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 [4] 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 [2, 36], we have

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

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 [23] and introducing the growth function $\Pi_{\mathcal{H}}$ [2, 36, 56] 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. $\square$

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* *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 $1-\delta$:*

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

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

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

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 [23] and introducing the growth function $\Pi_{\mathcal{H}}$ [2, 36, 56] at the same time; Eqn. 31 is achieved by using the fact $\Pi_{\mathcal{H}}(2N) \le (e \cdot 2N/d)^d \le (2N)^d$, 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. $\square$

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 [4] 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 $\mathcal{D}, \mathcal{D}'$,*

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

Proof. By definition, we have

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

$\square$

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:*

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

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

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:*

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

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

Proof. By applying the triangle inequality [4] of the 0-1 loss function, we have

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

$\square$

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:*

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

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

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

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

$\square$

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:*

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

where $\widehat{\epsilon}_{\mathcal{D}_i}(h, H_{t-1}) = \frac{1}{\widetilde{N}_i} \sum_{\boldsymbol{x}\in\widetilde{\mathcal{X}}_i} \mathbb{1}_{h(\boldsymbol{x})\ne H_{t-1}(\boldsymbol{x})}$, $\widehat{\epsilon}_{\mathcal{D}_t}(h, H_{t-1}) = \frac{1}{N_t} \sum_{\boldsymbol{x}\in\mathcal{X}_i} \mathbb{1}_{h(\boldsymbol{x})\ne H_{t-1}(\boldsymbol{x})}$, 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

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

Table 4: Unification of Existing Methods under UDIL.

	α_i	β_i	γ_i	**Transformed Objective**	**Condition**
UDIL (Ours)	$[0,1]$	$[0,1]$	$[0,1]$	-	-
LwF [29]	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 [46]	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++ [6]	$1/2$	0	$1/2$	$\mathcal{L}_{\text{DER++}}(h) = \widehat{\ell}_{B_t}(h) + \frac{1}{2} \sum_{i=1}^{t-1} \frac{\lvert B_t' \rvert/(t-1)}{\lvert B_t \rvert} [\widehat{\ell}_{B_i'}(h) + \widehat{\ell}_{B_i'}(h, H_{t-1})]$	$\lvert B_t \rvert = \frac{\lvert B_t' \rvert}{(t-1)}$
CLS-ER [3]	$\lambda/\lambda+1$	0	$1/\lambda+1$	$\mathcal{L}_{\text{CLS-ER}}(h) = \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \frac{1}{t-1} \widehat{\ell}_{B_i'}(h) + \sum_{i=1}^{t-1} \frac{\lambda}{t-1} \widehat{\ell}_{B_i'}(h, H_{t-1})$	$\lambda = t-2$
ESM-ER [50]	$\lambda/\lambda+1$	0	$1/\lambda+1$	$\mathcal{L}_{\text{CLS-ER}}(h) = \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \frac{1}{r(t-1)} \widehat{\ell}_{B_i'}(h) + \sum_{i=1}^{t-1} \frac{\lambda}{r(t-1)} \widehat{\ell}_{B_i'}(h, H_{t-1})$	$\begin{cases} \lambda = -1 + r(t-1) \\ r = 1 - e^{-1} \end{cases}$

Re-organizing the terms will give us

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

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) [29] 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:

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

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

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

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) [46] 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 [6], 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++) [6] includes an additional dark experience replay, i.e., knowledge distillation on the past domain exemplars, compared to ER [46]. 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).

Complementary Learning System based Experience Replay (CLS-ER) [3] 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*:

$$H_{t-1}^{(i)} \leftarrow \alpha^{(i)} \cdot H_{t-1}^{(i)} + (1 - \alpha^{(i)}) \cdot h, \qquad i \in \{p, s\}, \tag{46}$$

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:

$$H_{t-1}(\boldsymbol{x}) \triangleq \begin{cases} H_{t-1}^{(p)}(\boldsymbol{x}) & \text{if } [H_{t-1}^{(p)}(\boldsymbol{x})]_y > [H_{t-1}^{(s)}(\boldsymbol{x})]_y \\ H_{t-1}^{(s)}(\boldsymbol{x}) & \text{o.w.} \end{cases} \tag{47}$$

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 [50], 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 [20], 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:

$$\begin{aligned} \mathcal{L}_{\text{CLS-ER}}(h) &= \widehat{\ell}_{B_t}(h) + \widehat{\ell}_{B_t'}(h) + \lambda \widehat{\ell}_{B_t'}(h, H_{t-1}) && (48) \\ &= \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \tfrac{1}{t-1} \widehat{\ell}_{B_i'}(h) + \sum_{i=1}^{t-1} \tfrac{\lambda}{t-1} \widehat{\ell}_{B_i'}(h, H_{t-1}). && (49) \end{aligned}$$

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 5 in Table 4).

Error Sensitivity Modulation based Experience Replay (ESM-ER) [50] 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:

$$\ell(\boldsymbol{x}, y) = -\log([H_{t-1}^{(s)}(\boldsymbol{x})]_y). \tag{50}$$

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:

$$\lambda(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \ell(\boldsymbol{x}, y) \leq \beta \cdot \mu \\ \frac{\mu}{\ell(\boldsymbol{x}, y)} & \text{o.w.} \end{cases} \tag{51}$$

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:

$$\lambda^\star(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \ell(\boldsymbol{x}, y) \leq \mathbb{E}_{\boldsymbol{x},y}[\ell(\boldsymbol{x}, y)] \\ 0 & \text{o.w.} \end{cases} \tag{52}$$

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:

$$r = \int_0^{\frac{1}{\lambda_0}} 1 \cdot \lambda_0 e^{-\lambda_0 \ell(\boldsymbol{x},y)}\, d\ell(\boldsymbol{x},y) = \int_0^1 e^{-y} dy = (1 - e^{-1}). \tag{53}$$

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:

$$\begin{aligned}
\mathcal{L}_{\text{ESM-ER}}(h) &= r \cdot \widehat{\ell}_{B_t}(h) + \widehat{\ell}_{B'_t}(h) + \lambda \widehat{\ell}_{B'_t}(h, H_{t-1}) && (54)\\
&= r \cdot \widehat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \tfrac{1}{t-1} \widehat{\ell}_{B'_i}(h) + \sum_{i=1}^{t-1} \tfrac{\lambda}{t-1} \widehat{\ell}_{B'_i}(h, H_{t-1}). && (55)
\end{aligned}$$

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 6 in Table 4).

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_ϕ;
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 η;
Require: domain alignment strength coefficient λ_d, hyperparameter for generalization effect C.

```
h_θ ← H_{t-1}                                                  ▷ Initialization of the current model.
Ω ≜ {α_i, β_i, γ_i} ← {1/3, 1/3, 1/3}, for ∀i ∈ [t−1]          ▷ Initialization of the replay coefficient Ω.
for s = 1, ··· , S do
    B_t ~ S_t; B_i ~ M_i, ∀i ∈ [t−1]                           ▷ Sample a mini-batch of data from all domains.
    φ ← φ − η · λ_d · ∇_φ V_d(d, e, Ω̊)                          ▷ Discriminator training with Eqn. 16.
    Ω ← Ω − η · ∇_Ω V_{0-1}(h̊, Ω)                               ▷ Find a tighter bound with Eqn. 15.
    θ ← θ − η · ∇_θ(V_l(h_θ, Ω̊) − λ_d V_d(d, e, Ω̊))             ▷ Model training with Eqn. 14 and Eqn. 16.
end for
H_t ← h
M ← BalancedSampling(M, S_t)
return H_t                                                     ▷ For training on domain t + 1.
```

C Implementation Details of UDIL

This section delves into the implementation details of the UDIL algorithm. The algorithmic description of UDIL is presented in Alg. 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. Next, Sec. C.3 will cover the evaluation metrics used in this paper. Finally, Sec. C.4 and Sec. C.5 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:

$$\begin{bmatrix}\alpha_i \\ \beta_i \\ \gamma_i\end{bmatrix} = \operatorname{softmax}\left(\begin{bmatrix}\bar{\alpha}_i \\ \bar{\beta}_i \\ \bar{\gamma}_i\end{bmatrix}\right) = \begin{bmatrix}\exp(\bar{\alpha}_i)/Z_i \\ \exp(\bar{\beta}_i)/Z_i \\ \exp(\bar{\gamma}_i)/Z_i\end{bmatrix}, \tag{56}$$

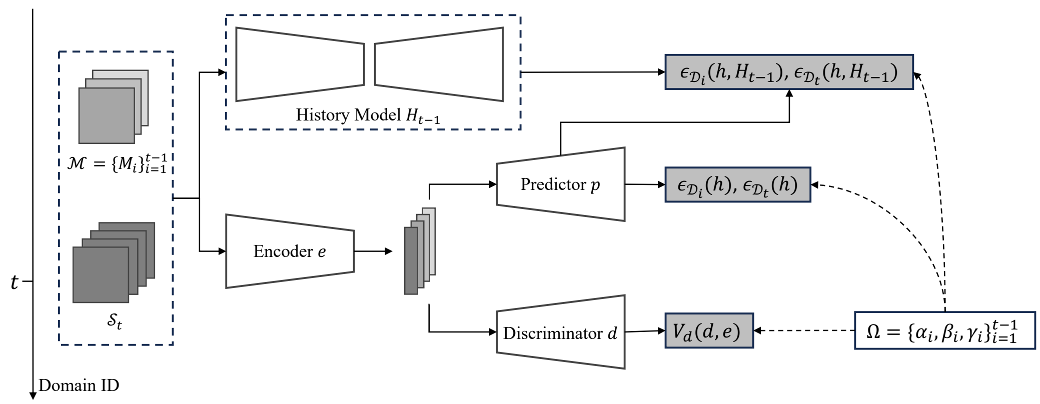


Figure 2: **Diagram of UDIL.** $\mathcal{M} = M_{i}{}_{i=1}^{t-1}$ represents the memory bank that stores all the past exemplars. $\mathcal{S}_t$ corresponds to the dataset from the current domain t, and the current model $h = p \circ e$ is depicted separately in the diagram. The three different categories 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.

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++ [6] and its following work [3, 50] that use reservoir sampling [58] 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 Alg. 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 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:

$$A_t \triangleq \tfrac{1}{t}\sum_{i=1}^{t} R_{t,i}. \tag{57}$$

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:

$$A_{t_1:t_2} \triangleq \tfrac{1}{t_2-t_1+1}\sum_{i=t_1}^{t_2} A_i. \tag{58}$$

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:

$$F_t \triangleq \tfrac{1}{t-1}\sum_{j=1}^{t-1} f_t(j), \tag{59}$$

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

$$f_t(j) \triangleq \max_{l\in[t-1]}\{R_{l,j} - R_{t,j}\}. \tag{60}$$

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:

$$W_t \triangleq \tfrac{1}{t-1}\sum_{i=2}^{t} R_{i-1,i} - r_i, \tag{61}$$

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.4 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**) [51], Synaptic Intelligence (**SI**) [66], and Learning without Forgetting (**LwF**) [29]. Memory-based domain incremental learning baselines include Gradient Episodic Memory (**GEM**) [34], Averaged Gradient Episodic Memory (**A-GEM**) [8], Experience Replay (**ER**) [46], Dark Experience Replay (**DER++**) [6], and two recent methods, Complementary Learning System based Experience Replay (**CLS-ER**) [3] and Error Senesitivity Modulation based Experience Replay (**ESM-ER**) [50]. In addition, we implement the fine-tuning (**Fine-tune**) [29] 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** [34]: 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** [8]: The improved baseline method where the constraints of GEM are averaged as one, which shortens the computational time significantly.
- **ER** [46]: The fundamental memory-based domain incremental learning framework where the mini-batch of the memory is regularly replayed with the current domain data.
- **DER++** [6]: 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** [3]: 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** [50]: An improved version of CLS-ER, where the effect of large errors when learning the current domain is reduced, dubbed "error sensitivity modulation".

C.5 Training Schemes

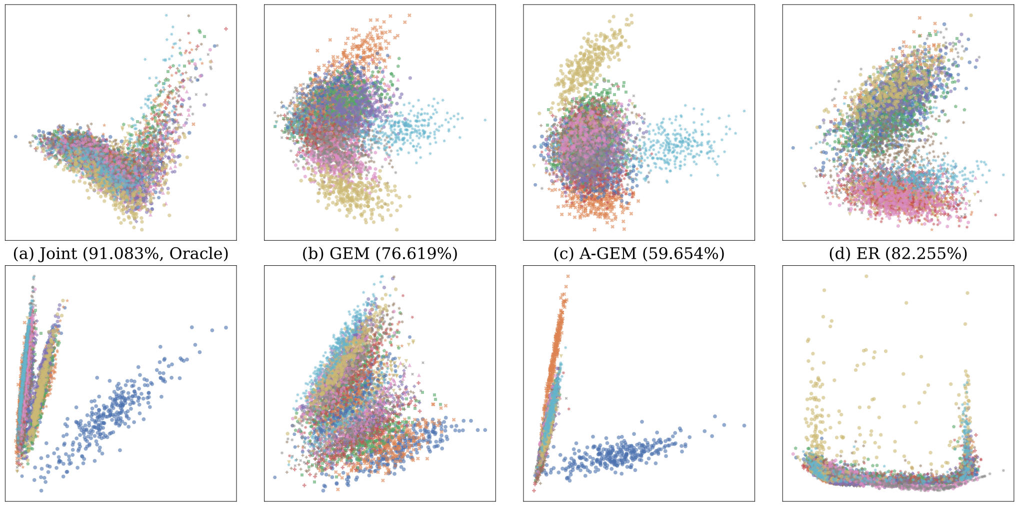
Training Process. For each group of experiments, we run three rounds with different seeds and report the mean and standard deviation of the results. We follow the optimal configurations (epochs and learning rate) stated in [6, 50] for the baselines in *P-MNIST* and *R-MNIST* dataset. For *HD-Balls* and *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 [59], 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++ [6] with hidden dimension set to 800 as well. In *Seq-CORe50*, we use the ResNet18 [19] 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.

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.



(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 [5]. **(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.

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.

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 [51]	-	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 [66]	-	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 [29]	-	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 [34]	200	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 [8]	200	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 [46]	200	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++ [6]	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 [3]	200	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 [50]	200	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)	200	**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 [34]	400	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 [8]	400	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 [46]	400	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++ [6]	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 [3]	400	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 [50]	400	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)	400	**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 [34]	800	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 [8]	800	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 [46]	800	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++ [6]	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 [3]	800	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 [50]	800	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)	800	**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	-

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 [51]	-	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 [66]	-	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 [29]	-	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 [34]	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 [8]	200	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 [46]	200	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++ [6]	200	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 [3]	200	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 [50]	200	**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)	200	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 [34]	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 [8]	400	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 [46]	400	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++ [6]	400	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 [3]	400	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 [50]	400	**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)	400	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 [34]	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 [8]	800	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 [46]	800	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++ [6]	800	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 [3]	800	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 [50]	800	**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)	800	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	-

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 <u>underlining</u> 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 [51]	-	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 [66]	-	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 [29]	-	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 [34]	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 [8]	500	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 [46]	500	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++ [6]	500	<u>92.444±1.764</u>	<u>88.652±1.854</u>	<u>80.391±0.107</u>	<u>78.038±0.591</u>	<u>78.629±0.753</u>	<u>21.910±1.094</u>
CLS-ER [3]	500	89.834±1.323	78.909±1.724	70.591±0.322	⋆	⋆	⋆
ESM-ER [50]	500	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)	500	**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 [34]	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 [8]	1000	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 [46]	1000	<u>90.048±2.699</u>	84.668±1.988	77.561±1.281	72.268±0.720	72.988±0.566	25.997±0.694
DER++ [6]	1000	89.510±5.726	<u>92.492±0.902</u>	<u>88.883±0.794</u>	<u>86.108±0.284</u>	<u>86.392±0.714</u>	<u>13.128±0.474</u>
CLS-ER [3]	1000	92.004±0.894	85.044±1.276	⋆	⋆	⋆	⋆
ESM-ER [50]	1000	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)	1000	**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	-

References

[1] 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.

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

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

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

[24] 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.

[25] 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.

[26] 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.

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

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

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

[30] 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.

[31] 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.

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

[33] 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.

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

[35] 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.

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

[37] 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.

[38] 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.

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

[40] 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.

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

[42] 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.

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

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

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

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

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

[56] 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.

[57] 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.

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

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

[60] 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.

[61] 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.

[62] 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.

[63] 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.

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

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

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

[67] 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.

[68] 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.