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 \(\alpha\)-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 \(1 - \delta\):

\[
\sum_j \alpha_j \epsilon_{\mathcal{D}_j}(h) \leq \sum_j \alpha_j \tilde{\epsilon}_{\mathcal{D}_j}(h) + \sqrt{\left(\sum_j \frac{\alpha_j^2}{N_j}\right)\left(8d \log \left(\frac{2eN}{d}\right) + 8 \log \left(\frac{2}{\delta}\right)\right)}. \tag{17}
\]

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

\[
\tilde{\epsilon}_\alpha(h) = \sum_j \alpha_j \tilde{\epsilon}_{\mathcal{D}_j}(h) = \sum_j \alpha_j \frac{1}{N_j} \sum_{x \in \mathcal{X}_j} \mathbb{1}_{h(x) \neq f_j(x)} = \frac{1}{N} \sum_j \sum_k R_{j,k}, \tag{18}
\]

where \(R_{j,k} = \left(\frac{\alpha_j N_j}{N}\right) \cdot \mathbb{1}_{h(x_k) \neq f_j(x_k)}\) is a random variable that takes the values in \(\{\alpha_j N_j, 0\}\). By the linearity of the expectation, we have \(\epsilon_\alpha(h) = E[\tilde{\epsilon}_\alpha(h)]\). Following [2, 36], we have

\[
P \left\{ \exists h \in \mathcal{H}, \text{s.t. } |\tilde{\epsilon}_\alpha(h) - \epsilon_\alpha(h)| \geq \epsilon \right\} \leq 2 \cdot P \left\{ \sup_{h \in \mathcal{H}} |\tilde{\epsilon}_\alpha(h) - \epsilon_\alpha(h)| \geq \frac{\epsilon}{2} \right\} \tag{19}
\]

\[
\leq 2 \cdot \mathbb{P} \left\{ \bigcup_{R_{j,k}, R'_{j,k}} \frac{1}{N} \sum_j \sum_k (R_{j,k} - R'_{j,k}) \geq \frac{\epsilon}{2} \right\} \tag{20}
\]

\[
\leq 2 \Pi_{\mathcal{H}}(2N) \exp \left\{ -\frac{2(\epsilon/2)^2}{\sum_j (N_j)^2} \right\} \tag{21}
\]

\[
= 2 \Pi_{\mathcal{H}}(2N) \exp \left\{ -\frac{\epsilon^2}{8 \sum_j (\alpha_j^2 / N_j)} \right\} \tag{22}
\]

\[
\leq 2(2N)^d \exp \left\{ -\frac{\epsilon^2}{8 \sum_j (\alpha_j^2 / N_j)} \right\}. \tag{23}
\]

where in Eqn. 20, \(\tilde{\epsilon}_\alpha(h)\) is the \(\alpha\)-weighted empirical loss evaluated on the “ghost” set of examples \(\{X^*_j\}\); 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 \(\Pi_{\mathcal{H}}(2N) \leq (\epsilon^{-2N/d})^d \leq (2N)^d\), where \(d\) is the VC-dimension of the hypothesis set \(\mathcal{H}\). Finally, by setting Eqn. 24 to \(\delta\) and solve for the error tolerance \(\epsilon\) will complete the proof.

**Lemma A.2 (Generalization Bound of \(\beta\)-weighted Labeling Functions).** Let \(\mathcal{D}\) be a single domain and \(\mathcal{X} = \{x_i\}_i^{N}\) be a collection of samples drawn from \(\mathcal{D}\). \(\mathcal{H}\) is a hypothesis space of VC dimension...
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 \( \overline{N}_t \) data points from each previous domain \( i < t \) in the memory bank.

With probability at least \( 1 - \delta \), we have:

\[
\sum_{i=1}^{t} \epsilon_{D_i}(h) \leq \sum_{i=1}^{t} \overline{\epsilon}_{D_i}(h) + \sqrt{\left( \frac{1}{N_t} + \sum_{i=1}^{t-1} \frac{1}{N_i} \right) \left( 8d \log \left( \frac{2eN}{d} \right) + 8 \log \left( \frac{2}{\delta} \right) \right)}. \tag{33}
\]
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 \( \epsilon_{D_i}(h) \) on the previous domain \( i \) has an upper bound:

\[
\epsilon_{D_i}(h) \leq \epsilon_{D_i}(h, H_{t-1}) + \epsilon_{D_i}(H_{t-1}),
\]

(34)

where \( \epsilon_{D_i}(h, H_{t-1}) \triangleq \mathbb{E}_{x \sim D_i}[h(x) \neq H_{t-1}(x)] \).

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

\[
\epsilon_{D_i}(h) = \epsilon_{D_i}(h, f_i)
\]

\[
\leq \epsilon_{D_i}(h, H_{t-1}) + \epsilon_{D_i}(H_{t-1}, f_i)
\]

\[
= \epsilon_{D_i}(h, H_{t-1}) + \epsilon_{D_i}(H_{t-1}). \quad \square
\]

**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 \( \epsilon_{D_i}(h) \) evaluated on the previous domain \( i \) then has an upper bound:

\[
\epsilon_{D_i}(h) \leq \epsilon_{D_i}(h, H_{t-1}) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(D_i, D_t) + \epsilon_{D_i}(H_{t-1}),
\]

(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 \( \epsilon_{D_i}(h, H_{t-1}) \triangleq \mathbb{E}_{x \sim D_i}[h(x) \neq H_{t-1}(x)] \).

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

\[
\epsilon_{D_i}(h) \leq \epsilon_{D_i}(h, H_{t-1}) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(D_i, D_t) + \left( \sum_{i=1}^{t-1} \beta_i d_{\mathcal{H}\Delta\mathcal{H}}(D_i, D_t) + \sum_{i=1}^{t-1} (\alpha_i + \beta_i) \epsilon_{D_i}(H_{t-1}) \right)
\]

\[
+ \sqrt{\left( \frac{1 + \sum_{i=1}^{t-1} \beta_i}{N_i} \right) \left( \sum_{i=1}^{t-1} \left( \alpha_i + \beta_i \right)^2 \right) (8d \log \left( \frac{2eN}{d} \right) + 8 \log \left( \frac{2}{\delta} \right))}
\]

\[
\triangleq g(h, H_{t-1}, \Omega), \quad (36)
\]

where \( \hat{\epsilon}_{D_i}(h, H_{t-1}) = \frac{1}{N_i} \sum_{x \in X_i} \mathbbm{1}_{h(x) \neq H_{t-1}(x)} \), \( \hat{\epsilon}_{D_i}(h, H_{t-1}) = \frac{1}{N_i} \sum_{x \in X_i} \mathbbm{1}_{h(x) \neq H_{t-1}(x)} \), 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

\[
\epsilon_{D_i}(h) = (\alpha_i + \beta_i + \gamma_i) \epsilon_{D_i}(h)
\]

\[
\leq \gamma_i \epsilon_{D_i}(h) + \alpha_i [\epsilon_{D_i}(h, H_{t-1}) + \epsilon_{D_i}(H_{t-1})]
\]

\[
+ \beta_i [\epsilon_{D_i}(h, H_{t-1}) + \epsilon_{D_i}(H_{t-1}) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(D_i, D_t)].
\]

Theorem 3.4 (Unified Generalization Bound for All Domains). Let \( \mathcal{H} \) be a hypothesis space of VC dimension \( d \). Let \( N = N_t + \sum_{i=1}^{t-1} N_i \) denote the total number of data points available to the training of current domain \( t \), where \( N_t \) and \( N_i \) 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:

\[
\sum_{i=1}^{t} \epsilon_{D_i}(h) \leq \left( \sum_{i=1}^{t-1} \gamma_i \epsilon_{D_i}(h) + \alpha_i \epsilon_{D_i}(h, H_{t-1}) \right) + \left( \sum_{i=1}^{t-1} \beta_i \epsilon_{D_i}(H_{t-1}) \right)
\]

\[
+ \sqrt{\left( \frac{1 + \sum_{i=1}^{t-1} \beta_i}{N_i} \right) \left( \sum_{i=1}^{t-1} (\alpha_i + \beta_i)^2 \right) (8d \log \left( \frac{2eN}{d} \right) + 8 \log \left( \frac{2}{\delta} \right))}
\]

\[
\triangleq g(h, H_{t-1}, \Omega), \quad (36)
\]

where \( \hat{\epsilon}_{D_i}(h, H_{t-1}) = \frac{1}{N_i} \sum_{x \in X_i} \mathbbm{1}_{h(x) \neq H_{t-1}(x)} \), \( \hat{\epsilon}_{D_i}(h, H_{t-1}) = \frac{1}{N_i} \sum_{x \in X_i} \mathbbm{1}_{h(x) \neq H_{t-1}(x)} \), and

\[
\Omega \triangleq \{ \alpha_i, \beta_i, \gamma_i \}_{i=1}^{t-1}.
\]
Re-organizing the terms will give us

\[
\sum_{i=1}^{t} \epsilon_{D_i}(h) \leq \left\{ \sum_{i=1}^{t-1} \left[ \gamma_i \epsilon_{D_i}(h) + \alpha_i \epsilon_{D_i}(h, H_{t-1}) \right] \right\} + \left\{ \epsilon_{D_t}(h) + \sum_{i=1}^{t-1} \beta_i \epsilon_{D_i}(h, H_{t-1}) \right\} \\
+ \frac{1}{2} \sum_{i=1}^{t-1} \beta_i d_{\mathcal{H}\Delta\mathcal{H}}(D_i, D_t) + \sum_{i=1}^{t-1} (\alpha_i + \beta_i) \epsilon_{D_i}(H_{t-1}).
\] (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 \(X_t\) at the beginning of the new task \(t\). Subsequently, knowledge distillation (as defined in Definition 4.2) is performed to mitigate forgetting:

\[
\mathcal{L}_{old}(h, H_{t-1}) \triangleq -\frac{1}{N_t} \sum_{x \in X_t} \sum_{k=1}^{K} [H_{t-1}(x)]_k \cdot \log([h(x)]_k) = \hat{\mathcal{L}}_{X_t}(h, H_{t-1}),
\] (38)

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

\[
\mathcal{L}_{new}(h) \triangleq -\frac{1}{N_t} \sum_{(x,y) \in \mathcal{S}_t} \sum_{k=1}^{K} \mathbb{1}_{y=k} \cdot \log([h(x)]_k) = \hat{\mathcal{L}}_{X_t}(h).
\] (39)

LwF uses a “loss balance weight” \(\lambda_o\) to balance two losses, which gives us its final loss for training:

\[
\mathcal{L}_{LwF}(h) \triangleq \mathcal{L}_{new}(h) + \lambda_o \cdot \mathcal{L}_{old}(h, H_{t-1}).
\] (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
where, 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, \( \lambda_0 \) 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:

\[
\mathcal{L}_{\text{ER}}(h) = \frac{1}{|B_t| + |B'_t|} \sum_{(x,y) \in B_t \cup B'_t} \sum_{y=k}^{K} \mathbb{I}_{y=k} \cdot [\log([h(x)]_k)]
\]

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:

\[
\mathcal{L}_{\text{ER}}(h) = \frac{|B_t|}{|B_t| + |B'_t|} \hat{L}_{B_t}(h) + \frac{|B'_t|}{|B_t| + |B'_t|} \hat{L}_{B'_t}(h).
\]

where \( B'_t = \{(x,y);(x,y) \in (B'_t \cap M_t)\} \) 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:

\[
\mathcal{L}_{\text{DER++}}(h) = \frac{|B_t|}{|B_t| + |B'_t|} \hat{L}_{B_t}(h) + \frac{1}{2} \sum_{k=1}^{K} \mathbb{I}_{y=k} \cdot [\log([h(x)]_k)] + \frac{1}{2} \sum_{k=1}^{K} \mathbb{I}_{y=k} \cdot [\log([h(x)]_k)]
\]

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 \( H^{(p)}_{t-1} \) and the stable model \( H^{(s)}_{t-1} \), 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:

\[
H^{(i)}_{t-1} \leftarrow \alpha^{(i)} \cdot H^{(i)}_{t-1} + (1 - \alpha^{(i)}) \cdot h, \quad i \in \{p,s\},
\]

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. 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:

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

where $(x, y) \in \mathcal{M}$ 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 $\hat{\ell}_{B_t}(h) + \hat{\ell}_{B'_t}(h)$ instead of $\hat{\ell}_{B_t\cup B'_t}(h)$. Based on the assumption from [20], the MSE loss on the logits is equivalent to the entropy-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:

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

$$= \hat{\ell}_{B_t}(h) + \sum_{i=1}^{t-1} \frac{1}{\ell_{i}} \hat{\ell}_{B'_t}(h) + \sum_{i=1}^{t-1} \frac{1}{\ell_{i}} \hat{\ell}_{B'_t}(h, H_{t-1}).$$  \hspace{1cm} (49)$$

Therefore, by imposing the constraint $\alpha_i + \beta_i + \gamma_i = 1$, we find that $\lambda = t - 2$. Substituting this value back into $\lambda/\gamma_{t-1}$ 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 $(x, y) \sim \mathcal{D}_t$, 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:

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

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

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

where $\beta$ 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 $\mathbb{E}_{x, y} [\ell(x, y)]$ is used instead of the running estimate $\mu$; (iii) a hard screening mechanism is employed instead of the current re-scaling approach. Based on these assumptions, we determine the sample-wise weights $\lambda^*$ according to the following rule:

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

Under the assumption that the loss value $\ell(x, y)$ follows an exponential distribution, denoted as $\ell(x, y) \sim \text{Exp}(\lambda_0)$, where the probability density function is given by $f(\ell(x, y), \lambda_0) = \lambda_0 e^{-\lambda_0 \ell(x, y)}$, we can calculate the expectation of the loss as $\mathbb{E}_{x, y} [\ell(x, y)] = 1 / \lambda_0$. Based on
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 Eqn. (10), we find that ESM-ER is equivalent to \( \{ \alpha_i = \lambda/\lambda+1, \beta_i = 0, \gamma_i = 1/\lambda+1 \} \) in UDIL, where \( \lambda = r \cdot (t - 1) - 1 \). Consequently, the original training loss of ESM-ER can be transformed as follows:

\[
L_{\text{ESM-ER}}(h) = r \cdot \hat{L}_{B_i}(h) + \lambda \hat{L}_{B_i}(h, H_{t-1})
\]

\[
= r \cdot \hat{L}_{B_i}(h) + \sum_{i=1}^{t-1} \frac{1}{\lambda_i} \hat{L}_{B_i}(h) + \sum_{i=1}^{t-1} \frac{1}{\lambda_i} \hat{L}_{B_i}(h, H_{t-1}).
\]

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 Eqn. (10), we find that ESM-ER is equivalent to \( \{ \alpha_i = \lambda/\lambda+1, \beta_i = 0, \gamma_i = 1/\lambda+1 \} \) in UDIL, where \( \lambda = r \cdot (t - 1) - 1 \). Consequently, the original training loss of ESM-ER can be transformed as follows:

\[
L_{\text{ESM-ER}}(h) = r \cdot \hat{L}_{B_i}(h) + \lambda \hat{L}_{B_i}(h, H_{t-1})
\]

\[
= r \cdot \hat{L}_{B_i}(h) + \sum_{i=1}^{t-1} \frac{1}{\lambda_i} \hat{L}_{B_i}(h) + \sum_{i=1}^{t-1} \frac{1}{\lambda_i} \hat{L}_{B_i}(h, H_{t-1}).
\]

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

**Require:** history model \( H_{t-1} = P_{t-1} \circ \epsilon_{t-1} \), current model \( h_\theta = p \circ e, \) discriminator model \( d_\phi \); memory bank \( M = \{ M_i \}_{i=1}^{t-1} \); batch size \( B \), learning rate \( \eta \);

\begin{enumerate}
  \item \( h_\theta \leftarrow H_{t-1} \) \hfill \triangleright \text{Initialization of the current model}.
  \item \( \Omega \leftarrow \{ \alpha_i, \beta_i, \gamma_i \} \leftarrow \{1/3, 1/3, 1/3\} \), for \( \forall i \in [t-1] \) \hfill \triangleright \text{Initialization of the replay coefficient } \Omega.
  \item for \( s = 1, \ldots, S \) do
    \item \( B_i \sim S; B_i \sim M_i, \forall i \in [t-1] \) \hfill \triangleright \text{Sample a mini-batch of data from all domains}.
    \item \( \phi \leftarrow \phi - \eta \cdot \lambda d \cdot \nabla_\phi V_d(d, e, \Omega) \) \hfill \triangleright \text{Discriminator training with Eqn. 16}.
    \item \( \Omega \leftarrow \Omega - \eta \cdot \nabla_\Omega V_{0:1}(h, \Omega) \) \hfill \triangleright \text{Find a tighter bound with Eqn. 15}.
    \item \( \theta \leftarrow \theta - \eta \cdot \nabla_\theta \left( V_i(h_\theta, \Omega) - \lambda_d V_d(d, e, \Omega) \right) \) \hfill \triangleright \text{Model training with Eqn. 14 and Eqn. 16}.
  \end{enumerate}

\begin{enumerate}
  \item \( H_t \leftarrow h \)
  \item \( \mathcal{M} \leftarrow \text{BalancedSampling}(\mathcal{M}, S_t) \)
  \item \( \text{return } H_t \) \hfill \triangleright \text{For training on domain } t + 1.
\end{enumerate}

### 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 \( \{ \alpha_i, \beta_i, \gamma_i \} \in \mathbb{R}^2 \) 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 \( \Omega \) 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 \( \{ \tilde{\alpha}_i, \tilde{\beta}_i, \tilde{\gamma}_i \} \in \mathbb{R}^3 \) and the softmax function to indirectly calculate \( \Omega \) during training. Concretely, we have:

\[
\begin{bmatrix}
\alpha_i \\
\beta_i \\
\gamma_i
\end{bmatrix} = \text{softmax} \left( \begin{bmatrix}
\tilde{\alpha}_i & \tilde{\beta}_i & \tilde{\gamma}_i
\end{bmatrix} \right) = \begin{bmatrix}
\exp(\tilde{\alpha}_i)/Z_i \\
\exp(\tilde{\beta}_i)/Z_i \\
\exp(\tilde{\gamma}_i)/Z_i
\end{bmatrix}, \tag{56}
\]
where $Z_t = \exp(\alpha_t) + \exp(\tilde{\beta}_t) + \exp(\gamma_t)$ is the normalizing constant. At the beginning of training on domain $t$, the logit variables $\{\alpha_t, \tilde{\beta}_t, \gamma_t\} = \{0, 0, 0\}$ 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 $\lceil |\mathcal{M}|/t \rceil$ 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 $S_t$, domain ID $t$.

1. for $i = 1, \ldots , t - 1$ do
2.   for $j = 1, \ldots , \lceil |\mathcal{M}|/t-1 - |\mathcal{M}|/t \rceil$ do
3.     $(x, y) \leftarrow \text{RandomSample}(M_i)$
4.     $(x', y') \leftarrow \text{RandomSample}(S_t)$
5.     Swap $(x', y')$ into $\mathcal{M}$, replacing $(x, y)$
6.   end for
7. end for
8. return $\mathcal{M}$

### C.3 Evaluation Metrics

In continual learning, many evaluation metrics are based on the **Accuracy Matrix** $R \in \mathbb{R}^{T \times T}$, where $T$ represents the total number of tasks (domains). In the accuracy matrix $R$, the entry $R_{t,i,j}$ 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:

$$A_t \triangleq \frac{1}{t} \sum_{i=1}^{t} R_{t,i}. \quad (57)$$

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:

$$A_{t_1:t_2} \triangleq \frac{1}{t_2-t_1+1} \sum_{i=t_1}^{t_2} A_i. \quad (58)$$

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:

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

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

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

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 $P$ 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 $S$ has a similar average accuracy to $P$ but significantly lower average forgetting, we can infer that the model $S$ 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:

$$W_t \triangleq \frac{1}{t} \sum_{i=2}^{t} R_{t-1,i} - r_i, \quad (61)$$

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 Sensitivitivy 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 $\lambda_d$ and parameter $C$ that models the combined effect of VC-dimension $d$ and error tolerance $\delta$, 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.
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 applicable”.

<table>
<thead>
<tr>
<th>Method</th>
<th>Buffer</th>
<th>(D_{1:5})</th>
<th>(D_{5:10})</th>
<th>(D_{11:15})</th>
<th>(D_{16:20})</th>
<th>Overall Avg. Acc (Avg.)</th>
<th>Forgetting ((\pm))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tune</td>
<td>-</td>
<td>92.506 ±0.062</td>
<td>87.088 ±1.337</td>
<td>81.295 ±2.372</td>
<td>72.807 ±1.817</td>
<td>70.102 ±2.945</td>
<td>27.522 ±3.042</td>
</tr>
<tr>
<td>oEWC [51]</td>
<td>-</td>
<td>92.415 ±0.016</td>
<td>87.988 ±1.607</td>
<td>83.098 ±1.843</td>
<td>78.670 ±0.902</td>
<td>78.476 ±1.223</td>
<td>18.068 ±1.321</td>
</tr>
<tr>
<td>SI [55]</td>
<td>-</td>
<td>92.282 ±0.062</td>
<td>87.986 ±1.622</td>
<td>83.695 ±1.220</td>
<td>79.609 ±0.709</td>
<td>79.045 ±1.357</td>
<td>17.409 ±1.446</td>
</tr>
<tr>
<td>LwF [29]</td>
<td>-</td>
<td>95.025 ±0.487</td>
<td>91.402 ±1.546</td>
<td>83.984 ±2.103</td>
<td>76.046 ±2.004</td>
<td>73.545 ±2.646</td>
<td>24.556 ±2.789</td>
</tr>
<tr>
<td>GEM [34]</td>
<td></td>
<td>93.310 ±0.774</td>
<td>91.900 ±0.456</td>
<td>89.813 ±0.914</td>
<td>87.251 ±0.524</td>
<td>86.729 ±0.203</td>
<td>9.430 ±0.156</td>
</tr>
<tr>
<td>A-GEM [8]</td>
<td></td>
<td>93.326 ±0.363</td>
<td>91.466 ±0.605</td>
<td>89.048 ±0.005</td>
<td>86.518 ±0.604</td>
<td>85.712 ±0.228</td>
<td>10.485 ±0.196</td>
</tr>
<tr>
<td>ER [46]</td>
<td></td>
<td>94.087 ±0.762</td>
<td>92.397 ±0.444</td>
<td>89.999 ±0.100</td>
<td>87.492 ±0.448</td>
<td>86.963 ±0.303</td>
<td>9.273 ±0.255</td>
</tr>
<tr>
<td>DER++ [6]</td>
<td>200</td>
<td>94.708 ±0.451</td>
<td>94.582 ±0.158</td>
<td>93.271 ±0.585</td>
<td>90.980 ±0.610</td>
<td>90.333 ±0.587</td>
<td>6.110 ±0.545</td>
</tr>
<tr>
<td>CLS-ER [3]</td>
<td></td>
<td>94.761 ±0.340</td>
<td>93.943 ±0.197</td>
<td>92.725 ±0.566</td>
<td>91.150 ±0.357</td>
<td>90.726 ±0.218</td>
<td>5.428 ±0.252</td>
</tr>
<tr>
<td>ESM-ER [50]</td>
<td></td>
<td>95.198 ±0.236</td>
<td>94.029 ±0.427</td>
<td>91.710 ±1.056</td>
<td>88.181 ±1.021</td>
<td>86.851 ±0.838</td>
<td>10.007 ±0.864</td>
</tr>
<tr>
<td>UDIL (Ours)</td>
<td></td>
<td>95.747 ±3.486</td>
<td>94.695 ±0.256</td>
<td>93.756 ±0.343</td>
<td>92.254 ±0.564</td>
<td>91.483 ±0.270</td>
<td>4.399 ±0.314</td>
</tr>
<tr>
<td>GEM [34]</td>
<td></td>
<td>93.557 ±0.225</td>
<td>92.635 ±0.306</td>
<td>91.246 ±0.492</td>
<td>89.565 ±0.342</td>
<td>89.097 ±0.149</td>
<td>6.975 ±0.167</td>
</tr>
<tr>
<td>A-GEM [8]</td>
<td></td>
<td>93.432 ±0.333</td>
<td>92.064 ±0.439</td>
<td>90.038 ±0.726</td>
<td>87.988 ±0.335</td>
<td>87.560 ±0.087</td>
<td>8.577 ±0.053</td>
</tr>
<tr>
<td>ER [46]</td>
<td></td>
<td>93.525 ±1.011</td>
<td>91.649 ±0.562</td>
<td>90.426 ±0.456</td>
<td>88.728 ±0.353</td>
<td>88.339 ±0.044</td>
<td>7.180 ±0.029</td>
</tr>
<tr>
<td>DER++ [6]</td>
<td>400</td>
<td>94.952 ±0.403</td>
<td>95.089 ±0.075</td>
<td>94.458 ±0.328</td>
<td>93.257 ±0.249</td>
<td>92.950 ±0.261</td>
<td>3.378 ±0.245</td>
</tr>
<tr>
<td>CLS-ER [3]</td>
<td></td>
<td>94.262 ±0.449</td>
<td>93.195 ±0.148</td>
<td>92.623 ±0.195</td>
<td>91.839 ±0.187</td>
<td>91.598 ±0.117</td>
<td>3.795 ±0.144</td>
</tr>
<tr>
<td>ESM-ER [50]</td>
<td></td>
<td>95.413 ±0.139</td>
<td>94.654 ±0.311</td>
<td>93.353 ±0.588</td>
<td>91.022 ±0.781</td>
<td>89.829 ±0.698</td>
<td>6.888 ±0.738</td>
</tr>
<tr>
<td>UDIL (Ours)</td>
<td></td>
<td>95.992 ±3.349</td>
<td>95.026 ±0.250</td>
<td>94.212 ±2.280</td>
<td>93.094 ±0.526</td>
<td>92.666 ±0.108</td>
<td>2.853 ±1.017</td>
</tr>
<tr>
<td>GEM [34]</td>
<td></td>
<td>93.717 ±0.177</td>
<td>93.116 ±0.206</td>
<td>92.166 ±0.335</td>
<td>91.076 ±0.342</td>
<td>90.609 ±0.364</td>
<td>5.393 ±0.417</td>
</tr>
<tr>
<td>A-GEM [8]</td>
<td></td>
<td>93.612 ±0.241</td>
<td>92.523 ±0.375</td>
<td>90.718 ±0.739</td>
<td>88.543 ±0.391</td>
<td>88.020 ±0.831</td>
<td>8.081 ±0.867</td>
</tr>
<tr>
<td>ER [46]</td>
<td></td>
<td>93.827 ±0.871</td>
<td>92.457 ±0.217</td>
<td>91.688 ±0.277</td>
<td>90.617 ±0.289</td>
<td>90.252 ±0.056</td>
<td>5.188 ±0.045</td>
</tr>
<tr>
<td>DER++ [6]</td>
<td>800</td>
<td>95.295 ±0.317</td>
<td>95.539 ±0.041</td>
<td>95.099 ±0.187</td>
<td>94.423 ±0.151</td>
<td>94.227 ±0.261</td>
<td>2.106 ±0.161</td>
</tr>
<tr>
<td>CLS-ER [3]</td>
<td></td>
<td>94.463 ±0.537</td>
<td>93.567 ±0.093</td>
<td>93.182 ±0.137</td>
<td>92.744 ±0.112</td>
<td>92.578 ±0.152</td>
<td>2.803 ±0.183</td>
</tr>
<tr>
<td>ESM-ER [50]</td>
<td></td>
<td>95.567 ±0.150</td>
<td>95.136 ±0.202</td>
<td>94.301 ±0.347</td>
<td>92.981 ±0.597</td>
<td>92.408 ±0.387</td>
<td>4.170 ±0.357</td>
</tr>
<tr>
<td>UDIL (Ours)</td>
<td></td>
<td>96.082 ±3.313</td>
<td>95.207 ±0.196</td>
<td>94.642 ±0.156</td>
<td>93.997 ±0.194</td>
<td>93.724 ±0.043</td>
<td>1.633 ±0.038</td>
</tr>
</tbody>
</table>

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 applicable".

<table>
<thead>
<tr>
<th>Method</th>
<th>Buffer</th>
<th>(D_{1.5})</th>
<th>(D_{6.10})</th>
<th>(D_{11.15})</th>
<th>(D_{16.20})</th>
<th>Overall Avg. Acc (%)</th>
<th>Forgetting (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tune</td>
<td>-</td>
<td>92.961 ±0.283</td>
<td>76.617 ±0.011</td>
<td>60.212 ±3.688</td>
<td>49.793 ±1.552</td>
<td>47.803 ±1.703</td>
<td>52.281 ±1.797</td>
</tr>
<tr>
<td>eEWC [51]</td>
<td>-</td>
<td>91.765 ±2.236</td>
<td>76.226 ±7.622</td>
<td>60.320 ±3.892</td>
<td>50.505 ±1.712</td>
<td>48.203 ±0.827</td>
<td>51.181 ±0.867</td>
</tr>
<tr>
<td>LwF [29]</td>
<td>-</td>
<td>95.174 ±1.154</td>
<td>83.044 ±9.935</td>
<td>65.899 ±4.061</td>
<td>55.980 ±1.296</td>
<td>54.709 ±0.515</td>
<td>45.473 ±0.565</td>
</tr>
<tr>
<td>GEM [34]</td>
<td>-</td>
<td>93.441 ±0.610</td>
<td>88.620 ±2.381</td>
<td>81.034 ±2.704</td>
<td>73.112 ±1.922</td>
<td>70.545 ±0.623</td>
<td>27.684 ±0.645</td>
</tr>
<tr>
<td>A-GEM [8]</td>
<td>-</td>
<td>92.667 ±1.352</td>
<td>82.772 ±5.503</td>
<td>70.579 ±4.028</td>
<td>60.462 ±2.001</td>
<td>57.958 ±0.579</td>
<td>40.969 ±0.580</td>
</tr>
<tr>
<td>ER [16]</td>
<td>-</td>
<td>94.705 ±0.790</td>
<td>89.171 ±2.883</td>
<td>79.962 ±3.365</td>
<td>71.787 ±1.608</td>
<td>69.627 ±0.911</td>
<td>28.749 ±0.993</td>
</tr>
<tr>
<td>DER++ [6]</td>
<td>400</td>
<td>94.904 ±0.414</td>
<td>91.637 ±1.871</td>
<td>84.915 ±2.315</td>
<td>78.373 ±1.244</td>
<td>76.671 ±0.931</td>
<td>21.743 ±0.409</td>
</tr>
<tr>
<td>CLS-ER [3]</td>
<td>-</td>
<td>95.131 ±0.523</td>
<td>91.421 ±1.732</td>
<td>84.773 ±2.665</td>
<td>77.733 ±1.480</td>
<td>75.609 ±0.418</td>
<td>22.483 ±0.456</td>
</tr>
<tr>
<td>ESM-ER [50]</td>
<td>-</td>
<td>95.378 ±0.531</td>
<td>90.800 ±2.528</td>
<td>83.438 ±2.581</td>
<td>76.987 ±1.219</td>
<td>75.203 ±0.143</td>
<td>23.564 ±0.157</td>
</tr>
<tr>
<td>UDIL (Ours)</td>
<td>-</td>
<td>95.097 ±0.447</td>
<td>93.101 ±1.365</td>
<td>89.194 ±1.472</td>
<td>84.704 ±1.722</td>
<td>82.796 ±1.882</td>
<td>12.971 ±2.389</td>
</tr>
<tr>
<td>GEM [34]</td>
<td>-</td>
<td>93.842 ±0.313</td>
<td>90.663 ±1.856</td>
<td>85.392 ±1.856</td>
<td>79.061 ±1.578</td>
<td>76.619 ±0.581</td>
<td>21.289 ±0.579</td>
</tr>
<tr>
<td>A-GEM [8]</td>
<td>-</td>
<td>92.820 ±1.274</td>
<td>83.564 ±0.502</td>
<td>72.616 ±3.865</td>
<td>62.223 ±2.081</td>
<td>59.654 ±0.122</td>
<td>39.196 ±0.171</td>
</tr>
<tr>
<td>ER [16]</td>
<td>-</td>
<td>94.916 ±0.457</td>
<td>91.491 ±1.878</td>
<td>86.029 ±2.176</td>
<td>78.688 ±1.323</td>
<td>76.794 ±0.606</td>
<td>20.696 ±0.744</td>
</tr>
<tr>
<td>DER++ [6]</td>
<td>400</td>
<td>95.246 ±0.228</td>
<td>93.627 ±1.147</td>
<td>90.011 ±1.289</td>
<td>85.601 ±0.982</td>
<td>84.258 ±0.544</td>
<td>13.692 ±0.550</td>
</tr>
<tr>
<td>CLS-ER [3]</td>
<td>-</td>
<td>95.233 ±0.271</td>
<td>92.740 ±1.268</td>
<td>89.111 ±1.385</td>
<td>83.678 ±1.388</td>
<td>81.771 ±0.374</td>
<td>15.455 ±0.356</td>
</tr>
<tr>
<td>ESM-ER [50]</td>
<td>-</td>
<td>95.825 ±0.303</td>
<td>93.378 ±1.480</td>
<td>89.290 ±1.604</td>
<td>83.868 ±1.163</td>
<td>82.192 ±0.164</td>
<td>16.195 ±0.150</td>
</tr>
<tr>
<td>UDIL (Ours)</td>
<td>-</td>
<td>95.274 ±0.469</td>
<td>94.043 ±0.759</td>
<td>91.511 ±0.990</td>
<td>87.809 ±0.849</td>
<td>86.635 ±0.686</td>
<td>8.506 ±1.181</td>
</tr>
<tr>
<td>GEM [34]</td>
<td>-</td>
<td>94.212 ±0.322</td>
<td>92.482 ±1.125</td>
<td>89.191 ±1.346</td>
<td>84.866 ±1.317</td>
<td>82.772 ±0.107</td>
<td>14.781 ±1.104</td>
</tr>
<tr>
<td>A-GEM [8]</td>
<td>-</td>
<td>92.902 ±1.194</td>
<td>84.611 ±4.431</td>
<td>75.150 ±3.421</td>
<td>64.510 ±2.437</td>
<td>61.240 ±1.026</td>
<td>37.528 ±1.089</td>
</tr>
<tr>
<td>ER [16]</td>
<td>-</td>
<td>94.144 ±0.281</td>
<td>92.997 ±1.195</td>
<td>89.319 ±1.365</td>
<td>84.352 ±1.681</td>
<td>81.877 ±1.157</td>
<td>15.285 ±1.196</td>
</tr>
<tr>
<td>DER++ [6]</td>
<td>800</td>
<td>95.496 ±0.561</td>
<td>94.960 ±0.568</td>
<td>93.013 ±0.689</td>
<td>90.820 ±0.687</td>
<td>89.746 ±0.356</td>
<td>7.821 ±0.371</td>
</tr>
<tr>
<td>CLS-ER [3]</td>
<td>-</td>
<td>95.462 ±0.174</td>
<td>93.927 ±0.880</td>
<td>91.275 ±0.930</td>
<td>87.816 ±0.988</td>
<td>86.418 ±0.215</td>
<td>10.598 ±0.228</td>
</tr>
<tr>
<td>ESM-ER [50]</td>
<td>-</td>
<td>96.086 ±0.361</td>
<td>94.746 ±0.915</td>
<td>92.393 ±0.974</td>
<td>89.745 ±0.712</td>
<td>86.662 ±0.263</td>
<td>9.409 ±0.255</td>
</tr>
<tr>
<td>UDIL (Ours)</td>
<td>-</td>
<td>95.354 ±0.480</td>
<td>94.711 ±0.503</td>
<td>92.776 ±0.695</td>
<td>90.399 ±0.755</td>
<td>89.191 ±0.645</td>
<td>6.351 ±1.304</td>
</tr>
</tbody>
</table>

Joint (Oracle) \(\infty\) - - - - 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 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Buffer</th>
<th>(D_{1:3})</th>
<th>(D_{4:6})</th>
<th>(D_{7:9})</th>
<th>(D_{10:11})</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg. Acc (%)</td>
<td></td>
<td></td>
<td></td>
<td>Avg. Acc (%)</td>
</tr>
<tr>
<td>SI [66]</td>
<td>-</td>
<td>74.661±14.162</td>
<td>34.345±1.001</td>
<td>30.127±2.971</td>
<td>28.839±3.631</td>
<td>32.469±1.315</td>
</tr>
<tr>
<td>LwF [29]</td>
<td>-</td>
<td>80.383±10.190</td>
<td>28.357±1.143</td>
<td>31.386±0.787</td>
<td>28.711±2.981</td>
<td>31.692±0.768</td>
</tr>
<tr>
<td>GEM [34]</td>
<td>-</td>
<td>79.852±6.864</td>
<td>38.961±1.718</td>
<td>39.258±2.644</td>
<td>36.859±0.842</td>
<td>37.701±0.273</td>
</tr>
<tr>
<td>ER [46]</td>
<td>-</td>
<td>90.838±2.177</td>
<td>79.343±2.699</td>
<td>68.151±1.206</td>
<td>65.034±1.571</td>
<td>66.605±0.214</td>
</tr>
<tr>
<td>DER++ [6]</td>
<td>500</td>
<td>92.444±1.764</td>
<td>88.652±1.854</td>
<td>80.391±1.017</td>
<td>78.038±0.591</td>
<td>78.629±0.753</td>
</tr>
<tr>
<td>CLS-ER [3]</td>
<td>-</td>
<td>89.834±1.323</td>
<td>78.909±1.724</td>
<td>70.591±0.322</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>ESM-ER [50]</td>
<td>-</td>
<td>84.905±6.471</td>
<td>51.905±3.257</td>
<td>53.815±1.779</td>
<td>50.178±2.574</td>
<td>52.751±1.296</td>
</tr>
<tr>
<td>UDIL (Ours)</td>
<td>-</td>
<td>98.152±1.665</td>
<td>89.814±2.382</td>
<td>83.052±0.151</td>
<td>81.547±0.269</td>
<td>82.103±0.279</td>
</tr>
<tr>
<td>GEM [34]</td>
<td>-</td>
<td>78.717±4.831</td>
<td>43.269±3.419</td>
<td>40.908±2.300</td>
<td>40.408±1.168</td>
<td>41.576±1.599</td>
</tr>
<tr>
<td>A-GEM [6]</td>
<td>-</td>
<td>78.917±8.984</td>
<td>41.172±4.293</td>
<td>44.576±1.701</td>
<td>38.960±3.867</td>
<td>42.827±1.659</td>
</tr>
<tr>
<td>ER [16]</td>
<td>-</td>
<td>90.048±2.699</td>
<td>84.668±1.988</td>
<td>77.561±1.281</td>
<td>72.268±1.720</td>
<td>72.988±0.566</td>
</tr>
<tr>
<td>DER++ [6]</td>
<td>1000</td>
<td>89.510±5.726</td>
<td>92.492±0.902</td>
<td>88.883±0.784</td>
<td>86.108±0.284</td>
<td>86.392±0.714</td>
</tr>
<tr>
<td>CLS-ER [3]</td>
<td>-</td>
<td>92.004±0.984</td>
<td>85.044±1.276</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>ESM-ER [50]</td>
<td>-</td>
<td>85.120±4.339</td>
<td>54.852±5.511</td>
<td>61.714±1.840</td>
<td>55.098±3.834</td>
<td>58.932±0.959</td>
</tr>
<tr>
<td>UDIL (Ours)</td>
<td>-</td>
<td>98.648±1.174</td>
<td>93.447±1.114</td>
<td>90.545±0.780</td>
<td>87.923±0.232</td>
<td>88.155±0.445</td>
</tr>
</tbody>
</table>

Joint (Oracle) ∞ - - - - 99.137±0.049 -
References


