GENERAL INCREMENTAL LEARNING WITH DOMAIN-AWARE CATEGORICAL REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Incremental learning is necessary to achieve human-like intelligence system since the model must continuously accumulate knowledge in response to real-world streaming data. 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 vary over sequential sessions. Apart from the challenges discussed extensively in the class incremental learning, the problem also faces an intra-class stability-plasticity dilemma and intra-class domain imbalance issue. To address above issues, we develop a novel domain-aware learning framework. Concretely, we introduce a flexible class representation based on the von Mises-Fisher mixture model to capture the intra-class structure as well as a bi-level balanced memory to deal with data imbalances within and between classes. In particular, we build a mixture model on deep features of each class and devise an expansion-and-reduction strategy for dynamically increasing the number of components according to the concept complexity. Combining with distillation loss, our design encourages the model to learn a domain-ware representation, which aids in achieving inter- and intra-class stability-plasticity trade-off. We conduct exhaustive experiments on three benchmarks, each with three representative splits. The results show that our method consistently outperforms other methods with a significant margin, suggesting its superiority.

1 INTRODUCTION

In order to build a human-like intelligent system, it is indispensable to continuously accumulate knowledge over time for adapting to novel environments, known as incremental learning (Chalup, 2002). To cope with real-world circumstances, we consider a general and yet under-explored incremental learning problem, where the class distribution and class-specific domain distributions of the incoming data continuously change across sequential learning sessions. This requires a model to recognize not only novel concepts but also new variants of previously-learned concepts. Take the continuously evolving object grasping systems (Chen & Wen, 2020) as an example. We suppose a robot can pick up certain types of cups. In a real-world scenario, it may need to grasp new objects such as boxes, and even also pick up cups of new types, with new backgrounds or styles in future.

Stability-plasticity dilemma (Grossberg, 2013) and data imbalance problem are two critical challenges in incremental learning. The former means model usually forgets learned concepts when assimilating novel knowledge, and the latter is caused by a limited memory size in a typical system. While the class incremental learning have been intensively explored, our general incremental learning additionally faces an *intra-class stability-plasticity dilemma* which refers to the trade-off between adapting novel examples and preserving current knowledge about the class, and an *intra-class domain imbalance* problem, where the model is biased toward incoming domains. The intra-class problem are particularly challenging since the domain labels are usually unknown in practice.

The majority of existing research (Rebuffi et al., 2017; Delange et al., 2021) focuses on improving the inter-class trade-off between stability and plasticity. While some recent efforts (Tao et al., 2020; Volpi et al., 2021) attempt to tackle the intra-class stability-plasticity dilemma, they typically ignore the intra-class structure of data distribution. In particular, these methods utilize the same discriminative features for the data from both incoming and existing domains of the same class, which makes it difficult to learn new domains without interference with previously-learned representation of the target 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 challenges in an 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 particular, we build a vMF mixture model on deep features of each class and design an expansion-and-reduction strategy to dynamically increase the number of its components in new sessions. Combining with inter- and intra-class forgetting resistence strategies like distillation, our design encourages the model to learn a domain-aware representation, which in turn help achieve 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 develop an iterative procedure for model update at each session. Specifically, when new data comes, we first inherit the learned model from the 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 in which we treat the cluster assignments of input data as latent variables. We incorporate inter-class strategies overcoming forgetting like Hou et al. (2019) and adopt intra-class knowledge distillation for alleviating inter- and intra-class catastrophic forgetting, respectively. After the model update, we perform a mixture reduction step based on hierarchical clustering to maintain a compact class representation. During inference, we first extract input features via the backbone network and then infer its mixture assignment in each class, followed by taking the class with the maximal component probability as its prediction.

We validate our approach by extensively comparing our method with different incremental learning methods using three benchmarks: iDigits, iCIFAR100, iDomainNet. For each benchmark, we conduct experiments on splits with varying class and domain distributions over time. The empirical results and ablation study demonstrate that our strategy consistently outperforms other approaches across all benchmarks.

To conclude, the main contributions of our work can be summarized as follows:

1. We formulate a novel general offline incremental learning problem in which both class distribution and intra-class domain distribution continuously change over time.
2. We propose a vMF mixture model to learn a domain-aware representation to tackle the general stability-plasticity dilemma and develop a bi-level balanced memory strategy to mitigate both the inter- and intra-class data imbalance problem.
3. Extensive experiments on three benchmarks show that our strategy consistently outperforms existing methods by a sizable margin.

2 Related Works

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

Problem Settings Van de Ven & Tolias (2018) proposes three incremental learning scenarios including task incremental learning, domain incremental learning and class incremental learning. Task incremental learning requires the task identity during inference, which is often not practical in real applications. Domain incremental learning (Tao et al., 2020; Volpi et al., 2021) describes the situation that the domain distribution changes continuously while the class space remains unchanged. Class incremental learning (Wu et al., 2019; Zhao et al., 2020) means that there are new classes coming at each session while the class conditional domain distribution is invariant. In this work, we focus on the general incremental learning, where domain and class incremental learning are the special cases of our discussed problem. There are also some works (Aljundi et al., 2019; Buzzega et al., 2020) studying the online incremental learning, where the model constantly learns from an online stream of data. In contrast, we study the offline learning setting which allows multiple pass of incoming data at each session. This paradigm is important in many real world applications (Pierre, 2018) in which offline learning usually performs better than online learning. Although the works (Lomonaco & Maltoni, 2017; Buzzega et al., 2020) handle both shifts of class and domain distribution in online scenario, to the best of our knowledge, we are the first to tackle the general incremental learning when allowing offline training at different sessions.

Stability-Plasticity Dilemma To alleviate the forgetting of representation, current methods can be mainly grouped into three categories. *Regularization*-based methods (Kirkpatrick et al., 2017; Chaudhry et al., 2018) add regularization on the parameters directly to mitigate dramatic changes of the important parameters. EWC (Kirkpatrick et al., 2017) regularizes the parameters by penalizing the changes on important parameters where the importance scores are calculated via the diagonal of the Fisher information matrix. UCB (Ebrahimi et al., 2019) adjust learning rates for each parameter according to its uncertainty in the neural network. *Distillation*-based (Castro et al., 2018; Hou et al., 2019; Douillard et al., 2020; Xiaoyu et al., 2020; Tao et al., 2020) methods introduces knowledge distillation to preserve the representation by penalizing the difference between outputs of last model and current model. To restrict the change of model, PODNet (Douillard et al., 2020) adopts a spatial-based distillation loss applied on both the final feature and the intermediate features. GeoDL (Simon et al., 2021a) introduces knowledge distillation between the model responses of previous and current session, which measures the dissimilarity along the geodesic connecting the projected low-dimensional manifolds. *Parameter isolation* methods allocate isolate parameters for each session and keep the parameters related to previous sessions fixed to avoid forgetting. DER (Yan et al., 2021) propose a dynamically expandable representation where the model freezes previously learned representation and augment it with novel parameterized representation by creating a new feature extractor. However, they focus on the domain invariant representation learning, 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 adopt a two-stage learning strategy, which adjusts the classifier to eliminate the bias after the learning of representation. EEIL(Castro et al., 2018) finetunes the classifier by constructing a class-balanced subset from training data. BiC (Wu et al., 2019) adds a bias correction layer after the last fully connected layer to reduce the bias, and train the layer on a small separate validation set. WA (Zhao et al., 2020) corrects the weights for the biased classifier by matching the norms of the weight vectors for new classes to those for old classes weight vectors. UCIR (Hou et al., 2019) adopts cosine normalization and inter-class separation loss to mitigate the inter-class imbalance. We note that these works mainly focus on solving the inter-class imbalance problem. Moreover, they can not deal 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 an EM framework.

3 METHOD

In this work, our goal is to address the general incremental learning problem. To this end, we propose an approach to learn a domain-aware representation to achieve a better stability-plasticity trade-off at both intra- and inter-class level. Concretely, we introduce a mixture model for each class to model the intra-class structure and learn the mixture model with the EM 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. 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. In real world, each class has multiple domains representing different variants of the class, e.g. backgrounds, styles, shape variations, etc. We denote the underlying domain label as z_t^i for the data point $(\boldsymbol{x}_t^i, y_t^i)$ where $z_t^i \in \mathbb{Z}_t^c$. The 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 the loss function $\mathcal{L}(y, y')$ 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 over $\mathrm{y} \in \tilde{\mathbb{C}}_t$, $P(\mathrm{z}|\mathrm{y}, t)$ refers to the class-specific domain distribution over $\mathrm{z} \in \tilde{\mathbb{Z}}_t^{\mathrm{y}}$, 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 $p(\mathrm{x}|\mathrm{y}, \mathrm{z}, t)$ is not changed over sessions in this work, which also often holds in real-world scenarios. Moreover, the class distribution $P(\mathrm{y}|t)$ and the class-specific domain distribution $P(\mathrm{z}|\mathrm{x}, t)$ takes the uniform distribution during test. At session t, due to the storage 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 $\tilde{\boldsymbol{v}} = \boldsymbol{v}/\|\boldsymbol{v}\|$, following the practices used in (Hou et al., 2019) in order to alleviate the influence of imbalance. For each class, we model the feature distribution over $\tilde{\mathbf{v}}$ with a mixture model as follows:

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

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 $P(\mathrm{z} = k|\mathrm{y}) = \frac{1}{K_{\mathrm{y}}}$ as uniform distribution to mitigate the intra-class domain imbalance issue. Moreover, $p(\tilde{\mathbf{v}}|\mathrm{z}, \mathrm{y})$ follows the von Mises-Fisher(vMF) distribution (Banerjee et al., 2005), which can be considered as multivariate normal distribution for directional features on the hyper-sphere. The probability density function within each component is given by $p(\tilde{\mathbf{v}}|\mathrm{z}, \mathrm{y}) = C_d(\kappa)e^{\kappa\tilde{\boldsymbol{\mu}}_{\mathrm{y},\mathrm{z}}^{\top}\tilde{\mathbf{v}}}$, where 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

For the incremental learning, the main challenge in building our mixture model is to identify whether incoming data belong to existing components or they requires creating new sub-class clusters. To ease the model learning, we adopt an expansion-and-reduction strategy to dynamically determine the number of components and introduce an EM framework to learn the model, treating cluster assignments as latent variables. Concretely, we expand the mixture models for initialization at each session, and then jointly learn the backbone and mixture models within the EM framework. Finally, we perform mixture model reduction to maintain a compact representation. The overview of proposed method is shown in Fig. 1.

Expansion-and-Reduction At the beginning of each session t, the model $\mathcal{M}$ is inherited from the last session and 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 where the newly-added components are randomly initialized.

After the model learning (describe below), we perform a mixture model reduction step to prevent the model complexity from explosive growth. We group the vMF components and re-represent each group by a new single vMF density, which results in a compact representation while still maintaining the original component structure. We can view vMF component clustering as a standard data

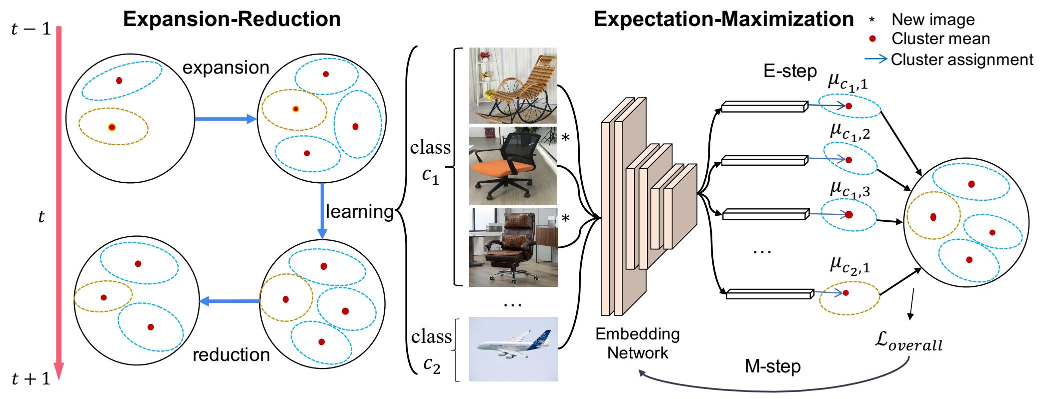


Figure 1: **Method Overview:** At session t, the model starts with the state at last session. It observes incoming images and map them into unit hyper-sphere in the feature space. For the classes in the incoming categories like c_1 in the figure, we first expand the mixture model for each incoming class, followed by the learning with an EM framework. In E-step, we perform a cluster 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.

clustering with additional requirement that data points sharing the same original vMF component should end up in the same output component. In this work, we adopt a hierarchical clustering by regarding original component as data points, which works reasonably well in practice. Concretely, we recursively merge pairs of clusters with linkage distance smaller than a predefined threshold δ.

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

$$\begin{aligned}\log P(\mathrm{y}|\boldsymbol{x},\Theta) &= \sum_{k=1}^{K_\mathrm{y}} Q(\mathrm{z}=k)\log\frac{P(\mathrm{z}=k,\mathrm{y}|\boldsymbol{x};\Theta)}{Q(\mathrm{z}=k)}d\mathrm{z} + KL\left[Q(\mathrm{z}=k)||P(\mathrm{z}=k|\mathrm{y},\boldsymbol{x};\Theta)\right] \\ &\geq \mathbb{E}_{Q(\mathrm{z})}\left[\log\frac{P(\mathrm{z},\mathrm{y}|\boldsymbol{x};\Theta)}{Q(\mathrm{z})}\right]\end{aligned} \tag{3}$$

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

E-STEP In E-step, we compute a new estimate of the cluster assignment using the learned parameters Θ' from last E-step. It is worth noting that the equality in Eq. 3 holds when $Q(\mathrm{z})$ is equal to the posterior of cluster assignment $P(\mathrm{z}|\mathrm{y},\boldsymbol{x};\Theta')$. In this work, we adopt a hard-EM approximation. Concretely, $Q^*(\mathrm{z}) = P(\mathrm{z}|\mathrm{y},\boldsymbol{x};\Theta')$ takes one-hot distribution and $P(\mathrm{z}=\hat{z}|\mathrm{y},\boldsymbol{x};\Theta')=1$ with

$$\begin{aligned}\hat{z} &= \arg\max_k P(\mathrm{z}=k|\mathbf{x},\mathrm{y};\Theta') \\ &= \arg\max_k \frac{p(\boldsymbol{x}|\mathrm{z}=k,\mathrm{y};\Theta')P(\mathrm{z}=k)}{\sum_{l=1}^{K_\mathrm{y}} p(\boldsymbol{x}|\mathrm{z}=l,\mathrm{y};\Theta')P(\mathrm{z}=l)} \\ &= \arg\max_k \tilde{\boldsymbol{\mu}}_{\mathrm{y},k}^\top\tilde{\boldsymbol{v}}\end{aligned} \tag{4}$$

where $\tilde{\boldsymbol{\mu}}_{\mathrm{y},k}$ is the mean of the k-th component for class y. In other words, we update the cluster assignment of image $\boldsymbol{x}$ within class y by taking the component with the closest mean feature $\tilde{\boldsymbol{\mu}}$ on the hyper-sphere.

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

$$\min_\Theta \mathbb{E}_{(\mathrm{x},y)\sim p(\mathrm{x},y)}\left[KL\left[Q^*(\mathrm{z})||P(\mathrm{z}|\mathrm{y},\boldsymbol{x};\Theta)\right] - \log P(\mathrm{y}|\boldsymbol{x};\Theta)\right] \tag{5}$$

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

$$\mathcal{L}_{\text{clf}} = \mathcal{L}_{\text{clf}}^{\text{inter}} + \lambda\mathcal{L}_{\text{clf}}^{\text{intra}} = -\frac{1}{|\tilde{\mathbb{D}}|}\sum_{i=1}^{|\tilde{\mathbb{D}}|}(\log(P(\mathrm{y}=y_i|\boldsymbol{x}_i;\Theta)) + \lambda\log(P(\mathrm{z}=\hat{z}_i|\boldsymbol{x}_i,y_i;\Theta))) \tag{6}$$

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, the posterior of cluster assignment $P(\mathrm{z}=k|\mathrm{y},\boldsymbol{x};\Theta)$ is equal to $\frac{e^{\kappa\tilde{\boldsymbol{\mu}}_{\mathrm{y},k}^{\top}\tilde{\boldsymbol{v}}}}{\sum_{l=1}^{K_{\mathrm{y}}}e^{\kappa\tilde{\boldsymbol{\mu}}_{\mathrm{y},l}^{\top}\tilde{\boldsymbol{v}}}}$, and λ is the hyper-parameter to balance these two losses. It is worth noting that the cluster assignment z is dependent on the prediction of label y. So 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

$$P(\mathrm{y}=c|\boldsymbol{x};\Theta) = \frac{p(\boldsymbol{x}|\mathrm{y}=c)P(\mathrm{y}=c)}{\sum_{m=1}^{|\tilde{\mathbb{C}}_t|}\sum_{n=1}^{K_m}p(\boldsymbol{x}|\mathrm{z}=n,\mathrm{y}=m)} = \frac{\frac{1}{K_c}\sum_{i=1}^{K_c}e^{\kappa\tilde{\boldsymbol{\mu}}_{c,i}^{\top}\tilde{\boldsymbol{v}}}}{\sum_{n=1}^{|\tilde{\mathbb{C}}_t|}\sum_{k=1}^{K_m}\frac{1}{K_n}e^{\kappa\tilde{\boldsymbol{\mu}}_{n,k}^{\top}\tilde{\boldsymbol{v}}}} \tag{7}$$

To prevent intra-class forgetting and preserve the learned intra-class structure, we employ knowledge distillation on the memory and the corresponding loss is

$$\mathcal{L}_{\text{dis}} = \frac{1}{|\mathbb{M}|}\sum_{(\boldsymbol{x}_i,y_i)\in\mathbb{M}}KL(P(\mathrm{z}|y_i,\boldsymbol{x}_i;\Theta)||P(\mathrm{z}|y_i,\boldsymbol{x}_i;\Theta_{\text{old}})) \tag{8}$$

where Θ_{old} is the parameters of learned model at last session. Moreover, to encourage the model to learn a compact representation, we introduce a cluster regularization loss (Shah & Koltun, 2017) on the mixture model of each class as follows

$$\mathcal{L}_{\text{reg}} = -\frac{1}{|\tilde{\mathbb{C}}|}\sum_{y\in\tilde{\mathbb{C}}}\sum_{i=1}^{K_y}\sum_{j=i+1}^{K_y}\frac{1}{K_y*(K_y-1)}\tilde{\boldsymbol{\mu}}_{y,i}^{\top}\tilde{\boldsymbol{\mu}}_{y,j} \tag{9}$$

The overall loss function in M-step is the linear combination of these losses, defined as follows

$$\mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{clf}} + \beta\mathcal{L}_{\text{dis}} + \eta\mathcal{L}_{\text{reg}} \tag{10}$$

where β,η are the loss weighting coefficients.

3.4 Memory Selection

To tackle the data imbalance problem, we introduce a bi-level balanced memory strategy when encountering new data. Concretely, we select $m=\mathcal{B}/|\tilde{\mathbb{C}}|$ exemplars for each class where $\mathcal{B}$ is the number of exemplars that can be saved to ensure the inter-class class balance. Furthermore, given the mixture model for each class c, we select $m/|K_c|$ samples from each component of class c to achieve an intra-class domain balance.

3.5 Model Inference

During inference, given an image $\mathbf{x}$, we obtain its normalized feature $\tilde{\boldsymbol{v}}$ and predict the label $\hat{y}$ by taking the class of the closest component in the feature space, which can be written as

$$\hat{y} = \arg\max_{c}\max_{k}\ \tilde{\boldsymbol{\mu}}_{c,k}^{\top}\tilde{\boldsymbol{v}} \tag{11}$$

4 Experiments

We conduct a series of experiments to validate the effectiveness of our method. Firstly, we 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 exhaustive experimental results in Sec.4.3. In the end, we demonstrate the analysis of our method to provide more insights in Sec. 4.4.

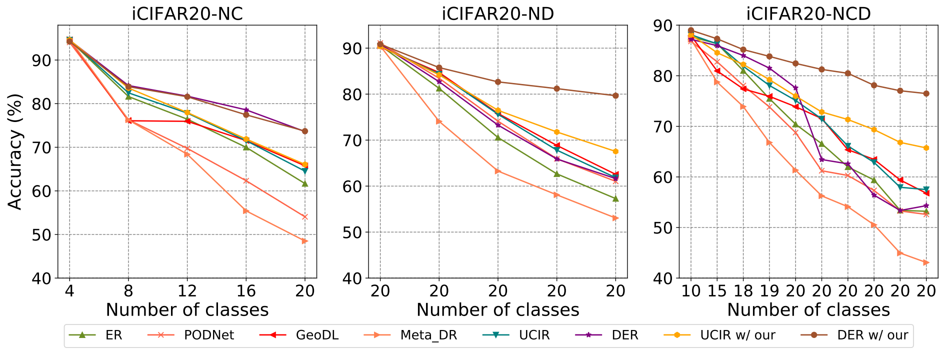


Figure 2: **Performances w.r.t sessions** on iCIFAR-20 benchmark with three splits

4.1 Experiment Setup

We compare previous methods with our approach on three benchmarks, including iCIFAR-20, iDomainNet and iDigits:

- *iCIFAR-20:* It is based on CIFAR-100 (Krizhevsky & Hinton, 2009), which consists of 20 super classes and each super class has 5 subclasses. We take these subclasses as different domains of the same class and require the model to predict super class labels for the recognition task.
- *iDomainNet:* It is constructed from DomainNet (Peng et al., 2019), which is a well-known domain adaptation dataset. It contains six domains, which are *Clipart*, *Infograph*, *Painting*, *Quickdraw*, *Real* and *Sketch*. Each domain contains 345 categories of common objects. We sample 100 classes from them which contains 132,673 training data in total.
- *iDigits:* We follow Volpi et al. (2021) to construct a digit recognition benchmark, which includes four datasets: MNIST (Yann et al., 2011), SVHN (Netzer et al., 2011), MNIST-M (Ganin & Lempitsky, 2015) and SYN (Ganin & Lempitsky, 2015). Each dataset is built from a different domain.

For every benchmark, we evaluate our proposed method on three representative splits to simulate different scenarios in which the class and domain distributions shift:

- *New Class(NC)*: The incoming data contains images from new categories only.
- *New Domain(ND)*: The incoming data contains images from new domains only.
- *New Class and Domain(NCD)*: There are images arriving from either new classes or domains.

For this *NC* split, we build 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. For the *ND* split, each class has an incoming domain at session t, where iCIFAR-20, iDigits and iDomainNet are splitted into 5, 4 and 6 sessions, respectively. For the *NCD* split, we divide all domains in the dataset into ten sessions for each of the three datasets.

We adopt Replay, UCIR (Hou et al., 2019), PODNet (Douillard et al., 2020), GeoDL (Simon et al., 2021b), DER (Yan et al., 2021) and Meta-DR (Volpi et al., 2021) as the comparison methods. Here Replay refers that the model is fine-tuned using both the memory and incoming data. It is noteworthy that UCIR, PODNet, GeoDL and DER are all designed to address the class incremental learning problem, whereas Meta-DR are proposed to resovle domain incremental learning problem. In this work, we focus on solving the intra-class stability-plasticity dilemma, which is compatible with previous class incremental learning methods that address inter-class stability-plasticity dilemma. As a result, we conduct experiments in which our method is combined with existing incremental learning methods.

Table 1: **Results:** Average incremental accuracy(%) over sessions on iCIFAR-20, iDomainNet, and iDigits with three representative splits.

Methods	iCIFAR-20			iDomainNet			iDigits		
	NC	ND	NCD	NC	ND	NCD	NC	ND	NCD
Replay	76.85	72.47	69.54	45.73	47.40	44.53	88.28	92.75	80.24
PODNet (Douillard et al., 2020)	71.29	75.13	67.49	56.23	50.86	54.75	78.65	94.39	75.50
GeoDL (Simon et al., 2021b)	76.79	76.43	71.26	51.64	47.14	45.12	85.39	93.44	80.07
Meta-DR (Volpi et al., 2021)	68.66	67.79	61.69	32.23	37.66	30.64	86.48	93.43	83.69
UCIR (Hou et al., 2019)	78.19	76.01	72.54	50.17	49.25	44.53	89.41	93.13	86.29
UCIR w/ ours	78.82	78.08	75.62	50.52	52.85	49.81	90.51	94.39	**90.42**
DER (Yan et al., 2021)	82.07	74.87	70.64	66.58	46.76	50.00	88.90	84.96	66.27
DER w/ ours	**82.77**	**84.11**	**82.17**	**66.85**	**61.05**	**57.07**	**91.32**	**97.32**	88.65

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

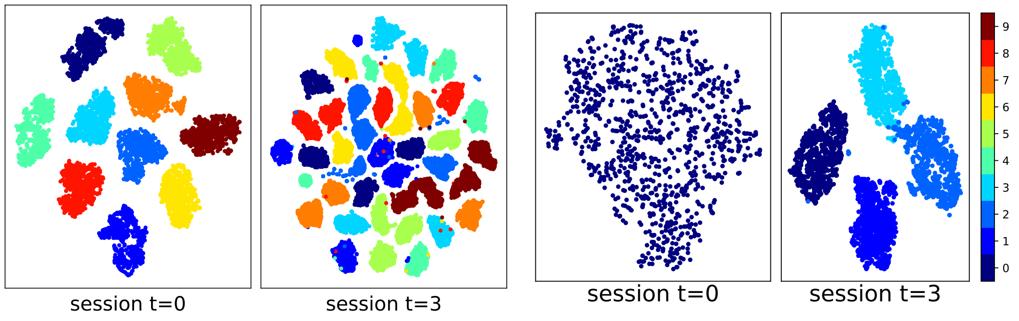
Components			NC		ND		NCD	
Mixture model	**Expansion-Reduction**	**Bi-level Memory**	**Final(%)**	**Avg(%)**	**Final(%)**	**Avg(%)**	**Final(%)**	**Avg(%)**
✗	✗	✗	72.06	82.07	67.35	77.18	54.32	70.64
✓	✗	✗	72.39	81.38	68.98	80.39	68.47	78.71
✓	✓	✗	74.14	82.46	71.92	82.04	71.56	80.97
✓	✓	✓	**74.49**	**82.77**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

4.2 Implementation Details

All these methods are implemented with PyTorch (Paszke et al., 2017). For iCIFAR-20 and iDigits benchmarks, we resize the images to 32x32 for each involved dataset. For iDomainNet, we resize the image to 112x112. For the iCIFAR-20 and iDomainNet benchmarks, we follow DER (Yan et al., 2021) and adopt the standard ResNet18 (He et al., 2016) architecture as the feature extractor. We use SGD optimizer to train the network with 200 epochs in total. 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 a modified 32-layer ResNet which is used in Hou et al. (2019). We train the methods with SGD optimizer for 70 epochs for the iDigits benchmarks 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, and the loss weighting coefficients $\beta = 1, \eta = 0.1$. Following Douillard et al. (2020), these hyper-parameters are tuned on a validation set built from the original training data. 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 E-step to update the cluster assignments at the beginning of each epoch. The coefficient λ in Eq. 6 linearly increases from 0 to 0.1 for the first 10 epochs and then is fixed at 0.1.

4.3 Experimental Results

Tab. 1 summarizes the average accuracy over sessions for all splits. We can observe that our method regularly outperforms than other methods on three different type of data distribution shifts for each benchmarks, demonstrating our method's superiority. Particularly, DER w/ ours consistently achieves the highest average accuracy in the majority of cases, e.g. 84.11% on iCIFAR20-ND, and UCIR w/ ours performs best on iDigits-NCD split at 90.42% accuracy. As demonstrated in Fig. 2, it is observed that our method consistently performs better than other method at each session for different splits. Specifically, the final session accuracy is boosted from 54.32% to $76.40\%(+\mathbf{20.08}\%)$ on iCIFAR20-NCD split by incorporating our method into DER which again proves the effectiveness of our method. Additionally, we find that integrating our method can significantly increase 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. Particularly, we get a substantial increase on average incremental accuracy from 46.76 to $61.05(+\mathbf{14.29}\%)$ by introducing our method based on DER for iDomainNet-ND benchmark.



(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 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

4.4 Analysis

Ablation Study Tab. 2 summarizes the results of our ablative experiments on iCIFAR-20, starting with DER. We can find that with the help of introduced mixture model, our method achieve 8.07% improvement on average incremental accuracy for the NCD split. Furthermore, 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 1.25% gain on the ND split. 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 (Van der Maaten & Hinton, 2008) 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 class, 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 specific class for further analysis, which is 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 little variation of the threshold. See Appendix C for more sensitive study.

5 Conclusion

In this work, we propose and formulate the general incremental learning problem, which is important in many real applications. To tackle the challenges in this problem, we introduce a domain-aware learning framework. Concretely, we propose a flexible class representation based on the mixture model and a bi-level balanced memory selection strategy that in combination to solve the intra-class stability-plasticity dilemma and domain imbalance. Specifically, we learn the compact domain-aware representation by adopting the expansion-reduction strategy and an EM framework. We conduct exhaustive experiments on three benchmarks to validate the effectiveness of our method. The experimental results demonstrates that our method consistently outperforms than other methods on three representative splits for each dataset.

REFERENCES

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.

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.

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 (eds.), *Advances in neural information processing systems (NeurIPS)*, 2020.

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.

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

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.

Shuyang Chen and John T Wen. Adaptive neural trajectory tracking control for flexible-joint robots with online learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

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.

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.

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.

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

Stephen Grossberg. Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks*, 2013.

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.

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.

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.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

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

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.

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.

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.

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 (eds.), *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

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.

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

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)*, 2021a.

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)*, 2021b.

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.

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *Advances in Neural Information Processing Systems (NeurIPS) Workshop*, 2018.

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

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.

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.

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.

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.

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.

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.

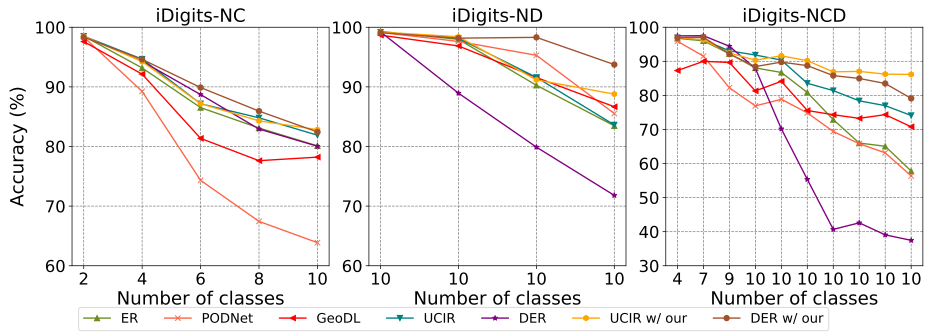


Figure 4: **Performances w.r.t sessions** on iDigits benchmark with three splits

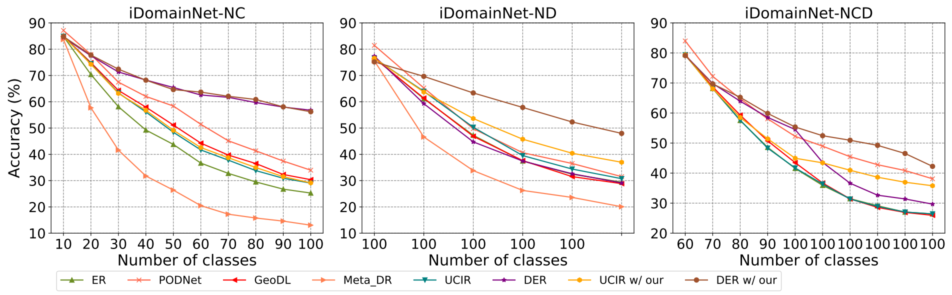


Figure 5: **Performances w.r.t sessions** on iDomainNet benchmark with three splits

A More Implementation Details

We use Nvidia Titan XP, Nvidia RTX 2080 and Nvidia Titan RTX as the computation platforms with CUDA 10.1. Our python is 3.7 and PyTorch is 1.71. We use seed 1993 for all the experiments.

Data Augmentation For iDigits benchmark, we only resize every image to 32x32. For iCIFAR-20 benchmark, we use RandomCrop with shape 32 and padding 4. We also use RandomHorizontalFlip, ColorJitter with brightness as 63/255 and Normalization with mean (0.5071,0.4867,0.4408) and std (0.2675,0.2565,0.2761). For iDomainNet, We apply RandomResizedCrop with size as 112 , RandomHorizontalFlip and Normalization with 0.5 mean and 0.5 std for all the channels.

B More curves

We include more curves of performance w.r.t. sessions on iDigits and iDomainNet benchmarks with three splits, which are shown in Fig. 4 and Fig. 5. We can see our method consistently perform better than other methods.

C More Sensitive study

We also conduct sensitive study on memory size. As shown in Tab. 4, the performance of the model continuously improves as the memory size increases.

D More t-SNE visualization

We provide more t-SNE visualization results of different digits in the iDigits ND split across different sessions. We can see that for most digits, our method can inference the correct domain labels.

Table 4: **Sensitive Study** Influence of memory size on our method for iCIFAR-20 NCD split.

Memory size	$M = 500$		$M = 1000$		$M = 1500$		$M = 2000$		$M = 2500$		$M = 3000$	
	Final	Avg	Final	Avg	Final	Avg	Final	Avg	Final	Avg	Final	Avg
DER w/ ours	66.45	75.04	72.47	78.02	75.13	80.63	76.40	82.17	77.56	82.61	78.80	82.80

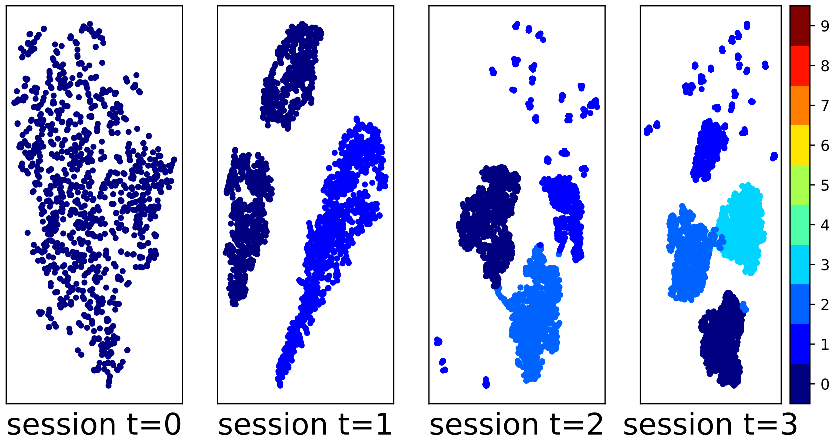


Figure 6: t-SNE visualization of digits 1.

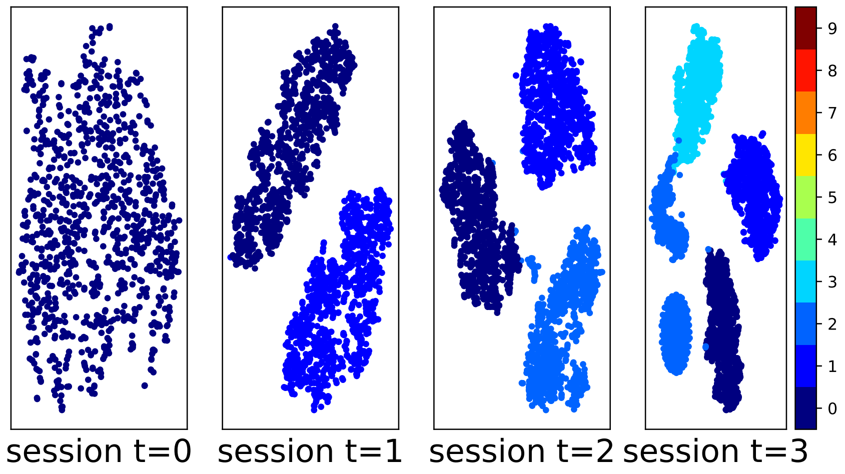


Figure 7: t-SNE visualization of digits 3.

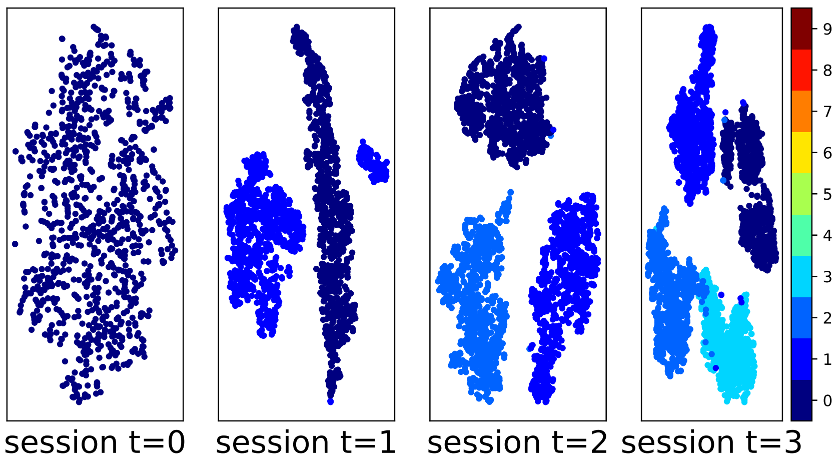


Figure 8: t-SNE visualization of digits 5.

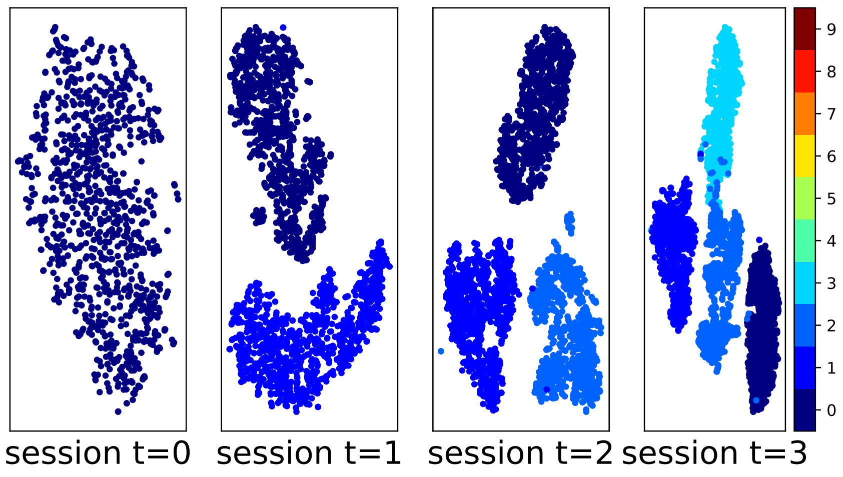


Figure 9: t-SNE visualization of digits 7.