LEARNING TO MANAGE MEMORY FOR DOMAIN INCREMENTAL LEARNING VIA ADAPTIVE REPLAY

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world data often evolves across domains, challenging models to adapt to new distributions while retaining previously acquired knowledge. Replay-based methods for Domain Incremental Learning (DIL) address this challenge by interleaving current data with samples from previous domains. However, as the model and domains evolve, some stored memories may become stale, redundant, or less useful for replay. The challenge is therefore not simply to preserve memory, but to keep it useful. Inspired by neuroscience studies that view forgetting as an active and functional process of memory regulation, we propose **R**einforcement learning-based **A**daptive **M**emory (**RAM**), a framework that treats memory as an adaptive resource rather than a static collection of raw samples. We further present a *Mutual-Information Sequential Adaptation Bound* that relates cross-domain error bounds to the mutual information between semantic-related representations and domain identity. This analysis motivates separating semantic- and domain-related representations, whose drift provides structured signals for assessing memory throughout training. Guided by drift-based priors and performance feedback across all seen domains, **RAM** learns a lightweight policy to actively *retain*, *repair*, or *remove* memory entries under a fixed budget. Experiments on four DIL benchmarks demonstrate that **RAM** consistently improves accuracy and reduces forgetting across different backbones, outperforming the evaluated baselines and highlighting the value of active memory regulation for continual learning.

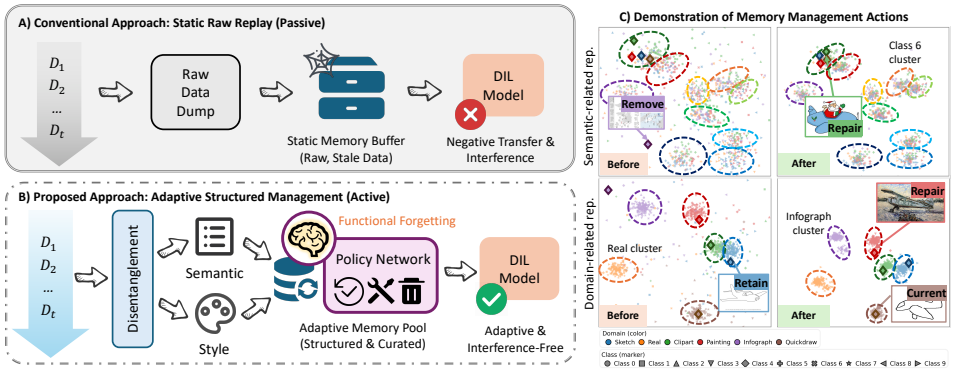


Figure 1: **Motivation and overview of adaptive memory management for domain-incremental learning.** (A) Static raw replay passively stores past samples, which may become stale and cause negative transfer. (B) Inspired by neuroscience views of forgetting as an active and functional process, RAM learns to *retain*, *repair*, or *remove* samples using semantic- and domain-related memory states. (C) The learned actions remove misleading samples, repair domain-shifted ones, and retain useful memories for adaptive replay.

1 INTRODUCTION

Continual learning (CL) aims to enable models to learn from a sequence of tasks or domains without losing previously acquired knowledge. A central challenge in CL is none other than *catastrophic for-*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

getting (McCloskey & Cohen, 1989; McClelland et al., 1995), where adapting to new data degrades performance on earlier tasks (van de Ven & Tolias, 2019; Wang et al., 2024a). Consequently, many CL methods are designed to preserve past knowledge, either by constraining parameter updates (Li & Hoiem, 2017; Kirkpatrick et al., 2017; Aljundi et al., 2018) or by replaying stored samples from previous tasks or domains (Rebuffi et al., 2017; Buzzega et al., 2020; Wang et al., 2025).

However, preventing knowledge loss does not require every stored experience to remain unchanged. Studies in neuroscience suggest that biological forgetting (here referring to functional memory loss rather than catastrophic forgetting) can be an active and functional process that interacts with memory persistence to support adaptive behavior (Gravitz, 2019; Migues et al., 2016; Akers et al., 2014). In particular, memory transience has been argued to improve flexibility by reducing the influence of outdated information and preventing overfitting to specific past experiences, thereby promoting generalization (Richards & Frankland, 2017). This perspective does not suggest that artificial continual learners should forget indiscriminately. Instead, it highlights a key principle: useful memory must be regulated. For continual learning, this raises an important question: how should a learner decide what to retain, refresh, or discard to preserve historical knowledge while adapting to new data?

We study this question in Domain Incremental Learning (DIL), where a model must adapt to a sequence of shifting domains while maintaining performance on previously seen domains, without access to the full historical data (van de Ven & Tolias, 2019; Wang et al., 2024a). Such settings naturally arise in real-world applications, including autonomous driving under changing weather conditions (Mirza et al., 2022; Nisar et al., 2023) and medical imaging across different hospitals (Bándi et al., 2019; Yang et al., 2022; Chen et al., 2025). In DIL, replay provides a natural mechanism for preserving past experience. Existing methods improve replay through selective retrieval and sample valuation (Aljundi et al., 2019; Mai et al., 2020). Nevertheless, selecting an example for storage does not by itself determine how that entry should be maintained throughout subsequent training. As the model and domains evolve, an entry's continued value for replay may change. Under a fixed memory budget, memory management must therefore consider how stored entries support both historical retention and new-domain adaptation. These considerations motivate treating replay memory as a resource to be actively maintained, rather than a fixed archive of past data.

A further challenge is to obtain informative signals for this ongoing maintenance. Raw samples alone do not explicitly describe how their semantic- and domain-related representations change during training. In DIL, a stored sample combines label-relevant semantic content with domain-specific factors, whose representations may evolve differently as the model adapts. For example, in driving scenes, object categories are task-relevant semantic cues, whereas weather, illumination, and camera conditions reflect domain-specific variations (Mirza et al., 2022; Yang et al., 2023). Tracking semantic- and domain-related representation drift separately provides structured signals for assessing these changes. We hypothesize that some entries with pronounced drift in both components are less stable replay representatives with limited additional value, particularly when their historical knowledge is already supported by other memories. This motivates a stability-aware prior that favors relatively stable entries while allowing selective updates and removal. Effective memory management thus concerns not only *which* samples to store, but also *how* stored entries should be maintained as learning proceeds.

To address these challenges, we propose **RAM**, a **R**einforcement learning–based **A**daptive **M**emory management framework for Domain Incremental Learning, as illustrated in Fig. 1. The key idea of **RAM** is to manage stored samples together with semantic- and domain-related representation states that track their changes during training. We provide a *Mutual-Information Sequential Adaptation Bound* to motivate learning label-predictive semantic representations with reduced dependence on domain identity. Building on this representation design, we construct complementary drift signals for memory assessment. A lightweight policy then selects sample-wise *retain*, *repair*, and *remove* actions, preserving entries, refreshing their associated representation states, or removing them from the buffer. The policy is trained using drift-guided priors together with validation feedback across all seen domains. In this way, **RAM** learns to maintain replay memory under a fixed budget, with the goal of balancing historical knowledge retention and adaptation to newly observed domains.

In summary, this work makes the following contributions:

- **Representation-drift-aware memory editing.** We propose **RAM**, a structured replay framework that manages memory entries through *retain*, *repair*, and *remove* operations for DIL.

- **Theory-motivated representation design.** We provide a *Mutual-Information Sequential Adaptation Bound* to motivate the representation design underlying semantic- and domain-related memory states.
- **Feedback-guided editing policy.** We formulate memory maintenance as an online decision process and learn a lightweight policy using drift-guided priors and performance feedback across seen domains.
- **Empirical validation.** We evaluate **RAM** on Rotated-MNIST, DomainNet, Camelyon17, and iWildCam across backbones of different scales, showing consistent gains over the compared baselines. On DomainNet with ResNet-18, **RAM** improves average accuracy by **3.3** percentage points and reduces forgetting by **6.2** percentage points relative to X-DER.

2 Related Works

2.1 Continual Learning

Continual Learning studies how to acquire new knowledge over time while retaining previously learned knowledge. Classical CL methods are often grouped into exemplar-free and replay-based approaches. **Exemplar-free** methods avoid storing past data and instead regularize parameter updates using weight-importance estimates, such as the Fisher Information Matrix (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018), or preserve prior knowledge through distillation from earlier model snapshots (Li & Hoiem, 2017; Rebuffi et al., 2017). Recent exemplar-free methods PINA (Wang et al., 2024b) fixes a unified classifier learned on the base domain as a shared concept space and aligns subsequent-domain features to it through domain-specific modules. **Replay-** or **memory-based** methods maintain a buffer of past samples or features for rehearsal. ER (Riemer et al., 2019) interleaves buffered samples with new data during training. DER++ (Buzzega et al., 2020) extends ER with feature- and output-space distillation to better preserve representations over time, while X-DER (Boschini et al., 2023) further adds a future-preparation loss and regularly updated memory to improve robustness under distribution shifts.

2.2 Memory Management & Prompt Update

In continual learning, another line of work studies how learned knowledge is stored, retrieved, and updated. iCaRL (Rebuffi et al., 2017) selects exemplars that approximate class means and performs classification using a nearest-mean-of-exemplars rule. AdaER (Li et al., 2024) adaptively replays memories most conflicting with the current input and updates the buffer via entropy-balanced reservoir sampling. Recent methods instead encode prior knowledge into compact parameterized memories. L2P (Wang et al., 2022) and its replay variant L2P-R introduce a pool of learnable prompt tokens appended to the input, allowing the model to retrieve task-relevant prompts and reuse prior knowledge across experiences. CP-Prompt (Feng et al., 2024) decomposes prompt memory into common prompts for domain-shared knowledge and personalized prompts for domain-specific knowledge. Beyond prompts, DUCT (Zhou et al., 2025) consolidates domain-adapted representations and classifiers into a unified feature-classifier space. However, these methods either rely on update mechanisms that are not directly human-interpretable or depend on hand-crafted heuristics.

3 Methodology

In this paper, we consider *Domain Incremental Learning*, where a model is trained on a sequence of domains $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$ that share the same label space $\mathcal{Y}$ but have different input distributions $P_t(x)$. At step t, the learner only accesses the current domain dataset $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$ and cannot revisit the full data from previous domains. We use $d_i \in \{1, \ldots, T\}$ to denote the domain index of sample x_i, with $d_i = t$ for samples in the current domain $\mathcal{D}_t$. The summary of notation is provided in Sec. K.

The learner h_θ is parameterized by an encoder with two branches that extract a disentangled representation for each input:

$$z_\tau = f_\tau(x), \qquad z_\delta = f_\delta(x),$$

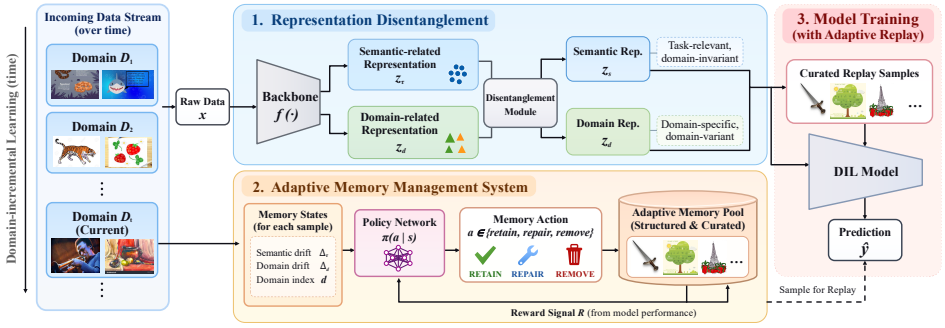


Figure 2: **Pipeline of the proposed adaptive memory management framework for domain-incremental learning.** As domains $\mathcal{D}_1, \mathcal{D}_2, \ldots$ arrive over time, each incoming raw sample is processed by a disentanglement module to obtain a semantic-related z_τ and a domain-related representation z_δ. The memory pool stores raw samples together with these representations, which serve as state signals for memory management. At each step, a policy network observes the current memory state and training signals, and selects a sample-wise action from {*RETAIN*, *REPAIR*, *REMOVE*}. The curated raw samples are then used for replay, and the model's performance provides reward signals for policy learning.

where z_τ captures *semantic-related* (label-relevant) factors and z_δ captures *domain-related* (style/domain-specific) factors. f_τ and f_δ correspond to the two encoder branches that extract complementary factors of variation.

To mitigate forgetting, the learner maintains a small episodic memory $\mathcal{M} = \{M_i\}_{i=1}^{t-1}$, where each $M_i \subseteq \mathcal{D}_i$ stores a small subset of examples from domain i, and the memory is updated online as new domains arrive. After training on domain t, we evaluate h_θ on the test sets of all observed domains and report the average accuracy $\bar{e}^{(t)} = \frac{1}{t}\sum_{k=1}^{t} e_k^{(t)}$, the final-domain accuracy $e_t^{(t)}$, and the forgetting metric $f^{(t)} = \frac{1}{t-1}\sum_{k=1}^{t-1}\left(\max_{1\le s\le t-1} e_k^{(s)} - e_k^{(t)}\right)$, where $e_k^{(t)}$ denotes the test accuracy on domain k after training up to domain t. The overall pipeline of **RAM** is illustrated in Fig. 2.

3.1 Mutual-Information Sequential Adaptation Bound

DIL can be viewed as a sequence of domain-adaptation steps: all domains share the same label space and prediction task, but their input marginals drift over time. To make this connection explicit, we combine a standard adaptation bound for consecutive domains with a mutual-information upper bound on domain discrepancy in a learned representation. This yields a sequential error bound whose dominant term depends on how much the *semantic-related representation* reveals the domain identity.

Lemma 3.1 (Mutual-Information Sequential Adaptation Bound)**.** *Consider a sequence of domains $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$ over a common label space $\mathcal{Y}$. Let P_t denote the* input *marginal of domain $\mathcal{D}_t$, and for any hypothesis h let $e_t(h)$ be its (population) error on $\mathcal{D}_t$. For two consecutive domains, the standard adaptation bound (Ben-David et al., 2010; Shi & Wang, 2023) gives, for all $t \ge 2$,*

$$e_t(h) \;\le\; e_{t-1}(h) \;+\; \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(P_t, P_{t-1}) \;+\; \lambda_{t-1,t}, \tag{1}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}$ is the $\mathcal{H}\Delta\mathcal{H}$-divergence and $\lambda_{t-1,t} = \min_{h\in\mathcal{H}}\left(e_{t-1}(h) + e_t(h)\right)$ is the joint optimal risk (see (Kifer et al., 2004; Ben-David et al., 2006; 2010)). Recursively applying equation 1 yields

$$e_t(h) \;\le\; e_1(h) \;+\; \frac{1}{2}\sum_{k=2}^{t} d_{\mathcal{H}\Delta\mathcal{H}}(P_k, P_{k-1}) \;+\; \lambda^*, \tag{2}$$

where λ^* aggregates the joint optimal risks along the path.

Now consider predictors that operate the semantic-related representation $Z_\tau = \phi_\tau(X)$, i.e., $h_\tau(x) = g(\phi_\tau(x))$ for some $g \in \mathcal{G}$, and let the induced hypothesis class be $\mathcal{H}_\tau \triangleq \{g \circ \phi_\tau : g \in \mathcal{G}\}$. For each step $k \ge 2$, let $D_k \in \{k-1, k\}$ denote the domain indicator, and let $I(Z_\tau; D_k)$ be the

mutual information under this mixture. Then (via the JS-divergence connection; proof in Sec. D.1),

$$e_t(h_\tau) \;\lesssim\; e_1(h_\tau) \;+\; \sum_{k=2}^{t} \sqrt{\ln 2 \cdot I(Z_\tau; D_k)} \;+\; \lambda_\tau^*. \tag{3}$$

Consequently, reducing $I(Z_\tau; D_k)$ tightens the bound on $e_t(h_\tau)$ and helps control error accumulation (e.g., catastrophic forgetting) across domains. Moreover, if the full representation decomposes as $Z = (Z_\tau, Z_\delta)$, where Z_τ is semantic-related features and Z_δ is domain-specific features, then by the chain rule,

$$I(Z; D_k) = I(Z_\tau; D_k) + I(Z_\delta; D_k \mid Z_\tau) \;\geq\; I(Z_\tau; D_k),$$

so $I(Z_\tau; D_k) \leq I(Z; D_k)$. This motivates enforcing domain-invariance specifically in the semantic component.

3.2 Disentangled Representation Learning

Motivated by Eq. (3), we learn semantic representations that are label-predictive while reducing dependence on domain identity.

At stage t, a two-branch encoder maps each input from $\mathcal{D}_t$ into a semantic-related representation $z_\tau \in \mathbb{R}^p$ and a domain-related representation $z_\delta \in \mathbb{R}^q$. The semantic representation z_τ is optimized for label prediction using a softmax classifier c_θ:

$$\mathcal{L}_{\text{task}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}\Big[\mathrm{CE}\big(c_\theta(z_\tau),\, y\big)\Big], \tag{4}$$

which encourages z_τ to preserve label-relevant information.

To capture domain-specific variation, we train a domain classifier g_ϕ on z_δ:

$$\mathcal{L}_{\text{dom}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}\Big[\mathrm{CE}\big(g_\phi(z_\delta),\, d\big)\Big], \tag{5}$$

where $d \in \{1, \ldots, t\}$ is the domain label.

To encourage domain invariance in z_τ, we apply a gradient reversal layer (GRL) (Ganin & Lempitsky, 2015),

$$\mathcal{R}_\eta(z) = z, \qquad \frac{\partial \mathcal{R}_\eta(z)}{\partial z} = -\eta I,$$

and define the adversarial loss

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}\Big[\mathrm{CE}\big(g_\phi(\mathcal{R}_\eta(z_\tau)),\, d\big)\Big]. \tag{6}$$

We further reduce correlation between the two branches with an orthogonality regularizer:

$$\mathcal{L}_{\text{ortho}} = \left\| \frac{Z_\tau^\top Z_\delta}{\|Z_\tau\|_F \, \|Z_\delta\|_F} \right\|_F^2, \tag{7}$$

where Z_τ and Z_δ are the batch-wise stacked representations.

The overall disentanglement loss is

$$\mathcal{L}_{\text{dis}} = \mathcal{L}_{\text{dom}} + \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{ortho}}, \tag{8}$$

which promotes label-relevant semantic features and domain-specific features with reduced overlap.

3.3 A Drift-Aware View of Episodic Memory

Memory representation. The learner maintains a limited episodic memory $\mathcal{M}$ of samples from previous domains. After training on $\mathcal{D}_t$, representative examples are added as

$$m = (x, y, z_\tau^{\text{old}}, z_\delta^{\text{old}}),$$

where $z_\tau^{\text{old}} = f_\tau(x)$ and $z_\delta^{\text{old}} = f_\delta(x)$ are the semantic- and domain-related representations at insertion time.

Drift characterization. As the encoder evolves, stored representations may become misaligned with the current model. For each memory entry, we recompute

$$z_\tau^{\text{new}} = f_\tau(x), \qquad z_\delta^{\text{new}} = f_\delta(x),$$

and measure normalized semantic and domain drift:

$$\Delta_\tau = \frac{\left\| z_\tau^{\text{new}} - z_\tau^{\text{old}} \right\|_2}{\|z_\tau^{\text{old}}\|_2 + \varepsilon}, \qquad \Delta_\delta = \frac{\left\| z_\delta^{\text{new}} - z_\delta^{\text{old}} \right\|_2}{\|z_\delta^{\text{old}}\|_2 + \varepsilon}, \tag{9}$$

where ε ensures numerical stability.

Semantic–domain drift plane. For interpretability, we center the drift scores by their batch medians:

$$u = \Delta_\tau - \text{median}(\Delta_\tau), \qquad v = \Delta_\delta - \text{median}(\Delta_\delta), \tag{10}$$

where positive values indicate above-median drift and negative values indicate relative stability in the corresponding component. The resulting semantic–domain drift plane is visualized in Fig. 4.

3.4 Adaptive Memory Management

Markov Decision Process. We cast memory maintenance as an online decision process. At the end of each training epoch, the policy evaluates a batch of memory items and assigns an action to each of them.

State. For a memory entry $m = (x, y, z_\tau^{\text{old}}, z_\delta^{\text{old}})$, we define its state as

$$s(m) = \big[\Delta_\tau(m),\, \Delta_\delta(m),\, d\big], \tag{11}$$

where $\Delta_\tau(m)$ and $\Delta_\delta(m)$ are computed by Eq. (9), and $d \in \{1, \ldots, t\}$ denotes the domain index of the sample.

Action. Inspired by the balance between memory persistence and transience in biological memory (Richards & Frankland, 2017), we allow each memory item to be actively preserved, refreshed, or forgotten through an action $a \in \mathcal{A}$:

$$\mathcal{A} = \{\textsc{Retain}, \textsc{Repair}, \textsc{Remove}\}. \tag{12}$$

Remove deletes the item from $\mathcal{M}$. Retain keeps the item unchanged. Repair refreshes the stored representations by moving them toward the current encoder outputs. Concretely, for each component $z \in \{z_\tau, z_\delta\}$, we update

$$z^{\text{old}} \leftarrow (1 - \beta)\, z^{\text{old}} + \beta\, z^{\text{new}}, \tag{13}$$

where z^{new} is recomputed by the current encoder, and $\beta \in (0, 1]$ controls the update rate. Given an action, the corresponding memory update is deterministic.

Reward. We compute one scalar reward after executing a batch of decisions $\{(s_i, a_i)\}_{i=1}^N$ within an epoch. The reward consists of three components.

(i) Validation improvement. We measure the performance change on a validation set covering all seen domains:

$$r_{\text{val}} = \text{Acc}^{\text{curr}} - \text{Acc}^{\text{prev}}. \tag{14}$$

Intuitively, a memory update is beneficial if it improves validation accuracy after replay. We provide further theoretical justification by relating r_{val} to changes in output-space distribution and the MI-bound term in Eq. (3) (see Lem. D.17 and Lem. D.18).

(ii) Action diversity. To avoid degenerate behavior (e.g., always selecting the same action), we encourage exploration by maximizing the entropy of the average action distribution:

$$\bar{\pi}(a) = \frac{1}{N} \sum_{i=1}^{N} \pi_\omega(a \mid s_i), \quad H(\bar{\pi}) = -\sum_{a \in \mathcal{A}} \bar{\pi}(a) \log \bar{\pi}(a), \tag{15}$$

and define

$$r_{\text{div}} = \frac{H(\bar{\pi})}{\log |\mathcal{A}|}. \tag{16}$$

(iii) Drift-guided alignment. The drift plane in Eq. (10) provides a simple, interpretable prior about which operation is likely appropriate. We first compute centered drifts (u_i, v_i) for each memory item, and define a target action $a_i^\star$ by

$$a^\star(u,v) = \begin{cases} \textsc{Remove}, & u > 0,\ v > 0, \\ \textsc{Repair}, & (u > 0,\ v < 0) \text{ or } (u < 0,\ v > 0), \\ \textsc{Retain}, & u < 0,\ v < 0. \end{cases} \tag{17}$$

This prior is not enforced as a hard rule; instead, it provides a lightweight guidance signal. We reward agreement between the sampled action and the target:

$$\mathrm{align}_i = \mathbb{1}\{a_i = a_i^\star\}, \qquad r_{\mathrm{align}} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{align}_i. \tag{18}$$

We further show that these operations tend to reduce feature-distribution discrepancy under mild assumptions (see Sec. D.2, Sec. D.3 and Lem. D.16).

Finally, the overall reward is defined as

$$R = \lambda_1\, r_{\mathrm{val}} + \lambda_2\, r_{\mathrm{div}} + \lambda_3\, r_{\mathrm{align}}, \tag{19}$$

where $\lambda_1, \lambda_2, \lambda_3 \geq 0$ control the trade-off among validation improvement, exploration, and drift-guided alignment. We parameterize the action policy $\pi_\omega(a \mid s)$ with a lightweight policy network and optimize it using REINFORCE (Williams, 1992). Details of the policy-gradient update, together with an empirical stability analysis, are provided in Sec. G.

Replay training with the curated memory. After applying RETAIN/REPAIR/REMOVE, we obtain a curated memory set $\mathcal{M}$ for replay. We sample a mini-batch $\mathcal{B}_m \subset \mathcal{M}$ and compute a replay loss using the semantic branch:

$$\mathcal{L}_{\mathrm{replay}} = \mathbb{E}_{(x,y,\cdot,\cdot)\sim\mathcal{B}_m}\Big[\mathrm{CE}\big(c_\theta(f_\tau(x)),\, y\big)\Big]. \tag{20}$$

This replay objective complements current-domain training by reinforcing label-relevant semantics from past domains using the curated memory.

Overall objective and inference. Combining current-domain learning, disentanglement regularization, and replay, the overall training objective becomes

$$\mathcal{L} = \mathcal{L}_{\mathrm{task}} + \alpha\, \mathcal{L}_{\mathrm{dis}} + \gamma\, \mathcal{L}_{\mathrm{replay}}, \tag{21}$$

where α balances label prediction and representation disentanglement, and γ controls the strength of replay. The full algorithm is provided in Sec. J. At inference time, predictions rely only on the semantic-related representation: given an input x, we output $\hat{y} = \arg\max c_\theta(f_\tau(x))$.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

We evaluate RAM on four DIL benchmarks with increasing domain shift and realism: **Rotated-MNIST** (Liu et al., 2025), **DomainNet** (Peng et al., 2019), **Camelyon17** (Bándi et al., 2019), and **iWildCam** (Beery et al., 2020). They cover controlled synthetic shifts, multi-domain recognition, and real-world medical and ecological domain shifts. We use a three-layer CNN for Rotated-MNIST, and evaluate the other three main benchmarks with both ResNet-18 and ViT-B/16 backbones. Recent prompt-based baselines are evaluated only with ViT-B/16, since they rely on pretrained transformer prompt mechanisms and are not directly applicable to ResNet-18. Rotated-MNIST results are reported in Sec. F. More experimental details and hyperparameter sensitivity analyses in Sec. E and Sec. I.

4.2 COMPARISON WITH BASELINE METHODS

RAM consistently achieves the best performance across datasets and backbones. As shown in Tab. 1, RAM achieves the best Final/Avg accuracy and the lowest Forgetting across DomainNet, Camelyon17, and iWildCam under both ResNet-18 and ViT-B/16 backbones. It consistently

Table 1: **Overall performance on DomainNet, Camelyon17, and iWildCam under different backbone settings.** We report Final Accuracy (Final%), Average Accuracy (Avg%), and Forgetting (Forget%; lower is better). All numbers are reported as mean$_{\pm}$ over **5** runs.

Method	DomainNet			Camelyon17			iWildCam		
	Final↑	Avg↑	Forget↓	Final↑	Avg↑	Forget↓	Final↑	Avg↑	Forget↓
Backbone: ResNet-18									
Memory-free / Non-replay									
LwF (Li & Hoiem, 2017)	28.9$_{\pm 0.74}$	32.4$_{\pm 0.83}$	41.8$_{\pm 1.06}$	75.1$_{\pm 0.89}$	72.5$_{\pm 0.97}$	19.7$_{\pm 1.18}$	49.2$_{\pm 0.92}$	50.5$_{\pm 0.88}$	18.4$_{\pm 1.10}$
EWC (Kirkpatrick et al., 2017)	27.1$_{\pm 0.86}$	30.7$_{\pm 0.93}$	44.3$_{\pm 1.14}$	73.4$_{\pm 1.02}$	71.8$_{\pm 1.07}$	21.4$_{\pm 1.29}$	47.5$_{\pm 1.04}$	48.9$_{\pm 0.97}$	20.3$_{\pm 1.21}$
MAS (Aljundi et al., 2018)	27.9$_{\pm 0.78}$	31.2$_{\pm 0.81}$	43.5$_{\pm 1.03}$	74.2$_{\pm 0.95}$	72.1$_{\pm 0.92}$	20.2$_{\pm 1.17}$	48.2$_{\pm 0.98}$	49.6$_{\pm 0.91}$	19.5$_{\pm 1.16}$
Replay-centric									
ER (Chaudhry et al., 2019)	41.2$_{\pm 0.64}$	42.6$_{\pm 0.59}$	29.5$_{\pm 0.86}$	81.0$_{\pm 0.74}$	79.3$_{\pm 0.71}$	13.1$_{\pm 0.83}$	53.9$_{\pm 0.80}$	54.8$_{\pm 0.76}$	16.2$_{\pm 0.94}$
DER++ (Buzzega et al., 2020)	43.0$_{\pm 0.57}$	44.9$_{\pm 0.62}$	24.8$_{\pm 0.73}$	83.1$_{\pm 0.63}$	81.2$_{\pm 0.58}$	11.6$_{\pm 0.69}$	55.0$_{\pm 0.72}$	55.9$_{\pm 0.69}$	14.8$_{\pm 0.85}$
SER (Zhuo et al., 2023)	44.9$_{\pm 0.49}$	46.7$_{\pm 0.52}$	22.1$_{\pm 0.66}$	84.2$_{\pm 0.57}$	82.0$_{\pm 0.53}$	10.9$_{\pm 0.62}$	55.8$_{\pm 0.66}$	56.6$_{\pm 0.63}$	14.1$_{\pm 0.79}$
X-DER (Boschini et al., 2023)	46.4$_{\pm 0.41}$	47.9$_{\pm 0.48}$	19.6$_{\pm 0.58}$	85.4$_{\pm 0.47}$	83.8$_{\pm 0.49}$	9.7$_{\pm 0.57}$	56.2$_{\pm 0.59}$	57.1$_{\pm 0.56}$	12.6$_{\pm 0.71}$
Memory / Prompt Updating									
iCaRL (Rebuffi et al., 2017)	41.8$_{\pm 0.79}$	43.6$_{\pm 0.84}$	26.8$_{\pm 1.03}$	82.7$_{\pm 0.76}$	80.9$_{\pm 0.73}$	12.2$_{\pm 0.87}$	51.6$_{\pm 0.86}$	52.8$_{\pm 0.82}$	15.2$_{\pm 0.90}$
AdaER (Li et al., 2024)	43.5$_{\pm 0.73}$	45.3$_{\pm 0.76}$	23.5$_{\pm 0.94}$	82.4$_{\pm 0.73}$	80.6$_{\pm 0.71}$	12.4$_{\pm 0.93}$	52.2$_{\pm 0.81}$	53.6$_{\pm 0.77}$	14.7$_{\pm 0.88}$
Structured Memory Policy									
RAM	**49.1**$_{\pm 0.39}$	**51.2**$_{\pm 0.44}$	**13.4**$_{\pm 0.48}$	**88.3**$_{\pm 0.43}$	**86.5**$_{\pm 0.46}$	**6.9**$_{\pm 0.49}$	**59.5**$_{\pm 0.52}$	**60.4**$_{\pm 0.49}$	**9.8**$_{\pm 0.61}$
Backbone: ViT-B/16									
Memory-free / Non-replay									
LwF (Li & Hoiem, 2017)	54.6$_{\pm 0.66}$	55.8$_{\pm 0.71}$	19.5$_{\pm 0.83}$	80.4$_{\pm 0.72}$	78.8$_{\pm 0.76}$	15.8$_{\pm 0.91}$	50.9$_{\pm 0.78}$	51.7$_{\pm 0.82}$	18.3$_{\pm 0.96}$
EWC (Kirkpatrick et al., 2017)	52.7$_{\pm 0.74}$	53.9$_{\pm 0.79}$	21.2$_{\pm 0.91}$	78.9$_{\pm 0.81}$	77.4$_{\pm 0.83}$	17.2$_{\pm 1.00}$	49.4$_{\pm 0.86}$	50.1$_{\pm 0.88}$	19.8$_{\pm 1.03}$
MAS (Aljundi et al., 2018)	53.5$_{\pm 0.69}$	54.6$_{\pm 0.73}$	20.4$_{\pm 0.87}$	79.6$_{\pm 0.77}$	78.1$_{\pm 0.80}$	16.5$_{\pm 0.95}$	50.0$_{\pm 0.81}$	50.8$_{\pm 0.84}$	19.1$_{\pm 0.99}$
PINA (Wang et al., 2024b)	64.9$_{\pm 0.44}$	65.7$_{\pm 0.48}$	9.7$_{\pm 0.57}$	87.1$_{\pm 0.46}$	85.8$_{\pm 0.49}$	8.6$_{\pm 0.55}$	60.9$_{\pm 0.52}$	61.6$_{\pm 0.56}$	10.4$_{\pm 0.63}$
Replay-centric									
ER (Chaudhry et al., 2019)	59.2$_{\pm 0.58}$	60.5$_{\pm 0.61}$	14.6$_{\pm 0.72}$	83.7$_{\pm 0.61}$	82.2$_{\pm 0.64}$	11.4$_{\pm 0.74}$	54.6$_{\pm 0.70}$	55.4$_{\pm 0.73}$	15.8$_{\pm 0.85}$
DER++ (Buzzega et al., 2020)	61.1$_{\pm 0.52}$	62.4$_{\pm 0.56}$	13.1$_{\pm 0.65}$	85.1$_{\pm 0.55}$	83.7$_{\pm 0.58}$	10.3$_{\pm 0.67}$	56.0$_{\pm 0.65}$	56.8$_{\pm 0.67}$	14.4$_{\pm 0.78}$
SER (Zhuo et al., 2023)	61.9$_{\pm 0.49}$	63.2$_{\pm 0.53}$	12.4$_{\pm 0.61}$	85.8$_{\pm 0.51}$	84.3$_{\pm 0.54}$	9.8$_{\pm 0.63}$	56.7$_{\pm 0.61}$	57.5$_{\pm 0.64}$	13.8$_{\pm 0.73}$
X-DER (Boschini et al., 2023)	63.0$_{\pm 0.46}$	65.1$_{\pm 0.50}$	10.4$_{\pm 0.58}$	86.7$_{\pm 0.47}$	85.1$_{\pm 0.50}$	8.9$_{\pm 0.57}$	57.9$_{\pm 0.58}$	58.7$_{\pm 0.61}$	13.0$_{\pm 0.69}$
Memory / Prompt Updating									
iCaRL (Rebuffi et al., 2017)	60.4$_{\pm 0.67}$	61.7$_{\pm 0.70}$	13.9$_{\pm 0.82}$	84.6$_{\pm 0.66}$	83.1$_{\pm 0.69}$	10.9$_{\pm 0.78}$	55.1$_{\pm 0.72}$	55.9$_{\pm 0.75}$	15.0$_{\pm 0.88}$
L2P-R (Wang et al., 2022)	63.7$_{\pm 0.55}$	64.5$_{\pm 0.59}$	11.5$_{\pm 0.67}$	86.2$_{\pm 0.52}$	84.9$_{\pm 0.55}$	9.5$_{\pm 0.64}$	59.8$_{\pm 0.57}$	60.5$_{\pm 0.60}$	11.5$_{\pm 0.70}$
AdaER (Li et al., 2024)	61.7$_{\pm 0.64}$	62.8$_{\pm 0.68}$	12.7$_{\pm 0.79}$	84.9$_{\pm 0.64}$	83.4$_{\pm 0.67}$	10.7$_{\pm 0.76}$	56.2$_{\pm 0.69}$	56.9$_{\pm 0.72}$	14.1$_{\pm 0.83}$
CP-Prompt (Feng et al., 2024)	63.1$_{\pm 0.47}$	63.9$_{\pm 0.51}$	11.3$_{\pm 0.62}$	85.9$_{\pm 0.50}$	84.6$_{\pm 0.53}$	9.6$_{\pm 0.61}$	59.0$_{\pm 0.55}$	59.8$_{\pm 0.59}$	12.1$_{\pm 0.68}$
DUCT (Zhou et al., 2025)	67.0$_{\pm 0.41}$	67.2$_{\pm 0.45}$	8.8$_{\pm 0.53}$	88.0$_{\pm 0.43}$	86.8$_{\pm 0.47}$	7.9$_{\pm 0.51}$	62.1$_{\pm 0.49}$	62.8$_{\pm 0.53}$	9.2$_{\pm 0.58}$
Structured Memory Policy									
RAM	**68.1**$_{\pm 0.39}$	**68.9**$_{\pm 0.43}$	**7.5**$_{\pm 0.48}$	**89.2**$_{\pm 0.40}$	**88.1**$_{\pm 0.44}$	**6.7**$_{\pm 0.47}$	**63.5**$_{\pm 0.46}$	**64.2**$_{\pm 0.50}$	**8.0**$_{\pm 0.54}$

Table 2: **Ablation study on DomainNet, Camelyon17, and iWildCam.** We report Final Accuracy (Final↑), Average Accuracy (Avg↑), and Forgetting (Forget↓), all in percentage. All results are reported as mean$_{\pm}$ over **5** runs.

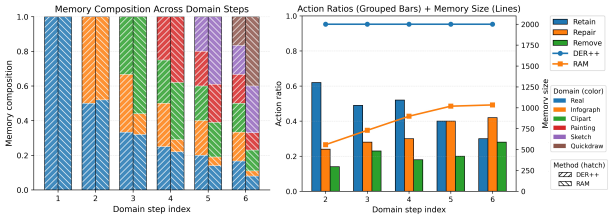
Method	DomainNet			Camelyon17			iWildCam		
	Final↑	Avg↑	Forget↓	Final↑	Avg↑	Forget↓	Final↑	Avg↑	Forget↓
Source Only	12.5$_{\pm 0.91}$	14.2$_{\pm 0.88}$	–	65.3$_{\pm 0.96}$	66.1$_{\pm 0.91}$	–	43.6$_{\pm 1.05}$	44.8$_{\pm 0.97}$	–
Sequential FT	25.4$_{\pm 0.83}$	27.1$_{\pm 0.86}$	50.2$_{\pm 1.21}$	68.1$_{\pm 0.88}$	69.7$_{\pm 0.92}$	25.3$_{\pm 1.18}$	46.8$_{\pm 0.94}$	48.1$_{\pm 0.89}$	24.6$_{\pm 1.13}$
RAM	**49.1**$_{\pm 0.39}$	**51.2**$_{\pm 0.44}$	**13.4**$_{\pm 0.48}$	**88.3**$_{\pm 0.43}$	**86.5**$_{\pm 0.46}$	**6.9**$_{\pm 0.49}$	**59.5**$_{\pm 0.52}$	**60.4**$_{\pm 0.49}$	**9.8**$_{\pm 0.61}$
w/o repair	46.3$_{\pm 0.51}$	48.0$_{\pm 0.56}$	17.8$_{\pm 0.62}$	85.4$_{\pm 0.51}$	83.1$_{\pm 0.54}$	10.8$_{\pm 0.63}$	57.0$_{\pm 0.59}$	57.8$_{\pm 0.55}$	12.1$_{\pm 0.70}$
w/o delete	44.7$_{\pm 0.58}$	46.1$_{\pm 0.61}$	19.6$_{\pm 0.69}$	84.2$_{\pm 0.57}$	82.0$_{\pm 0.59}$	12.2$_{\pm 0.72}$	55.6$_{\pm 0.66}$	56.5$_{\pm 0.62}$	13.4$_{\pm 0.78}$
w/o dynamic mgmt	41.2$_{\pm 0.64}$	42.6$_{\pm 0.59}$	29.5$_{\pm 0.86}$	81.0$_{\pm 0.74}$	79.3$_{\pm 0.71}$	13.1$_{\pm 0.83}$	53.9$_{\pm 0.80}$	54.8$_{\pm 0.76}$	16.2$_{\pm 0.94}$
w/o disentanglement	45.0$_{\pm 0.55}$	46.8$_{\pm 0.58}$	18.7$_{\pm 0.66}$	84.7$_{\pm 0.54}$	82.5$_{\pm 0.56}$	11.9$_{\pm 0.68}$	56.2$_{\pm 0.63}$	57.1$_{\pm 0.60}$	12.8$_{\pm 0.74}$

improves over the strongest baseline, with larger gains under ResNet-18 and competitive improvements over recent ViT-based methods such as PINA (Wang et al., 2024b) and DUCT (Zhou et al., 2025). These results show that structured memory management is effective across both conventional CNN and pretrained transformer settings. The sequential domain accuracy curve is shown in Sec. H.

Table 3: **Comparison of RL-based controller and hand-crafted memory policies on DomainNet.** We report Average Accuracy (Avg%), Final Accuracy (Final%), and Forgetting (Forget%; lower is better). **Bold** denotes the best and underline denotes the second best.

Method	Policy Type	Avg↑	Final↑	Forget↓
Fixed-ratio	Heur. (time)	44.8	43.0	24.6
Loss-based	Heur. (loss)	46.0	44.2	22.0
Grad.-conflict	Heur. (grad.)	47.1	45.3	20.1
Quadrant-rule	Heur. (feat.)	48.6	46.8	17.9
Ours	**Learned (RL)**	**51.2**	**49.1**	**13.4**

Figure 3: **Memory dynamics under domain shifts on DomainNet. Left:** Memory composition across domain steps comparing DER++ and **RAM**. **Right: RAM** operation statistics and replay-buffer size, contrasted with DER++'s fixed saturated buffer.



Takeaway #1: Replay is useful, but static replay is insufficient.

Replay-centric methods substantially outperform memory-free or non-replay baselines, confirming the importance of revisiting past data. However, their Forgetting remains high under larger domain shifts, suggesting that a fixed replay buffer can preserve stale or less useful samples.

4.3 Analysis of Learned Memory Control

We analyze whether learned control is necessary for adaptive replay management. As shown in Tab. 3, the RL controller in RAM outperforms hand-crafted policies on DomainNet, including fixed-ratio, loss-based, gradient-conflict, and semantic-domain quadrant rules. Compared with the strongest heuristic, quadrant-rule, RAM improves Avg accuracy by $+2.6$, Final accuracy by $+2.3$, and reduces Forgetting by 4.5, showing that memory management cannot be fully captured by a fixed rule. Fig. 3 further shows that RAM actively reshapes the replay buffer over time. Unlike DER++, which keeps a nearly saturated buffer, RAM allocates memory non-uniformly across domains and uses a mixture of *RETAIN*, *REPAIR*, and *REMOVE* operations. This indicates the learned policy improves replay quality by refreshing or discarding stale items, rather than simply preserving more samples.

Takeaway #2: Learned memory control improves replay quality.

Hand-crafted rules use useful but partial signals. RAM learns from feedback to better balance retaining useful samples, repairing misaligned ones, and removing harmful memory items.

4.4 Ablation Study

As shown in Tab. 2, RAM performs best across all three benchmarks. Removing any component consistently lowers accuracy and increases forgetting. The drops from *w/o repair* and *w/o delete* show the importance of refreshing drifting representations and removing harmful samples. Replacing the learned controller with ER causes the largest degradation, confirming that naive replay is insufficient under domain shift. Removing disentanglement further hurts performance, highlighting the value of semantic- and domain-related drift signals.

5 Conclusion

DIL requires replay memory to remain useful as models and domains evolve. We propose **RAM**, a representation-drift-aware framework that maintains memory through *retain*, *repair*, and *remove* operations. A *Mutual-Information Sequential Adaptation Bound* motivates the semantic- and domain-related state design, while a lightweight policy combines drift-guided priors with feedback across seen domains to adapt memory decisions over time. Experiments on multiple DIL benchmarks show improved accuracy and reduced forgetting over the evaluated baselines. Overall, **RAM** shows that preserving historical knowledge does not require keeping every memory entry unchanged, but rather maintaining replay memory to support both retention and adaptation.

AI USE STATEMENT

We used generative AI tools for language polishing and feedback on theoretical analysis, methodological explanations, and experimental design. The original research ideas and core method were conceived by the authors. The authors reviewed all AI-assisted content and take full responsibility for the final manuscript, including its claims and results.

REFERENCES

Katherine G Akers, Alonso Martinez-Canabal, Leonardo Restivo, Adelaide P Yiu, Antonietta De Cristofaro, Hwa-Lin Hsiang, Anne L Wheeler, Axel Guskjolen, Yosuke Niibori, Hirotaka Shoji, et al. Hippocampal neurogenesis regulates forgetting during adulthood and infancy. *Science*, 344(6184):598–602, 2014.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *CoRR*, abs/1908.04742, 2019.

Péter Bándi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermsen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, Quanzheng Li, Farhad Ghazvinian Zanjani, Svitlana Zinger, Keisuke Fukuta, Daisuke Komura, Vlado Ovtcharov, Shenghua Cheng, Shaoqun Zeng, Jeppe Thagaard, Anders B. Dahl, Huangjing Lin, Hao Chen, Ludwig Jacobsson, Martin Hedlund, Melih Çetin, Eren Halici, Hunter Jackson, Richard Chen, Fabian Both, Jörg Franke, Heidi Küsters-Vandevelde, Willem Vreuls, Peter Bult, Bram van Ginneken, Jeroen van der Laak, and Geert Litjens. From detection of individual metastases to classification of lymph node status at the patient level: The CAMELYON17 challenge. *IEEE Trans. Medical Imaging*, 38(2):550–560, 2019.

Sara Beery, Elijah Cole, and Arvi Gjoka. The iwildcam 2020 competition dataset. *arXiv preprint arXiv:2004.10340*, 2020.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(5):5497–5512, 2023. doi: 10.1109/TPAMI.2022.3206549.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Ziyang Chen, Yiwen Ye, Yongsheng Pan, and Yong Xia. Gradient alignment improves test-time adaptation for medical image segmentation. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 2429–2437. AAAI Press, 2025.

DC Dowson and BV666017 Landau. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.

Haozhe Feng, Zhaorui Yang, Hesun Chen, Tianyu Pang, Chao Du, Minfeng Zhu, Wei Chen, and Shuicheng Yan. Cosda: Continual source-free domain adaptation. *arXiv preprint arXiv:2304.06627*, 2023.

Yu Feng, Zhen Tian, Yifan Zhu, Zongfu Han, Haoran Luo, Guangwei Zhang, and Meina Song. Cp-prompt: Composition-based cross-modal prompting for domain-incremental continual learning. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 2729–2738, 2024.

Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1180–1189. JMLR.org, 2015.

Lauren Gravitz. The forgotten part of memory. *Nature*, 571(7766):S12–S12, 2019.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking FID: towards a better evaluation metric for image generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 9307–9315. IEEE, 2024.

Leonid V Kantorovich. Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422, 1960.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings 2004 VLDB Conference: The 30th International Conference on Very Large Databases (VLDB)*, pp. 180. Morgan Kaufmann, 2004.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In *2017 IEEE 58th annual symposium on foundations of computer science (FOCS)*, pp. 633–638. IEEE, 2017.

Xingyu Li, Bo Tang, and Haifeng Li. Adaer: An adaptive experience replay approach for continual lifelong learning. *Neurocomputing*, 572:127204, 2024.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Hanbing Liu, Huaze Tang, Yanru Wu, Yang Li, and Xiao-Ping Zhang. Reinforced domain selection for continuous domain adaptation. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Zheda Mai, Dongsub Shim, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Adversarial shapley value experience replay for task-free continual learning. *CoRR*, abs/2009.00093, 2020.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Paola Virginia Migues, Lidong Liu, Georgina EB Archbold, Einar Ö Einarsson, Jacinda Wong, Kyra Bonasia, Seung Hyun Ko, Yu Tian Wang, and Oliver Hardt. Blocking synaptic removal of glua2-containing ampa receptors prevents the natural forgetting of long-term memories. *Journal of Neuroscience*, 36(12):3481–3494, 2016.

Muhammad Jehanzeb Mirza, Marc Masana, Horst Possegger, and Horst Bischof. An efficient domain-incremental learning approach to drive in all weather conditions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, New Orleans, LA, USA, June 19-20, 2022*, pp. 3000–3010. IEEE, 2022. doi: 10.1109/CVPRW56347.2022.00339.

Laura F Nern, Harsh Raj, Maurice André Georgi, and Yash Sharma. On transfer of adversarial robustness from pretraining to downstream tasks. *Advances in neural information processing systems*, 36:59206–59226, 2023.

Barza Nisar, Hruday Vishal Kanna Anand, and Steven L. Waslander. Gradient-based maximally interfered retrieval for domain incremental 3d object detection. In *20th Conference on Robots and Vision, CRV 2023, Montreal, QC, Canada, June 6-8, 2023*, pp. 304–311. IEEE, 2023. doi: 10.1109/CRV60082.2023.00046.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Blake A Richards and Paul W Frankland. The persistence and transience of memory. *Neuron*, 94 (6):1071–1084, 2017.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

Haizhou Shi and Hao Wang. A unified approach to domain incremental learning with memory: Theory and algorithm. *Advances in Neural Information Processing Systems*, 36:15027–15059, 2023.

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *CoRR*, abs/1904.07734, 2019.

Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Hongtao Wang, Haohao Ye, Hui Zhang, Xiaofeng Liu, Qiang Wang, and Weifeng Xu. Ser: A continual learning framework with spacing effect-based review of forgotten samples. *Available at SSRN 5690557*, 2025.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024a.

Qiang Wang, Yuhang He, Songlin Dong, Xinyuan Gao, Shaokun Wang, and Yihong Gong. Non-exemplar domain incremental learning via cross-domain concept integration. In *European Conference on Computer Vision*, pp. 144–162. Springer, 2024b.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 139–149, 2022.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992. doi: 10.1007/BF00992696.

Hongzheng Yang, Cheng Chen, Meirui Jiang, Quande Liu, Jianfeng Cao, Pheng-Ann Heng, and Qi Dou. DLTTA: dynamic learning rate for test-time adaptation on cross-domain medical images. *IEEE Trans. Medical Imaging*, 41(12):3575–3586, 2022.

Wenchen Yang, Youting Zhao, Qiang Li, Feng Zhu, and Yu Su. Multi visual feature fusion based fog visibility estimation for expressway surveillance using deep learning network. *Expert Syst. Appl.*, 234:121151, 2023.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, Lijun Zhang, and De-Chuan Zhan. Dual consolidation for pre-trained model-based domain-incremental learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 20547–20557, 2025.

Tao Zhuo, Zhiyong Cheng, Zan Gao, Hehe Fan, and Mohan Kankanhalli. Continual learning with strong experience replay. *arXiv preprint arXiv:2305.13622*, 2023.

Contents of Appendix

A Limitations **15**

B Ethics Statement **15**

C Reproducibility Statement **15**

D Proof **15**

D.1 Error bound in Domain Incremental Learning 15

D.2 Wasserstein distance and Memory Management 17

D.3 Mean-dominant W_2 changes induced by Remove/Retain/Repair 18

D.4 Wasserstein-2 viewpoint on reward function 21

E Experimental Setting **24**

E.1 Datasets, Domains, and Domain Order 24

E.2 Train/Validation/Test Splits 25

E.3 Models and Input Preprocessing 25

E.4 Training and Optimization Details 25

E.5 Replay Buffer: Budget, Updates, and Sampling 25

E.6 Reproducibility 26

E.7 Dataset Licenses 26

F Results on Rotated-MNIST **26**

G Policy Optimization and Training Dynamics **27**

H Analysis of Domain Accuracy Evolution **28**

I Hyperparameter Sensitivity **28**

J Algorithm overview **29**

K Notations **29**

A LIMITATIONS

RAM introduces modest additional training cost compared with standard replay methods, as it includes disentangled representation learning, memory-state evaluation, and policy optimization. In practice, the policy network is lightweight, and the additional cost is accompanied by consistent gains in accuracy and forgetting reduction.

Our theoretical analysis is mainly intended to motivate semantic-domain separation and drift-aware memory updates. The Wasserstein-based discussion relies on simplifying assumptions, including approximately stable covariance during memory updates. In practice, covariance may vary as the encoder evolves across domains, so this analysis should be viewed as an intuitive approximation rather than a complete characterization of deep-network behavior.

B ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work uses publicly available benchmark datasets and involves no human subjects, private data, or new data collection. The proposed method is intended for research on continual learning and does not introduce additional ethical risks beyond standard considerations for responsible machine learning evaluation and deployment.

C REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce our results throughout the paper and appendix. Complete theoretical derivations are provided in Sec. D.1, experimental settings and implementation details in Sec. E, policy optimization details in Sec. G, and the full training procedure in Sec. J. We report results over multiple random seeds, and our source code is included in the Supplementary Material.

D PROOF

D.1 ERROR BOUND IN DOMAIN INCREMENTAL LEARNING

In this subsection, we will discuss the source and target domain error bound from (Ben-David et al., 2010) for a self-contained work. As given in (Ben-David et al., 2010), the inter domain error bound between domain t and $t-1$ is given by

$$e_t(h) \;\leq\; e_{t-1}(h) + d_{\mathcal{H}\Delta\mathcal{H}}(P_t, P_{t-1}) + \lambda, \tag{22}$$

where e_t denotes error for task t, h denotes hypothesis where $h : \mathcal{X} \to 0, 1.$ λ^* denotes joint optimal risk constant where $\lambda = \min\{e_t(h^*) + e_{t-1}(h^*)\} + 4\sqrt{\frac{2d\log(2m')+\log(2/\delta)}{m'}}$ and $h^* = \text{argmin}_{h\in\mathcal{H}} e_t(h) + e_{t-1}(h)$. This is where d is the VC dimension of h, m' is the sample size and for any $\delta \in (0, 1)$. $d_{\mathcal{H}\Delta\mathcal{H}}(P_t, P_{t-1})$ denotes $\mathcal{H}\Delta\mathcal{H}$-Divergence, in which we will define next.

Lemma D.1 ($\mathcal{H}\Delta\mathcal{H}$-Divergence (Kifer et al., 2004; Ben-David et al., 2006; 2010))**.** *A common measure of divergence between distribution is the L^1 divergence*

$$d_1(P_t, P_{t-1}) = 2\sup_{B\in\mathcal{B}}|\Pr_{P_t}[B] - \Pr_{P_{t-1}}[B]|, \tag{23}$$

where $\mathcal{B}$ is a measurable subsets under P_t and P_{t-1} (Ben-David et al., 2010). However, as pointed out by (Ben-David et al., 2010), L^1 divergence cannot be accurately estimated using finite samples from arbitrary distributions (Kifer et al., 2004), and the L^1 divergence is overly strict.

Thus, (Kifer et al., 2004) defined the $\mathcal{H}-$Divergence,

$$d_{\mathcal{H}}(P_t, P_{t-1}) = 2\sup_{h\in\mathcal{H}}|\Pr_{P_t}[I(h)] - \Pr_{P_{t-1}}[I(h)]|, \tag{24}$$

where $I(h)$ is the set for which $h \in \mathcal{H}$ is the characteristic function. In contrast to L^1-Divergence, $\mathcal{H}$-divergence can be estimated with finite sample and it is never greater than L^1 divergence for any hypothesis class $\mathcal{H}$. While $\mathcal{H}$-divergence enable estimation of divergence from unlabeled data,

in order to formulate a bound, the divergence measure that represent error in relative to other hypotheses is required, thus, (Ben-David et al., 2010) defined $\mathcal{H}\Delta\mathcal{H}$-Divergence as

$$d_{\mathcal{H}\Delta\mathcal{H}}(P_t, P_{t-1}) = 2 \sup_{h,h' \in \mathcal{H}} = |\Pr_{x\sim P_{t-1}}[h(x) \neq h'(x)] - \Pr_{x\sim P_t}[h(x) \neq h'(x)]|. \tag{25}$$

Lemma D.2 (Jensen-Shannon Divergence and mutual information)**.** *We will first show the relationship between Eq.* (28) *with Jensen-Shannon (JS) Divergence and mutual information. By definition, JS divergence is defined as,*

$$\mathrm{JS}(P||Q) = \frac{1}{2}\mathrm{KL}(P||M) + \frac{1}{2}\mathrm{KL}(Q||M) \quad \textit{where } M = \frac{1}{2}(P+Q),$$

for probability distributions P and Q.

Let $P := P_t$ and $Q := P_{t-1}$ be two distributions over a measurable space $\mathcal{X}$, and let

$$M \;=\; \tfrac{1}{2}(P+Q).$$

Define a domain indicator random variable $D \in \{0,1\}$ with $\Pr(D=0) = \Pr(D=1) = \frac{1}{2}$ and a random variable $Z \sim P(\cdot \mid D=0)$, $Z \sim Q(\cdot \mid D=1)$. Then:

$$\mathrm{JS}(P\|Q) = I(Z;D), \tag{26}$$

Proof. **(JS equals mutual information).** By definition of mutual information,

$$I(Z;D) = \sum_{d\in\{0,1\}} \Pr(D=d)\ \mathrm{KL}\big(p(Z \mid D=d)\,\big\|\,p(Z)\big).$$

Under the construction above, $p(Z \mid D=0) = P$, $p(Z \mid D=1) = Q$, and the marginal is

$$p(Z) = \tfrac{1}{2}P + \tfrac{1}{2}Q = M.$$

Therefore,

$$I(Z;D) = \tfrac{1}{2}\mathrm{KL}(P\|M) + \tfrac{1}{2}\mathrm{KL}(Q\|M) = \mathrm{JS}(P\|Q),$$

which proves equation 26. □

Lemma D.3 (Bounding $d_{\mathcal{H}\Delta\mathcal{H}}$ with Jensen-Shannon Divergence)**.** *Let $\mathcal{A}_{\mathcal{H}\Delta\mathcal{H}} := \{x \in \mathcal{X} : h(x) \neq h'(x)\ :\ h, h' \in \mathcal{H}\}$ denote the family of measurable disagreement sets. Then*

$$d_{\mathcal{H}\Delta\mathcal{H}}(P,Q) = 2 \sup_{A\in\mathcal{A}_{\mathcal{H}\Delta\mathcal{H}}} |P(A) - Q(A)| \;\le\; 2 \sup_{B\in\mathcal{B}} |P(B) - Q(B)| = d_1(P,Q),$$

and since $d_1(P,Q) = 2\,\mathrm{TV}(P,Q)$ where TV is total variation distance,

$$d_{\mathcal{H}\Delta\mathcal{H}}(P,Q) \le 2\,\mathrm{TV}(P,Q).$$

Next, by triangle inequality for TV and Pinsker's inequality,

$$\begin{aligned}
\mathrm{TV}(P,Q) &\le \mathrm{TV}(P,M) + \mathrm{TV}(Q,M) \\
&\le \sqrt{\tfrac{1}{2}\mathrm{KL}(P\|M)} + \sqrt{\tfrac{1}{2}\mathrm{KL}(Q\|M)} \\
&\le \sqrt{\mathrm{KL}(P\|M) + \mathrm{KL}(Q\|M)} = \sqrt{2\,\mathrm{JS}(P\|Q)}.
\end{aligned}$$

Thus, $d_{\mathcal{H}\Delta\mathcal{H}}$ is bounded by JS divergence by

$$d_{\mathcal{H}\Delta\mathcal{H}}(P,Q) \le 2\sqrt{2\mathrm{JS}(P\|Q)}. \tag{27}$$

Lemma D.4 (Domain Incremental Learning error bound)**.** *As we are interested in Domain Incremental Learning, instead of two domains, we require a notion of error bound across a series of domains. Hence, by recursively applying Eq.* (22) *over t domains yields*

$$e_t(h) \;\le\; e_1(h) + \sum_{k=2}^{t} d_{\mathcal{H}\Delta\mathcal{H}}(P_k, P_{k-1}) + \lambda^*, \tag{28}$$

where $\lambda_\tau^* \triangleq \min_{h\in\mathcal{H}_\tau}\big(e_1(h)+\ldots+e_t(h)\big)$. *Substituting Eq.* (27) *into Eq.* (28) *gives the mutual-information form of the DIL bound:*

$$e_t(h) \;\le\; e_1(h) + \sum_{k=2}^{t}\sqrt{2\ \mathrm{JS}(Z;D_k)} + \lambda^*. \tag{29}$$

As shown in Lem. D.2, we substitute the JS divergence term with the mutual information between Z *and* D *to achieve the error bounded by mutual information*

$$e_t(h) \;\le\; e_1(h) + \sum_{k=2}^{t}\sqrt{\ln 2\ I(Z;D_k)} + \lambda^*. \tag{30}$$

Lemma D.5 (Semantic-related representation and bound)**.** *The error bound in Eq.* (30) *utilized the learned general representation* Z*, which is composed of semantic-related and domain-specific information, which we will denote with the tuple*

$$Z = (Z_\tau, Z_\delta),$$

where Z_τ *captures semantic-relevant content and* Z_δ *captures domain-specific factors. As* Z_δ *often contains misleading domain-specific information, we instead propose to utilized* Z_τ *in DIL only, which we show in Sec. 3.2 on how to disentangle* Z_τ, Z_δ*. Thus, by the chain rule of mutual information,*

$$\begin{aligned} I(Z;D_k) &= I\big((Z_\tau,Z_\delta);D_k\big) \\ &= I(Z_\tau;D_k) + I(Z_\delta;D_k \mid Z_\tau), \end{aligned} \tag{31}$$

which separates the domain information carried by the task component from the additional domain information contained in Z_δ*. Since conditional mutual information is nonnegative, we have* $I(Z_\tau;D_k) \le I(Z;D_k)$*.*

Motivated by this decomposition, we consider predictors that depend only on the semantic component. Thus, let h_t *be the hypothesis depending only on* Z_τ *and denote the induced hypothesis class by* $\mathcal{H}_\tau$*. Hence, the sequential DIL bound for semantic-related representation is given by:*

$$e_t(h_\tau) \;\lesssim\; e_1(h_\tau) + \sum_{k=2}^{t}\sqrt{\ln 2\ I(Z_\tau;D_k)} + \lambda_\tau^*. \tag{32}$$

D.2 Wasserstein distance and Memory Management

Lemma D.6 (Wasserstein-p Distance (Kantorovich, 1960; Vaserstein, 1969))**.** *Let* $\mathcal{X}$ *and* $\mathcal{X}'$ *be probability measures on a metric space* $(\mathcal{M},d)$*, and let* $p\in[1,+\infty)$*. The* Wasserstein-p distance *is*

$$W_p(\mathcal{X},\mathcal{X}') = \left(\inf_{\gamma\in\Pi(\mathcal{X},\mathcal{X}')}\mathbb{E}_{(x,x')\sim\gamma}\big[d(x,x')^p\big]\right)^{1/p}, \tag{33}$$

where $\Pi(\mathcal{X},\mathcal{Y})$ *is the set of all couplings* γ *on* $\mathcal{M}\times\mathcal{M}$ *whose marginals are* $\mathcal{X}$ *and* $\mathcal{X}'$*, respectively.*

For empirical measures supported on $\{x_i, x_i'\in\mathbb{R}^d\}_{i=1}^n$*, computing the exact* W_2 *via the Hungarian algorithm (Kuhn, 1955) requires forming an* $n\times n$ *cost matrix (cost* $\mathcal{O}(n^2d)$*) and solving a linear assignment problem (cost* $\mathcal{O}(n^3)$*).*

Lemma D.7 (Wasserstein-p Distance (Dowson & Landau, 1982) between Gaussian Distributions)**.** *If* $\mathcal{X}=\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ *and* $\mathcal{X}'=\mathcal{N}(\boldsymbol{\mu}',\boldsymbol{\Sigma}')$ *are Gaussian on* $\mathbb{R}^d$*, then the squared* W_2 *distance admits the closed form*

$$W_2^2(\mathcal{X},\mathcal{X}') = \|\boldsymbol{\mu}-\boldsymbol{\mu}'\|_2^2 + \mathrm{Tr}\Big(\boldsymbol{\Sigma}+\boldsymbol{\Sigma}' - 2(\boldsymbol{\Sigma}^{1/2}\boldsymbol{\Sigma}'\boldsymbol{\Sigma}^{1/2})^{1/2}\Big), \tag{34}$$

where $\mathrm{Tr}(\cdot)$ *denotes the matrix trace and* $\boldsymbol{\Sigma}^{1/2}$ *is the unique symmetric positive-semidefinite square root of* $\boldsymbol{\Sigma}$*.* W_2 *distance between gaussian distribution is also known as the Frećhet Distance.*

Evaluating Eq. (34) *typically entails an eigen-decomposition or SVD, i.e.,* $\mathcal{O}(d^3)$*, and can be numerically unstable for ill-conditioned covariances.*

Lemma D.8 (Metric Space of Domain and Task Representation)**.** *As motivated by Eq.* (3)*, we aim to learn a representation whose semantic-related component preserves label information while remaining as independent as possible from domain identity. Thus, the features extracted are disentangled and broken down into two parts,*

$$Z_\tau = f_\tau(x), \qquad Z_\delta = f_\delta(x),$$

where $Z_\tau \in \mathbb{R}^p$ denotes the semantic-related representation, $Z_\delta \in \mathbb{R}^q$ denotes the domain-related representation, and f_τ, f_δ are the two branches of the encoder.

Let $\mathcal{P}, \mathcal{P}'$ be the feature distribution of two domains, following the well-established assumption of low-dimensional image features can be modeled as Gaussian distribution (Heusel et al., 2017). Hence, we define $\mathcal{P} = \mathcal{N}(\mu_\mathcal{P}, \Sigma_\mathcal{P})$, and $\mathcal{P}' = \mathcal{N}(\mu_{\mathcal{P}'}, \Sigma_{\mathcal{P}'})$.

For two domain $\mathcal{D}$ and $\mathcal{D}'$, we model the task and domain representations as gaussian distributions $\mathcal{P}_\tau, \mathcal{P}_\delta$ and $\mathcal{P}'_\tau, \mathcal{P}'_\delta$. In order to define a distance between $\mathcal{D}$ and $\mathcal{D}'$, let us define:

$$d(\mathcal{D}, \mathcal{D}') = \sqrt{W_2^2(\mathcal{P}_\tau, \mathcal{P}'_\tau) + W_2^2(\mathcal{P}_\delta, \mathcal{P}'_\delta)} \tag{35}$$

Lemma D.9 (Definition of misaligned sample)**.** *For the proposed method (Sec. 3), we maintain a small episodic memory $\mathcal{M}$ that stores entries of the form*

$$m = (x, y, \bar{Z}_\tau, \bar{Z}_\delta),$$

where (x, y) is the input–label pair, $(\bar{Z}_\tau, \bar{Z}_\delta)$ are the semantic-related and domain-related representations computed at insertion time.

In domain-incremental learning, the encoder updates as new domains arrive, so the stored representations may become misaligned *with respect to the current model. We quantify misalignment via normalized drift in each component:*

$$\Delta_\tau \;=\; \frac{\left\|Z_\tau^{\text{new}} - \bar{Z}_\tau\right\|}{\|\bar{Z}_\tau\| + \varepsilon}, \qquad \Delta_\delta \;=\; \frac{\left\|Z_\delta^{\text{new}} - \bar{Z}_\delta\right\|}{\|\bar{Z}_\delta\| + \varepsilon}, \tag{36}$$

where $Z_\tau^{\text{new}} = f_\tau(x)$ and $Z_\delta^{\text{new}} = f_\delta(x)$ are the current semantic/domain features for the same input x, and $\varepsilon > 0$ is a small constant.

To obtain an interpretable partition of memory behaviors, we center drifts by their batch medians and embed samples in a semantic–domain drift plane, as illustrated in Fig. 4:

$$u = \Delta_\tau - \text{median}(\Delta_\tau), \qquad v = \Delta_\delta - \text{median}(\Delta_\delta). \tag{37}$$

Let $\mathcal{T}(P)$ denote a typical *(high-density) region of a distribution P. We interpret $u > 0$ (resp. $v > 0$) as task (resp. domain) drift, meaning that the new feature falls outside the overlap of typical regions between the current and stored feature distributions:*

$$u > 0 \iff Z_\tau^{\text{new}} \notin \mathcal{T}(P_\tau^{\text{new}}) \cap \mathcal{T}(P_\tau), \qquad v > 0 \iff Z_\delta^{\text{new}} \notin \mathcal{T}(P_\delta^{\text{new}}) \cap \mathcal{T}(P_\delta),$$

and conversely $u < 0$/$v < 0$ indicates relative stability (membership in the corresponding intersection).

The sign pattern of (u, v) yields four quadrants with heuristic target actions:

$$a^\star(u, v) = \begin{cases} \textit{Delete}, & u > 0,\ v > 0 \quad \textit{(large task \& domain drift)}, \\ \textit{Repair}, & (u < 0,\ v > 0) \textit{ or } (u > 0,\ v < 0) \quad \textit{(one component drifts)}, \\ \textit{Retain}, & u < 0,\ v < 0 \quad \textit{(stable)}. \end{cases} \tag{38}$$

These targets are not enforced as hard rules; instead, they are used to shape the RL reward.

D.3 Mean-dominant W_2 changes induced by Remove/Retain/Repair

Setup. Fix a reference Gaussian $Q = \mathcal{N}(\nu, \Lambda)$ on $\mathbb{R}^d$. Given samples $x_1, \ldots, x_n \in \mathbb{R}^d$, define the (biased) moment-matching empirical Gaussian

$$G_P = \mathcal{N}(\mu, \Sigma), \qquad \mu = \frac{1}{n}\sum_{i=1}^n x_i, \qquad \Sigma = \frac{1}{n}\sum_{i=1}^n (x_i - \mu)(x_i - \mu)^\top.$$

For Gaussians, the squared Wasserstein–2 distance admits the closed form

$$W_2^2(\mathcal{N}(m_1, C_1), \mathcal{N}(m_2, C_2)) = \|m_1 - m_2\|_2^2 + \mathcal{B}(C_1, C_2), \tag{39}$$

where the Bures term is $\mathcal{B}(C_1, C_2) := \text{Tr}\big(C_1 + C_2 - 2(C_2^{1/2} C_1 C_2^{1/2})^{1/2}\big)$.

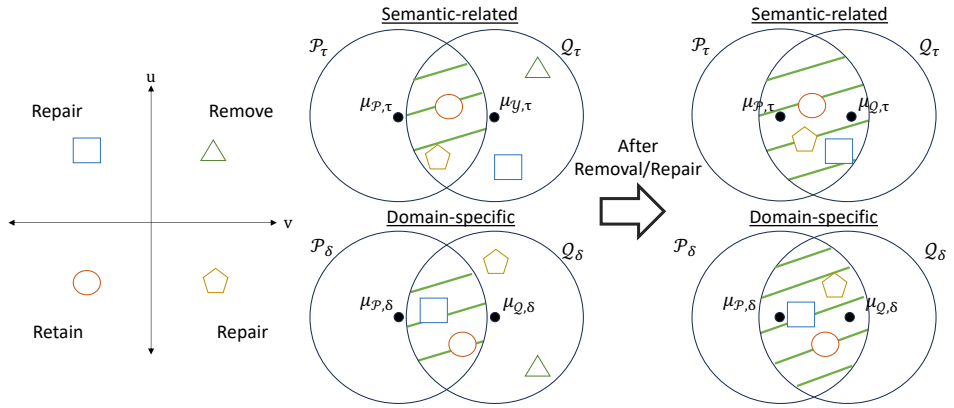


Figure 4: For better visualization, let $\mathcal{P} = (\mathcal{P}_\tau, \mathcal{P}_\delta) \in \mathbf{R}^2$ and $\mathcal{Q} = (\mathcal{Q}_\tau, \mathcal{Q}_\delta) \in \mathbf{R}^2$ be two isotropic gaussian distribution. We visualize our memory management actions to correspond to different quadrants on the semantic-task drift plane. We could interpret $u < 0$ features, $z_\tau \sim \mathcal{P}_\tau \cap \mathcal{Q}_\tau$ and $v < 0$ features, $z_\delta \sim \mathcal{P}_\delta \cap \mathcal{Q}_\delta$ and vice-versa. We visualize the effect of memory management through the *remove* and *repair* actions: *remove* discards outlier samples, while *repair* updates misaligned samples, jointly bringing $\mathcal{P}$ and $\mathcal{Q}$ closer in semantic and domain space.

Mean-dominant regime (used throughout). To keep the statements and proofs focused on the effect of mean shift, we work in an idealized regime where the covariance contribution is (approximately) unchanged by the update of interest:

$$\mathcal{B}(\Sigma_{\text{new}}, \Lambda) \approx \mathcal{B}(\Sigma, \Lambda). \tag{40}$$

In the limit $\Sigma_{\text{new}} = \Sigma$, all results below are exact.

Lemma D.10 (Mean-shift criterion for Gaussian W_2)**.** *Let $G = \mathcal{N}(\mu, \Sigma)$ and $Q = \mathcal{N}(\nu, \Lambda)$. Define $\delta := \mu - \nu$. For any update $\mu_\eta = \mu + \eta v$ (with $\eta \in \mathbb{R}$, $v \in \mathbb{R}^d$) and corresponding $G_\eta = \mathcal{N}(\mu_\eta, \Sigma)$, we have*

$$W_2^2(G_\eta, Q) - W_2^2(G, Q) = 2\eta\, \delta^\top v + \eta^2 \|v\|_2^2. \tag{41}$$

In particular, for $\eta > 0$,

$$W_2^2(G_\eta, Q) \le W_2^2(G, Q) \quad \Longleftrightarrow \quad \delta^\top v \le -\frac{\eta}{2}\|v\|_2^2, \tag{42}$$

with strict inequality whenever equation 42 is strict.

Proof. By equation 34 with fixed covariance Σ,

$$W_2^2(G_\eta, Q) - W_2^2(G, Q) = \|\mu_\eta - \nu\|_2^2 - \|\mu - \nu\|_2^2 = \|\delta + \eta v\|_2^2 - \|\delta\|_2^2,$$

which expands to equation 41 and yields equation 42. □

Theorem D.11 (Removal of a misaligned memory sample strictly decreases the W_2 distance)**.** *Let $Q = \mathcal{N}(\nu, \Lambda)$ be fixed. Let $G_P = \mathcal{N}(\mu, \Sigma)$ be built from $x_1, \ldots, x_n \in \mathbb{R}^d$, and pick an index $o \in \{1, \ldots, n\}$. Remove x_o and form the leave-one-out Gaussian $G_{P\setminus o} = \mathcal{N}(\mu', \Sigma')$ from $\{x_i\}_{i \ne o}$. Define*

$$\phi := x_o - \mu, \qquad \delta := \mu - \nu.$$

Assume:

(1) ***Mean-outlier condition:***

$$\delta^\top \phi > \frac{\|\phi\|_2^2}{2(n-1)}. \tag{43}$$

(2) ***Covariance stability:*** *equation 40 holds with $\Sigma_{new} = \Sigma'$.*

Then removing x_o strictly decreases the Wasserstein–2 distance to Q:

$$W_2\big(G_{P\setminus o}, Q\big) < W_2\big(G_P, Q\big).$$

Proof. A standard leave-one-out identity gives

$$\mu' = \frac{n\mu - x_o}{n-1} = \mu - \frac{\phi}{n-1}. \tag{44}$$

Apply Lemma D.10 with $\eta = \frac{1}{n-1}$ and $v = -\phi$:

$$W_2^2(G_{P\setminus o}, Q) - W_2^2(G_P, Q) \approx 2\eta\,\delta^\top(-\phi) + \eta^2\|\phi\|_2^2 = -\frac{2}{n-1}\delta^\top\phi + \frac{1}{(n-1)^2}\|\phi\|_2^2.$$

Under equation 43, the right-hand side is strictly negative, hence the claim. □

Lemma D.12 (Retain: aligned memory samples should not be removed)**.** *In the setting of Theorem D.11, assume instead the* ***mean-alignment condition***

$$\delta^\top\phi \le \frac{\|\phi\|_2^2}{2(n-1)}. \tag{45}$$

Together with covariance stability equation 40, removing x_o cannot strictly reduce the Wasserstein–2 distance to Q; more precisely,

$$W_2^2\big(G_{P\setminus o}, Q\big) \;\ge\; W_2^2\big(G_P, Q\big), \tag{46}$$

with equality if and only if equality holds in equation 45.

Proof. Use equation 44 and Lemma D.10 with $\eta = \frac{1}{n-1}$ and $v = -\phi$. Condition equation 45 is exactly the negation (non-strict) of equation 43, hence the mean term is non-decreasing, and by equation 40 so is W_2^2. □

Lemma D.13 (Repair: convex update of domain features towards the current encoder)**.** *Let $G_P = \mathcal{N}(\mu, \Sigma)$ be built from domain features $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$, and let $Q = \mathcal{N}(\nu, \Lambda)$ be a reference Gaussian. Define $\delta := \mu - \nu$. Fix one memory item x_o and let $z \in \mathbb{R}^d$ be its* current *domain feature under the updated encoder. For $\alpha \in (0, 1]$, define the repaired feature*

$$x_o^{(\alpha)} := (1-\alpha)x_o + \alpha z, \qquad \Delta := z - x_o,$$

and let $G_{P^{(\alpha)}} = \mathcal{N}(\mu_\alpha, \Sigma_\alpha)$ be the moment-matching Gaussian obtained by replacing x_o with $x_o^{(\alpha)}$. Then

$$\mu_\alpha = \mu + \frac{\alpha}{n}\Delta.$$

Assume:

(1) ***Repair direction condition:***

$$\delta^\top\Delta \le -\frac{\alpha}{2n}\|\Delta\|_2^2. \tag{47}$$

(2) ***Covariance stability:*** *equation 40 holds with $\Sigma_{new} = \Sigma_\alpha$.*

Then the Repair update does not increase the Wasserstein–2 distance:

$$W_2^2\big(G_{P^{(\alpha)}}, Q\big) \;\le\; W_2^2\big(G_P, Q\big), \tag{48}$$

with strict inequality whenever equation 47 is strict.

Proof. Apply Lemma D.10 with $\eta = \frac{\alpha}{n}$ and $v = \Delta$:

$$W_2^2(G_{P^{(\alpha)}}, Q) - W_2^2(G_P, Q) \approx 2\eta\,\delta^\top\Delta + \eta^2\|\Delta\|_2^2.$$

Condition equation 47 is exactly equation 42 for this choice of η and v, hence the right-hand side is ≤ 0 (strictly < 0 if equation 47 is strict). □

Corollary D.14 (Effect of memory actions on the overall distance d)**.** *Let*

$$d(\mathcal{D}, \mathcal{D}') := \sqrt{W_2^2(\mathcal{P}_\tau, \mathcal{P}'_\tau) + W_2^2(\mathcal{P}_\delta, \mathcal{P}'_\delta)} \tag{49}$$

as in Lemma D.8. Consider an update that affects exactly one component distribution in equation 49 (e.g., $\mathcal{P}_\tau$ while $\mathcal{P}_\delta$ is unchanged), and choose Q as the corresponding reference Gaussian (e.g., $Q = \mathcal{P}'_\tau$). Then:

(i) ***Remove / Repair (distance-reducing actions).*** *Under the assumptions of Theorem D.11 (Remove), the affected Wasserstein term strictly decreases, hence $d(\mathcal{D}, \mathcal{D}')$ strictly decreases. Under the assumptions of Lemma D.13 (Repair), the affected term does not increase, hence $d(\mathcal{D}, \mathcal{D}')$ does not increase; moreover, the decrease is strict whenever equation 47 is strict.*

(ii) ***Retain (removing aligned samples is harmful).*** *Under the assumptions of Lemma D.12 (aligned/Retain), the affected Wasserstein term cannot decrease:*

$$W_2^2\big(G_{P\setminus o}, Q\big) \geq W_2^2\big(G_P, Q\big),$$

and therefore $d(\mathcal{D}, \mathcal{D}')$ cannot decrease. Furthermore, the inequality is strict, and thus $d(\mathcal{D}, \mathcal{D}')$ strictly increases whenever the mean-alignment condition is strict, i.e., $\delta^\top \phi < \frac{\|\phi\|_2^2}{2(n-1)}$ (equivalently, whenever equation 45 does not hold with equality).

D.4 Wasserstein-2 viewpoint on reward function

Lemma D.15 (Johnson–Lindenstrauss lemma (Larsen & Nelson, 2017))**.** *Let $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ be features extracted by feature encoder, and some linear map $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ where $\boldsymbol{A} \in \mathbb{R}^{k \times n}$ for $k < n$ and $k = \mathcal{O}\log(n)$. Thus, $f : \mathbb{R}^n \to \mathbb{R}^k$ is a linear mapping from high-dimensional features to a low-dimensional output space. Then, there exists a bound where:*

$$(1-\epsilon)\|\boldsymbol{u} - \boldsymbol{v}\|^2 \leq \|f(\boldsymbol{u}) - f(\boldsymbol{v})\|^2 \leq (1+\epsilon)\|\boldsymbol{u} - \boldsymbol{v}\|^2. \tag{50}$$

Thus, the Johnson–Lindenstrauss (JL) lemma is tight up to a constant factor of $\mathcal{O}\log(n)$.

Lemma D.16 (Wasserstein distance under JL Lemma)**.** *Define coupling π be a matrix where all its entries are non-negative, and a cost function $\mathrm{cost}(\cdot)$ be*

$$\mathrm{cost}(\pi) = \sum_{i,j} \pi_{ij} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2. \tag{51}$$

After some linear mapping to low-dimension, the new cost $\mathrm{cost}_{\boldsymbol{A}}(\pi)$ is:

$$\mathrm{cost}_{\boldsymbol{A}}(\pi) = \sum_{i,j} \pi_{ij} \|\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j\|_2^2. \tag{52}$$

If JL lemma holds for all $\boldsymbol{x}$, then

$$(1-\epsilon)\mathrm{cost}(\pi) \leq \mathrm{cost}_{\boldsymbol{A}}(\pi) \leq (1+\epsilon)\mathrm{cost}(\pi)$$

As mentioned in equation 33, the Wasserstein distance is essentially taking the infimum of all possible couplings. Thus, taking the infimum on all terms yields

$$(1-\epsilon)W_2^2(\boldsymbol{x}_i - \boldsymbol{x}_j) \leq W_2^2(\boldsymbol{x}_i - \boldsymbol{x}_j) \leq (1+\epsilon)W_2^2(\boldsymbol{x}_i - \boldsymbol{x}_j) \tag{53}$$

replacing x_i and x_j with μ and ν:

$$(1-\epsilon)W_2^2(\mu - \nu) \leq W_2^2(\boldsymbol{A}_{\#}\mu - \boldsymbol{A}_{\#}\nu) \leq (1+\epsilon)W_2^2(\mu - \nu), \tag{54}$$

showing that Wasserstein-2 distance is preserved under JL lemma with some pushforwards $\boldsymbol{A}_{\#}$.

Thus, Lem. D.16 shows that decreasing Wasserstein distance in feature space also decrease Wasserstein distance in the output space bounded by $1 \pm \epsilon$. This is done by approximating the classifier h_θ and the domain classifier g_ϕ as a linear mapping and ignoring the activation functions between linear layers in the MLP classifiers.

Lemma D.17 (Wasserstein Distance bound on reward)**.** *For classification loss (cross-entropy with softmax) is Lipschitz continuous with respect to output logits (Nern et al., 2023). Let $\ell(o, y)$ be the classification loss for logit o, then,*

$$|\ell(o, y) - \ell(o', y) \leq \boldsymbol{L}\|o - o'\|. \tag{55}$$

Let us define O, O' be two distribution of logits and the expected risk of the domain to be

$$e(h) = \mathbb{E}_{(o,y)\sim O}[\ell(o, y)] \tag{56}$$

Hence, due to Eq. (55) *and Hölder's inequality (Villani et al., 2009), then*

$$|e(h) - e(h')| \leq \boldsymbol{L} W_1(O, O') \leq \boldsymbol{L} W_2(O, O'), \tag{57}$$

and combining with Eq. (54)*, this yields*

$$|e(h) - e(h')| \lesssim \boldsymbol{L}\sqrt{1+\epsilon} W_2(\mu, \nu). \tag{58}$$

By incorporating the overall distance Eq. (49) *and define a constant* $\boldsymbol{C}$ *that depends on* $\boldsymbol{L}$ *and* ϵ*, we obtain*

$$|e(h) - e(h')| \lesssim \boldsymbol{C} d(D, D'). \tag{59}$$

As $\mathrm{Acc} = 1 - e(h)$*, then the reward function* $r_{val} = -(e(h) - e(h'))$*, thus, allowing us to define the bound*

$$|r_{val}| \lesssim \boldsymbol{C} d(D, D'). \tag{60}$$

Lemma D.18 (Link between Wasserstein distance and mutual information)**.** *As discussed in Eq.* (3) *under Sec. 3.1,*

$$e_t(h) \;\lesssim\; e_1(h) + \sum_{k=2}^{t} 2\sqrt{\ln 2 \; I(Z_\tau; D_k)} + \lambda^*,$$

the accumulated error in domain-incremental learning is determined by the mutual information between the learned representation Z *and the domain identity* D_k*. By definition,*

$$I(Z_\tau; D_k) = \sum_k \omega_k D_{KL}(P_{\tau,k} || P_{Z_\tau}), \tag{61}$$

where $\omega_k = \Pr(D = k)$*. Following the assumption that the domain's semantic-relevant feature distribution is Gaussian-distributed in Lem. D.8, then*

$$\begin{aligned} D_{KL}(P_{\tau,k} \| P_{Z_\tau}) &\approx D_{KL}(\mathcal{N}(\mu_{\tau,k}, \Sigma_{\tau,k}) \| \mathcal{N}(\mu_\tau, \Sigma_\tau)) \\ &= \frac{1}{2}\Big[\mathrm{tr}(\Sigma_\tau^{-1}\Sigma_{\tau,k}) + (\mu_\tau - \mu_{\tau,k})^\top \Sigma_\tau^{-1}(\mu_\tau - \mu_{\tau,k}) - d + \log \frac{\det \Sigma_\tau}{\det \Sigma_{\tau,k}}\Big]. \end{aligned}$$

and for simplicity $\Sigma_{\tau,k} = \Sigma_\tau = \Sigma,\ \forall k$*,*

$$D_{KL}(P_{\tau,k} || P_{Z_\tau}) \approx D_{KL}(\mathcal{N}(\mu_{\tau,k}, \Sigma_{\tau,k}) || \mathcal{N}(\mu_\tau, \Sigma_\tau)) = \frac{1}{2}\mu_\tau - \mu_{\tau,k})^\top \Sigma_\tau^{-1}(\mu_\tau - \mu_{\tau,k}),$$

and this is exactly in the form of Mahalanobis distance. As the spectral bound of a symmetric positive definite matrix $\boldsymbol{A}$ *is defined as:*

$$\boldsymbol{v}^\top \boldsymbol{A} \boldsymbol{v} \leq \lambda_{max}(\boldsymbol{A}) \|\boldsymbol{v}\|_2^2, \tag{62}$$

where $\lambda_{max}(\boldsymbol{A})$ *is the maximum eigenvalue of* $\boldsymbol{A}$*, then, for* $\boldsymbol{C}_{\Sigma_\tau} := \frac{1}{2}\lambda_{max}(\Sigma_\tau)$

$$D_{KL}(P_{\tau,j} \,\|\, P_{Z_\tau}) \lesssim C_\Sigma \, \|\mu_\tau - \mu_{\tau,j}\|_2^2. \tag{63}$$

Thus, as the Wasserstein-2 distance between Gaussian distributions under the assumption of same covariance is

$$W_2^2\big(P_{\tau,j}, P_{Z_\tau}\big) \approx W_2^2\big(\mathcal{N}(\mu_{\tau,j}, \Sigma), \mathcal{N}(\mu_\tau, \Sigma)\big) = \|\mu_\tau - \mu_{\tau,j}\|_2^2, \tag{64}$$

then combining Eq. (63)*, Eq.* (34) *into Eq.* (61)*, this obtains the bound*

$$I(Z_\tau; D_k) \lesssim C_\Sigma \sum_j \omega_j W_2^2\big(P_{\tau,j}, P_{Z_\tau}\big). \tag{65}$$

Thus, the mutual information between the task-relevant representation Z_τ *and the domain identity* D_k *is upper bounded (up to a constant depending on* Σ*) by the* ω_j*-weighted dispersion of the per-domain Gaussian feature distributions around their barycenter, measured in Wasserstein-2 distance.*

In particular, any memory management policy that contracts the Wasserstein-2 distances $\{W_2(P_{\tau,j}, P_{Z_\tau})\}_j$ *across domains will decrease* $I(Z_\tau; D_k)$*, thereby tightening the DIL generalization bound in Eq.* (30)*.*

Remark D.19. *Limitations: While we analytically show how memory management can contribute to the DIL error bound, our derivations have several limitations:*

1. *We assume that the feature distribution can be modeled as a Gaussian. While this assumption is common in the image generation literature (Heusel et al., 2017), a recent study on CMMD (Jayasumana et al., 2024) suggests it may not always hold.*

2. *For simplicity, we assume that memory operations do not affect the covariance. In practice, accounting for covariance terms would substantially increase the analytical complexity and may change the effects of memory operations.*

E EXPERIMENTAL SETTING

This appendix describes our experimental protocol, including dataset construction, domain ordering, data splits, model architectures and preprocessing, optimization details, replay buffer management, and evaluation.

E.1 DATASETS, DOMAINS, AND DOMAIN ORDER

Rotated-MNIST. We construct Rotated-MNIST from MNIST, which contains 60,000 training and 10,000 test grayscale images of size 28×28.[1] Each domain corresponds to a fixed rotation angle applied to all images in that domain. We use **11** rotation domains with angles

$$\mathcal{A} = \{0^\circ, 18^\circ, 36^\circ, \ldots, 180^\circ\}.$$

For efficiency, we create a subset with 20,000 training and 4,000 test images via class-stratified sampling from the original MNIST train/test splits. We then assign samples to the 11 domains as evenly as possible:

- **Training:** 9 domains receive 1,818 images and 2 domains receive 1,819 images (total 20,000).
- **Test:** 4 domains receive 363 images and 7 domains receive 364 images (total 4,000).

Rotation is implemented as a deterministic domain-specific transform and does not change labels.

DomainNet. We use the cleaned DomainNet dataset distributed by TensorFlow Datasets.[2] DomainNet contains 6 domains (Real, Infograph, Clipart, Painting, Sketch, Quickdraw) and 345 classes; images have varying native resolutions (Peng et al., 2019). We adopt the fixed domain sequence:

$$\text{Real} \to \text{Infograph} \to \text{Clipart} \to \text{Painting} \to \text{Sketch} \to \text{Quickdraw}.$$

This ordering starts from natural images and progressively moves to more abstract and stylized renderings, and it is consistent with prior sequential/continual domain adaptation protocols that choose a shortest-path order based on inter-domain distances (Feng et al., 2023). To form a training stream of 120,000 images, we sample 20,000 images from the `train` split of each domain using class-stratified sampling whenever possible. For evaluation, we use each domain's official `test` split.

Camelyon17-WILDS. We use Camelyon17-wilds from WILDS.[3] Each example is a histopathology patch of size 96×96 with a binary tumor label, and the domain corresponds to the hospital (5 hospitals). We treat hospitals as domains and use the hospital id order:

$$0 \to 1 \to 2 \to 3 \to 4.$$

To construct a balanced subset of 100,000 patches, we sample 20,000 patches per hospital with label balance (10,000 tumor and 10,000 non-tumor) within each hospital.

iWildCam-WILDS. We use the iWildCam species-classification benchmark from WILDS (Beery et al., 2020).[4] Each example is a single camera-trap image, the label is one of 182 animal species, and the domain corresponds to the camera-trap location ID. This benchmark captures realistic domain shifts across camera locations, including changes in background, lighting, vegetation, viewpoint, and species frequency.

To adapt iWildCam to DIL, we keep the 182-way label space fixed across all domains and treat camera locations as domains. Since location sizes are highly imbalanced, we select **10** training locations with sufficient samples and sample **600** images from each location, resulting in a 10-domain incremental sequence:

$$d_1 \to d_2 \to \cdots \to d_{10}.$$

This setup tests whether a model can continually adapt to new camera locations while retaining species-discrimination knowledge from previous locations.

[1] `https://keras.io/2/api/datasets/mnist/`
[2] `https://www.tensorflow.org/datasets/catalog/domainnet`
[3] `https://wilds.stanford.edu/datasets/#camelyon17`
[4] `https://wilds.stanford.edu/datasets/#iwildcam`

E.2 TRAIN/VALIDATION/TEST SPLITS

We construct splits *independently within each domain*. For each domain, we partition the available data into **train/validation/test** with an **80/10/10** ratio using a fixed random seed. The validation split is used for model selection (e.g., early stopping and hyperparameter tuning), and the test split is used only for reporting results.

Camelyon17 (slide-level splitting). Camelyon17 contains many patches extracted from the same whole-slide image (WSI). We therefore perform **slide-level** splitting within each hospital domain. Specifically, for each hospital, we collect the set of slide IDs associated with that hospital, randomly split the **slides** into **80/10/10** train/validation/test groups using the same fixed seed, and then assign each patch to the split of its slide ID. Consequently, **all patches from the same slide appear in exactly one split**. When constructing the balanced subset, we perform label-balanced subsampling *after* slide-level splitting and independently within each split.

E.3 MODELS AND INPUT PREPROCESSING

For **Rotated-MNIST**, we use a compact three-layer CNN designed for 28×28 grayscale inputs. The network consists of two convolutional layers followed by ReLU activation and max-pooling, and one fully connected classifier for digit prediction. This backbone is sufficient for the controlled rotation-based shifts in Rotated-MNIST while keeping the model capacity substantially smaller than the ResNet-18 and ViT-B/16 backbones used for the larger image benchmarks. Rotations are applied on-the-fly as deterministic domain-specific transforms. Since Rotated-MNIST is a controlled synthetic benchmark, we report its results in Appendix Sec. F.

For the three main benchmarks, **DomainNet**, **Camelyon17**, and **iWildCam**, we evaluate all methods with both ResNet-18 and ViT-B/16 backbones. For DomainNet and iWildCam, images are resized to 224×224 and normalized using ImageNet statistics. For Camelyon17, we keep the native 96×96 patch resolution for ResNet-18, and resize images to 224×224 when using ViT-B/16. Unless otherwise specified, all other preprocessing steps follow the standard dataset protocols.

E.4 TRAINING AND OPTIMIZATION DETAILS

Unless otherwise stated, we optimize all models using SGD with momentum and keep the optimization settings identical across methods; only the continual-learning strategy differs:

- **Optimizer:** SGD with momentum 0.9.
- **Learning rate:** 0.1 with a standard decay schedule (cosine or step), shared across methods.
- **Weight decay:** 5×10^{-4} for Rotated-MNIST; 1×10^{-4} for DomainNet, Camelyon17 and iWildCam.
- **Batch size:** 128.
- **Epochs per domain:** 20 for Rotated-MNIST; 20–30 for DomainNet, Camelyon17 and iWildCam.
- **EMA repair coefficient:** $\beta = 0.7$.
- **Gradient reversal coefficient:** $\eta = 1.0$.

E.5 REPLAY BUFFER: BUDGET, UPDATES, AND SAMPLING

Memory budget. We adopt a replay buffer budget M (maximum number of stored samples): M=200 for Rotated-MNIST, M=2000 for DomainNet, M=1000 for Camelyon17, and M=1000 for iWildCam. Importantly, M is an *upper bound*. Since **RAM** performs *retain/repair/remove* on the buffer, the memory size is *not* a simple cumulative sum of per-domain insertions and can vary over time. In practice, **RAM** often stays *below* M due to deletions, whereas conventional replay baselines typically keep the buffer saturated at M once enough domains have been seen. This design highlights that **RAM** can outperform prior methods even with a smaller effective memory footprint, making it more memory-efficient in practice.

How many samples are added after each domain. Let N be the number of domains. After finishing training on domain t, we add approximately

$$k \approx \left\lfloor \frac{M}{N} \right\rfloor$$

new samples from the current domain into the replay buffer. Concretely, we use:

- **Rotated-MNIST** (N=11, M=200): k=10 samples per domain.
- **DomainNet** (N=6, M=2000): k=300 samples per domain.
- **Camelyon17** (N=5, M=1000): k=200 samples per domain.
- **iWildCam** (N=10, M=1000): k=100 samples per domain.

Note that the *effective* buffer size under **RAM** can remain below M because the controller may delete low-utility or harmful entries before inserting new ones.

Which samples are added. New samples are selected from the *training split* of the current domain:

- **Rotated-MNIST:** class-stratified sampling with approximately equal samples per class.
- **DomainNet:** class-stratified sampling over 345 classes whenever possible; if a class has fewer available instances, we sample uniformly from its available images.
- **Camelyon17:** label-balanced sampling within each hospital to maintain a balanced replay set.
- **iWildCam:** each selected camera-location domain contains 600 training images. We sample k=100 replay candidates per domain from these 600 images using class-aware sampling when possible. Since the 182-way species labels are naturally imbalanced within each location, if strict class stratification is infeasible, we sample uniformly from the available images while preserving the fixed label space.

E.6 Reproducibility

All experiments are run on two NVIDIA A800 GPUs. For each dataset and backbone setting, we keep the data splits, domain order, memory budget, optimizer, and training schedule fixed across methods, and vary only the continual-learning strategy. We run each main experiment with 5 random seeds and report the mean and standard deviation. Random seeds are fixed for dataset sampling, train/validation/test splitting, model initialization, minibatch shuffling, and replay-buffer sampling. All hyperparameters used by RAM, including the memory budget, EMA repair coefficient, and gradient reversal coefficient, are reported in Secs. E.4 and E.5.

E.7 Dataset Licenses

We use only publicly available benchmark datasets and follow their corresponding terms of use. MNIST is distributed through common dataset interfaces and is listed with a CC-BY-SA-3.0 license in public dataset documentation. For DomainNet, we use the cleaned version distributed by TensorFlow Datasets; the TFDS catalog notes that the dataset page content is under CC-BY 4.0 and code samples are under Apache 2.0, while the original DomainNet data are provided for research use. Camelyon17-WILDS is distributed in the public domain under CC0 1.0. iWildCam-WILDS is distributed under the Community Data License Agreement, Permissive, Version 1.0. We do not redistribute the datasets and use them only for research evaluation.

F Results on Rotated-MNIST

Rotated-MNIST provides a controlled DIL setting where each domain corresponds to a fixed rotation angle. Following our experimental setup, we use the same compact three-layer CNN for all methods, consisting of two convolutional layers with ReLU activation and max-pooling, followed by one fully connected classifier for digit prediction. This dataset is simpler than real-world domain-shift

Table 4: **Results on Rotated-MNIST.** We report Final Accuracy (Final%), Average Accuracy (Avg%), and Forgetting (Forget%; lower is better). All results are reported as mean$_{\pm}$ over multiple runs. **Bold** denotes the best mean and underline denotes the second best mean.

Method	Final↑	Avg↑	Forget↓
Memory Free			
LwF (Li & Hoiem, 2017)	89.3$_{\pm 0.43}$	84.1$_{\pm 0.57}$	13.9$_{\pm 0.68}$
EWC (Kirkpatrick et al., 2017)	87.8$_{\pm 0.56}$	82.9$_{\pm 0.62}$	14.8$_{\pm 0.79}$
MAS (Aljundi et al., 2018)	88.5$_{\pm 0.47}$	83.5$_{\pm 0.54}$	14.4$_{\pm 0.71}$
Replay-centric			
ER (Chaudhry et al., 2019)	92.7$_{\pm 0.33}$	90.8$_{\pm 0.31}$	7.6$_{\pm 0.46}$
DER++ (Buzzega et al., 2020)	94.3$_{\pm 0.27}$	92.1$_{\pm 0.28}$	6.7$_{\pm 0.37}$
SER (Zhuo et al., 2023)	94.8$_{\pm 0.21}$	92.7$_{\pm 0.26}$	6.1$_{\pm 0.32}$
X-DER (Boschini et al., 2023)	<u>95.6</u>$_{\pm 0.19}$	93.6$_{\pm 0.23}$	5.4$_{\pm 0.27}$
Memory Updating			
iCaRL (Rebuffi et al., 2017)	93.3$_{\pm 0.34}$	91.0$_{\pm 0.39}$	7.5$_{\pm 0.51}$
L2P-R (Wang et al., 2022)	95.0$_{\pm 0.24}$	<u>93.8</u>$_{\pm 0.26}$	<u>5.0</u>$_{\pm 0.31}$
AdaER (Li et al., 2024)	93.2$_{\pm 0.31}$	91.4$_{\pm 0.33}$	7.1$_{\pm 0.46}$
Structured Memory Policy			
RAM	**96.9**$_{\pm 0.17}$	**95.3**$_{\pm 0.19}$	**4.2**$_{\pm 0.22}$

benchmarks, but it allows us to isolate progressive rotation shifts and evaluate whether adaptive memory management reduces forgetting under a well-controlled transformation sequence. Tab. 4 reports the results.

The results show that replay-based methods substantially improve over memory-free regularization methods, confirming the importance of explicit memory in the Rotated-MNIST DIL setting. Among prior methods, X-DER achieves the strongest final accuracy, while L2P-R obtains the second-best average accuracy and forgetting. RAM consistently outperforms all baselines, improving final accuracy from 95.6% to 96.9%, average accuracy from 93.8% to 95.3%, and reducing forgetting from 5.0% to 4.2%. These gains indicate that even under controlled rotation shifts, actively curating replay memory through sample-wise *retain*, *repair*, and *remove* decisions provides more stable continual adaptation than static or heuristic memory updates.

G Policy Optimization and Training Dynamics

We parameterize the action policy $\pi_\omega(a \mid s)$ with a lightweight two-layer MLP policy network. The network takes the semantic drift and domain drift of each memory entry as input, uses a hidden dimension of 64 with ReLU activations, and outputs three action logits corresponding to RETAIN, REPAIR, and REMOVE. The policy network is optimized using REINFORCE (Williams, 1992). Given the epoch-level reward R, the policy gradient is

$$\nabla_\omega J(\omega) = \mathbb{E}\Big[R \sum_{i=1}^{N} \nabla_\omega \log \pi_\omega(a_i \mid s_i)\Big]. \tag{66}$$

Here, R summarizes the effect of all memory operations performed within an epoch, encouraging the policy to select RETAIN, REPAIR, and REMOVE actions that improve continual accuracy while maintaining stable and drift-aware memory updates.

We further examine the training dynamics and stability of the learned policy on **Camelyon17** in Fig. 5. The policy reward increases rapidly during the early stage and then gradually stabilizes, suggesting that the controller learns effective and stable memory-management behavior over training. The action distribution also exhibits a clear adaptation pattern: the policy initially assigns a relatively high probability to *Repair*, reflecting the need to refresh unstable or misaligned memory representations, while the probability of *Retain* increases as the memory pool becomes more reliable. Meanwhile, *Remove* remains lower but non-negligible, indicating that the policy selectively discards stale or harmful samples rather than preserving all stored data. The smooth reward curve and non-collapsed action distribution indicate that policy optimization remains stable instead of degenerating into a single dominant action. These results support our motivation that replay memory should be actively regulated through learned sample-wise decisions.

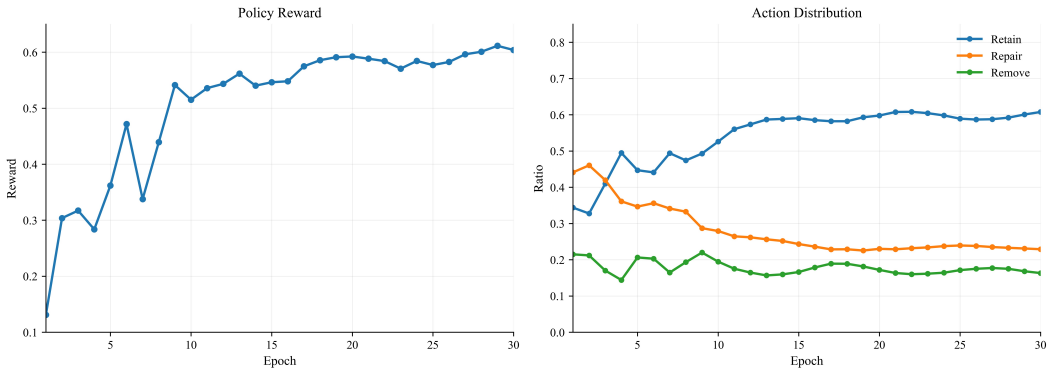


Figure 5: **Training dynamics of the learned memory policy on Camelyon17.** Left: policy reward over training epochs. Right: action distribution of *Retain*, *Repair*, and *Remove*. The reward steadily improves and stabilizes, while the policy gradually shifts toward retaining reliable memory items, while still using *Repair* and *Remove* to refresh or discard unstable memory items.

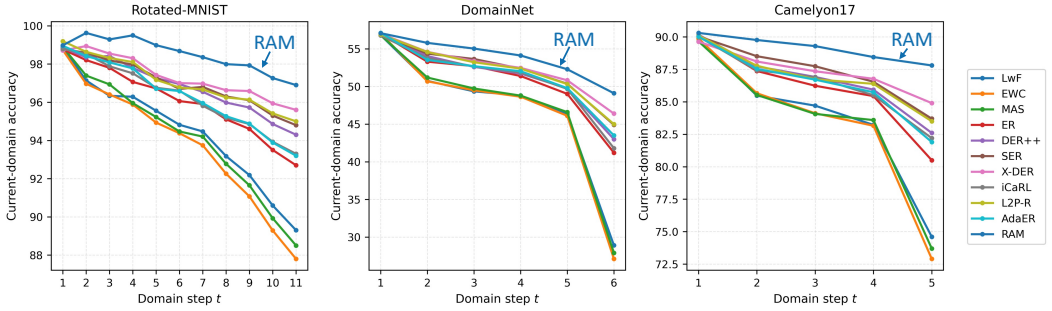


Figure 6: **Evolution of domain accuracy.** Current-domain test accuracy $e_t^{(t)}$ over sequential domains on Rotated-MNIST, DomainNet, and Camelyon17. Replay-based methods alleviate performance degradation compared to memory-free baselines, while **RAM** achieves the most stable adaptation and the strongest performance across domain steps, with the clearest gains under larger domain shifts (DomainNet and Camelyon17).

H Analysis of Domain Accuracy Evolution

Fig. 6 analyzes how current-domain accuracy $e_t^{(t)}$ evolves as domains arrive. Overall performance declines over time as the domain shift becomes more pronounced, and the separation between methods becomes clearer in later steps. Memory-free baselines suffer the fastest drop, reflecting limited adaptability under non-stationary inputs. Replay-based methods slow down this degradation, but they still exhibit noticeable drops on the most challenging domains, suggesting that a static buffer may replay samples that are less aligned with the current distribution. In contrast, **RAM** consistently maintains higher $e_t^{(t)}$ with smoother transitions, especially on DomainNet and Camelyon17, supporting the benefit of actively curating replay to better match the evolving domain distribution.

I Hyperparameter Sensitivity

We study the sensitivity of **RAM** to (i) the reward weights in the RL objective, $R = \lambda_1 r_{\text{val}} + \lambda_2 r_{\text{div}} + \lambda_3 r_{\text{align}}$, and (ii) the loss weights in training, $\mathcal{L} = \mathcal{L}_{\text{task}} + \alpha \mathcal{L}_{\text{dis}} + \gamma \mathcal{L}_{\text{replay}}$. Unless otherwise specified, we report DomainNet results (Avg/Final/Forget, %) as mean±std over 3 seeds.

Analysis. Across both sweeps, performance changes only mildly: Avg/Final fluctuate within about ∼0.4 points and Forgetting within ∼0.5 points (Tabs. 5 and 6). This suggests that **RAM** is not tightly tuned to a particular reward decomposition or a specific balance between disentanglement and replay. Intuitively, the controller receives consistent learning signals from validation improvement

Table 5: **Sensitivity to reward weights** in the RL controller on DomainNet. We vary one weight at a time around the default setting (shaded) and keep other settings unchanged.

λ_1	λ_2	λ_3	Avg↑	Final↑	Forget↓
0.8	0.2	0.5	51.0$_{\pm 0.35}$	48.9$_{\pm 0.31}$	13.6$_{\pm 0.42}$
1.2	0.2	0.5	51.1$_{\pm 0.33}$	49.0$_{\pm 0.29}$	13.5$_{\pm 0.41}$
1.0	0.1	0.5	50.9$_{\pm 0.37}$	48.8$_{\pm 0.32}$	13.7$_{\pm 0.45}$
1.0	0.2	0.5	**51.2**$_{\pm 0.44}$	**49.1**$_{\pm 0.39}$	**13.4**$_{\pm 0.48}$
1.0	0.3	0.5	51.1$_{\pm 0.36}$	49.0$_{\pm 0.33}$	13.5$_{\pm 0.44}$
1.0	0.2	0.3	50.9$_{\pm 0.39}$	48.9$_{\pm 0.35}$	13.8$_{\pm 0.50}$
1.0	0.2	0.7	51.3$_{\pm 0.34}$	49.2$_{\pm 0.30}$	13.3$_{\pm 0.46}$

Table 6: **Sensitivity to loss weights** on DomainNet. We vary α (disentanglement strength) and γ (replay strength) around the default (shaded).

α	γ	Avg↑	Final↑	Forget↓
0.2	1.0	51.0$_{\pm 0.38}$	48.9$_{\pm 0.34}$	13.6$_{\pm 0.47}$
0.8	1.0	51.1$_{\pm 0.36}$	49.0$_{\pm 0.31}$	13.5$_{\pm 0.46}$
1.0	0.5	50.8$_{\pm 0.40}$	48.7$_{\pm 0.36}$	13.9$_{\pm 0.52}$
0.5	1.0	**51.2**$_{\pm 0.44}$	**49.1**$_{\pm 0.39}$	**13.4**$_{\pm 0.48}$
0.5	1.5	51.1$_{\pm 0.37}$	49.0$_{\pm 0.33}$	13.5$_{\pm 0.49}$
0.5	2.0	51.0$_{\pm 0.39}$	48.9$_{\pm 0.35}$	13.6$_{\pm 0.50}$

and drift alignment, so moderate reweighting of $\lambda_1, \lambda_2, \lambda_3$ does not alter the learned behavior substantially. Similarly, as long as α is sufficient to preserve a stable semantic factor and γ maintains effective replay, the memory edits remain reliable under domain shift, leading to robust performance under a broad range of hyperparameter choices.

J ALGORITHM OVERVIEW

We summarize the overall training and memory-management procedure of **RAM** in Alg. 1. At a high level, **RAM** jointly learns (i) a disentangled encoder that separates semantic-related and domain-related factors, and (ii) a lightweight policy network that decides how to update stored memory items under domain shift. This algorithmic view clarifies the interaction between supervised learning, replay, and policy optimization within each domain step.

K NOTATIONS

Tab. 7 provides a grouped overview of the notation. We first define the Domain Incremental Learning setting and the sequential-domain quantities used in the generalization bound, then introduce the semantic/domain encoders and task/domain predictors. Finally, we list the symbols for the replay memory, drift-based guidance, and the RL policy and reward terms that govern retain/repair/remove operations.

Algorithm 1 RAM: RL-based Adaptive Memory for Domain Incremental Learning

Require: Domain sequence $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$; memory budget M; encoders (f_τ, f_δ); task head c_θ; domain head g_ϕ; policy π_ω; loss weights (α, γ); reward weights $(\lambda_1, \lambda_2, \lambda_3)$; EMA rate β; small ε.

Ensure: Trained predictor $h_\theta(x) = \arg\max_{y \in \mathcal{Y}} c_\theta(f_\tau(x))$ and curated memory $\mathcal{M}$.

Initialize model parameters (θ, ϕ) and policy parameters ω.
Initialize episodic memory $\mathcal{M} \leftarrow \emptyset$.
for $t = 1$ **to** T **do**
  for each training epoch on $\mathcal{D}_t$ **do**
    Sample current minibatch $B_t \sim \mathcal{D}_t$.
    Compute $(z_\tau, z_\delta) = (f_\tau(x), f_\delta(x))$ for $x \in B_t$.
    Compute task loss $\mathcal{L}_{\text{task}}$ using $c_\theta(z_\tau)$.
    Compute disentanglement loss $\mathcal{L}_{\text{dis}} = \mathcal{L}_{\text{dom}} + \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{ortho}}$.
    Set replay loss $\mathcal{L}_{\text{replay}} \leftarrow 0$.
    Set policy objective accumulator $G \leftarrow 0$.
    if $\mathcal{M} \neq \emptyset$ **then**
      Sample memory batch $\mathcal{B} = \{m_i\}_{i=1}^N \subset \mathcal{M}$.
      Each item $m_i = (x_i, y_i, z_{\tau,i}^{\text{old}}, z_{\delta,i}^{\text{old}}, d_i)$ stores features and domain index.
      for $i = 1$ **to** N **do**
        Recompute $z_{\tau,i}^{\text{new}} = f_\tau(x_i)$ and $z_{\delta,i}^{\text{new}} = f_\delta(x_i)$.
        Compute feature drift $\Delta_{\tau,i} = \frac{\|z_{\tau,i}^{\text{new}} - z_{\tau,i}^{\text{old}}\|_2}{\|z_{\tau,i}^{\text{old}}\|_2 + \varepsilon}$, $\Delta_{\delta,i} = \frac{\|z_{\delta,i}^{\text{new}} - z_{\delta,i}^{\text{old}}\|_2}{\|z_{\delta,i}^{\text{old}}\|_2 + \varepsilon}$.
        Form state $s_i = (\Delta_{\tau,i}, \Delta_{\delta,i}, d_i)$.
      end for
      Center drifts by batch medians: $u_i = \Delta_{\tau,i} - \text{median}(\{\Delta_{\tau,j}\}_{j=1}^N)$ and $v_i = \Delta_{\delta,i} - \text{median}(\{\Delta_{\delta,j}\}_{j=1}^N)$.
      for $i = 1$ **to** N **do**
        Sample action $a_i \sim \pi_\omega(\cdot \mid s_i)$ from $\mathcal{A} = \{\text{RETAIN}, \text{REPAIR}, \text{REMOVE}\}$.
        if $a_i = \text{REMOVE}$ **then**
          Delete m_i from $\mathcal{M}$.
        else
          if $a_i = \text{REPAIR}$ **then**
            Update stored features by EMA: $\bar{z}_{\tau,i} \leftarrow (1-\beta)\bar{z}_{\tau,i} + \beta z_{\tau,i}^{\text{new}}$.
            Update stored features by EMA: $\bar{z}_{\delta,i} \leftarrow (1-\beta)\bar{z}_{\delta,i} + \beta z_{\delta,i}^{\text{new}}$.
          end if
        end if
      end for
      Build replay set $\mathcal{B}_{\text{rep}} = \{(x_i, y_i) \in \mathcal{B} \mid a_i \in \{\text{RETAIN}, \text{REPAIR}\}\}$.
      if $|\mathcal{B}_{\text{rep}}| > 0$ **then**
        Compute replay loss $\mathcal{L}_{\text{replay}} = \mathbb{E}_{(x,y)\sim\mathcal{B}_{\text{rep}}}\big[\text{CE}(c_\theta(f_\tau(x)), y)\big]$.
      end if
      Update (θ, ϕ) by minimizing $\mathcal{L} = \mathcal{L}_{\text{task}} + \alpha\,\mathcal{L}_{\text{dis}} + \gamma\,\mathcal{L}_{\text{replay}}$.
      Compute validation-improvement reward r_{val}.
      Compute action-diversity reward $r_{\text{div}} = \frac{H(\bar{\pi})}{\log|\mathcal{A}|}$ where $\bar{\pi}(a) = \frac{1}{N}\sum_{i=1}^N \pi_\omega(a \mid s_i)$.
      Define drift-guided target action $a^\star(u, v)$ as follows.
      If $(u_i > 0, v_i > 0)$ then $a^\star = \text{REMOVE}$. If $(u_i > 0, v_i < 0)$ or $(u_i < 0, v_i > 0)$ then $a^\star = \text{REPAIR}$. If $(u_i < 0, v_i < 0)$ then $a^\star = \text{RETAIN}$.
      Compute alignment reward $r_{\text{align}} = \frac{1}{N}\sum_{i=1}^N \mathbf{1}[a_i = a^\star(u_i, v_i)]$.
      Aggregate reward $R = \lambda_1 r_{\text{val}} + \lambda_2 r_{\text{div}} + \lambda_3 r_{\text{align}}$.
      Update policy with REINFORCE: $G \leftarrow R\sum_{i=1}^N \nabla_\omega \log \pi_\omega(a_i \mid s_i)$, and take a gradient step on ω using G.
    else
      Update (θ, ϕ) by minimizing $\mathcal{L} = \mathcal{L}_{\text{task}} + \alpha\,\mathcal{L}_{\text{dis}}$.
    end if
  end for
  Select representative samples from $\mathcal{D}_t$ and insert into $\mathcal{M}$ as $(x, y, f_\tau(x), f_\delta(x), t)$ (kept within budget M).
end for

Table 7: Summary of notation used in this paper.

Notation	Description
Problem Setup and Sequential Domains	
$\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$	Sequence of domains in Domain Incremental Learning (DIL).
$\mathcal{Y}$	Shared label space across domains.
$\mathcal{D}$	Set of domain identities (domain index space).
$D_t = \{(x_i, y_i)\}_{i=1}^{N_t}$	Training set of domain $\mathcal{D}_t$ with N_t samples.
$x,\ y,\ d$	Input, class label, and domain identity, respectively.
P_t	Input marginal distribution of domain $\mathcal{D}_t$.
Generalization Bound Notation	
h	A predictor (hypothesis) used for classification.
$e_t(h)$	Population error of h on domain $\mathcal{D}_t$.
$d_{\mathcal{H}\Delta\mathcal{H}}(P_t, P_{t-1})$	$\mathcal{H}\Delta\mathcal{H}$-divergence between consecutive domains.
$\lambda_{t-1,t},\ \lambda^\star$	Joint optimal risk for two domains; aggregated path-wise term in the sequential bound.
$D_k \in \{k-1, k\}$	Domain indicator for the two-domain mixture at step k in the bound.
$I(Z_\tau; D_k)$	Mutual information between semantic representation and domain identity.
Disentangled Representation and Predictors	
$f_\tau,\ f_\delta$	Encoder branches extracting semantic-related and domain-related factors.
$z_\tau = f_\tau(x) \in \mathbb{R}^p$	Semantic-related representation.
$z_\delta = f_\delta(x) \in \mathbb{R}^q$	Domain-related representation.
$Z_\tau,\ Z_\delta$	Batch-wise stacked representations (matrices) of z_τ and z_δ.
$h_\theta : \mathbb{R}^p \to \Delta(\mathcal{Y})$	Semantic classifier (main task head).
$g_\phi : \mathbb{R}^q \to \Delta(\mathcal{D})$	Domain classifier used for modeling domain-specific factors.
$\mathrm{GRL}(\cdot)$	Gradient reversal layer for promoting domain invariance in z_τ.
Replay Memory and Stored Representations	
$\mathcal{M}$	Replay memory (dynamic memory pool).
M	Memory budget (maximum capacity), i.e., an upper bound on $\lvert\mathcal{M}\rvert$.
$z_\tau^{\mathrm{old}},\ z_\delta^{\mathrm{old}}$	Stored (possibly stale) semantic/domain representations in memory.
$z_\tau^{\mathrm{new}},\ z_\delta^{\mathrm{new}}$	Current representations recomputed by the latest encoder for the same x.
$B_m \subset \mathcal{M}$	Mini-batch sampled from memory for replay.
Drift Signals and Heuristic Guidance	
$\Delta_\tau,\ \Delta_\delta$	Normalized drift of semantic/domain components (stored vs. current mismatch).
$u,\ v$	Centered drifts defining the task–domain drift plane (from $\Delta_\tau, \Delta_\delta$).
$a^\star(u, v)$	Drift-based target operation (retain/repair/remove) used as a guidance signal.
RL Policy and Rewards	
$\mathcal{A} = \{\textsc{retain}, \textsc{repair}, \textsc{remove}\}$	Action space for memory operations.
$s_i,\ a_i$	State observed for a memory item and the selected action, respectively.
$\pi_\omega(a \mid s)$	Action policy parameterized by the policy network with parameters ω.
$r_{\mathrm{val}},\ r_{\mathrm{div}},\ r_{\mathrm{align}}$	Reward components: validation gain, action diversity, and drift-alignment.
$R = \lambda_1 r_{\mathrm{val}} + \lambda_2 r_{\mathrm{div}} + \lambda_3 r_{\mathrm{align}}$	Overall reward with weights $\lambda_1, \lambda_2, \lambda_3 \geq 0$.