCIR w/ ours performs best on iDigits-NCD split with 90.42% accuracy. As demonstrated in Fig. 2, we observe that our method consistently performs better than other method at each session for different splits. Specifically, the final session accuracy is boosted from 60.04% to $76.40\%(+\mathbf{16.36}\%)$ on iCIFAR20-NCD split by incorporating our method into DER. Additionally, we find that integrating our method can significantly increases the performance of existing class incremental learning methods such as UCIR and DER on ND and NCD splits, while maintaining comparable performance on the NC split.

With regard to the ND split, UCIR and GeoDL perform better than other baselines, which is because their distillation is based on features. However, iCaRL and EE2L, which use logit-based distillation, perform worse because they penalize the change of predictions for data of old classes from new domains. As for DER, the old feature extractors cannot recognize data of old classes in new domains, which affects its prediction. For the NCD split, DER is the best for iCFIAR-20 and iDomainNet, since it performs much better on the NC split. Furthermore, Meta-DR performs well on the iDomainNet and iDigits but does not perform well on iCIFAR-20. This is because one domain in iCIFAR-20 represents one semantic subclass and domain randomization cannot solve the this domain gap.

4.4. Analysis

Ablation Study Tab. 2 summarizes the results of our ablative experiments on iCIFAR-20, starting with DER. We can find that our method achieve 8.07% improvement on average incremental accuracy for the NCD split by mixture model. We also show that the performance of the model is consistently improved over three different types of distribution shifts with our expansion and reduction strategy, especially achieving 0.89% gain on the ND split. Furthermore, it shows that we can obtain 0.76% improvement after adding component regularization loss. Finally, our method further improve the accuracy by 2.07% for the ND split and 1.20% for the NCD split, after adding the bi-level memory sampling approach.

Visualization We utilize t-SNE [31] to visualize the feature embeddings on the iDigits ND split at different sessions, shownd in Fig. 3. As the number of sessions increases, each cluster mainly contains only examples from the domain, implying a high degree of purity for each cluster. It is noteworthy that each class in this split has four domains in the last session ($t = 3$) and our method can separate most classes into four groups. Furthermore, we take images of one class for further analysis, shown on the right side of Fig. 3. It reveals our method is able to assign most instances to their respective domain labels, demonstrating the effectiveness of latent variable estimation.

Sensitive Study We conduct sensitive study on the influence of threshold δ in the mixture model reduction step, as shown in Tab. 3, which shows our method is robust to small variations of the threshold. We also study the influence of different memory sizes and number of newly added components m, which are shown in the appendix.

5. Conclusion and Discussion

In this work, we propose and formulate the offline general incremental learning problem, which has many real-world applications. To address the challenges, we introduce a domain-aware learning framework. Concretely, we propose a flexible class representation based on the mixture model to solve the stability-plasticity dilemma, which is learned by an expansion-reduction strategy and the EM algorithm. Furthermore, we also develop a bi-level balanced memory selection strategy based on the learned mixture model for the imbalance challenge. We conduct exhaustive experiments on three benchmarks to validate the effectiveness of our method. The experimental results demonstrate that our method consistently outperforms than other methods on three representative splits for each benchmark. Furthermore, it is meaningful to apply our method to other vision tasks like video classification [18] and semantic segmentation [8] as future work.

Limitations and Negative Impact Our method is designed for allowing access to limited old examples, and cannot be used to the setting without memory. As our method continuously updates the model with incoming data, it can be leveraged by malicious applications to upgrade its model with new data.

References

[1] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[2] Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *J. Mach. Learn. Res.*, 2005. 3

[3] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in neural information processing systems (NeurIPS)*, 2020. 1, 2

[4] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2, 3, 7

[5] Stephan K. Chalup. Incremental learning in biological and machine learning systems. *International Journal of Neural Systems*, 2002. 1

[6] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2

[7] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence(TPAMI)*, 2021. 1

[8] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. PLOP: learning without forgetting for continual semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, 2021. 8

[9] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2, 7

[10] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. In *International Conference on Learning Representations(ICLR)*, 2019. 2

[11] Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning (ICML)*, 2015. 6

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016. 7

[13] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2019. 2, 3, 6, 7

[14] Dapeng Hu, Shipeng Yan, Qizhengqiu Lu, Lanqing HONG, Hailin Hu, Yifan Zhang, Zhenguo Li, Xinchao Wang, and Jiashi Feng. How well does self-supervised pre-training perform with streaming data? In *International Conference on Learning Representations(ICLR)*, 2022. 2

[15] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences (PNAS)*, 2017. 2

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 6

[17] Jogendra Nath Kundu, Rahul Mysore Venkatesh, Naveen Venkat, Ambareesh Revanur, and R. Venkatesh Babu. Class-incremental domain adaptation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[18] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *1st Annual Conference on Robot Learning (CoRL)*, 2017. 1, 2, 8

[19] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 6

[20] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 7

[21] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *International Conference on Computer Vision (ICCV)*, 2019. 6

[22] John M. Pierre. Incremental lifelong deep learning for autonomous vehicles. In Wei-Bin Zhang, Alexandre M. Bayen, Javier J. Sánchez Medina, and Matthew J. Barth, editors, *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018. 2

[23] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[24] Jathushan Rajasegaran, Munawar Hayat, Salman H. Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[25] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017. 1, 7

[26] Sohil Atul Shah and Vladlen Koltun. Robust continuous clustering. *Proceedings of the National Academy of Sciences*, 2017. 5

[27] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2021. 2, 6, 7

[28] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, 2021. 2

[29] Shixiang Tang, Peng Su, Dapeng Chen, and Wanli Ouyang. Gradient regularized contrastive learning for continual domain adaptation. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, (AAAI)*, 2021. 1, 2

[30] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, and Yihong Gong. Bi-objective continual learning: Learning 'new' while consolidating 'known'. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, (AAAI)*, 2020. 1, 2

[31] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *J. Mach. Learn. Res.*, 2008. 8

[32] Vinay Kumar Verma, Kevin J. Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient feature transformations for discriminative and generative continual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2021. 2

[33] Riccardo Volpi, Diane Larlus, and Grégory Rogez. Continual adaptation of visual representations via domain randomization and meta-learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2021. 1, 2, 6, 7

[34] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2019. 3

[35] Tao Xiaoyu, Chang Xinyuan, Hong Xiaopeng, Wei Xing, and Gong Yihong. Topology-preserving class-incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[36] Mengya Xu, Mobarakol Islam, Chwee Ming Lim, and Hongliang Ren. Class-incremental domain adaptation with smoothing and calibration for surgical report generation. In *Medical Image Computing and Computer Assisted Intervention MICCAI*, 2021. 2

[37] Shipeng Yan, Jiangwei Xie, and Xuming He. DER: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, 2021. 2, 6, 7

[38] Shipeng Yan, Jiale Zhou, Jiangwei Xie, Songyang Zhang, and Xuming He. An em framework for online incremental learning of semantic segmentation. In *Proceedings of the 29th ACM International Conference on Multimedia(ACM MM)*, 2021. 2

[39] LeCun Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 6

[40] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2020. 2

[41] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2020. 3

[42] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2021. 2