A Unified Approach to Domain Incremental Learning with Memory: Theory and Algorithm

Haizhou Shi
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
haizhou.shi@rutgers.edu

Hao Wang
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
hw488@cs.rutgers.edu

Abstract

Domain incremental learning aims to adapt to a sequence of domains with access to only a small subset of data (i.e., memory) from previous domains. Various methods have been proposed for this problem, but it is still unclear how they are related and when practitioners should choose one method over another. In response, we propose a unified framework, dubbed Unified Domain Incremental Learning (UDIL), for domain incremental learning with memory. Our UDIL unifies various existing methods, and our theoretical analysis shows that UDIL always achieves a tighter generalization error bound compared to these methods. The key insight is that different existing methods correspond to our bound with different *fixed* coefficients; based on insights from this unification, our UDIL allows *adaptive* coefficients during training, thereby always achieving the tightest bound. Empirical results show that our UDIL outperforms the state-of-the-art domain incremental learning methods on both synthetic and real-world datasets. Code will be available at https://github.com/Wang-ML-Lab/unified-continual-learning.

1 Introduction

Despite recent success of large-scale machine learning models [35, 48, 36, 28, 92, 22, 33], continually learning from evolving environments remains a longstanding challenge. Unlike the conventional machine learning paradigms where learning is performed on a static dataset, *domain incremental learning, i.e., continual learning with evolving domains*, hopes to accommodate the model to the dynamically changing data distributions, while retaining the knowledge learned from previous domains [90, 60, 41, 97, 27]. Naive methods, such as continually finetuning the model on new-coming domains, will suffer a substantial performance drop on the previous domains; this is referred to as "catastrophic forgetting" [46, 58, 81, 105, 52]. In general, domain incremental learning algorithms aim to minimize the total risk of *all* domains, i.e.,

$$\mathcal{L}^*(\theta) = \mathcal{L}_t(\theta) + \mathcal{L}_{1:t-1}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell(y, h_\theta(x))] + \sum_{i=1}^{t-1} \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\ell(y, h_\theta(x))], \tag{1}$$

where $\mathcal{L}_t$ calculates model h_θ's expected prediction error ℓ over the current domain's data distribution $\mathcal{D}_t$. $\mathcal{L}_{1:t-1}$ is the total error evaluated on the past $t-1$ domains' data distributions, i.e., $\{\mathcal{D}_i\}_{i=1}^{t-1}$.

The main challenge of domain incremental learning comes from the practical *memory constraint* that no (or only very limited) access to the past domains' data is allowed [52, 46, 105, 74]. Under such a constraint, it is difficult, if not impossible, to accurately estimate and optimize the past error $\mathcal{L}_{1:t-1}$. Therefore the main focus of recent domain incremental learning methods has been to develop effective surrogate learning objectives for $\mathcal{L}_{1:t-1}$. Among these methods [46, 81, 2, 105, 58, 10, 75,

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

https://doi.org/10.52202/075280-0660

77, 21, 25, 65, 66, 9, 72, 82, 95, 53], replay-based methods, which replay a small set of old exemplars during training [90, 75, 8, 4, 80, 11], has consistently shown promise and is therefore commonly used in practice.

One typical example is ER [75], which stores a set of exemplars $\mathcal{M}$ and uses a replay loss $\mathcal{L}_{\text{replay}}$ as the surrogate of $\mathcal{L}_{1:t-1}$. In addition, a fixed, predetermined coefficient β is used to balance current domain learning and past sample replay. Specifically,

$$\widetilde{\mathcal{L}}(\theta) = \mathcal{L}_t(\theta) + \beta \cdot \mathcal{L}_{\text{replay}}(\theta) = \mathcal{L}_t(\theta) + \beta \cdot \mathbb{E}_{(x',y')\sim\mathcal{M}}[\ell(y', h_\theta(x'))]. \tag{2}$$

While such methods are popular in practice, there is still a gap between the surrogate loss ($\beta\mathcal{L}_{\text{replay}}$) and the true objective ($\mathcal{L}_{1:t-1}$), rendering them lacking in theoretical support and therefore calling into question their reliability. Besides, different methods use different schemes of setting β [75, 8, 4, 80], and it is unclear how they are related and when practitioners should choose one method over another.

To address these challenges, we develop a unified generalization error bound and theoretically show that different existing methods are actually minimizing the same error bound with different *fixed* coefficients (more details in Table 1 later). Based on such insights, we then develop an algorithm that allows *adaptive* coefficients during training, thereby always achieving the tightest bound and improving the performance. Our contributions are as follows:

- We propose a unified framework, dubbed Unified Domain Incremental Learning (UDIL), for domain incremental learning with memory to unify various existing methods.
- Our theoretical analysis shows that different existing methods are equivalent to minimizing the same error bound with different *fixed* coefficients. Based on insights from this unification, our UDIL allows *adaptive* coefficients during training, thereby always achieving the tightest bound and improving the performance.
- Empirical results show that our UDIL outperforms the state-of-the-art domain incremental learning methods on both synthetic and real-world datasets.

2 Related Work

Continual Learning. Prior work on continual learning can be roughly categorized into three scenarios [90, 15]: (i) task-incremental learning, where task indices are available during both training and testing [52, 46, 90], (ii) class-incremental learning, where new classes are incrementally included for the classifier [74, 100, 30, 45, 44], and (iii) domain-incremental learning, where the data distribution's incremental shift is explicitly modeled [60, 41, 97, 27]. Regardless of scenarios, the main challenge of continual learning is to alleviate catastrophic forgetting with only limited access to the previous data; therefore methods in one scenario can often be easily adapted for another. Many methods have been proposed to tackle this challenge, including functional and parameter regularization [52, 46, 81, 2], constraining the optimization process [77, 21, 58, 10], developing incrementally updated components [104, 38, 53], designing modularized model architectures [73, 95], improving representation learning with additional inductive biases [9, 66, 65, 25], and Bayesian approaches [24, 63, 49, 1]. Among them, replaying a small set of old exemplars, i.e., memory, during training has shown great promise as it is easy to deploy, *applicable in all three scenarios*, and, most importantly, achieves impressive performance [90, 75, 8, 4, 80, 11]. Therefore in this paper, we focus on *domain incremental learning* with *memory*, aiming to provide a principled theoretical framework to unify these existing methods.

Domain Adaptation and Domain Incremental Learning. Loosely related to our work are domain adaptation (DA) methods, which adapt a model trained on *labeled* source domains to *unlabeled* target domains [68, 67, 57, 78, 79, 108, 71, 16, 17, 64, 94, 51]. Much prior work on DA focuses on matching the distribution of the source and target domains by directly matching the statistical attributions [67, 89, 87, 71, 64] or adversarial training [108, 57, 26, 109, 17, 102, 101, 54, 94]. Compared to DA's popularity, domain incremental learning (DIL) has received limited attention in the past. However, it is now gaining significant traction in the research community [90, 60, 41, 97, 27]. These studies predominantly focus on the practical applications of DIL, such as semantic segmentation [27], object detection for autonomous driving [60], and learning continually in an open-world setting [18]. Inspired by the theoretical foundation of adversarial DA [5, 57], we propose, to the best of our knowledge, **the first unified upper bound for DIL**. Most related to our work are previous DA methods that flexibly align different domains according to their associated given or inferred domain index [94, 101], domain graph [102], and domain taxonomy [54]. The main difference between DA and DIL is that the former focuses on improving the accuracy of the *target domains*, while the latter focuses on the total error of *all domains*, with additional measures taken to alleviate forgetting on the previous domains. More importantly, DA methods typically require access to target domain data to match the distributions, and therefore are not directly applicable to DIL.

3 Theory: Unifying Domain Incremental Learning

In this section, we formalize the problem of domain incremental learning, provide the generalization bound of naively applying empirical risk minimization (ERM) on the memory bank, derive two error bounds (i.e., intra-domain and cross-domain error bounds) more suited for domain incremental learning, and then unify these three bounds to provide our final adaptive error bound. We then develop an algorithm inspired by this bound in Sec. 4. **All proofs of lemmas, theorems, and corollaries can be found in Appendix A.**

Problem Setting and Notation. We consider the problem of domain incremental learning with T domains arriving one by one. Each domain i contains N_i data points $\mathcal{S}_i = \{(\boldsymbol{x}_j^{(i)}, y_j^{(i)})\}_{j=1}^{N_i}$, where $(\boldsymbol{x}_j^{(i)}, y_j^{(i)})$ is sampled from domain i's data distribution $\mathcal{D}_i$. Assume that when domain $t \in [T] \triangleq \{1, 2, \dots, T\}$ arrives at time t, one has access to (1) the current domain t's data $\mathcal{S}_t$, (2) a memory bank $\mathcal{M} = \{M_i\}_{i=1}^{t-1}$, where $M_i = \{(\widetilde{\boldsymbol{x}}_j^{(i)}, \widetilde{y}_j^{(i)})\}_{j=1}^{\widetilde{N}_i}$ is a small subset ($\widetilde{N}_i \ll N_i$) randomly sampled from $\mathcal{S}_i$, and (3) the history model H_{t-1} after training on the previous $t-1$ domains. For convenience we use shorthand notation $\mathcal{X}_i \triangleq \{\boldsymbol{x}_j^{(i)}\}_{j=1}^{N_i}$ and $\widetilde{\mathcal{X}}_i \triangleq \{\widetilde{\boldsymbol{x}}_j^{(i)}\}_{j=1}^{\widetilde{N}_i}$. The goal is to learn the optimal model (hypothesis) h^* that minimizes the prediction error over all t domains after each domain t arrives. Formally,

$$h^* = \arg\min_h \sum_{i=1}^{t} \epsilon_{\mathcal{D}_i}(h), \qquad \epsilon_{\mathcal{D}_i}(h) \triangleq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_i}[h(\boldsymbol{x}) \neq f_i(\boldsymbol{x})], \tag{3}$$

where for domain i, we assume the labels $y \in \mathcal{Y} = \{0, 1\}$ are produced by an unknown deterministic function $y = f_i(\boldsymbol{x})$ and $\epsilon_{\mathcal{D}_i}(h)$ denotes the expected error of domain i.

3.1 Naive Generalization Bound Based on ERM

Definition 3.1 (Domain-Specific Empirical Risks). *When domain t arrives, model h's empirical risk $\widehat{\epsilon}_{\mathcal{D}_i}(h)$ for each domain i (where $i \le t$) is computed on the available data at time t, i.e.,*

$$\widehat{\epsilon}_{\mathcal{D}_i}(h) = \begin{cases} \frac{1}{N_i}\sum_{\boldsymbol{x}\in X_i} \mathbb{1}_{h(\boldsymbol{x})\neq f_i(\boldsymbol{x})} & \textit{if } i = t, \\ \frac{1}{\widetilde{N}_i}\sum_{\boldsymbol{x}\in \widetilde{X}_i} \mathbb{1}_{h(\boldsymbol{x})\neq f_i(\boldsymbol{x})} & \textit{if } i < t. \end{cases} \tag{4}$$

Note that at time t, only a small subset of data from previous domains ($i < t$) is available in the memory bank ($\widetilde{N}_i \ll N_i$). Therefore empirical risks for previous domains ($\widehat{\epsilon}_{\mathcal{D}_i}(h)$ with $i < t$) can deviate a lot from the true risk $\epsilon_{\mathcal{D}_i}(h)$ (defined in Eqn. 3); this is reflected in Lemma 3.1 below.

Lemma 3.1 (ERM-Based Generalization Bound). *Let $\mathcal{H}$ be a hypothesis space of VC dimension d. When domain t arrives, there are N_t data points from domain t and $\widetilde{N}_i$ data points from each previous domain $i < t$ in the memory bank. With probability at least $1-\delta$, we have:*

$$\sum_{i=1}^{t} \epsilon_{\mathcal{D}_i}(h) \le \sum_{i=1}^{t} \widehat{\epsilon}_{\mathcal{D}_i}(h) + \sqrt{\left(\frac{1}{N_t} + \sum_{i=1}^{t-1}\frac{1}{\widetilde{N}_i}\right)\left(8d\log\left(\frac{2eN}{d}\right) + 8\log\left(\frac{2}{\delta}\right)\right)}. \tag{5}$$

Lemma 3.1 shows that naively using ERM to learn h is equivalent to minimizing a loose generalization bound in Eqn. 33. Since $\widetilde{N}_i \ll N_i$, there is a large constant $\sum_{i=1}^{t-1}\frac{1}{\widetilde{N}_i}$ compared to $\frac{1}{N_t}$, making the second term of Eqn. 33 much larger and leading to a looser bound.

https://doi.org/10.52202/075280-0660

3.2 Intra-Domain and Cross-Domain Model-Based Bounds

In domain incremental learning, one has access to the history model H_{t-1} besides the memory bank $\{M_i\}_{i=1}^{t-1}$; this offers an opportunity to derive tighter error bounds, potentially leading to better algorithms. In this section, we will derive two such bounds, an intra-domain error bound (Lemma 3.2) and a cross-domain error bound (Lemma 3.3), and then integrate them two with the ERM-based bound in Eqn. 33 to arrive at our final adaptive bound (Theorem 3.4).

Lemma 3.2 (Intra-Domain Model-Based Bound). *Let $h \in \mathcal{H}$ be an arbitrary function in the hypothesis space $\mathcal{H}$, and H_{t-1} be the model trained after domain $t-1$. The domain-specific error $\epsilon_{\mathcal{D}_i}(h)$ on the previous domain i has an upper bound:*

$$\epsilon_{\mathcal{D}_i}(h) \leq \epsilon_{\mathcal{D}_i}(h, H_{t-1}) + \epsilon_{\mathcal{D}_i}(H_{t-1}), \tag{6}$$

where $\epsilon_{\mathcal{D}_i}(h, H_{t-1}) \triangleq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_i}[h(\boldsymbol{x}) \neq H_{t-1}(\boldsymbol{x})]$.

Lemma 3.2 shows that the current model h's error on domain i is bounded by the discrepancy between h and the history model H_{t-1} plus the error of H_{t-1} on domain i.

One potential issue with the bound Eqn. 34 is that only a limited number of data is available for each previous domain i in the memory bank, making empirical estimation of $\epsilon_{\mathcal{D}_i}(h, H_{t-1}) + \epsilon_{\mathcal{D}_i}(H_{t-1})$ challenging. Lemma 3.3 therefore provides an alternative bound.

Lemma 3.3 (Cross-Domain Model-Based Bound). *Let $h \in \mathcal{H}$ be an arbitrary function in the hypothesis space $\mathcal{H}$, and H_{t-1} be the function trained after domain $t-1$. The domain-specific error $\epsilon_{\mathcal{D}_i}(h)$ evaluated on the previous domain i then has an upper bound:*

$$\epsilon_{\mathcal{D}_i}(h) \leq \epsilon_{\mathcal{D}_t}(h, H_{t-1}) + \tfrac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t) + \epsilon_{\mathcal{D}_i}(H_{t-1}), \tag{7}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{P}, \mathcal{Q}) = 2\sup_{h\in\mathcal{H}\Delta\mathcal{H}} |\Pr_{x\sim\mathcal{P}}[h(x)=1] - \Pr_{x\sim\mathcal{Q}}[h(x)=1]|$ denotes the $\mathcal{H}\Delta\mathcal{H}$-divergence between distribution $\mathcal{P}$ and $\mathcal{Q}$, and $\epsilon_{\mathcal{D}_t}(h, H_{t-1}) \triangleq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}[h(\boldsymbol{x}) \neq H_{t-1}(\boldsymbol{x})]$.

Lemma 3.3 shows that if the divergence between domain i and domain t, i.e., $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t)$, is small enough, one can use H_{t-1}'s predictions evaluated on the current domain $\mathcal{D}_t$ as a surrogate loss to prevent catastrophic forgetting. Compared to the error bound Eqn. 34 which is hindered by limited data from previous domains, Eqn. 35 relies on the current domain t which contains abundant data and therefore enjoys much lower generalization error. Our lemma also justifies LwF-like cross-domain distillation loss $\epsilon_{\mathcal{D}_t}(h, H_{t-1})$ which are widely adopted [52, 23, 100].

3.3 A Unified and Adaptive Generalization Error Bound

Our Lemma 3.1, Lemma 3.2, and Lemma 3.3 provide three different ways to bound the true risk $\sum_{i=1}^{t} \epsilon_{\mathcal{D}_i}(h)$; each has its own advantages and disadvantages. Lemma 3.1 overly relies on the limited number of data points from previous domains $i < t$ in the memory bank to compute the empirical risk; Lemma 3.2 leverages the history model H_{t-1} for knowledge distillation, but is still hindered by the limited number of data points in the memory bank; Lemma 3.3 improves over Lemma 3.2 by leveraging the abundant data $\mathcal{D}_t$ in the current domain t, but only works well if the divergence between domain i and domain t, i.e., $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t)$, is small. Therefore, we propose to integrate these three bounds using coefficients $\{\alpha_i, \beta_i, \gamma_i\}_{i=1}^{t-1}$ (with $\alpha_i + \beta_i + \gamma_i = 1$) in the theorem below.

Theorem 3.4 (Unified Generalization Bound for All Domains). *Let $\mathcal{H}$ be a hypothesis space of VC dimension d. Let $N = N_t + \sum_{i}^{t-1} \widetilde{N}_i$ denoting the total number of data points available to the training of current domain t, where N_t and $\widetilde{N}_i$ denote the numbers of data points collected at domain t and data points from the previous domain i in the memory bank, respectively. With probability at*

https://doi.org/10.52202/075280-0660

Table 1: **UDIL as a unified framework** for domain incremental learning with memory. Three methods (LwF [52], ER [75], and DER++ [8]) are by default compatible with DIL setting. For the remaining four CIL methods (iCaRL [74], CLS-ER [4], EMS-ER [80], and BiC [100]), we adapt their original training objective to DIL settings before the analysis. For CLS-ER [4] and EMS-ER [80], λ and λ' are the intensity coefficients of the logits distillation. For BiC [100], t is the current number of the incremental domain. The conditions under which the unification of each method is achieved are provided in detail in Appendix B.

	UDIL (Ours)	LwF [52]	ER [75]	DER++ [8]	iCaRL [74]	CLS-ER [4]	EMS-ER [80]	BiC [100]
α_i	$[0, 1]$	0	0	0.5	1	$^{\lambda}/_{(1+\lambda)}$	$^{\lambda'}/_{(1+\lambda')}$	$^{1}/_{(2t-1)}$
β_i	$[0, 1]$	1	0	0	0	0	0	$^{(t-1)}/_{(2t-1)}$
γ_i	$[0, 1]$	0	1	0.5	0	$^{1}/_{(1+\lambda)}$	$^{1}/_{(1+\lambda')}$	$^{t-1}/_{(2t-1)}$

least $1 - \delta$*, we have:*

$$\begin{aligned}
\sum_{i=1}^{t} \epsilon_{\mathcal{D}_i}(h) \leq & \left\{ \sum_{i=1}^{t-1} [\gamma_i \widehat{\epsilon}_{\mathcal{D}_i}(h) + \alpha_i \widehat{\epsilon}_{\mathcal{D}_i}(h, H_{t-1})] \right\} + \left\{ \widehat{\epsilon}_{\mathcal{D}_t}(h) + (\sum_{i=1}^{t-1} \beta_i) \widehat{\epsilon}_{\mathcal{D}_t}(h, H_{t-1}) \right\} \\
& + \tfrac{1}{2} \sum_{i=1}^{t-1} \beta_i d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t) + \sum_{i=1}^{t-1} (\alpha_i + \beta_i) \epsilon_{\mathcal{D}_i}(H_{t-1}) \\
& + \sqrt{\left(\tfrac{(1+\sum_{i=1}^{t-1} \beta_i)^2}{N_t} + \sum_{i=1}^{t-1} \tfrac{(\gamma_i + \alpha_i)^2}{\widetilde{N}_i} \right) \left(8d \log\left(\tfrac{2eN}{d}\right) + 8 \log\left(\tfrac{2}{\delta}\right) \right)} \\
& \triangleq g(h, H_{t-1}, \Omega),
\end{aligned} \tag{8}$$

where $\widehat{\epsilon}_{\mathcal{D}_i}(h, H_{t-1}) = \frac{1}{\widetilde{N}_i} \sum_{\boldsymbol{x} \in \widetilde{\mathcal{X}}_i} \mathbb{1}_{h(\boldsymbol{x}) \neq H_{t-1}(\boldsymbol{x})}$, $\widehat{\epsilon}_{\mathcal{D}_t}(h, H_{t-1}) = \frac{1}{N_t} \sum_{\boldsymbol{x} \in \mathcal{X}_i} \mathbb{1}_{h(\boldsymbol{x}) \neq H_{t-1}(\boldsymbol{x})}$*, and* $\Omega \triangleq \{\alpha_i, \beta_i, \gamma_i\}_{i=1}^{t-1}$.

Theorem 3.4 offers the opportunity of adaptively adjusting the coefficients (α_i, β_i, and γ_i) according to the data (current domain data $\mathcal{S}_t$ and the memory bank $\mathcal{M} = \{M_i\}_{i=1}^{t-1}$) and history model ($H_{t-1}$) at hand, thereby achieving the tightest bound. For example, when the $\mathcal{H}\Delta\mathcal{H}$ divergence between domain i and domain t, i.e., $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t)$, is small, minimizing this unified bound (Eqn. 8) leads to a large coefficient β_i and therefore naturally puts on more focus on cross-domain bound in Eqn. 35 which leverages the current domain t's data to estimate the true risk.

UDIL as a Unified Framework. Interestingly, Eqn. 8 unifies various domain incremental learning methods. Table 1 shows that different methods are equivalent to fixing the coefficients $\{\alpha_i, \beta_i, \gamma_i\}_{i=1}^{t-1}$ to different values (see Appendix B for a detailed discussion). For example, assuming default configurations, LwF [52] corresponds to Eqn. 8 with *fixed* coefficients $\{\alpha_i = \gamma_i = 0, \beta_i = 1\}$; ER [75] corresponds to Eqn. 8 with *fixed* coefficients $\{\alpha_i = \beta_i = 0, \gamma_i = 1\}$, and DER++ [8] corresponds to Eqn. 8 with *fixed* coefficients $\{\alpha_i = \gamma_i = 0.5, \beta_i = 0\}$, under certain achievable conditions. Inspired by this unification, our UDIL adaptively adjusts these coefficients to search for the tightest bound in the range $[0, 1]$ when each domain arrives during domain incremental learning, thereby improving performance. Corollary 3.4.1 below shows that such *adaptive* bound is always tighter, or at least as tight as, any bounds with *fixed* coefficients.

Corollary 3.4.1. *For any bound* $g(h, H_{t-1}, \Omega_{\textit{fixed}})$ *(defined in Eqn. 8) with* fixed *coefficients* $\Omega_{\textit{fixed}}$*, e.g.,* $\Omega_{\textit{fixed}} = \Omega_{\textit{ER}} = \{\alpha_i = \beta_i = 0, \gamma_i = 1\}_{i=1}^{t-1}$ *for ER [75], we have*

$$\sum\nolimits_{i=1}^{t} \epsilon_{\mathcal{D}_i}(h) \leq \min_{\Omega} g(h, H_{t-1}, \Omega) \leq g(h, H_{t-1}, \Omega_{\textit{fixed}}), \quad \forall h, H_{t-1} \in \mathcal{H}. \tag{9}$$

Corollary 3.4.1 shows that the unified bound Eqn. 8 with *adaptive* coefficients is always preferable to other bounds with *fixed* coefficients. We therefore use it to develop a better domain incremental learning algorithm in Sec. 4 below.

https://doi.org/10.52202/075280-0660

4 Method: Adaptively Minimizing the Tightest Bound in UDIL

Although Theorem 3.4 provides a unified perspective for domain incremental learning, it does not immediately translate to a practical objective function to train a model. It is also unclear what coefficients Ω for Eqn. 8 would be the best choice. In fact, a *static* and *fixed* setting will not suffice, as different problems may involve different sequences of domains with dynamic changes; therefore ideally Ω should be *dynamic* (e.g., $\alpha_i \neq \alpha_{i+1}$) and *adaptive* (i.e., learnable from data). In this section, we start by mapping the unified bound in Eqn. 8 to concrete loss terms, discuss how the coefficients Ω are learned, and then provide a final objective function to learn the optimal model.

4.1 From Theory to Practice: Translating the Bound in Eqn. 8 to Differentiable Loss Terms

(1) ERM Terms. We use the cross-entropy classification loss in Definition 4.1 below to optimize domain t's ERM term $\widehat{\epsilon}_{\mathcal{D}_t}(h)$ and memory replay ERM terms $\{\gamma_i \widehat{\epsilon}_{\mathcal{D}_i}(h)\}_{i=1}^{t-1}$ in Eqn. 8.

Definition 4.1 (Classification Loss). *Let $h : \mathbb{R}^n \to \mathbb{S}^{K-1}$ be a function that maps the input $\boldsymbol{x} \in \mathbb{R}^n$ to the space of K-class probability simplex, i.e., $\mathbb{S}^{K-1} \triangleq \{\boldsymbol{z} \in \mathbb{R}^K : z_i \geq 0, \sum_i z_i = 1\}$; let $\mathcal{X}$ be a collection of samples drawn from an arbitrary data distribution and $f : \mathbb{R}^n \to [K]$ be the function that maps the input to the true label. The classification loss is defined as the average cross-entropy between the true label $f(\boldsymbol{x})$ and the predicted probability $h(\boldsymbol{x})$, i.e.,*

$$\widehat{\ell}_{\mathcal{X}}(h) \triangleq \tfrac{1}{|\mathcal{X}|} \sum\nolimits_{\boldsymbol{x}\in\mathcal{X}} \Big[-\sum\nolimits_{j=1}^{K} \mathbb{1}_{f(\boldsymbol{x})=j} \cdot \log\Big([h(\boldsymbol{x})]_j\Big)\Big]. \qquad (10)$$

Following Definition 4.1, we replace $\widehat{\epsilon}_{\mathcal{D}_t}(h)$ and $\widehat{\epsilon}_{\mathcal{D}_i}(h)$ in Eqn. 8 with $\widehat{\ell}_{\mathcal{X}_t}(h)$ and $\widehat{\ell}_{\mathcal{X}_i}(h)$.

(2) Intra- and Cross-Domain Terms. We use the distillation loss below to optimize intra-domain ($\{\widehat{\epsilon}_{\mathcal{D}_i}(h, H_{t-1})\}_{i=1}^{t-1}$) and cross-domain ($\widehat{\epsilon}_{\mathcal{D}_t}(h, H_{t-1})$) model-based error terms in Eqn. 8.

Definition 4.2 (Distillation Loss). *Let $h, H_{t-1} : \mathbb{R}^n \to \mathbb{S}^{K-1}$ both be functions that map the input $\boldsymbol{x} \in \mathbb{R}^n$ to the space of K-class probability simplex as defined in Definition 4.1; let $\mathcal{X}$ be a collection of samples drawn from an arbitrary data distribution. The distillation loss is defined as the average cross-entropy between the target probability $H_{t-1}(\boldsymbol{x})$ and the predicted probability $h(\boldsymbol{x})$, i.e.,*

$$\widehat{\ell}_{\mathcal{X}}(h, H_{t-1}) \triangleq \tfrac{1}{|\mathcal{X}|} \sum\nolimits_{\boldsymbol{x}\in\mathcal{X}} \Big[-\sum\nolimits_{j=1}^{K} [H_{t-1}(\boldsymbol{x})]_j \cdot \log\Big([h(\boldsymbol{x})]_j\Big)\Big]. \qquad (11)$$

Accordingly, we replace $\widehat{\epsilon}_{\mathcal{D}_i}(h, H_{t-1})$ with $\widehat{\ell}_{\mathcal{X}_i}(h, H_{t-1})$ and $\widehat{\epsilon}_{\mathcal{D}_t}(h, H_{t-1})$ with $\widehat{\ell}_{\mathcal{X}_t}(h, H_{t-1})$.

(3) Constant Term. The error term $\sum_{i=1}^{t-1}(\alpha_i + \beta_i)\epsilon_{\mathcal{D}_i}(H_{t-1})$ in Eqn. 8 is a constant and contains no trainable parameters (since H_{t-1} is a fixed history model); therefore it does not need a loss term.

(4) Divergence Term. In Eqn. 8, $\sum_{i=1}^{t-1} \beta_i d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t)$ measures the weighted average of the dissimilarity between domain i's and domain t's data distributions. Inspired by existing adversarial domain adaptation methods [57, 26, 109, 17, 102, 101, 94], we can further tighten this divergence term by considering the *embedding distributions* instead of *data distributions* using an learnable encoder. Specifically, given an encoder $e : \mathbb{R}^n \to \mathbb{R}^m$ and a family of domain discriminators (classifiers) $\mathcal{H}_d$, we have the empirical estimate of the divergence term as follows:

$$\sum_{i=1}^{t-1} \beta_i \widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(e(\mathcal{U}_i), e(\mathcal{U}_t)) = 2\sum_{i=1}^{t-1} \beta_i - 2 \min_{d\in\mathcal{H}_d} \sum_{i=1}^{t-1} \beta_i \left[\tfrac{1}{|\mathcal{U}_i|} \sum_{\boldsymbol{x}\in\mathcal{U}_i} \mathbb{1}_{\Delta_i(\boldsymbol{x})\geq 0} + \tfrac{1}{|\mathcal{U}_t|} \sum_{\boldsymbol{x}\in\mathcal{U}_t} \mathbb{1}_{\Delta_i(\boldsymbol{x})<0} \right],$$

where $\mathcal{U}_i$ (and $\mathcal{U}_t$) is a set of samples drawn from domain $\mathcal{D}_i$ (and $\mathcal{D}_t$), $d : \mathbb{R}^m \to \mathbb{S}^{t-1}$ is a domain classifier, and $\Delta_i(\boldsymbol{x}) = [d(e(\boldsymbol{x}))]_i - [d(e(\boldsymbol{x}))]_t$ is the difference between the probability of $\boldsymbol{x}$ belonging to domain i and domain t. Replacing the indicator function with the differentiable cross-entropy loss, $\sum_{i=1}^{t-1} \beta_i \widehat{d}_{\mathcal{H}\Delta\mathcal{H}}(e(\mathcal{U}_i), e(\mathcal{U}_t))$ above then becomes

$$2\sum_{i=1}^{t-1} \beta_i - 2 \min_{d\in\mathcal{H}_d} \sum_{i=1}^{t-1} \beta_i \left[\tfrac{1}{N_i} \sum_{\boldsymbol{x}\in\mathcal{X}_i} [-\log([d(e(\boldsymbol{x}))]_i)] + \tfrac{1}{N_t} \sum_{\boldsymbol{x}\in\mathcal{S}_t} [-\log([d(e(\boldsymbol{x}))]_t)] \right]. \qquad (12)$$

Algorithm 1 Unified Domain Incremental Learning (UDIL) for Domain t Training

```
Require: history model $H_{t-1} = P_{t-1} \circ E_{t-1}$, current model $h_\theta = p \circ e$, discriminator model $d_\phi$;
Require: dataset from the current domain $\mathcal{S}_t$, memory bank $\mathcal{M} = \{M_i\}_{i=1}^{t-1}$;
Require: training steps $S$, batch size $B$, learning rate $\eta$;
Require: domain alignment strength coefficient $\lambda_d$, hyperparameter for generalization effect $C$.
$h_\theta \leftarrow H_{t-1}$                                                    ▷ Initialization of the current model.
$\Omega \triangleq \{\alpha_i, \beta_i, \gamma_i\} \leftarrow \{1/3, 1/3, 1/3\}$, for $\forall i \in [t-1]$    ▷ Initialization of the replay coefficient $\Omega$.
for $s = 1, \cdots, S$ do
    $B_t \sim \mathcal{S}_t; B_i \sim M_i, \forall i \in [t-1]$                  ▷ Sample a mini-batch of data from all domains.
    $\phi \leftarrow \phi - \eta \cdot \lambda_d \cdot \nabla_\phi V_d(d, e, \mathring{\Omega})$               ▷ Discriminator training with Eqn. 16.
    $\Omega \leftarrow \Omega - \eta \cdot \nabla_\Omega V_{\text{0-1}}(\mathring{h}, \Omega)$                    ▷ Find a tighter bound with Eqn. 15.
    $\theta \leftarrow \theta - \eta \cdot \nabla_\theta (V_l(h_\theta, \mathring{\Omega}) - \lambda_d V_d(d, e, \mathring{\Omega}))$   ▷ Model training with Eqn. 14 and Eqn. 16.
end for
$H_t \leftarrow h$
$\mathcal{M} \leftarrow \text{BalancedSampling}(\mathcal{M}, \mathcal{S}_t)$
return $H_t$                                                         ▷ For training on domain $t+1$.
```

4.2 Putting Everything Together: UDIL Training Algorithm

Objective Function. With these differentiable loss terms above, we can derive an algorithm that learns the optimal model by minimizing the tightest bound in Eqn. 8. As mentioned above, to achieve a tighter $d_{\mathcal{H}\Delta\mathcal{H}}$, we decompose the hypothesis as $h = p \circ e$, where $e : \mathbb{R}^n \to \mathbb{R}^m$ and $p : \mathbb{R}^m \to \mathbb{S}^{K-1}$ are the encoder and predictor, respectively. To find and to minimize the tightest bound in Theorem 3.4, we treat $\Omega = \{\alpha_i, \beta_i, \gamma_i\}_{i=1}^{t-1}$ as learnable parameters and seek to optimize the following objective (we denote as $\mathring{x} = \text{sg}(x)$ the 'copy-weights-and-stop-gradients' operation):

$$\begin{aligned} \min_{\{\Omega, h = p\circ e\}} \; \max_{d} \quad & V_l(h, \mathring{\Omega}) + V_{\text{0-1}}(\mathring{h}, \Omega) - \lambda_d V_d(d, e, \mathring{\Omega}) \\ \text{s.t.} \quad & \alpha_i + \beta_i + \gamma_i = 1, \qquad \forall i \in \{1, 2, \dots, t-1\} \\ & \alpha_i, \beta_i, \gamma_i \geq 0, \qquad \forall i \in \{1, 2, \dots, t-1\} \end{aligned} \tag{13}$$

Details of V_l, $V_{\text{0-1}}$, and V_d. V_l is the loss for **learning the model** h, where the terms $\widehat{\ell}_\cdot(\cdot)$ are *differentiable cross-entropy losses* as defined in Eqn. 10 and Eqn. 11:

$$V_l(h, \mathring{\Omega}) = \sum\nolimits_{i=1}^{t-1} \left[\mathring{\gamma}_i \widehat{\ell}_{\mathcal{X}_i}(h) + \mathring{\alpha}_i \widehat{\ell}_{\mathcal{X}_i}(h, H_{t-1}) \right] + \widehat{\ell}_{\mathcal{S}_t}(h) + (\sum\nolimits_{i=1}^{t-1} \mathring{\beta}_i) \widehat{\ell}_{\mathcal{S}_t}(h, H_{t-1}). \tag{14}$$

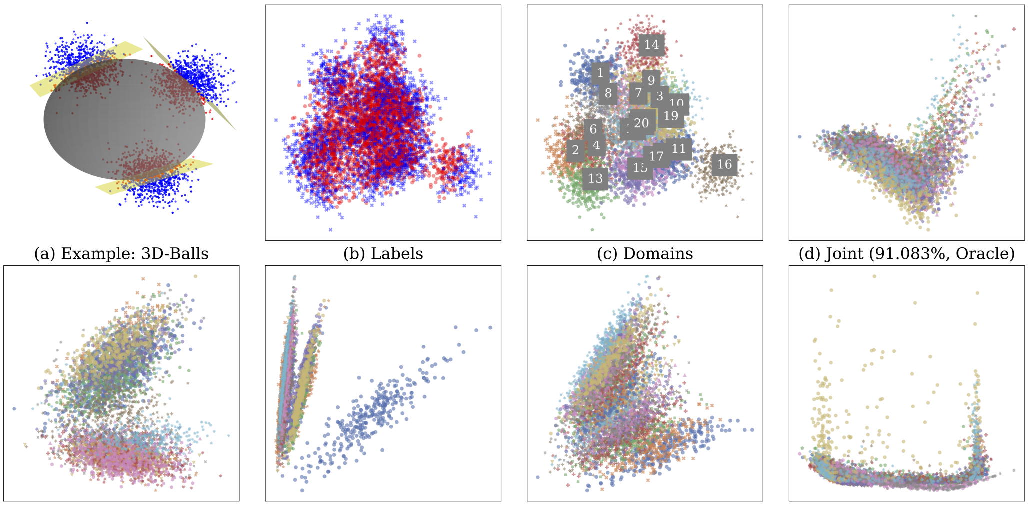
$V_{\text{0-1}}$ is the loss for **finding the optimal coefficient set** Ω. Its loss terms use Definition 3.1 and Eqn. 12 to estimate ERM terms and $\mathcal{H}\Delta\mathcal{H}$-divergence, respectively:

$$\begin{aligned} V_{\text{0-1}}(\mathring{h}, \Omega) = & \sum\nolimits_{i=1}^{t-1} \left[\gamma_i \widehat{\epsilon}_{\mathcal{D}_i}(\mathring{h}) + \alpha_i \widehat{\epsilon}_{\mathcal{D}_i}(\mathring{h}, H_{t-1}) \right] + (\sum\nolimits_{i=1}^{t-1} \beta_i) \widehat{\epsilon}_{\mathcal{D}_t}(\mathring{h}, H_{t-1}) \\ & + \tfrac{1}{2} \sum\nolimits_{i=1}^{t-1} \beta_i \widehat{d}_{\mathcal{H}\Delta\mathcal{H}} \left(\mathring{e}(\mathcal{X}_i), \mathring{e}(\mathcal{S}_t) \right) + \sum\nolimits_{i=1}^{t-1} (\alpha_i + \beta_i) \widehat{\epsilon}_{\mathcal{D}_i}(H_{t-1}) \\ & + C \cdot \sqrt{\left(\tfrac{(1 + \sum_{i=1}^{t-1} \beta_i)^2}{N_t} + \sum\nolimits_{i=1}^{t-1} \tfrac{(\gamma_i + \alpha_i)^2}{\widetilde{N}_i} \right)}. \end{aligned} \tag{15}$$

In Eqn. 15, $\widehat{\epsilon}_\cdot(\cdot)$ uses *discrete 0-1 loss*, which is different from Eqn. 14, and a hyper-parameter $C = \sqrt{8d \log(2eN/d) + 8 \log(2/\delta)}$ is introduced to model the combined influence of $\mathcal{H}$'s VC-dimension and δ.

V_d follows Eqn. 12 to **minimize the divergence between different domains' embedding distributions** (i.e., aligning domains) by the minimax game between e and d with the value function:

$$V_d(d, e, \mathring{\Omega}) = \left(\sum_{i=1}^{t-1} \mathring{\beta}_i \right) \tfrac{1}{N_i} \sum_{\boldsymbol{x} \in \mathcal{S}_t} \left[-\log\left([d(e(\boldsymbol{x}))]_t \right) \right] + \sum_{i=1}^{t-1} \tfrac{\mathring{\beta}_i}{\widetilde{N}_i} \sum_{\boldsymbol{x} \in \widetilde{\mathcal{X}}_i} \left[-\log\left([d(e(\boldsymbol{x}))]_i \right) \right]. \tag{16}$$



(a) Example: 3D-Balls (b) Labels (c) Domains (d) Joint (91.083%, Oracle)

(e) ER (82.255%) (f) DER++ (79.322%) (g) CLS-ER (85.844%) (h) UDIL (86.872%, Ours)

Figure 1: Results on *HD-Balls*. In (a-b), data is colored according to labels; in (c-h), data is colored according to domain ID. All data is plotted after PCA [6]. **(a)** Simplified *HD-Balls* dataset with 3 domains in the 3D space (for visualization purposes only). **(b-c)** Embeddings of *HD-Balls*'s raw data colored by labels and domain ID. **(d-h)** Accuracy and embeddings learned by Joint (oracle), UDIL, and three best baselines (more in Appendix C.5). Joint, as the *oracle*, naturally aligns different domains, and UDIL outperforms all baselines in terms of embedding alignment and accuracy.

Here in Eqn. 16, if an optimal d^* and a fixed Ω is given, maximizing $V_d(d^*, e, \Omega)$ with respect to the encoder e is equivalent to minimizing the weighted sum of the divergence $\sum_{i=1}^{t-1} \beta_i d_{\mathcal{H}\Delta\mathcal{H}}(e(\mathcal{D}_i), e(\mathcal{D}_t))$. This result indicates that the divergence between two domains' *embedding distributions* can be actually minimized. Intuitively this minimax game learns an encoder e that aligns the embedding distributions of different domains so that their domain IDs can not be predicted (distinguished) by a powerful discriminator given an embedding $e(\boldsymbol{x})$. Algorithm 1 below outlines how UDIL minimizes the tightest bound. Please refer to Appendix C for more implementation details, including a model diagram in Fig. 2.

5 Experiments

In this section, we compare UDIL with existing methods on both synthetic and real-world datasets.

5.1 Baselines and Implementation Details

We compare UDIL with the state-of-the-art continual learning methods that are either specifically designed for domain incremental learning or can be easily adapted to the domain incremental learning setting. For fair comparison, we do not consider methods that leverage large-scale pre-training or prompt-tuning [99, 98, 53, 88]. Exemplar-free baselines include online Elastic Weight Consolidation (**oEWC**) [81], Synaptic Intelligence (**SI**) [105], and Learning without Forgetting (**LwF**) [52]. Memory-based domain incremental learning baselines include Gradient Episodic Memory (**GEM**) [58], Averaged Gradient Episodic Memory (**A-GEM**) [10], Experience Replay (**ER**) [75], Dark Experience Replay (**DER++**) [8], and two recent methods, Complementary Learning System based Experience Replay (**CLS-ER**) [4] and Error Senesitivity Modulation based Experience Replay (**ESM-ER**) [80] (see Appendix C.5 for more detailed introduction to the baseline methods above). In addition, we implement the fine-tuning (**Fine-tune**) [52] and joint-training (**Joint**) as the performance lower bound and upper bound (Oracle).

We train all models using three different random seeds and report the mean and standard deviation. All methods are implemented with PyTorch [70], based on the mammoth code base [7, 8], and run on a single NVIDIA RTX A5000 GPU. For fair comparison, within the same dataset, all methods adopt the same neural network architecture, and the memory sampling strategy is set to random

Table 2: **Performances (%) on *HD-Balls*, *P-MNIST*, and *R-MNIST*.** We use two metrics, Average Accuracy and Forgetting, to evaluate the methods' effectiveness. "↑" and "↓" mean higher and lower numbers are better, respectively. We use **boldface** and underlining to denote the best and the second-best performance, respectively. We use "-" to denote "not appliable".

Method	**Buffer**	***HD-Balls*** Avg. Acc (↑)	***HD-Balls*** Forgetting (↓)	***P-MNIST*** Avg. Acc (↑)	***P-MNIST*** Forgetting (↓)	***R-MNIST*** Avg. Acc (↑)	***R-MNIST*** Forgetting (↓)
Fine-tune	-	52.319±0.024	43.520±0.079	70.102±2.945	27.522±3.042	47.803±1.703	52.281±1.797
oEWC [81]	-	54.131±0.193	39.743±1.388	78.476±1.223	18.068±1.321	48.203±0.827	51.181±0.867
SI [105]	-	52.303±0.037	43.175±0.041	79.045±1.357	17.409±1.446	48.251±1.381	51.053±1.507
LwF [52]	-	51.523±0.065	25.155±0.264	73.545±2.646	24.556±2.789	54.709±0.515	45.473±0.565
GEM [58]	400	69.747±0.656	13.591±0.779	89.097±0.149	6.975±0.167	76.619±0.581	21.289±0.579
A-GEM [10]		62.777±0.295	12.878±1.588	87.560±0.087	8.577±0.053	59.654±0.122	39.196±0.171
ER [75]		82.255±1.552	9.524±1.655	88.339±0.044	7.180±0.029	76.794±0.696	20.696±0.744
DER++ [8]		79.332±1.347	13.762±1.514	**92.950±0.361**	3.378±0.245	84.258±0.544	13.692±0.560
CLS-ER [4]		85.844±0.165	5.297±0.281	91.598±0.117	3.795±0.144	81.771±0.354	15.455±0.356
ESM-ER [80]		71.995±3.833	13.245±5.397	89.829±0.698	6.888±0.738	82.192±0.164	16.195±0.150
UDIL (Ours)		**86.872±0.195**	**3.428±0.359**	92.666±0.108	**2.853±0.107**	**86.635±0.686**	**8.506±1.181**
Joint (Oracle)	∞	91.083±0.332	-	96.368±0.042	-	97.150±0.036	-

balanced sampling (see Appendix C.2 and Appendix C.6 for more implementation details on training). We evaluate all methods with standard continual learning metrics including 'average accuracy', 'forgetting', and 'forward transfer' (see Appendix C.4 for detailed definitions).

5.2 Toy Dataset: High-Dimensional Balls

To gain insight into UDIL, we start with a toy dataset, high dimensional balls on a sphere (referred to as *HD-Balls* below), for domain incremental learning. *HD-Balls* includes 20 domains, each containing 2,000 data points sampled from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, 0.2^2\boldsymbol{I})$. The mean $\boldsymbol{\mu}$ is randomly sampled from a 100-dimensional unit sphere, i.e., $\{\boldsymbol{\mu} \in \mathbb{R}^{100} : \|\boldsymbol{\mu}\|_2 = 1\}$; the covariance matrix Σ is fixed. In *HD-Balls*, each domain represents a binary classification task, where the decision boundary is the hyperplane that passes the center $\boldsymbol{\mu}$ and is tangent to the unit sphere. Fig. 1(a-c) shows some visualization on *HD-Balls*.

Column 3 and 4 of Table 2 compare the performance of our UDIL with different baselines. We can see that UDIL achieves the highest final average accuracy and the lowest forgetting. Fig. 1(d-h) shows the embedding distributions (i.e., $e(\boldsymbol{x})$) for different methods. We can see better embedding alignment across domains generally leads to better performance. Specifically, Joint, as the oracle, naturally aligns different domains' embedding distributions and achieves an accuracy upper bound of 91.083%. Similarly, our UDIL can adaptively adjust the coefficients of different loss terms, including Eqn. 12, successfully align different domains, and thereby outperform all baselines.

5.3 Permutation MNIST

We further evaluate our method on the Permutation MNIST (*P-MNIST*) dataset [50]. *P-MNIST* includes 20 sequential domains, with each domain constructed by applying a fixed random permutation to the pixels in the images. Column 5 and 6 of Table 2 show the results of different methods. Our UDIL achieves the second best (92.666%) final average accuracy, which is only 0.284% lower than the best baseline DER++. We believe this is because (i) there is not much space for improvement as the gap between joint-training (oracle) and most methods are small; (ii) under the permutation, different domains' data distributions are too distinct from each other, lacking the meaningful relations among the domains, and therefore weakens the effect of embedding alignment in our method. Nevertheless, UDIL still achieves best performance in terms of forgetting (2.853%). This is mainly because our unified UDIL framework (i) is directly derived from the total loss of *all* domains, and (ii) uses adaptive coefficients to achieve a more balanced trade-off between learning the current domain and avoiding forgetting previous domains.

https://doi.org/10.52202/075280-0660

Table 3: **Performances (%) evaluated on *Seq-CORe50*.** We use three metrics, Average Accuracy, Forgetting, and Forward Transfer, to evaluate the methods' effectiveness. "↑" and "↓" mean higher and lower numbers are better, respectively. We use **boldface** and underlining to denote the best and the second-best performance, respectively. We use "-" to denote "not appliable" and "⋆" to denote out-of-memory (*OOM*) error when running the experiments.

Method	Buffer	$\mathcal{D}_{1:3}$	$\mathcal{D}_{4:6}$	$\mathcal{D}_{7:9}$	$\mathcal{D}_{10:11}$	Overall		
		Avg. Acc (↑)				Avg. Acc (↑)	Forgetting (↓)	Fwd. Transfer (↑)
Fine-tune	-	73.707±13.144	34.551±1.254	29.406±2.579	28.689±3.144	31.832±1.034	73.296±1.399	15.153±0.255
oEWC [81]	-	74.567±13.360	35.915±0.260	30.174±3.195	28.291±2.522	30.813±1.154	74.563±0.937	15.041±0.249
SI [105]	-	74.661±14.162	34.345±1.001	30.127±2.971	28.839±3.631	32.469±1.315	73.144±1.588	14.837±1.005
LwF [52]	-	80.383±10.190	28.357±1.143	31.386±0.787	28.711±2.981	31.692±0.768	72.990±1.350	15.356±0.750
GEM [58]	500	79.852±6.864	38.961±1.718	39.258±2.614	36.859±0.842	37.701±0.273	22.724±1.554	19.030±0.936
A-GEM [10]		80.348±9.394	41.472±3.394	43.213±1.542	39.181±3.999	43.181±2.025	33.775±3.003	19.033±0.792
ER [75]		90.838±2.177	79.343±2.699	68.151±0.226	65.034±1.571	66.605±0.214	32.750±0.455	21.735±0.802
DER++ [8]		92.444±1.764	88.652±1.854	80.391±0.107	78.038±0.591	78.629±0.753	21.910±1.094	22.488±1.049
CLS-ER [4]		89.834±1.323	78.909±1.724	70.591±0.322	⋆	⋆	⋆	⋆
ESM-ER [80]		84.905±6.471	51.905±3.257	53.815±1.770	50.178±2.574	52.751±1.296	25.444±0.580	21.435±1.018
UDIL (Ours)		**98.152±1.665**	**89.814±2.302**	**83.052±0.151**	**81.547±0.269**	**82.103±0.279**	**19.589±0.303**	**31.215±0.831**
Joint (Oracle)	∞	-	-	-	-	99.137±0.049	-	-

5.4 Rotating MNIST

We also evaluate our method on the Rotating MNIST dataset (*R-MNIST*) containing 20 sequential domains. Different from *P-MNIST* where shift from domain t to domain $t+1$ is abrupt, *R-MNIST*'s domain shift is gradual. Specifically, domain t's images are rotated by an angle randomly sampled from the range $[9° \cdot (t-1), 9° \cdot t)$. Column 7 and 8 of Table 2 show that our UDIL achieves the highest average accuracy (86.635%) and the lowest forgetting (8.506%) simultaneously, significantly improving on the best baseline DER++ (average accuracy of 84.258% and forgetting of 13.692%). Interestingly, such improvement is achieved when our UDIL's β_i is high, which further verifies that UDIL indeed leverages the similarities shared across different domains so that the generalization error is reduced.

5.5 Sequential CORe50

CORe50 [55, 56] is a real-world continual object recognition dataset that contains 50 domestic objects collected from 11 domains (120,000 images in total). Prior work has used CORe50 for settings such as domain generalization (e.g., train a model on only 8 domains and test it on 3 domains), which is different from our domain-incremental learning setting. To focus the evaluation on alleviating catastrophic forgetting, we retain 20% of the data as the test set and continually train the model on these 11 domains; we therefore call this dataset variant *Seq-CORe50*. Table 3 shows that our UDIL outperforms all baselines in every aspect on *Seq-CORe50*. Besides the average accuracy over all domains, we also report average accuracy over different domain intervals (e.g., $\mathcal{D}_{1:3}$ denotes average accuracy from domain 1 to domain 3) to show how different model's performance drops over time. The results show that our UDIL consistently achieves the highest average accuracy until the end. It is also worth noting that UDIL also achieves the best performance on another two metrics, i.e., forgetting and forward transfer.

6 Conclusion

In this paper, we propose a principled framework, UDIL, for domain incremental learning with memory to unify various existing methods. Our theoretical analysis shows that different existing methods are equivalent to minimizing the same error bound with different *fixed* coefficients. With this unification, our UDIL allows *adaptive* coefficients during training, thereby always achieving the tightest bound and improving the performance. Empirical results show that our UDIL outperforms the state-of-the-art domain incremental learning methods on both synthetic and real-world datasets. One limitation of this work is the implicit *i.i.d.* exemplar assumption, which may not hold if memory is selected using specific strategies. Addressing this limitation can lead to a more powerful unified framework and algorithms, which would be interesting future work.

https://doi.org/10.52202/075280-0660

Acknowledgement

The authors thank the reviewers/AC for the constructive comments to improve the paper. HS and HW are partially supported by NSF Grant IIS-2127918. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

[1] H. Ahn, S. Cha, D. Lee, and T. Moon. Uncertainty-based continual learning with adaptive regularization. *Advances in neural information processing systems*, 32, 2019.

[2] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[3] M. Anthony, P. L. Bartlett, P. L. Bartlett, et al. *Neural network learning: Theoretical foundations*, volume 9. cambridge university press Cambridge, 1999.

[4] E. Arani, F. Sarfraz, and B. Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*, 2022.

[5] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.

[6] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[7] M. Boschini, L. Bonicelli, P. Buzzega, A. Porrello, and S. Calderara. Class-incremental continual learning into the extended der-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[8] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

[9] H. Cha, J. Lee, and J. Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 9516–9525, 2021.

[10] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[11] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[13] T. Chen, H. Shi, S. Tang, Z. Chen, F. Wu, and Y. Zhuang. Cil: Contrastive instance learning framework for distantly supervised relation extraction. *arXiv preprint arXiv:2106.10855*, 2021.

[14] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[15] Z. Chen and B. Liu. *Lifelong machine learning*, volume 1. Springer, 2018.

[16] Z. Chen, J. Zhuang, X. Liang, and L. Lin. Blending-target domain adaptation by adversarial meta-adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2248–2257, 2019.

[17] S. Dai, K. Sohn, Y.-H. Tsai, L. Carin, and M. Chandraker. Adaptation across extreme variations using unlabeled domain bridges. *arXiv preprint arXiv:1906.02238*, 2019.

[18] Y. Dai, H. Lang, Y. Zheng, B. Yu, F. Huang, and Y. Li. Domain incremental lifelong learning in an open world. *arXiv preprint arXiv:2305.06555*, 2023.

[19] M. Davari, N. Asadi, S. Mudur, R. Aljundi, and E. Belilovsky. Probing representation forgetting in supervised and unsupervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16712–16721, 2022.

https://doi.org/10.52202/075280-0660

[20] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[21] D. Deng, G. Chen, J. Hao, Q. Wang, and P.-A. Heng. Flattening sharpness for dynamic gradient projection memory benefits continual learning. *Advances in Neural Information Processing Systems*, 34:18710–18721, 2021.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[23] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5138–5146, 2019.

[24] S. Ebrahimi, M. Elhoseiny, T. Darrell, and M. Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. *arXiv preprint arXiv:1906.02425*, 2019.

[25] J. Gallardo, T. L. Hayes, and C. Kanan. Self-supervised training enhances online continual learning. *arXiv preprint arXiv:2103.14010*, 2021.

[26] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[27] P. Garg, R. Saluja, V. N. Balasubramanian, C. Arora, A. Subramanian, and C. Jawahar. Multi-domain incremental learning for semantic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 761–771, 2022.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[29] F. Graf, C. Hofer, M. Niethammer, and R. Kwitt. Dissecting supervised contrastive learning. In *International Conference on Machine Learning*, pages 3821–3830. PMLR, 2021.

[30] Y. Guo, B. Liu, and D. Zhao. Online continual learning through mutual information maximization. In *International Conference on Machine Learning*, pages 8109–8126. PMLR, 2022.

[31] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 87–102. Springer, 2016.

[32] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[33] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[35] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[36] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[37] W. Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

[38] C.-Y. Hung, C.-H. Tu, C.-E. Wu, C.-H. Chen, Y.-M. Chan, and C.-S. Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[39] D. Jung, D. Lee, S. Hong, H. Jang, H. Bae, and S. Yoon. New insights for the stability-plasticity dilemma in online continual learning. *arXiv preprint arXiv:2302.08741*, 2023.

[40] H. Jung, J. Ju, M. Jung, and J. Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016.

https://doi.org/10.52202/075280-0660

[41] T. Kalb, M. Roschani, M. Ruf, and J. Beyerer. Continual learning for class-and domain-incremental semantic segmentation. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 1345–1351. IEEE, 2021.

[42] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[43] D. Kim and B. Han. On the stability-plasticity dilemma of class-incremental learning. *arXiv preprint arXiv:2304.01663*, 2023.

[44] G. Kim, C. Xiao, T. Konishi, Z. Ke, and B. Liu. A theoretical study on solving continual learning. *Advances in Neural Information Processing Systems*, 35:5065–5079, 2022.

[45] G. Kim, C. Xiao, T. Konishi, and B. Liu. Learnability and algorithm for continual learning. *arXiv preprint arXiv:2306.12646*, 2023.

[46] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[47] V. Kothapalli, E. Rasromani, and V. Awatramani. Neural collapse: A review on modelling principles and generalization. *arXiv preprint arXiv:2206.04041*, 2022.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[49] R. Kurle, B. Cseke, A. Klushyn, P. Van Der Smagt, and S. Günnemann. Continual learning with bayesian neural networks for non-stationary data. In *International Conference on Learning Representations*, 2019.

[50] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[51] M. Li, H. Zhang, J. Li, Z. Zhao, W. Zhang, S. Zhang, S. Pu, Y. Zhuang, and F. Wu. Unsupervised domain adaptation for video object grounding with cascaded debiasing learning. In *Proceedings of the 31th ACM International Conference on Multimedia*, 2023.

[52] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[53] Z. Li, L. Zhao, Z. Zhang, H. Zhang, D. Liu, T. Liu, and D. N. Metaxas. Steering prototype with prompt-tuning for rehearsal-free continual learning. *arXiv preprint arXiv:2303.09447*, 2023.

[54] T. Liu, Z. Xu, H. He, G. Hao, G.-H. Lee, and H. Wang. Taxonomy-structured domain adaptation. In *ICML*, 2023.

[55] V. Lomonaco and D. Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In S. Levine, V. Vanhoucke, and K. Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR, 13–15 Nov 2017.

[56] V. Lomonaco, D. Maltoni, and L. Pellegrini. Rehearsal-free continual learning over small non-iid batches. In *CVPR Workshops*, volume 1, page 3, 2020.

[57] M. Long, Z. CAO, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[58] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[59] U. Michieli and P. Zanuttigh. Knowledge distillation for incremental learning in semantic segmentation. *Computer Vision and Image Understanding*, 205:103167, 2021.

[60] M. J. Mirza, M. Masana, H. Possegger, and H. Bischof. An efficient domain-incremental learning approach to drive in all weather conditions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3001–3011, 2022.

[61] S. I. Mirzadeh, M. Farajtabar, R. Pascanu, and H. Ghasemzadeh. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33:7308–7320, 2020.

https://doi.org/10.52202/075280-0660

[62] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[63] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

[64] L. T. Nguyen-Meidine, A. Belal, M. Kiran, J. Dolz, L.-A. Blais-Morin, and E. Granger. Unsupervised multi-target domain adaptation through knowledge distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1339–1347, 2021.

[65] Z. Ni, H. Shi, S. Tang, L. Wei, Q. Tian, and Y. Zhuang. Revisiting catastrophic forgetting in class incremental learning. *arXiv preprint arXiv:2107.12308*, 2021.

[66] Z. Ni, L. Wei, S. Tang, Y. Zhuang, and Q. Tian. Continual vision-language representation learning with off-diagonal information, 2023.

[67] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE transactions on neural networks*, 22(2):199–210, 2010.

[68] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[69] V. Papyan, X. Han, and D. L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[70] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[71] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019.

[72] Q. Pham, C. Liu, and S. Hoi. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34:16131–16144, 2021.

[73] R. Ramesh and P. Chaudhari. Model zoo: A growing" brain" that learns continually. *arXiv preprint arXiv:2106.03027*, 2021.

[74] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[75] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[76] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[77] G. Saha, I. Garg, and K. Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021.

[78] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3723–3732, 2018.

[79] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8503–8512, 2018.

[80] F. Sarfraz, E. Arani, and B. Zonooz. Error sensitivity modulation based experience replay: Mitigating abrupt representation drift in continual learning. *arXiv preprint arXiv:2302.11344*, 2023.

[81] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pages 4528–4537. PMLR, 2018.

[82] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

https://doi.org/10.52202/075280-0660

[83] H. Shi, D. Luo, S. Tang, J. Wang, and Y. Zhuang. Run away from your teacher: Understanding byol by a novel self-supervised approach. *arXiv preprint arXiv:2011.10944*, 2020.

[84] H. Shi, Y. Zhang, Z. Shen, S. Tang, Y. Li, Y. Guo, and Y. Zhuang. Towards communication-efficient and privacy-preserving federated representation learning. *arXiv preprint arXiv:2109.14611*, 2021.

[85] H. Shi, Y. Zhang, S. Tang, W. Zhu, Y. Li, Y. Guo, and Y. Zhuang. On the efficacy of small self-supervised contrastive models without distillation signals. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2225–2234, 2022.

[86] M. S. Sorower. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18(1):25, 2010.

[87] B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 443–450. Springer, 2016.

[88] V. Thengane, S. Khan, M. Hayat, and F. Khan. Clip model is an efficient continual learner. *arXiv preprint arXiv:2210.03114*, 2022.

[89] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[90] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, pages 1–13, 2022.

[91] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Measures of complexity: festschrift for alexey chervonenkis*, pages 11–30, 2015.

[92] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[93] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[94] H. Wang, H. He, and D. Katabi. Continuously indexed domain adaptation. *arXiv preprint arXiv:2007.01807*, 2020.

[95] L. Wang, X. Zhang, Q. Li, J. Zhu, and Y. Zhong. Coscl: Cooperation of small continual learners is stronger than a big one. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*, pages 254–271. Springer, 2022.

[96] L. Wang, X. Zhang, H. Su, and J. Zhu. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*, 2023.

[97] Y. Wang, Z. Huang, and X. Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *arXiv preprint arXiv:2207.12819*, 2022.

[98] Z. Wang, Z. Zhang, S. Ebrahimi, R. Sun, H. Zhang, C.-Y. Lee, X. Ren, G. Su, V. Perot, J. Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer, 2022.

[99] Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022.

[100] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.

[101] Z. Xu, G.-Y. Hao, H. He, and H. Wang. Domain-indexing variational bayes: Interpretable domain index for domain adaptation. In *International Conference on Learning Representations*, 2023.

[102] Z. Xu, G.-H. Lee, Y. Wang, H. Wang, et al. Graph-relational domain adaptation. *arXiv preprint arXiv:2202.03628*, 2022.

[103] C. Yaras, P. Wang, Z. Zhu, L. Balzano, and Q. Qu. Neural collapse with normalized features: A geometric analysis over the riemannian manifold. *Advances in neural information processing systems*, 35:11547–11560, 2022.

https://doi.org/10.52202/075280-0660

[104] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[105] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

[106] M.-L. Zhang and Z.-H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[107] M.-L. Zhang and Z.-H. Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[108] Y. Zhang, T. Liu, M. Long, and M. Jordan. Bridging theory and algorithm for domain adaptation. In *International conference on machine learning*, pages 7404–7413. PMLR, 2019.

[109] M. Zhao, S. Yue, D. Katabi, T. S. Jaakkola, and M. T. Bianchi. Learning sleep stages from radio signals: A conditional adversarial architecture. In *International Conference on Machine Learning*, pages 4100–4109. PMLR, 2017.

[110] J. Zhou, C. You, X. Li, K. Liu, S. Liu, Q. Qu, and Z. Zhu. Are all losses created equal: A neural collapse perspective. *Advances in Neural Information Processing Systems*, 35:31697–31710, 2022.

[111] Z. Zhu, T. Ding, J. Zhou, X. Li, C. You, J. Sulam, and Q. Qu. A geometric analysis of neural collapse with unconstrained features. *Advances in Neural Information Processing Systems*, 34:29820–29834, 2021.

https://doi.org/10.52202/075280-0660

Appendix

In Sec. A, we present the proofs for the lemmas, theorems, and corollaries presented in the main body of our work. Sec. B discusses the correspondence of existing methods to specific cases within our framework. In Sec. C, we provide a detailed presentation of our final algorithm, UDIL, including an algorithmic description, a visual diagram, and implementation details. We introduce the experimental settings, including the evaluation metrics and specific training schemes. Finally, in Sec. D, we present additional empirical results with varying memory sizes and provide more visualization results.

A Proofs of Lemmas, Theorems, and Corollaries

Before proceeding to prove any lemmas or theorems, we first introduce three crucial additional lemmas that will be utilized in the subsequent sections. Among these, Lemma A.1 offers a generalization bound for any weighted summation of ERM losses across multiple domains. Furthermore, Lemma A.2 provides a generalization bound for a weighted summation of *labeling functions* within a given domain. Lastly, we highlight Lemma 3 in [5] as Lemma A.3, which will be used to establish the upper bound for Lemma 3.3.

Lemma A.1 (**Generalization Bound of α-weighted Domains**). *Let $\mathcal{H}$ be a hypothesis space of VC dimension d. Assume N_j denotes the number of the samples collected from domain j, and $N = \sum_j N_j$ is the total number of the examples collected from all domains. Then for any $\alpha_j > 0$ and $\delta \in (0,1)$, with probability at least $1-\delta$:*

$$\sum_j \alpha_j \epsilon_{\mathcal{D}_j}(h) \leq \sum_j \alpha_j \widehat{\epsilon}_{\mathcal{D}_j}(h) + \sqrt{\big(\sum_j \tfrac{\alpha_j^2}{N_j}\big)\left(8d\log\left(\tfrac{2eN}{d}\right) + 8\log\left(\tfrac{2}{\delta}\right)\right)}. \tag{17}$$

Proof. Suppose each domain $\mathcal{D}_j$ has a deterministic ground-truth labeling function $f_j : \mathbb{R}^n \to \{0,1\}$. Denote as $\widehat{\epsilon}_\alpha \triangleq \sum_j \alpha_j \widehat{\epsilon}_{\mathcal{D}_j}(h)$ the α-weighted empirical loss evaluated on different domains. Hence,

$$\widehat{\epsilon}_\alpha(h) = \sum_j \alpha_j \widehat{\epsilon}_{\mathcal{D}_j}(h) = \sum_j \alpha_j \tfrac{1}{N_j} \sum_{\boldsymbol{x}\in\mathcal{X}_j} \mathbb{1}_{h(\boldsymbol{x})\neq f_j(\boldsymbol{x})} = \tfrac{1}{N}\sum_j \sum_{k=1}^{N_j} R_{j,k}, \tag{18}$$

where $R_{j,k} = (\frac{\alpha_j N_j}{N}) \cdot \mathbb{1}_{h(\boldsymbol{x}_k)\neq f_j(\boldsymbol{x}_k)}$ is a random variable that takes the values in $\{\frac{\alpha_j N_j}{N}, 0\}$. By the linearity of the expectation, we have $\epsilon_\alpha(h) = \mathbb{E}[\widehat{\epsilon}_\alpha(h)]$. Following [3, 62], we have

$$\begin{aligned}
& \mathbb{P}\left\{\exists h \in \mathcal{H}, \text{s.t. } |\widehat{\epsilon}_\alpha(h) - \epsilon_\alpha(h)| \geq \epsilon\right\} && (19)\\
\leq\; & 2\cdot\mathbb{P}\left\{\sup_{h\in\mathcal{H}} |\widehat{\epsilon}_\alpha(h) - \widehat{\epsilon}'_\alpha(h)| \geq \tfrac{\epsilon}{2}\right\} && (20)\\
\leq\; & 2\cdot\mathbb{P}\left\{\bigcup_{R_{j,k},R'_{j,k}} \tfrac{1}{N}\left|\sum_j\sum_{k=1}^{N_j}(R_{j,k} - R'_{j,k})\right| \geq \tfrac{\epsilon}{2}\right\} && (21)\\
\leq\; & 2\Pi_{\mathcal{H}}(2N)\exp\left\{\frac{-2(^{N\epsilon}/_2)^2}{\sum_j (N_j)(^{2\alpha_j N}/_{N_j})^2}\right\} && (22)\\
=\; & 2\Pi_{\mathcal{H}}(2N)\exp\left\{-\frac{\epsilon^2}{8\sum_j(^{\alpha_j^2}/_{N_j})}\right\} && (23)\\
\leq\; & 2(2N)^d\exp\left\{-\frac{\epsilon^2}{8\sum_j(^{\alpha_j^2}/_{N_j})}\right\}, && (24)
\end{aligned}$$

where in Eqn. 20, $\widehat{\epsilon}'_\alpha(h)$ is the α-weighted empirical loss evaluated on the "ghost" set of examples $\{\mathcal{X}'_j\}$; Eqn. 22 is yielded by applying Hoeffding's inequalities [37] and introducing the growth function $\Pi_{\mathcal{H}}$ [3, 62, 91] at the same time; Eqn. 24 is achieved by using the fact $\Pi_{\mathcal{H}}(2N) \leq (^{e\cdot 2N}/_d)^d \leq (2N)^d$, where d is the VC-dimension of the hypothesis set $\mathcal{H}$. Finally, by setting Eqn. 24 to δ and solve for the error tolerance ϵ will complete the proof. □

Lemma A.2 (**Generalization Bound of β-weighted Labeling Functions**). *Let $\mathcal{D}$ be a single domain and $\mathcal{X} = \{\boldsymbol{x}_i\}_i^N$ be a collection of samples drawn from $\mathcal{D}$; $\mathcal{H}$ is a hypothesis space of VC dimension*

https://doi.org/10.52202/075280-0660

d. Suppose $\{f_j : \mathbb{R}^n \to \{0,1\}\}_j$ is a set of different labeling functions. Then for any $\beta_j > 0$ and $\delta \in (0,1)$, with probability at least $1-\delta$:

$$\sum_j \beta_j \epsilon_{\mathcal{D}}(h, f_j) \leq \sum_j \beta_j \widehat{\epsilon}_{\mathcal{D}}(h, f_j) + (\sum_j \beta_j)\sqrt{\tfrac{1}{N}\left(8d\log\left(\tfrac{2eN}{d}\right) + 8\log\left(\tfrac{2}{\delta}\right)\right)}. \tag{25}$$

Proof. Denote as $\epsilon_\beta(h) \triangleq \sum_j \beta_j \epsilon_{\mathcal{D}}(h, f_j)$ the β-weighted error on domain $\mathcal{D}$ and $\{f_j\}_j$ the set of the labeling functions , and $\widehat{\epsilon}_\beta \triangleq \sum_j \beta_j \widehat{\epsilon}_{\mathcal{D}}(h, f_j)$ as the β-weighted empirical loss evaluated on different labeling functions. We have

$$\widehat{\epsilon}_\beta(h) = \sum_j \beta_j \tfrac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{h(\boldsymbol{x}_i)\neq f_j(\boldsymbol{x}_i)} = \tfrac{1}{N}\sum_{i=1}^{N}\sum_j \beta_j \mathbb{1}_{h(\boldsymbol{x}_i)\neq f_j(\boldsymbol{x}_i)} \triangleq \tfrac{1}{N}\sum_{i=1}^{N} R_i, \tag{26}$$

where $R_i = \sum_j \beta_j \mathbb{1}_{h(\boldsymbol{x}_i)\neq f_j(\boldsymbol{x}_i)} \in [0, \sum_j \beta_j]$ is a new random variable.

Then we have

$$\begin{aligned}
& \mathbb{P}\left\{\exists h \in \mathcal{H}, \text{s.t. } |\widehat{\epsilon}_\beta(h) - \epsilon_\beta(h)| \geq \epsilon\right\} && (27)\\
\leq\quad & 2\cdot\mathbb{P}\left\{\sup_{h\in\mathcal{H}} |\widehat{\epsilon}_\beta(h) - \widehat{\epsilon}'_\beta(h)| \geq \tfrac{\epsilon}{2}\right\} && (28)\\
\leq\quad & 2\cdot\mathbb{P}\left\{\bigcup_{R_i, R'_i} \tfrac{1}{N}\left|\sum_{i=1}^{N}(R_i - R'_i)\right| \geq \tfrac{\epsilon}{2}\right\} && (29)\\
\leq\quad & 2\Pi_{\mathcal{H}}(2N)\exp\left\{\tfrac{-2(N\epsilon/2)^2}{N\cdot(2\sum_j \beta_j)^2}\right\} && (30)\\
\leq\quad & 2(2N)^d\exp\left\{-\tfrac{N\epsilon^2}{8(\sum_j \beta_j)^2}\right\}, && (31)
\end{aligned}$$

where in Eqn. 28, $\widehat{\epsilon}'_\beta(h)$ is the β-weighted empirical loss evaluated on the "ghost" set of examples $\mathcal{X}'$; Eqn. 30 is yielded by applying Hoeffding's inequalities [37] and introducing the growth function $\Pi_{\mathcal{H}}$ [3, 62, 91] at the same time; Eqn. 31 is achieved by using the fact $\Pi_{\mathcal{H}}(2N) \leq (^{e\cdot 2N}/_{d})^d \leq (2N)^d$, where d is the VC-dimension of the hypothesis set $\mathcal{H}$. Finally, by setting Eqn. 31 to δ and solve for the error tolerance ϵ will complete the proof. □

Lemma A.2 asserts that altering or merging multiple target functions does not impact the generalization error term, as long as the sum of the weights for each loss $\sum_j \beta_j$ remains constant and the same dataset $\mathcal{X}$ is used for estimation. Next we highligt the Lemma 3 in [5] again, as it will be utilized for proving 3.3.

Lemma A.3. *For any hypothesis $h, h' \in \mathcal{H}$ and any two different domains $\mathcal{D}, \mathcal{D}'$,*

$$|\epsilon_{\mathcal{D}}(h,h') - \epsilon_{\mathcal{D}'}(h,h')| \leq \tfrac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}, \mathcal{D}'). \tag{32}$$

Proof. By definition, we have

$$\begin{aligned}
d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}, \mathcal{D}') &= 2\sup_{h,h'\in\mathcal{H}} |\mathbb{P}_{\boldsymbol{x}\sim\mathcal{D}}[h(\boldsymbol{x}) \neq h'(\boldsymbol{x})] - \mathbb{P}_{\boldsymbol{x}\sim\mathcal{D}'}[h(\boldsymbol{x}) \neq h'(\boldsymbol{x})]| \\
&= 2\sup_{h,h'\in\mathcal{H}} |\epsilon_{\mathcal{D}}(h,h') - \epsilon_{\mathcal{D}'}(h,h')| \\
&\geq 2\,|\epsilon_{\mathcal{D}}(h,h') - \epsilon_{\mathcal{D}'}(h,h')|\,.
\end{aligned}$$

□

Now we are ready to prove the main lemmas and theorems in the main body of our work.

Lemma 3.1 (ERM-Based Generalization Bound). *Let $\mathcal{H}$ be a hypothesis space of VC dimension d. When domain t arrives, there are N_t data points from domain t and $\widetilde{N}_i$ data points from each previous domain $i < t$ in the memory bank. With probability at least $1-\delta$, we have:*

$$\sum_{i=1}^{t} \epsilon_{\mathcal{D}_i}(h) \leq \sum_{i=1}^{t} \widehat{\epsilon}_{\mathcal{D}_i}(h) + \sqrt{\left(\tfrac{1}{N_t} + \sum_{i=1}^{t-1}\tfrac{1}{\widetilde{N}_i}\right)\left(8d\log\left(\tfrac{2eN}{d}\right) + 8\log\left(\tfrac{2}{\delta}\right)\right)}. \tag{33}$$

Proof. Simply using Lemma A.1 and setting $\alpha_i = 1$ for every $i \in [t]$ completes the proof. □

Lemma 3.2 (Intra-Domain Model-Based Bound). *Let $h \in \mathcal{H}$ be an arbitrary function in the hypothesis space $\mathcal{H}$, and H_{t-1} be the model trained after domain $t-1$. The domain-specific error $\epsilon_{\mathcal{D}_i}(h)$ on the previous domain i has an upper bound:*

$$\epsilon_{\mathcal{D}_i}(h) \leq \epsilon_{\mathcal{D}_i}(h, H_{t-1}) + \epsilon_{\mathcal{D}_i}(H_{t-1}), \tag{34}$$

where $\epsilon_{\mathcal{D}_i}(h, H_{t-1}) \triangleq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_i}[h(\boldsymbol{x}) \neq H_{t-1}(\boldsymbol{x})]$.

Proof. By applying the triangle inequality [5] of the 0-1 loss function, we have

$$\begin{aligned}\epsilon_{\mathcal{D}_i}(h) &= \epsilon_{\mathcal{D}_i}(h, f_i)\\ &\leq \epsilon_{\mathcal{D}_i}(h, H_{t-1}) + \epsilon_{\mathcal{D}_i}(H_{t-1}, f_i)\\ &= \epsilon_{\mathcal{D}_i}(h, H_{t-1}) + \epsilon_{\mathcal{D}_i}(H_{t-1}).\end{aligned}$$

□

Lemma 3.3 (Cross-Domain Model-Based Bound). *Let $h \in \mathcal{H}$ be an arbitrary function in the hypothesis space $\mathcal{H}$, and H_{t-1} be the function trained after domain $t-1$. The domain-specific error $\epsilon_{\mathcal{D}_i}(h)$ evaluated on the previous domain i then has an upper bound:*

$$\epsilon_{\mathcal{D}_i}(h) \leq \epsilon_{\mathcal{D}_t}(h, H_{t-1}) + \tfrac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t) + \epsilon_{\mathcal{D}_i}(H_{t-1}), \tag{35}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{P}, \mathcal{Q}) = 2\sup_{h\in\mathcal{H}\Delta\mathcal{H}}|\Pr_{x\sim\mathcal{P}}[h(x)=1] - \Pr_{x\sim\mathcal{Q}}[h(x)=1]|$ denotes the $\mathcal{H}\Delta\mathcal{H}$-divergence between distribution $\mathcal{P}$ and $\mathcal{Q}$, and $\epsilon_{\mathcal{D}_t}(h, H_{t-1}) \triangleq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}[h(\boldsymbol{x}) \neq H_{t-1}(\boldsymbol{x})]$.

Proof. By the triangle inequality used above and Lemma A.3, we have

$$\begin{aligned}\epsilon_{\mathcal{D}_i}(h) &\leq \epsilon_{\mathcal{D}_i}(h, H_{t-1}) + \epsilon_{\mathcal{D}_i}(H_{t-1})\\ &= \epsilon_{\mathcal{D}_i}(h, H_{t-1}) + \epsilon_{\mathcal{D}_i}(H_{t-1}) - \epsilon_{\mathcal{D}_t}(h, H_{t-1}) + \epsilon_{\mathcal{D}_t}(h, H_{t-1})\\ &\leq \epsilon_{\mathcal{D}_i}(H_{t-1}) + |\epsilon_{\mathcal{D}_i}(h, H_{t-1}) - \epsilon_{\mathcal{D}_t}(h, H_{t-1})| + \epsilon_{\mathcal{D}_t}(h, H_{t-1})\\ &\leq \epsilon_{\mathcal{D}_t}(h, H_{t-1}) + \tfrac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t) + \epsilon_{\mathcal{D}_i}(H_{t-1}).\end{aligned}$$

□

Theorem 3.4 (Unified Generalization Bound for All Domains). *Let $\mathcal{H}$ be a hypothesis space of VC dimension d. Let $N = N_t + \sum_i^{t-1}\widetilde{N}_i$ denoting the total number of data points available to the training of current domain t, where N_t and $\widetilde{N}_i$ denote the numbers of data points collected at domain t and data points from the previous domain i in the memory bank, respectively. With probability at least $1-\delta$, we have:*

$$\begin{aligned}\sum_{i=1}^{t}\epsilon_{\mathcal{D}_i}(h) \leq &\left\{\sum_{i=1}^{t-1}[\gamma_i\widehat{\epsilon}_{\mathcal{D}_i}(h) + \alpha_i\widehat{\epsilon}_{\mathcal{D}_i}(h, H_{t-1})]\right\} + \left\{\widehat{\epsilon}_{\mathcal{D}_t}(h) + (\sum_{i=1}^{t-1}\beta_i)\widehat{\epsilon}_{\mathcal{D}_t}(h, H_{t-1})\right\}\\ &+ \tfrac{1}{2}\sum_{i=1}^{t-1}\beta_i d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t) + \sum_{i=1}^{t-1}(\alpha_i + \beta_i)\epsilon_{\mathcal{D}_i}(H_{t-1})\\ &+ \sqrt{\left(\frac{(1+\sum_{i=1}^{t-1}\beta_i)^2}{N_t} + \sum_{i=1}^{t-1}\frac{(\gamma_i+\alpha_i)^2}{\widetilde{N}_i}\right)\left(8d\log\left(\frac{2eN}{d}\right) + 8\log\left(\frac{2}{\delta}\right)\right)}\\ &\triangleq g(h, H_{t-1}, \Omega),\end{aligned} \tag{36}$$

where $\widehat{\epsilon}_{\mathcal{D}_i}(h, H_{t-1}) = \frac{1}{\widetilde{N}_i}\sum_{\boldsymbol{x}\in\widetilde{\mathcal{X}}_i}\mathbb{1}_{h(\boldsymbol{x})\neq H_{t-1}(\boldsymbol{x})}$, $\widehat{\epsilon}_{\mathcal{D}_t}(h, H_{t-1}) = \frac{1}{N_t}\sum_{\boldsymbol{x}\in\mathcal{X}_i}\mathbb{1}_{h(\boldsymbol{x})\neq H_{t-1}(\boldsymbol{x})}$, and $\Omega \triangleq \{\alpha_i, \beta_i, \gamma_i\}_{i=1}^{t-1}$.

Proof. By applying Lemma 3.2 and Lemma 3.3 to each of the past domains, we have

$$\begin{aligned}\epsilon_{\mathcal{D}_i}(h) &= (\alpha_i + \beta_i + \gamma_i)\epsilon_{\mathcal{D}_i}(h)\\ &\leq \gamma_i\epsilon_{\mathcal{D}_i}(h) + \alpha_i[\epsilon_{\mathcal{D}_i}(h, H_{t-1}) + \epsilon_{\mathcal{D}_i}(H_{t-1})]\\ &\quad + \beta_i[\epsilon_{\mathcal{D}_i}(h, H_{t-1}) + \epsilon_{\mathcal{D}_t}(h, H_{t-1}) + \tfrac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t)].\end{aligned}$$

Table 4: Unification of existing methods under UDIL, when certain conditions are achieved.

	α_i	β_i	γ_i	**Transformed Objective**	**Condition**
UDIL (Ours)	$[0,1]$	$[0,1]$	$[0,1]$	-	-
LwF [52]	0	1	0	$\mathcal{L}_{\text{LwF}}(h) = \widehat{\ell}_{\mathcal{X}_t}(h) + \lambda_o \widehat{\ell}_{\mathcal{X}_t}(h, H_{t-1})$	$\lambda_o = t-1$
ER [75]	0	0	1	$\mathcal{L}_{\text{ER}}(h) = \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \frac{\lvert B'_t \rvert/(t-1)}{\lvert B_t \rvert} \widehat{\ell}_{B'_i}(h)$	$\lvert B_t \rvert = \frac{\lvert B'_t \rvert}{(t-1)}$
DER++ [8]	$1/2$	0	$1/2$	$\mathcal{L}_{\text{DER++}}(h) = \widehat{\ell}_{B_t}(h) + \frac{1}{2} \sum_{i=1}^{t-1} \frac{\lvert B'_t \rvert/(t-1)}{\lvert B_t \rvert} [\widehat{\ell}_{B'_i}(h) + \widehat{\ell}_{B'_i}(h, H_{t-1})]$	$\lvert B_t \rvert = \frac{\lvert B'_t \rvert}{(t-1)}$
iCaRL [74]	1	0	0	$\mathcal{L}_{\text{iCaRL}}(h) = \widehat{\ell'}_{B_t}(h) + \sum_{i=1}^{t-1} \frac{\lvert B'_t \rvert/(t-1)}{\lvert B_t \rvert} \widehat{\ell'}_{B'_i}(h, H_{t-1})$	$\lvert B_t \rvert = \frac{\lvert B'_t \rvert}{(t-1)}$
CLS-ER [4]	$\frac{\lambda}{\lambda+1}$	0	$\frac{1}{\lambda+1}$	$\mathcal{L}_{\text{CLS-ER}}(h) = \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \frac{1}{t-1} \widehat{\ell}_{B'_i}(h) + \sum_{i=1}^{t-1} \frac{\lambda}{t-1} \widehat{\ell}_{B'_i}(h, H_{t-1})$	$\lambda = t-2$
ESM-ER [80]	$\frac{\lambda}{\lambda+1}$	0	$\frac{1}{\lambda+1}$	$\mathcal{L}_{\text{ESM-ER}}(h) = \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \frac{1}{r(t-1)} \widehat{\ell}_{B'_i}(h) + \sum_{i=1}^{t-1} \frac{\lambda}{r(t-1)} \widehat{\ell}_{B'_i}(h, H_{t-1})$	$\begin{cases} \lambda = -1 + r(t-1) \\ r = 1 - e^{-1} \end{cases}$
BiC [100]	$\frac{t-1}{2t-1}$	$\frac{t-1}{2t-1}$	$\frac{1}{2t-1}$	$\mathcal{L}_{\text{BiC}}(h) = \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \frac{(t-1)\lvert B_i \rvert}{\lvert B_t \rvert} \widehat{\ell}_{B'_i}(h, H_{t-1})$ $+ (t-1)\widehat{\ell}_{B_t}(h, H_{t-1}) + \sum_{i=1}^{t-1} \frac{\lvert B_i \rvert}{\lvert B_t \rvert} \widehat{\ell}_{B'_i}(h)$	$\lvert B_i \rvert = \lvert B_t \rvert$

Re-organizing the terms will give us

$$\sum_{i=1}^{t} \epsilon_{\mathcal{D}_i}(h) \leq \left\{ \sum_{i=1}^{t-1} [\gamma_i \epsilon_{\mathcal{D}_i}(h) + \alpha_i \epsilon_{\mathcal{D}_i}(h, H_{t-1})] \right\} + \left\{ \epsilon_{\mathcal{D}_t}(h) + (\sum_{i=1}^{t-1} \beta_i) \epsilon_{\mathcal{D}_t}(h, H_{t-1}) \right\}$$
$$+ \tfrac{1}{2} \sum_{i=1}^{t-1} \beta_i d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_t) + \sum_{i=1}^{t-1} (\alpha_i + \beta_i) \epsilon_{\mathcal{D}_i}(H_{t-1}). \tag{37}$$

Then applying Lemma A.1 and Lemma A.2 jointly to Eqn. 37 will complete the proof. □

B UDIL as a Unified Framework

In this section, we will delve into a comprehensive discussion of our UDIL framework, which serves as a unification of numerous existing methods. It is important to note that we incorporate methods designed for task incremental and class incremental scenarios that can be easily adapted to our domain incremental learning. To provide clarity, we will present the corresponding coefficients $\{\alpha_i, \beta_i, \gamma_i\}$ of each method within our UDIL framework (refer to Table 4). Furthermore, we will explore the conditions under which these coefficients are included in this unification process.

Learning without Forgetting (LwF) [52] was initially proposed for task-incremental learning, incorporating a combination of *shared parameters* and *task-specific parameters*. This framework can be readily extended to domain incremental learning by setting all "domain-specific" parameters to be the same in a static model architecture. LwF was designed for the strict continual learning setting, where no data from past tasks is accessible. To overcome this limitation, LwF records the predictions of the history model H_{t-1} on the current data $\mathcal{X}_t$ at the beginning of the new task t. Subsequently, knowledge distillation (as defined in Definition 4.2) is performed to mitigate forgetting:

$$\mathcal{L}_{\text{old}}(h, H_{t-1}) \triangleq -\tfrac{1}{N_t} \sum_{\boldsymbol{x} \in \mathcal{X}_t} \sum_{k=1}^{K} [H_{t-1}(\boldsymbol{x})]_k \cdot [\log([h(\boldsymbol{x})]_k)] = \widehat{\ell}_{\mathcal{X}_t}(h, H_{t-1}), \tag{38}$$

where $H_{t-1}(\boldsymbol{x}), h(\boldsymbol{x}) \in \mathbb{R}^K$ are the class distribution of $\boldsymbol{x}$ over K classes produced by the history model and current model, respectively. The loss for learning the current task $\mathcal{L}_{\text{new}}$ is defined as

$$\mathcal{L}_{\text{new}}(h) \triangleq -\tfrac{1}{N_t} \sum_{(\boldsymbol{x}, y) \in \mathcal{S}_t} \sum_{k=1}^{K} \mathbb{1}_{y=k} \cdot [\log([h(\boldsymbol{x})]_k)] = \widehat{\ell}_{\mathcal{X}_t}(h). \tag{39}$$

https://doi.org/10.52202/075280-0660

LwF uses a "loss balance weight" λ_o to balance two losses, which gives us its final loss for training:

$$\mathcal{L}_{\text{LwF}}(h) \triangleq \mathcal{L}_{\text{new}}(h) + \lambda_o \cdot \mathcal{L}_{\text{old}}(h, H_{t-1}). \tag{40}$$

In LwF, the default setting assumes the presence of two domains (tasks) with $\lambda_o = 1$. However, it is possible to learn multiple domains continuously using LwF's default configuration. To achieve this, the current domain t can be weighed against the number of previous domains (1 versus $t-1$). Specifically, if there is no preference for any particular domain, λ_o should be set to $t-1$. Remarkably, this is equivalent to setting $\{\beta_i = 1, \alpha_i = \gamma_i = 0\}$ in our UDIL framework (Row 2 in Table 4).

Experience Replay (ER) [75] serves as the fundamental operation for replay-based continual learning methods. It involves storing and replaying a subset of examples from past domains during training. Following the description and implementation provided by [8], ER operates as follows: during each training iteration on domain t, a mini-batch B_t of examples is sampled from the current domain, along with a mini-batch B'_t from the memory. These two mini-batches are then concatenated into a larger mini-batch $(B_t \cup B'_t)$, upon which average gradient descent is performed:

$$\mathcal{L}_{\text{ER}}(h) = \widehat{\ell}_{B_t \cup B'_t}(h) \tag{41}$$

$$= \tfrac{1}{|B_t|+|B'_t|} \sum_{(\boldsymbol{x},y)\in B_t \cup B'_t} \sum_{k=1}^{K} \mathbb{1}_{y=k} \cdot [\log([h(\boldsymbol{x})]_k)] \tag{42}$$

$$= \tfrac{|B_t|}{|B_t|+|B'_t|} \widehat{\ell}_{B_t}(h) + \tfrac{|B'_t|}{|B_t|+|B'_t|} \widehat{\ell}_{B'_t}(h). \tag{43}$$

Suppose that each time the mini-batch of past-domain data is perfectly balanced, meaning that each domain has the same number of examples in B'_t. In this case, Eqn. 43 can be further decomposed as follows:

$$\mathcal{L}_{\text{ER}}(h) = \tfrac{|B_t|}{|B_t|+|B'_t|} \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \tfrac{|B'_t|/(t-1)}{|B_t|+|B'_t|} \widehat{\ell}_{B'_i}(h), \tag{44}$$

where $B'_i = \{(\boldsymbol{x}, y) | (\boldsymbol{x}, y) \in (B'_t \cap M_i)\}$ is the subset of the mini-batch that belongs to domain i.

Now, by dividing both sides of Eqn. 44 by $({}^{|B_t|+|B'_t|}/_{|B_t|})$ and comparing it to Theorem 3.4, we can include ER in our UDIL framework when the condition $|B_t| = {}^{|B'_t|}/_{(t-1)}$ is satisfied. In this case, ER is equivalent to $\{\alpha_i = \beta_i = 0, \gamma_i = 1\}$ in UDIL (Row 3 in Table 4). It is important to note that this condition is not commonly met throughout the entire process of continual learning. It can be achieved by linearly scaling up the size of the mini-batch from the memory (which is feasible in the early domains) or by linearly scaling down the mini-batch from the current-domain data (which may cause a drop in model performance). It is worth mentioning that this incongruence *highlights the intrinsic bias of the original ER setting towards current domain learning* and cannot be rectified by adjusting the batch sizes of the current domain or the memory. However, it does not weaken our claim of unification.

Dark Experience Replay (DER++) [8] includes an additional dark experience replay, i.e., knowledge distillation on the past domain exemplars, compared to ER [75]. Now under the same assumptions (balanced sampling strategy and $|B_t| = {}^{|B'_t|}/_{(t-1)}$) as discussed for ER, we can utilize Eqn. 44 to transform the DER++ loss as follows:

$$\mathcal{L}_{\text{DER++}}(h) = \tfrac{|B_t|}{|B_t|+|B'_t|} \widehat{\ell}_{B_t}(h) + \tfrac{1}{2} \sum_{i=1}^{t-1} \tfrac{|B'_t|/(t-1)}{|B_t|+|B'_t|} \widehat{\ell}_{B'_i}(h) + \tfrac{1}{2} \sum_{i=1}^{t-1} \tfrac{|B'_t|/(t-1)}{|B_t|+|B'_t|} \widehat{\ell}_{B'_i}(h, H_{t-1}). \tag{45}$$

In this scenario, DER++ is equivalent to $\{\alpha_i = \gamma_i = {}^1/_2, \beta_i = 0\}$ in UDIL (Row 4 in Table 4).

Incremental Classifier and Representation Learning (iCaRL) [74] was initially proposed for class incremental learning. It decouples learning the representations and final classifier into two individual phases. During training, iCaRL adopts an incrementally increasing linear classifier. Different from traditional design choice of multi-class classification where the $\mathrm{softmax}$ activation layer and multi-class cross entropy loss are used jointly, iCaRL models multi-class classification as multi-label classification [107, 86, 106]. Suppose there are K classes in total, and we denote the one-hot label vector of the input $\boldsymbol{x}$ as $\boldsymbol{y} \in \mathbb{R}^K$ where $y_j \triangleq \mathbb{1}_{f(\boldsymbol{x})=j}$. Then the *multi-label learning objective* treats each dimension of the output logits as a score for binary classification, which is computed as follows:

$$\widehat{\ell}'_{\mathcal{X}}(h) \triangleq \tfrac{-1}{|\mathcal{X}|} \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{j=1}^{K} \left[y_j \log\left([h(\boldsymbol{x})]_j\right) + (1-y_j) \log\left(1 - [h(\boldsymbol{x})]_j\right) \right]. \tag{46}$$

When a new task that contains K' new classes is presented, iCaRL adds additional K' new dimensions to the final linear classifier.

In iCaRL training, the new classes and the old classes are treated differently. For new classes, it trains the representations of the network with the ground-truth labels ($f(\boldsymbol{x})$ in the original paper), and for old classes, it uses the history model's output ($H_{t-1}(\boldsymbol{x})$) as the learning target (i.e., pseudo labels). Each data point is treated with the same level of importance during iCaRL's training procedure. Hence after translating the loss function of iCaRL in the context of class-incremental learning to domain-incremental learning, we have

$$\mathcal{L}_{\text{iCaRL}}(h) = N_t \cdot \widehat{\ell}'_{\mathcal{X}_t}(h) + \sum_{i=1}^{t-1} \widetilde{N}_i \cdot \widehat{\ell}'_{\mathcal{X}_i}(h, H_{t-1}), \tag{47}$$

where in $\widehat{\ell}'_{\mathcal{X}_i}(h, H_{t-1})$, we follow the same definition as in the distillation loss in Eqn. 4.2 and replace the ground-truth label with the soft label for iCaRL.

Similar to ER [75] and DER++ [8] as analyzed before, the equation above does not naturally fall into the realm of unification provided in UDIL. To achieve this, we need to sample the data from the current domain and exemplars in the memory independently, i.e., assuming B_t and B_i. By applying the same rule of $\alpha_i + \beta_i + \gamma_i = 1$, we now have that for iCaRL, the corresponding coefficients are $\{\alpha_i = {}^{B_i}\!/_{B_t} = 1, \beta_i = \gamma_i = 0\}$ (Row 5 in Table 4).

Complementary Learning System based Experience Replay (CLS-ER) [4] involves the maintenance of two history models, namely the plastic model $H_{t-1}^{(p)}$ and the stable model $H_{t-1}^{(s)}$, throughout the continual training process of the working model h. Following each update of the working model, the two history models are stochastically updated at different rates using exponential moving averages (EMA) of the working model's *parameters*:

$$H_{t-1}^{(i)} \leftarrow \alpha^{(i)} \cdot H_{t-1}^{(i)} + (1 - \alpha^{(i)}) \cdot h, \qquad i \in \{p, s\}, \tag{48}$$

where $\alpha^{(p)} \leq \alpha^{(s)}$ is set such that the plastic model undergoes rapid updates, allowing it to swiftly adapt to newly acquired knowledge, while the stable model maintains a "long-term memory" spanning multiple tasks. Throughout training, CLS-ER assesses the certainty generated by both history models and employs the logits from the more certain model as the target for knowledge distillation.

In the general formulation of the UDIL framework, the history model H_{t-1} is not required to be a single model with the same architecture as the current model h. In fact, if there are no constraints on memory consumption, we have the flexibility to train and preserve a domain-specific model H_i for each domain i. During testing, we can simply select the prediction with the highest certainty from each domain-specific model. From this perspective, the "two-history-model system" employed in CLS-ER can be viewed as a specific and limited version of the all-domain history models. Consequently, we can combine the two models used in CLS-ER into a single history model H_{t-1} as follows:

$$H_{t-1}(\boldsymbol{x}) \triangleq \begin{cases} H_{t-1}^{(p)}(\boldsymbol{x}) & \text{if } [H_{t-1}^{(p)}(\boldsymbol{x})]_y > [H_{t-1}^{(s)}(\boldsymbol{x})]_y \\ H_{t-1}^{(s)}(\boldsymbol{x}) & \text{o.w.} \end{cases} \tag{49}$$

where $(\boldsymbol{x}, y) \in \mathcal{M}$ is an arbitrary exemplar stored in the memory bank.

At each iteration of training, CLS-ER samples a mini-batch B_t from the current domain and a mini-batch B'_t from the episodic memory. It then concatenates B_t and B'_t for the cross entropy loss minimization with the ground-truth labels, and uses B'_t to minimize the MSE loss between the logits of h and H_{t-1}. To align the loss formulation of CLS-ER with that of ESM-ER [80], here we consider the scenarios where the losses evaluated on B_t and B'_t are individually calculated, i.e., we consider $\widehat{\ell}_{B_t}(h) + \widehat{\ell}_{B'_t}(h)$ instead of $\widehat{\ell}_{B_t \cup B'_t}(h)$. Based on the assumption from [34], the MSE loss on the logits is equivalent to the cross-entropy loss on the predictions under certain conditions. Therefore, following the same balanced sampling strategy assumptions as in ER, the original CLS-ER training objective can be transformed as follows:

$$\mathcal{L}_{\text{CLS-ER}}(h) = \widehat{\ell}_{B_t}(h) + \widehat{\ell}_{B'_t}(h) + \lambda \widehat{\ell}_{B'_t}(h, H_{t-1}) \tag{50}$$

$$= \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \tfrac{1}{t-1} \widehat{\ell}_{B'_i}(h) + \sum_{i=1}^{t-1} \tfrac{\lambda}{t-1} \widehat{\ell}_{B'_i}(h, H_{t-1}). \tag{51}$$

Therefore, by imposing the constraint $\alpha_i+\beta_i+\gamma_i=1$, we find that $\lambda=t-2$. Substituting this value back into ${}^{\lambda}\!/_{t-1}$ yields the equivalence that CLS-ER corresponds to $\{\alpha_i={}^{\lambda}\!/_{\lambda+1}, \beta_i=0, \gamma_i={}^{1}\!/_{\lambda+1}\}$ in UDIL, where $\lambda=t-2$ (Row 6 in Table 4).

Error Sensitivity Modulation based Experience Replay (ESM-ER) [80] builds upon CLS-ER by incorporating an additional error sensitivity modulation module. The primary goal of ESM-ER is to mitigate sudden representation drift caused by excessively large loss values during current-domain learning. Let's consider $(\boldsymbol{x},y)\sim\mathcal{D}_t$, which represents a sample from the current domain batch. In ESM-ER, the cross-entropy loss value of this sample is evaluated using the stable model $H_{t-1}^{(s)}$ and can be expressed as:

$$\ell(\boldsymbol{x},y)=-\log([H_{t-1}^{(s)}(\boldsymbol{x})]_y). \tag{52}$$

To screen out those samples with a high loss value, ESM-ER assigns each sample a weight λ by comparing the loss with their expectation value, for which ESM-ER uses a running estimate μ as its replacement. This can be formulated as follows:

$$\lambda(\boldsymbol{x})=\begin{cases}1 & \text{if } \ell(\boldsymbol{x},y)\le\beta\cdot\mu\\ \frac{\mu}{\ell(\boldsymbol{x},y)} & \text{o.w.}\end{cases} \tag{53}$$

where β is a hyperparameter that determines the margin for a sample to be classified as low-loss. For the sake of analysis, we make the following assumptions: (i) $\beta=1$; (ii) the actual expected loss value $\mathbb{E}_{\boldsymbol{x},y}[\ell(\boldsymbol{x},y)]$ is used instead of the running estimate μ; (iii) a *hard screening mechanism* is employed instead of the current re-scaling approach. Based on these assumptions, we determine the sample-wise weights $\lambda^\star$ according to the following rule:

$$\lambda^\star(\boldsymbol{x})=\begin{cases}1 & \text{if } \ell(\boldsymbol{x},y)\le\mathbb{E}_{\boldsymbol{x},y}[\ell(\boldsymbol{x},y)]\\ 0 & \text{o.w.}\end{cases} \tag{54}$$

Under the assumption that the loss value $\ell(\boldsymbol{x},y)$ follows an exponential distribution, denoted as $\ell(\boldsymbol{x},y)\sim\text{Exp}(\lambda_0)$, where the probability density function is given by $f(\ell(\boldsymbol{x},y),\lambda_0)=\lambda_0 e^{-\lambda_0\ell(\boldsymbol{x},y)}$, we can calculate the expectation of the loss as $\mathbb{E}_{\boldsymbol{x},y}[\ell(\boldsymbol{x},y)]={}^{1}\!/_{\lambda_0}$. Based on this, we can now determine the expected ratio r of the *unscreened samples* in a mini-batch using the following equation:

$$r=\int_0^{\frac{1}{\lambda_0}}1\cdot\lambda_0 e^{-\lambda_0\ell(\boldsymbol{x},y)}\,d\ell(\boldsymbol{x},y)=\int_0^1 e^{-y}dy=(1-e^{-1}). \tag{55}$$

The ratio r represents the proportion of effective samples in the current-domain batch, as the weights $\lambda^\star(\boldsymbol{x})$ of the remaining samples are set to 0 due to their high loss value. Consequently, the original training loss of ESM-ER can be transformed as follows:

$$\mathcal{L}_{\text{ESM-ER}}(h)=r\cdot\widehat{\ell}_{B_t}(h)+\widehat{\ell}_{B_t'}(h)+\lambda\widehat{\ell}_{B_t'}(h,H_{t-1}) \tag{56}$$

$$=r\cdot\widehat{\ell}_{B_t}(h)+\sum_{i=1}^{t-1}\tfrac{1}{t-1}\widehat{\ell}_{B_i'}(h)+\sum_{i=1}^{t-1}\tfrac{\lambda}{t-1}\widehat{\ell}_{B_i'}(h,H_{t-1}). \tag{57}$$

After applying the constraint of $\alpha_i+\beta_i+\gamma_i=1$, we obtain $\lambda=r\cdot(t-1)-1$. Substituting this value back into ${}^{\lambda}\!/_{r(t-1)}$, we find that ESM-ER is equivalent to $\{\alpha_i={}^{\lambda}\!/_{\lambda+1}, \beta_i=0, \gamma_i={}^{1}\!/_{\lambda+1}\}$ in UDIL, where $\lambda=r\cdot(t-1)-1=(1-e^{-1})(t-1)-1$ should be set (Row 7 in Table 4).

Bias Correction (BiC) [100] was the first to apply class incremental learning at a large scale, covering extensive image classification datasets including ImageNet (1,000 classes, [76]) and MS-Celeb-1M (10,000 classes, [31]). Similar to iCaRL [74], BiC treats each data example (from new classes' data and the memory) with the same level of importance. Different from its previous work, the distillation loss in BiC (L_d in the original paper) implicitly includes the cross-domain distillation loss described by Lemma 3.3, as it computes the old classifier's output evaluated on the new data (considering only old classes). In addition, BiC further re-balances the classification loss (L_c in the original paper) and the distillation loss (L_d) based on the number of old classes n and new classes m as follows:

$$\mathcal{L}_{\text{BiC}}=\tfrac{n}{n+m}\cdot L_d+\tfrac{m}{n+m}\cdot L_c. \tag{58}$$

https://doi.org/10.52202/075280-0660

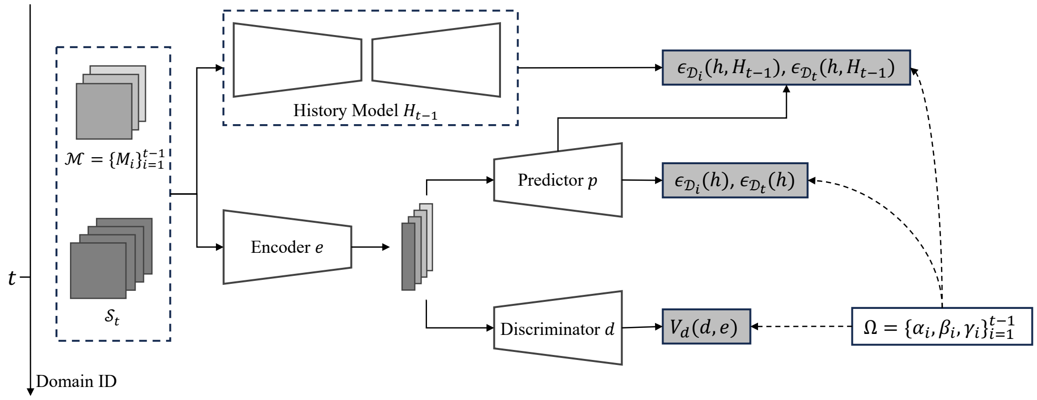

History Model H_{t-1}
$\epsilon_{\mathcal{D}_i}(h, H_{t-1}), \epsilon_{\mathcal{D}_t}(h, H_{t-1})$
$\mathcal{M} = \{M_i\}_{i=1}^{t-1}$
Predictor p
$\epsilon_{\mathcal{D}_i}(h), \epsilon_{\mathcal{D}_t}(h)$
Encoder e
t
$\mathcal{S}_t$
Discriminator d
$V_d(d, e)$
$\Omega = \{\alpha_i, \beta_i, \gamma_i\}_{i=1}^{t-1}$
Domain ID

Figure 2: **Diagram of UDIL.** $\mathcal{M} = \{M_i\}_{i=1}^{t-1}$ represents the memory bank that stores all the past exemplars. $\mathcal{S}_t$ corresponds to the dataset from the current domain t, and the current model $h = p \circ e$ is depicted separately in the diagram. The three different categories of losses are illustrated in the dark rectangles, while the weighting effect of the learned replay coefficient $\Omega = \{\alpha_i, \beta_i, \gamma_i\}_{i=1}^{t-1}$ is depicted using dashed lines.

By interpreting the distillation loss L_d as the combination of the intra-domain loss (Lemma 3.2) and the cross-domain distillation loss (Lemma 3.3), and substituting (n, m) with $(t-1, 1)$ due to the consistent number of classes in each domain, we arrive at the BiC model for DIL:

$$\begin{aligned}\mathcal{L}_{\text{BiC}}(h) =& \tfrac{t-1}{t}\left[N_t \cdot \widehat{\ell}_{\mathcal{X}_t}(h, H_{t-1}) + \sum_{i=1}^{t-1} \widetilde{N}_i \cdot \widehat{\ell}_{\mathcal{X}_i}(h, H_{t-1})\right] \\ &+ \tfrac{1}{t}\left[N_t \cdot \widehat{\ell}_{\mathcal{X}_t}(h) + \sum_{i=1}^{t-1} \widetilde{N}_i \cdot \widehat{\ell}_{\mathcal{X}_i}(h)\right]. \qquad (59)\end{aligned}$$

The equation above relies on the number of the current-domain examples and the exemplars (i.e., memory), which is not constant in practice. Hence we replace $(N_t, \widetilde{N}_i)$ with the mini-batch size $(|B_t|, |B_i|)$. After re-organizing the equation above, we have

$$\begin{aligned}\mathcal{L}_{\text{BiC}}(h) =& \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \tfrac{(t-1)|B_i|}{|B_t|} \cdot \widehat{\ell}_{B_i'}(h, H_{t-1}) \\ &+ (t-1) \cdot \widehat{\ell}_{B_t}(h, H_{t-1}) + \sum_{i=1}^{t-1} \tfrac{|B_i|}{|B_t|} \cdot \widehat{\ell}_{B_i'}(h). \qquad (60)\end{aligned}$$

The immediate observatio is that *BiC exhibits a significant bias towards retaining knowledge from past domains*. This is evident in the coefficient of coefficient of cross-domain distillation summed over the past domains $\sum_{i=1}^{t-1} \beta_i = (t-1)$, which violates the constraints posed in this work ($\alpha_i + \beta_i + \gamma_i = 1$). However, if we slightly relax BiC's formulation and focus on the relative ratios of the three coefficient, we get $\alpha_i : \beta_i : \gamma_i = (t-1)B_i : (t-1)B_t : B_i$. Further applying the constraint $\alpha_i + \beta_i + \gamma_i = 1$ and assuming $B_i = B_t$ to enforce a balanced sampling strategy over different domains, we arrive at $\{\alpha_i = \beta_i = {}^{t-1}/_{2t-1}, \gamma_i = {}^{1}/_{2t-1}\}$ (Row 8 in Table 4).

C Implementation Details of UDIL

This section delves into the implementation details of the UDIL algorithm. The algorithmic description of UDIL is presented in Algorithm 1 and a diagram is presented in Fig. 2. However, there are several practical issues to be further addressed here, including (i) how to exert the constraints of probability simplex ($[\alpha_i, \beta_i, \gamma_i] \in \mathbb{S}^2$) and (ii) how the memory is maintained. These two problems will be addressed in Sec. C.1 and Sec. C.2. Then, Sec. C.3 will introduce two auxiliary losses that improve the stability and domain alignment for the encoder during training. Next, Sec. C.4

https://doi.org/10.52202/075280-0660

will cover the evaluation metrics used in this paper. Finally, Sec. C.5 and Sec. C.6 will present a detailed introduction to the main baselines and the specific training schemes we follow for empirical evaluation.

C.1 Modeling the Replay Coefficients $\Omega = \{\alpha_i, \beta_i, \gamma_i\}$

Instead of directly modeling Ω in a way such that it can be updated by gradient descent and satisfies the constraints that $\alpha_i + \beta_i + \gamma_i = 1$ and $\alpha_i, \beta_i, \gamma_i \geq 0$ at the same time, we use a set of logit variables $\{\bar{\alpha}_i, \bar{\beta}_i, \bar{\gamma}_i\} \in \mathbb{R}^3$ and the *softmax* function to indirectly calculate Ω during training. Concretely, we have:

$$\begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} \bar{\alpha}_i \\ \bar{\beta}_i \\ \bar{\gamma}_i \end{bmatrix} \right) = \begin{bmatrix} {}^{\exp(\bar{\alpha}_i)}/_{Z_i} \\ {}^{\exp(\bar{\beta}_i)}/_{Z_i} \\ {}^{\exp(\bar{\gamma}_i)}/_{Z_i} \end{bmatrix}, \tag{61}$$

where $Z_i = \exp(\bar{\alpha}_i) + \exp(\bar{\beta}_i) + \exp(\bar{\gamma}_i)$ is the normalizing constant. At the beginning of training on domain t, the logit variables $\{\bar{\alpha}_i, \bar{\beta}_i, \bar{\gamma}_i\} = \{0, 0, 0\}$ are initialized to all zeros, since we do not have any bias towards any upper bound. During training, they are updated in the same way as the other parameters with gradient descent.

C.2 Memory Maintenance with Balanced Sampling

Different from DER++ [8] and its following work [4, 80] that use reservoir sampling [93] to maintain the episodic memory, UDIL adopts a random balanced sampling after training on each domain. To be more concrete, given a memory bank with fixed size $|\mathcal{M}|$, after domain t's training is complete, we will assign each domain a quota of ${}^{|\mathcal{M}|}/_{t}$. For the current domain t, we will randomly sample $\lfloor {}^{|\mathcal{M}|}/_{t} \rfloor$ exemplars from its dataset; for all the previous domains $i \in [t-1]$, we will randomly swap out $\lceil {}^{|\mathcal{M}|}/_{t-1} - {}^{|\mathcal{M}|}/_{t} \rceil$ exemplars from the memory to make sure each domain has roughly the same number of exemplars. To ensure a fair comparison, we use the same random balanced sampling strategy for all the other baselines. The following Algorithm 2 shows the detailed procedure of random balanced sampling.

Algorithm 2 Balanced Sampling for UDIL

Require: memory bank $\mathcal{M} = \{M_i\}_{i=1}^{t-1}$, current domain dataset $\mathcal{S}_t$, domain ID t.

```
for $i = 1, \cdots, t-1$ do
    for $j = 1, \cdots, \lceil {}^{|\mathcal{M}|}/_{t-1} - {}^{|\mathcal{M}|}/_{t} \rceil$ do
        $(\boldsymbol{x}, y) \leftarrow \text{RandomSample}(M_i)$
        $(\boldsymbol{x}', y') \leftarrow \text{RandomSample}(\mathcal{S}_t)$
        Swap $(\boldsymbol{x}', y')$ into $\mathcal{M}$, replacing $(\boldsymbol{x}, y)$
    end for
end for
return $\mathcal{M}$
```

C.3 Improving Stability and Domain Alignment for the Embedding Distribution

In this section, we will examine the training loss employed to align the embedding distribution across distinct domains. As discussed in Sec. 4.2, we decompose the model h into an encoder e and a predictor p as $h = p \circ e$. In order to establish a tighter upper bound proposed in Theorem 3.4, we introduce a discriminator d that aims to distinguish embeddings based on domain IDs. Specifically, during the training process of UDIL, given a set of coefficients $\Omega = \{\alpha_i, \beta_i, \gamma_i\}$, both the encoder e and the discriminator d engage in the following minimax game:

$$\min_e \max_d -\lambda_d V_d(d, e, \mathring{\Omega}), \tag{62}$$

where $(\mathring{\cdot})$ represents the "copying-weights-and-stopping-gradients" operation, and λ_d is a hyperparameter introduced to control the strength of domain embedding alignment. More specifically, the value function of the mini-max game V_d is defined as follows:

$$V_d(d, e, \overset{\circ}{\Omega}) = \left(\sum_{i=1}^{t-1} \overset{\circ}{\beta}_i\right) \tfrac{1}{N_i} \sum_{\boldsymbol{x} \in \mathcal{S}_t} \left[-\log\left(\left[d(e(\boldsymbol{x}))\right]_t\right)\right] + \sum_{i=1}^{t-1} \tfrac{\overset{\circ}{\beta}_i}{\widetilde{N}_i} \sum_{\boldsymbol{x} \in \widetilde{\mathcal{X}}_i} \left[-\log\left(\left[d(e(\boldsymbol{x}))\right]_i\right)\right]. \tag{63}$$

As previously mentioned, the practical effect of Eqn. 62 is to align the embedding distribution of different domains, thus enhancing the model's generalization ability to both previously encountered domains and those that may be encountered in the future.

However, actively altering the embedding space can lead to the well-known stability-plasticity dilemma [43, 96, 20, 39, 61]. This dilemma arises when the model needs to modify a significant number of parameters to achieve domain alignment for a new domain, potentially resulting in a mismatch between the predictor p and the encoder e, which, in turn, can lead to catastrophic forgetting. Furthermore, it is worth noting that the adversarial training scheme described in Eqn. 62 is primarily designed for unsupervised domain alignment [108, 57, 26, 109, 17, 102, 101, 94], where the semantic labels in the target domain(s) are not available during training. This implies that the current adversarial training technique does not fully leverage the label information in domain incremental learning.

To tackle the aforementioned challenges, i.e., (i) maintaining a stable embedding distribution across all domains and (ii) accelerating domain alignment with label information, we incorporate two additional auxiliary losses: (i) the *past embedding distillation loss* [40, 65] and (ii) the *supervised contrastive loss* [42, 29]. It's important to note that these auxiliary losses, in conjunction with the adversarial feature alignment loss V_d, operate exclusively on the encoder space of the model and do not impact the original objectives outlined in Theorem 3.4, provided that encoder e and predictor p remain sufficiently strong. Therefore, the two losses are used to simply stabilize the training, without compromising the theoretical significance of our work.

In **past embedding distillation**, also known as *representation(al) distillation* or *embedding distillation* in previous work on continual learning [40, 65, 59], the model stores the embeddings of the memory after being trained on domain $t-1$. It then uses them to constrain the encoder's behavior: the features produced on these samples should not change too much during the current domain training. After decoupling the history model $H_{t-1} = P_{t-1} \circ E_{t-1}$, where E_{t-1} is the history encoder and P_{t-1} is history predictor, we define the past embedding distillation loss V_p as

$$V_p(e) = \sum_{i=1}^{t-1} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_i} \left[\|e(\boldsymbol{x}) - E_{t-1}(\boldsymbol{x})\|_2^2\right]. \tag{64}$$

The loss above promotes the stability of the embedding distribution from past domains. When combined with the adversarial embedding alignment loss in Eqn. 62, it encourages the embedding distribution of the current domain to match those of the previous domains, but not vice versa.

As supervised variations of contrastive learning [12, 32, 14, 83, 65, 13, 84, 85], the **supervised contrastive loss** [42, 29] will compensate for the fact that Eqn. 62 does not utilize the label information to align different domains, and therefore lack in the efficiency of aligning two domains' embedding distribution. The supervised contrastive learning pulls together the embeddings of the same label and pushes apart those with distinct labels. Notably, it is done in a "domain-agnostic" manner, i.e., the domain labels are not considered. Denoting as $\mathcal{P} = \frac{1}{t}\sum_i^t \mathcal{D}_i$ the combined data distribution of all domains and as $\mathcal{P}_{\cdot|y}$ the data distribution given the class label $y \in [K]$, the supervised contrastive loss for embedding alignment is defined as follows:

$$V_s(e) = \mathbb{E}_y \mathbb{E}_{\substack{(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \mathcal{P}_{\cdot|y} \\ \{\boldsymbol{u}_i\}_{i=1}^B \sim \mathcal{P}}} \left[-\log \frac{\exp\{-s_e(\boldsymbol{x}_1, \boldsymbol{x}_2)\}}{\exp\{-s_e(\boldsymbol{x}_1, \boldsymbol{x}_2)\} + \sum_{i=1}^B \exp\{-s_e(\boldsymbol{x}_1, \boldsymbol{u}_i)\}}\right], \tag{65}$$

where $s_e(\boldsymbol{u}, \boldsymbol{v}) = \|e(\boldsymbol{u}) - e(\boldsymbol{v})\|_2^2$ is the squared Euclidean distance for any $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{X}$.

Introducing the supervised contrastive loss to continual learning is novel, as existing methods often harness its ability to create a compact representation space, thereby mitigating representation overlapping in class incremental learning [65, 30, 9, 96]. However, in this work, the primary motivation for using the supervised contrastive loss to facilitate domain alignment lies in its optimal solution [29, 110, 19]. This optimal point referred to as *neural collapse* [47, 69, 111, 103], where the embeddings of the same class collapse to the same point, and those of different classes are sparsely distributed. It is easy to envision that when *the optimal state of the supervised constrastive loss* is attained across different domains, it concurrently achieves *perfect domain alignment*.

The loss function that fosters stability and domain alignment in the encoder e can be summarized as follows:

$$\mathcal{L}_{\text{enc}}(e) = -\lambda_d V_d(d, e, \overset{\circ}{\Omega}) + \lambda_p V_p(e) + \lambda_s V_s(e). \tag{66}$$

Here, two hyper-parameters, λ_p and λ_s, balance the influence of each individual loss on the encoder's embedding distribution. In practice, the final performance of UDIL is not significantly affected by the values of λ_p and λ_s, and a wide range of values for these parameters can yield reasonable results.

C.4 Evaluation Metrics

In continual learning, many evaluation metrics are based on the **Accuracy Matrix** $\boldsymbol{R} \in \mathbb{R}^{T \times T}$, where T represents the total number of tasks (domains). In the accuracy matrix $\boldsymbol{R}$, the entry $R_{i,j}$ corresponds to the accuracy of the model when evaluated on task j *after* training on task i. With this definition in mind, we primarily focus on the following specific metrics:

Average Accuracy (Avg. Acc.) up until domain t represents the average accuracy of the first t domains after training on these domains. We denote it as A_t and define it as follows:

$$A_t \triangleq \tfrac{1}{t} \sum_{i=1}^{t} R_{t,i}. \tag{67}$$

In most of the continual learning literature, the final average accuracy A_T is usually reported. In our paper, this metric is reported in the column labeled "**overall**". The average accuracy of a model is a crucial metric as it directly corresponds to the primary optimization goal of minimizing the error on all domains, as defined in Eqn. 3.

Additionally, to better illustrate the learning (and forgetting) process of a model across multiple domains, we propose the use of the "**Avg. of Avg. Acc.**" metric $A_{t_1:t_2}$, which represents the average of average accuracies for a consecutive range of domains starting from domain t_1 and ending at domain t_2. Specifically, we define this metric as follows:

$$A_{t_1:t_2} \triangleq \tfrac{1}{t_2 - t_1 + 1} \sum_{i=t_1}^{t_2} A_i. \tag{68}$$

This metric provides a condensed representation of the trend in accuracy variation compared to directly displaying the entire series of average accuracies $\{A_1, A_2, \cdots, A_T\}$. We report this Avg. of Avg. Acc. metric in all tables (except in Table 2 due to the limit of space).

Average Forgetting (i.e., 'Forgetting' in the main paper) defines the average of the largest drop of accuracy for each domain up till domain t. We denote this metric as F_t and define it as follows:

$$F_t \triangleq \tfrac{1}{t-1} \sum_{j=1}^{t-1} f_t(j), \tag{69}$$

where $f_t(j)$ is the forgetting on domain j after the model completes the training on domain t, which is defined as:

$$f_t(j) \triangleq \max_{l \in [t-1]} \{R_{l,j} - R_{t,j}\}. \tag{70}$$

Typically, the average forgetting is reported after training on the last domain T. Measuring forgetting is of great practical significance, especially when two models have similar average accuracies. It indicates how a model balances *stability* and *plasticity*. If a model $\mathcal{P}$ achieves a reasonable final average accuracy across different domains but exhibits high forgetting, we can conclude that this model has high plasticity and low stability. It quickly adapts to new domains but at the expense of performance on past domains. On the other hand, if another model $\mathcal{S}$ has a similar average accuracy to $\mathcal{P}$ but significantly lower average forgetting, we can infer that the model $\mathcal{S}$ has high stability and low plasticity. It sacrifices performance on recent domains to maintain a reasonable performance on past domains. Hence, to gain a comprehensive understanding of model performance, we focus on evaluating two key metrics: *Avg. Acc.* and *Forgetting*. These metrics provide insights into how models balance stability and plasticity and allow us to assess their overall performance across different domains.

Forward Transfer W_t quantifies the extent to which learning from past $t-1$ domains contributes to the performance on the next domain t. It is defined as follows:

$$W_t \triangleq \tfrac{1}{t-1} \sum_{i=2}^{t} R_{i-1,i} - r_i, \tag{71}$$

where r_i is the accuracy of a randomly initialized model evaluated on domain i. For domain incremental learning, where the model does not have access to future domain data and does not explicitly optimize for higher Forward Transfer, the results of this metric are *typically random. Therefore, we do not report this metric in the complete tables presented in this section.*

C.5 Introduction to Baselines

We compare UDIL with the state-of-the-art continual learning methods that are either specifically designed for domain incremental learning or can be easily adapted to the domain incremental learning setting. Exemplar-free baselines include online Elastic Weight Consolidation (**oEWC**) [81], Synaptic Intelligence (**SI**) [105], and Learning without Forgetting (**LwF**) [52]. Memory-based domain incremental learning baselines include Gradient Episodic Memory (**GEM**) [58], Averaged Gradient Episodic Memory (**A-GEM**) [10], Experience Replay (**ER**) [75], Dark Experience Replay (**DER++**) [8], and two recent methods, Complementary Learning System based Experience Replay (**CLS-ER**) [4] and Error Senesitivity Modulation based Experience Replay (**ESM-ER**) [80]. In addition, we implement the fine-tuning (**Fine-tune**) [52] and joint-training (**Joint**) as the performance lower bound and upper bound (Oracle). Here we provide a short description of the primary idea of the memory-based domain incremental learning baselines.

- **GEM** [58]: The baseline method that uses the memory to provide additional optimization constraints during learning the current domain. Specifically, the update of the model cannot point towards the direction at which the loss of any exemplar increases.
- **A-GEM** [10]: The improved baseline method where the constraints of GEM are averaged as one, which shortens the computational time significantly.
- **ER** [75]: The fundamental memory-based domain incremental learning framework where the mini-batch of the memory is regularly replayed with the current domain data.
- **DER++** [8]: A simple yet effective replay-based method where an additional logits distillation (dubbed “dark experience replay”) is applied compared to the vanilla ER.
- **CLS-ER** [4]: A complementary learning system inspired replay method, where two exponential moving average models are used to serve as the semantic memory, which provides the logits distillation target during training.
- **ESM-ER** [80]: An improved version of CLS-ER, where the effect of large errors when learning the current domain is reduced, dubbed “error sensitivity modulation”.

C.6 Training Schemes

Training Process. For each group of experiments, we run three rounds with different seeds and report the mean and standard deviation of the results. We follow the optimal configurations (epochs and learning rate) stated in [8, 80] for the baselines in *P-MNIST* and *R-MNIST* dataset. For *HD-Balls* and *Seq-CORe50*, we first search for the optimal training configuration for the joint learning, and then grid-search the configuration in a small range near it for the baselines listed above. For our UDIL framework, as it involves adversarial training for the domain embedding alignment, we typically need a configuration that has larger number of epochs and smaller learning rate. We use a simple grid search to achieve the optimal configuration for it as well.

Model Architectures. For the baseline methods and UDIL in the same dataset, we adopt the same backbone neural architectures to ensure fair comparison. In *HD-Balls*, we adopt the same multi-layer perceptron with the same separation of encoder and decoder as in CIDA [94], where the hidden dimension is set to 800. In *P-MNIST* and *R-MNIST*, we adopt the same multi-layer perceptron architecture as in DER++ [8] with hidden dimension set to 800 as well. In *Seq-CORe50*, we use the ResNet18 [33] as our backbone architecture for all the methods, where the layers before the final average pooling are treated as the encoder e, and the remaining part is treated as the predictor p.

https://doi.org/10.52202/075280-0660

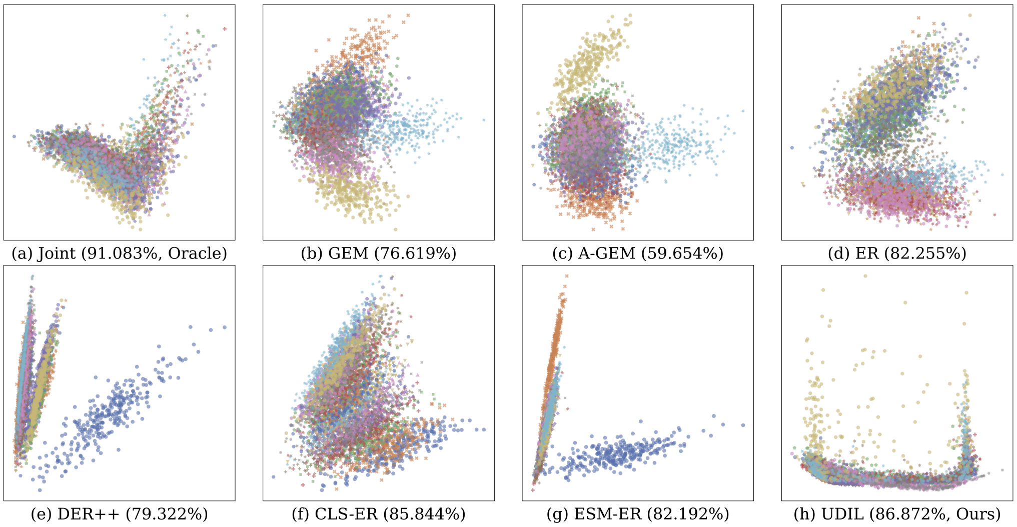

(a) Joint (91.083%, Oracle)
(b) GEM (76.619%)
(c) A-GEM (59.654%)
(d) ER (82.255%)
(e) DER++ (79.322%)
(f) CLS-ER (85.844%)
(g) ESM-ER (82.192%)
(h) UDIL (86.872%, Ours)

Figure 3: More Results on *HD-Balls*. Data is colored according to domain ID. All data is plotted after PCA [6]. **(a-h)** Accuracy and embeddings learned by Joint (oracle), UDIL, and six baselines with memory. Joint, as the *oracle*, naturally aligns different domains, and UDIL outperforms all baselines in terms of embedding alignment and accuracy.

Hyperparameter Setting. For setting the hyper-parameter embedding alignment strength coefficient λ_d and parameter C that models the combined effect of VC-dimension d and error tolerance δ, we use grid search for each dataset, where the range $\lambda_d \in [0.01, 100]$ and $C \in [0, 1000]$ are used.

D Additional Empirical Results

This section presents additional empirical results of the UDIL algorithm. Sec. D.1 will show the additional results on different constraints with varying memory sizes. Sec. D.2 provides additional qualitative results: visualization of embedding distributions, to showcase the importance of the embedding alignment across domains.

D.1 Empirical Results on Varying Memory Sizes

Here we present additional empirical results to validate the effectiveness of our UDIL framework using varying memory sizes. The evaluation is conducted on three real-world datasets, as shown in Table 5, Table 6, and Table 7. By increasing the memory size from 400 to 800 in Table 5 and 6 and from 500 to 1000 in Table 7, we can investigate the impact of having access to a larger pool of past experiences on the continual learning performance, which might occur when the constraint on memory capacity is relaxed. This allows us to study the benefits of a more extensive memory in terms of knowledge retention and performance improvement. On the other hand, by further decreasing the memory size to the extreme of 200 in Table 5 and 6, we can explore the consequences of severely limited memory capacity. This scenario simulates situations where memory constraints are extremely tight, and the model can only retain a small fraction of past domain data, for example, a model deployed on edge devices. To ensure a fair comparison, here we use the same best configuration found in the main body of this work.

The results in all three tables demonstrate a clear advantage of our UDIL framework when the memory size is limited. In *P-MNIST* and *R-MNIST*, when the memory size $|\mathcal{M}| = 200$, the overall performance of UDIL reaches 91.483% and 82.796% respectively, which outperforms the second best model DER++ by 0.757% and a remarkable 6.125%. In *Seq-CORe50*, when the memory size $|\mathcal{M}| = 500$ is set, UDIL holds a 3.474% lead compared to the second best result. When the memory size is larger, the gap between UDIL and the baseline models is smaller. This is because when the memory constraint is relaxed, all the continual learning models should be at least closer to the performance upper bound, i.e., joint learning or 'Joint (Oracle)' in the tables, causing the indistinguishable results among each other. Apparently, DER++ favors larger memory more than UDIL, while UDIL can still maintain a narrow lead in the large scale dataset *Seq-CORe50*.

D.2 Visualization of Embedding Spaces

Here we provide more embedding space visualization results for the baselines with the utilization of memory, shown in Fig. 3. As one of the primary objectives of our algorithm, embedding space alignment across multiple domains naturally follows the pattern shown in the joint learning and therefore leads to a higher performance.

https://doi.org/10.52202/075280-0660

Table 5: **Performances (%) evaluated on *P-MNIST*.** Average Accuracy (Avg. Acc.) and Forgetting are reported to measure the methods' performance. "↑" and "↓" mean higher and lower numbers are better, respectively. We use **boldface** and underlining to denote the best and the second-best performance, respectively. We use "-" to denote "not appliable".

Method	**Buffer**	$\mathcal{D}_{1:5}$	$\mathcal{D}_{6:10}$	$\mathcal{D}_{11:15}$	$\mathcal{D}_{16:20}$	**Overall**	
		Avg. Acc (↑)				Avg. Acc (↑)	Forgetting (↓)
Fine-tune	-	92.506±2.062	87.088±1.337	81.295±2.372	72.807±1.817	70.102±2.945	27.522±3.042
oEWC [81]	-	92.415±0.816	87.988±1.607	83.098±1.843	78.670±0.902	78.476±1.223	18.068±1.321
SI [105]	-	92.282±0.862	87.986±1.622	83.698±1.220	79.669±0.709	79.045±1.357	17.409±1.446
LwF [52]	-	95.025±0.487	91.402±1.546	83.984±2.103	76.046±2.004	73.545±2.646	24.556±2.789
GEM [58]		93.310±0.374	91.900±0.456	89.813±0.914	87.251±0.524	86.729±0.203	9.430±0.156
A-GEM [10]		93.326±0.363	91.466±0.605	89.048±1.005	86.518±0.604	85.712±0.228	10.485±0.196
ER [75]		94.087±0.762	92.397±0.464	89.999±1.060	87.492±0.448	86.963±0.303	9.273±0.255
DER++ [8]	200	94.708±0.451	94.582±0.158	93.271±0.585	90.980±0.610	90.333±0.587	6.110±0.545
CLS-ER [4]		94.761±0.340	93.943±0.197	92.725±0.566	91.150±0.357	90.726±0.218	5.428±0.252
ESM-ER [80]		95.198±0.236	94.029±0.427	91.710±1.056	88.181±1.021	86.851±0.858	10.007±0.864
UDIL (Ours)		**95.747±0.486**	**94.695±0.256**	**93.756±0.343**	**92.254±0.564**	**91.483±0.270**	**4.399±0.314**
GEM [58]		93.557±0.225	92.635±0.306	91.246±0.492	89.565±0.342	89.097±0.149	6.975±0.167
A-GEM [10]		93.432±0.333	92.064±0.439	90.038±0.726	87.988±0.335	87.560±0.087	8.577±0.053
ER [75]		93.525±1.101	91.649±0.362	90.426±0.456	88.728±0.353	88.339±0.044	7.180±0.029
DER++ [8]	400	94.952±0.403	**95.089±0.075**	**94.458±0.328**	**93.257±0.249**	**92.950±0.361**	3.378±0.245
CLS-ER [4]		94.262±0.649	93.195±0.148	92.623±0.195	91.839±0.187	91.598±0.117	3.795±0.144
ESM-ER [80]		95.413±0.139	94.654±0.314	93.353±0.588	91.022±0.781	89.829±0.698	6.888±0.738
UDIL (Ours)		**95.992±0.349**	95.026±0.250	94.212±0.280	93.094±0.326	92.666±0.108	**2.853±0.107**
GEM [58]		93.717±0.177	93.116±0.206	92.166±0.335	91.076±0.342	90.609±0.364	5.393±0.417
A-GEM [10]		93.612±0.241	92.523±0.375	90.718±0.739	88.543±0.391	88.020±0.851	8.081±0.867
ER [75]		93.827±0.871	92.457±0.217	91.688±0.277	90.617±0.289	90.252±0.056	5.188±0.045
DER++ [8]	800	95.295±0.317	**95.539±0.041**	**95.099±0.187**	**94.423±0.151**	**94.227±0.261**	2.106±0.161
CLS-ER [4]		94.463±0.537	93.567±0.093	93.182±0.137	92.744±0.112	92.578±0.152	2.803±0.183
ESM-ER [80]		95.567±0.150	95.136±0.202	94.301±0.347	92.981±0.397	92.408±0.387	4.170±0.357
UDIL (Ours)		**96.082±0.313**	95.207±0.196	94.642±0.156	93.997±0.194	93.724±0.043	**1.633±0.035**
Joint (Oracle)	∞	-	-	-	-	96.368±0.042	-

https://doi.org/10.52202/075280-0660

Table 6: **Performances (%) evaluated on *R-MNIST*.** Average Accuracy (Avg. Acc.) and Forgetting are reported to measure the methods' performance. "↑" and "↓" mean higher and lower numbers are better, respectively. We use **boldface** and underlining to denote the best and the second-best performance, respectively. We use "-" to denote "not appliable".

Method	**Buffer**	$\mathcal{D}_{1:5}$	$\mathcal{D}_{6:10}$	$\mathcal{D}_{11:15}$	$\mathcal{D}_{16:20}$	**Overall**	
		Avg. Acc (↑)				Avg. Acc (↑)	Forgetting (↓)
Fine-tune	-	92.961±2.683	76.617±8.011	60.212±3.688	49.793±1.552	47.803±1.703	52.281±1.797
oEWC [81]	-	91.765±2.286	76.226±7.622	60.320±3.892	50.505±1.772	48.203±0.827	51.181±0.867
SI [105]	-	91.867±2.272	76.801±7.391	60.956±3.504	50.301±1.538	48.251±1.381	51.053±1.507
LwF [52]	-	95.174±1.154	83.044±5.935	65.899±4.061	55.980±1.296	54.709±0.515	45.473±0.565
GEM [58]	200	93.441±0.610	88.620±2.381	81.034±2.704	73.112±1.922	70.545±0.623	27.684±0.645
A-GEM [10]		92.667±1.352	82.772±5.503	70.579±4.028	60.462±2.001	57.958±0.579	40.969±0.580
ER [75]		94.705±0.790	89.171±2.883	79.962±3.365	71.787±1.608	69.627±0.911	28.749±0.993
DER++ [8]		94.904±0.414	91.637±1.871	84.915±2.315	78.373±1.244	76.671±0.391	21.743±0.409
CLS-ER [4]		95.131±0.523	91.421±1.732	84.773±2.665	77.733±1.480	75.609±0.418	22.483±0.456
ESM-ER [80]		**95.378±0.531**	90.800±2.528	83.438±2.581	76.987±1.219	75.203±0.143	23.564±0.157
UDIL (Ours)		95.097±0.447	**93.101**±**1.305**	**89.194**±**1.472**	**84.704**±**1.722**	**82.796**±**1.882**	**12.971**±**2.389**
GEM [58]	400	93.842±0.313	90.663±1.856	85.392±1.856	79.061±1.578	76.619±0.581	21.289±0.579
A-GEM [10]		92.820±1.274	83.564±5.024	72.616±3.865	62.223±2.081	59.654±0.122	39.196±0.171
ER [75]		94.916±0.457	91.491±1.878	86.029±2.176	78.688±1.323	76.794±0.696	20.696±0.744
DER++ [8]		95.246±0.228	93.627±1.147	90.011±1.289	85.601±0.982	84.258±0.544	13.692±0.560
CLS-ER [4]		95.233±0.271	92.740±1.268	89.111±1.305	83.678±1.388	81.771±0.354	15.455±0.356
ESM-ER [80]		**95.825**±**0.303**	93.378±1.480	89.290±1.604	83.868±1.163	82.192±0.164	16.195±0.150
UDIL (Ours)		95.274±0.469	**94.043**±**0.759**	**91.511**±**0.990**	**87.809**±**0.849**	**86.635**±**0.686**	**8.506**±**1.181**
GEM [58]	800	94.212±0.322	92.482±1.125	89.191±1.346	84.866±1.317	82.772±1.079	14.781±1.104
A-GEM [10]		92.902±1.194	84.611±4.451	75.150±3.421	64.510±2.437	61.240±1.026	37.528±1.089
ER [75]		95.144±0.281	92.997±1.195	89.319±1.365	84.352±1.681	81.877±1.157	15.285±1.196
DER++ [8]		95.496±0.261	**94.960**±**0.568**	**93.013**±**0.689**	**90.820**±**0.687**	**89.746**±**0.356**	7.821±0.371
CLS-ER [4]		95.462±0.174	93.927±0.881	91.275±0.930	87.816±0.988	86.418±0.215	10.598±0.228
ESM-ER [80]		**96.086**±**0.361**	94.746±0.915	92.393±0.974	89.745±0.712	88.662±0.263	9.409±0.255
UDIL (Ours)		95.354±0.480	94.711±0.563	92.776±0.695	90.399±0.755	89.191±0.685	**6.351**±**1.304**
Joint (Oracle)	∞	-	-	-	-	97.150±0.036	-

https://doi.org/10.52202/075280-0660

Table 7: **Performances (%) evaluated on *Seq-CORe50*.** Avg. Acc. and Forgetting are reported to measure the methods' performance. "↑" and "↓" mean higher and lower numbers are better, respectively. We use **boldface** and underlining to denote the best and the second-best performance, respectively. We use "-" to denote "not appliable" and "⋆" to denote out-of-memory (*OOM*) error when running the experiments.

Method	**Buffer**	$\mathcal{D}_{1:3}$	$\mathcal{D}_{4:6}$	$\mathcal{D}_{7:9}$	$\mathcal{D}_{10:11}$	**Overall**	
		Avg. Acc (↑)				Avg. Acc (↑)	Forgetting (↓)
Fine-tune	-	73.707±13.144	34.551±1.254	29.406±2.579	28.689±3.144	31.832±1.034	73.296±1.399
oEWC [81]	-	74.567±13.360	35.915±0.260	30.174±3.195	28.291±2.522	30.813±1.154	74.563±0.937
SI [105]	-	74.661±14.162	34.345±1.001	30.127±2.971	28.839±3.631	32.469±1.315	73.144±1.588
LwF [52]	-	80.383±10.190	28.357±1.143	31.386±0.787	28.711±2.981	31.692±0.768	72.990±1.350
GEM [58]	500	79.852±6.864	38.961±1.718	39.258±2.614	36.859±0.842	37.701±0.273	22.724±1.554
A-GEM [10]		80.348±9.394	41.472±3.394	43.213±1.542	39.181±3.999	43.181±2.025	33.775±3.003
ER [75]		90.838±2.177	79.343±2.699	68.151±0.226	65.034±1.571	66.605±0.214	32.750±0.455
DER++ [8]		92.444±1.764	88.652±1.854	80.391±0.107	78.038±0.591	78.629±0.753	21.910±1.094
CLS-ER [4]		89.834±1.323	78.909±1.724	70.591±0.322	⋆	⋆	⋆
ESM-ER [80]		84.905±6.471	51.905±3.257	53.815±1.770	50.178±2.574	52.751±1.296	25.444±0.580
UDIL (Ours)		**98.152±1.665**	**89.814±2.302**	**83.052±0.151**	**81.547±0.269**	**82.103±0.279**	**19.589±0.303**
GEM [58]	1000	78.717±4.831	43.269±3.419	40.908±2.200	40.408±1.168	41.576±1.599	18.537±1.237
A-GEM [10]		78.917±8.984	41.172±4.293	44.576±1.701	38.960±3.867	42.827±1.659	33.800±1.847
ER [75]		90.048±2.699	84.668±1.988	77.561±1.281	72.268±0.720	72.988±0.566	25.997±0.694
DER++ [8]		89.510±5.726	92.492±0.902	88.883±0.794	86.108±0.284	86.392±0.714	13.128±0.474
CLS-ER [4]		92.004±0.894	85.044±1.276	⋆	⋆	⋆	⋆
ESM-ER [80]		85.120±4.339	54.852±5.511	61.714±1.840	55.098±3.834	58.932±0.959	20.134±0.643
UDIL (Ours)		**98.648±1.174**	**93.447±1.111**	**90.545±0.705**	**87.923±0.232**	**88.155±0.445**	**12.882±0.460**
Joint (Oracle)	∞	-	-	-	-	99.137±0.049	-

https://doi.org/10.52202/075280-0660