JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 31 No. 6, pp. 29-43, June 2026
https://doi.org/10.9708/jksci.2026.31.06.029

# Mitigating Domain Shift via Domain-Invariant Representation Learning in Domain-Incremental Learning

Min Taek Lim*, Joo Hyun Lee**, Jinyong Kim***, Yu Rang Park****

*Graduate Student, Dept. of Biomedical Systems Informatics, Yonsei University, Seoul, Korea
**M.S. Student, Dept. of Biomedical Systems Informatics, Yonsei University, Seoul, Korea
***M.S. Student, Dept. of Artificial Intelligence, Yonsei University, Seoul, Korea
****Professor, Dept. of Biomedical Systems Informatics, Yonsei University, Seoul, Korea

## [Abstract]

In Domain-Incremental Learning (DIL), models must adapt to domain shifts while mitigating catastrophic forgetting. While Pre-Trained Models (PTMs) are widely used, existing strategies often bias representations toward seen domains, failing to effectively mitigate domain shift in unseen domains and inducing representation drift. To address this, we propose a dual-encoder framework that learns domain-invariant and class-discriminative representations. A domain-specific encoder extracts prototypes, distilling stable knowledge into a trainable domain-invariant encoder. Prototype alignment and weight regularization prevent drift and preserve past knowledge. Furthermore, Representation squeezing and a Gradient Reversal Layer (GRL) form compact intra-class clusters and explicitly encourage domain-invariant learning. Extensive experiments on four benchmark datasets (Office-Home, DomainNet, CORe50, and PACS) show our framework improves unseen domain average accuracy by 1-6% across datasets. Notably, on Office-Home, it achieves the highest overall performance with 86.32% Last and 86.25% Avg accuracy. It also reaches an 83.17% Unseen Avg and successfully reduces the forgetting measure to 3.83%.

▸ **Key words**: Domain-Incremental Learning, Domain-Invariant, Domain Shift, Unseen Domain, Pre-trained Models

## [요 약]

도메인 점진적 학습 (DIL)에서 모델은 치명적 망각을 완화하여 도메인 이동에 적응해야 한다. 사전 학습 모델 (PTMs)이 널리 사용되나, 기존 전략들은 관찰된 도메인에 편향되어 미관찰 도메인에 도메인 쉬프트 문제를 효과적으로 줄이지 못해 표현 표류를 유발한다. 이를 해결하고자, 도메인 불변 및 클래스 변별적 표현을 학습하는 듀얼 인코더 프레임워크를 제안한다. 도메인 특이적 인코더로 프로토타입을 추출하고 안정적인 지식을 학습 가능한 도메인 불변 인코더로 증류한다. 프로토타입 정렬과 가중치 정규화는 표류를 방지하고 과거 지식을 보존한다. 또한 표현 압축과 GRL을 통해 클래스 내 군집을 조밀하게 형성하고 도메인 불변 학습을 유도한다. 4개 벤치마크 실험 결과, 본 프레임워크는 미관찰 도메인 평균 정확도를 1~6% 향상시켰다. 특히 Office-Home에서 Last 86.32%, Avg 86.25%를 달성했으며, 미관찰 평균 정확도 83.17%, 망각 수치 3.83%를 기록하였다.

▸ **주제어**: 도메인 점진적 학습, 도메인 불변성, 도메인 쉬프트, 미관측 도메인, 사전 학습 모델

• First Author: Min Taek Lim, Corresponding Author: Yu Rang Park
*Min Taek Lim (lmtna99@yuhs.ac), Dept. of Biomedical Systems Informatics, Yonsei University
**Joo Hyun Lee (wngus6860@yuhs.ac), Dept. of Biomedical Systems Informatics, Yonsei University
***Jinyong Kim (jnyngk@yonsei.ac.kr), Dept. of Artificial Intelligence, Yonsei University
****Yu Rang Park (yurangpark@yuhs.ac), Dept. of Biomedical Systems Informatics, Yonsei University
• Received: 2026. 04. 15, Revised: 2026. 05. 15, Accepted: 2026. 05. 29.


http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

## I. Introduction

Recent advances in deep learning have achieved strong performance across various real-world applications. However, in practical scenarios, data often arrives sequentially from changing domains. This domain shift frequently occurs due to differences in data distributions across domains, such as variations in lighting conditions or background environments[1, 2]. When a model is to adapt to these distribution changes, it may overwrite previously learned knowledge in order to adapt to the current domain. This leads to catastrophic forgetting, where the model's performance on earlier domains degrades significantly[3-7].

Domain-Incremental Learning(DIL) has been proposed to address this challenge. In this setting, models must deal with domain shift caused by distribution changes, while also mitigating catastrophic forgetting as new knowledge is learned. DIL aims to adapt to new domains while preserving previously learned knowledge.

To address catastrophic forgetting in DIL, recent studies have increasingly adopted pre-trained models(PTMs)[8-16], which learn generalizable representations from large-scale data. In typical PTM-based approaches, a pre-trained feature extractor is frozen to preserve previously learned knowledge, while lightweight trainable components are incorporated to adapt to new domain distributions. Prompt-based approaches, such as Learning to Prompt(L2P)[11] and DualPrompt[12], learn task-specific prompts to adapt the frozen feature representation. Other approaches adopt domain-specific modeling, where separate classifiers or components are maintained for each domain and selectively combined during inference, as in S-Prompts[13]. In addition, recent approaches explore integrating cross-domain knowledge through alignment or prototype-based representations to improve consistency across domains, such as DUCT[14] and PINA[15].

However, these approaches share common limitations. Although they are designed to preserve performance within observed domains, the learned representations often become biased toward seen domains due to domain-specific adaptation strategies. This leads to insufficient generalization across unseen domains. In addition, although the backbone is frozen, continually updated lightweight modules can induce representation drift, causing inconsistencies between representations and classifiers and leading to performance degradation on previously learned domains. These highlight that both domain generalization and representation stability remain key challenges in DIL.

To address these challenges, we propose a framework that learns domain-invariant representations through a dual-encoder architecture while explicitly leveraging the generalization capability of PTMs. Specifically, we extract prototypes from a frozen PTM, which serves as a domain-specific encoder for prototype extraction, to leverage its generalizable knowledge. These prototypes are distilled into a domain-invariant encoder, which is trained across all domains to learn domain-invariant representations. The domain-invariant encoder, on the other hand, is optimized to learn from current data while being guided by the distilled prototype knowledge. Therefore, the proposed framework enables learning representations that are both generalizable across domains and discriminative across classes. To further enhance representation quality, we incorporate representation squeezing[17], along with domain-adversarial learning using a gradient reversal layer (GRL)[18] to encourage domain-invariant representations. In addition, weight regularization is employed to maintain consistency with previously learned knowledge. This architecture facilitates learning representations that mitigate domain shift while preserving class discriminability and maintaining performance on previously learned domains.

Our contributions are summarized as follows:

1. We propose a dual-encoder framework for DIL that separates domain-specific and domain-invariant representations, enabling domain-invariant feature learning across domains.

2. We design an alignment strategy that integrates representation squeezing, domain-adversarial learning with a GRL, and prototype alignment to mitigate domain shift and encourage domain-invariant representations.

3. Extensive experiments on four benchmark datasets demonstrate that the proposed framework achieves competitive performance compared to existing comparison methods, with improved generalization across unseen domains.

## II. Preliminaries

### 1. Related works

#### 1.1 Domain-Incremental Learning

Early approaches addressed catastrophic forgetting using rehearsal-based approaches[19-21], which store a subset of samples from previously observed domains and replay them during training. Although effective, these approaches are limited by memory constraints and privacy concerns, thereby reducing their practicality in real-world scenarios.

To overcome these limitations, recent works have focused on non-rehearsal approaches that avoid storing past data. Regularization-based approaches[22-24] constrain parameter updates to preserve previously learned knowledge, such as Elastic Weight Consolidation (EWC)[4]. However, these approaches often struggle under large domain shifts, as they rely on parameter-level constraints rather than explicitly modeling domain variations.

More recently, PTMs have emerged as a promising paradigm for DIL due to their strong generalization capability learned from large-scale data. A common strategy is to freeze the pre-trained backbone and incorporate lightweight trainable components for domain-incremental adaptation. In addition, simple PTM-based approaches, such as SimpleCIL[8], directly learn a classifier on top of frozen pre-trained features without modifying the backbone. Prompt-based approaches, such as L2P[11] and DualPrompt[12], learn task-specific prompts to adapt frozen feature representation while preserving parameter efficiency and pre-trained knowledge. Domain-specific approaches further extend this idea by maintaining separate components for each domain. For example, S-Prompts[13] learns domain-specific prompts and employs a selection strategy during inference to apply the appropriate domain-related components.

Other approaches model cross-domain relationships or stabilize representations across sequential domains. For instance, PINA[15] adopts a shared classifier to unify semantic concepts across domains, improving consistency in a common representation space. In contrast, DUCT[14] focuses on consolidation by mitigating representation drift and classifier bias during incremental updates, thereby preserving representation stability across domains.

However, these approaches remain focused on modeling relationships within observed domains, which limits their ability to generalize under unseen domain shifts.

#### 1.2 Problem Definition

In DIL, the model is trained on a sequence of domains $\{D^1, D^2, \cdots, D^T\}$ where each task corresponds to a different domain. The dataset of the $t$-th domain is denoted as $D^t = \{(x_i, y_i)\}_{i=1}^{N_t}$ where $x_i \in \mathbb{R}^d$ is an input sample and $y_i \in Y$ is its corresponding label. The label space $Y$ remains fixed across all domains, meaning that the class label $y$ is shared and consistent across all domains, while the data distribution changes over time. That is, for domains $t \neq t'$, the marginal distributions differ as $p(x|D^t) \neq p(x|D^{t'})$ where $p(x|D^t)$ denotes

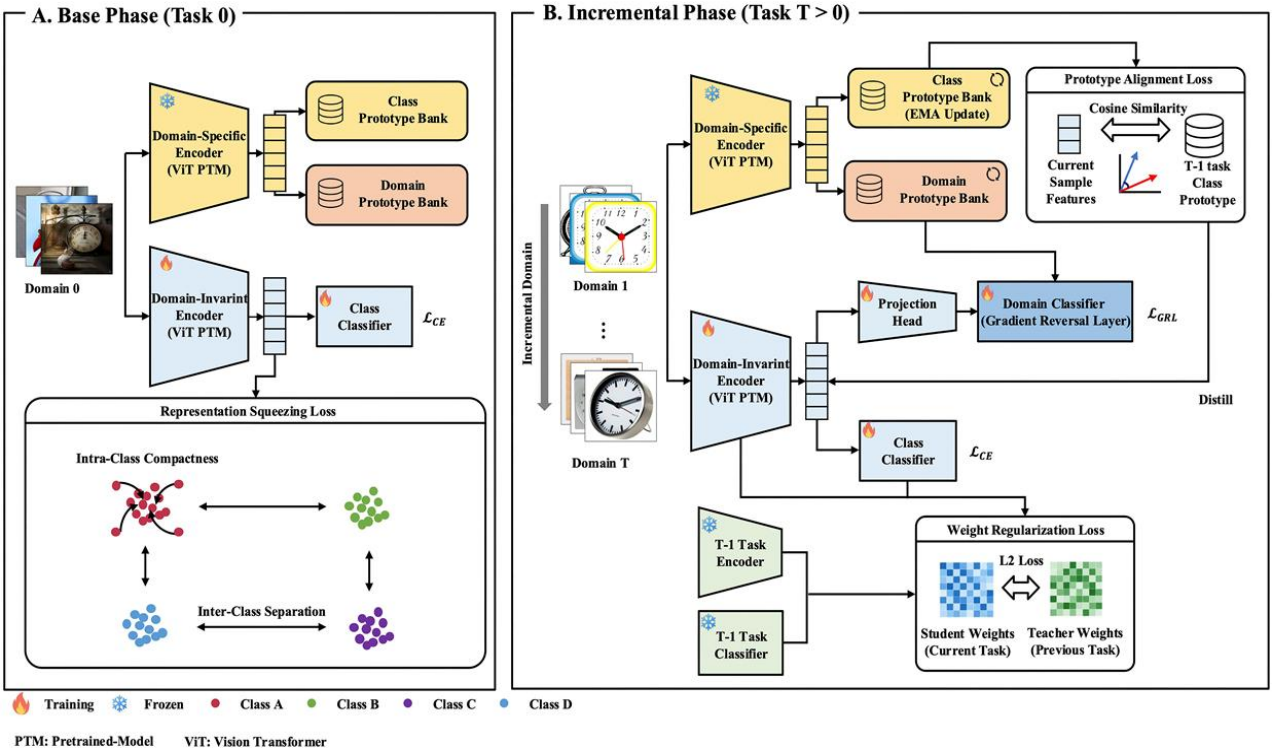



Fig. 1. Overall Framework

the marginal distribution of input $x$ in domain $t$. This discrepancy in marginal distributions is referred to as domain shift, reflecting variations in data characteristics across domains. During training, only data from the current domain is accessible, following the exemplar-free setting. The goal of DIL is to learn a model $f \in G$ that generalizes across domains while preserving previously acquired knowledge. Formally, the objective is defined as:

$$f^* = \arg\min_{f \in G} \mathbb{E}_{(x,y) \sim \cup_{t=1}^{T} D^t} [\, \| (y \neq f(x)) \,]$$

(1)

where G denotes the hypothesis space and $\| (\cdot)$ is the indicator function. This setting requires the model to handle both domain shift and catastrophic forgetting simultaneously.

## III. The Proposed Method

In this section, we propose a DIL framework that mitigates domain shift by learning domain-invariant and class-discriminative representations.

The proposed framework incorporates five key strategies. First, a dual-encoder architecture, consisting of a frozen domain-specific encoder and a trainable domain-invariant encoder, is utilized throughout the base and incremental phases to extract complementary representations. Second, class and domain prototypes are constructed and maintained using representations from the domain-specific encoder to capture class-level and domain-level knowledge across sequential domains. Third, an alignment strategy is applied to align current domain-invariant representations with stored class prototypes while simultaneously squeezing the representation space to enhance intra-class compactness. Fourth, domain-adversarial learning is performed by routing representations through a GRL to explicitly encourage the extraction of domain-invariant representation. Finally, weight regularization is applied to penalize the distance between current and previous model parameters, thereby constraining parameter updates and mitigating catastrophic forgetting. An overview of the

framework is illustrated in Fig. 1.

### 1. Overall Architecture

Our framework operates in two phases: a base phase and an incremental phase. In the base phase, the dual-encoder architecture is trained on data from the first domain of this phase. The core objective of this phase is representation squeezing, which constrains the representation space to minimize intra-class variance while maximizing inter-class margins, thereby initializing a discriminative representation space for subsequent domains.

During the incremental phase, the domain-invariant encoder and the class classifier are sequentially optimized to mitigate domain shift and catastrophic forgetting using data from each domain of this phase. This optimization process incorporates three strategies. Specifically, domain-adversarial learning with a GRL acquires information from the representations extracted by the domain-invariant encoder to learn domain-invariant representations.

Concurrently, prototype alignment and weight regularization are incorporated into the objective function to constrain the model in both the representation and parameter spaces to mitigate catastrophic forgetting.

The framework utilizes a dual-encoder architecture consisting of a domain-specific encoder $f_s$ and a domain-invariant encoder $f_{inv}$. Given an input sample $x$, the two encoders produce a domain-specific representation $z_s$ and a domain-invariant representation $z_{inv}$:

$$z_s = f_s(x),\ z_{inv} = f_{inv}(x) \tag{2}$$

Both $f_s$ and $f_{sh}$ are initialized with weights pre-trained on ImageNet. The domain-specific encoder $f_s$ is completely frozen to extract domain-specific representations. In contrast, the parameters of the domain-invariant encoder $f_{inv}$ are updated across all domains to learn domain-invariant representations. For class prediction, the domain-invariant representation $z_{inv}$ is fed into a class classifier $g$ to output the predicted label $\hat{y}$:

$$\hat{y} = g(z_{inv}) \tag{3}$$

Additionally, $z_{inv}$ is routed to a domain classifier that is adversarially trained with a GRL and incorporates a projection layer.

### 2. Prototype Representation and Update

We construct class and domain prototypes using the domain-specific representations $z_s$ extracted by the frozen encoder $f_s$. For a given domain $t$, the current class prototype $\tilde{P}_c^t$ is computed by averaging the representations of all samples belonging to class $c$:

$$\tilde{P}_c^t = \frac{1}{|D_c^t|} \sum_{x_i \in D_c^t} z_s(x_i) \tag{4}$$

where $D_c^t$ denotes the set of samples belonging to class $c$ in domain $t$. Similarly, the domain prototype $P_d^t$ is computed by averaging the representations of all samples belonging to domain $t$:

$$P_d^t = \frac{1}{|D^t|} \sum_{x_i \in D^t} z_s(x_i) \tag{5}$$

where $D^t$ represents the entire set of samples in domain $t$.

The class prototypes $P_c^t$ are sequentially updated across domains using an exponential moving average(EMA). Specifically, the class prototype for domain $t$ is updated by combining the class prototype from the previous domain $t-1$ with the current class prototype $\tilde{P}_c^t$:

$$P_c^t = \alpha P_c^{t-1} + (1-\alpha)\tilde{P}_c^t \tag{6}$$

where $\alpha \in [0, 1]$ is the momentum coefficient.

The current-domain prototype $\tilde{P}_c^t$ is computed from all samples of class $c$ in domain $t$, and the EMA update is applied once after training on domain $t$ is completed. As a result, a single prototype per class is maintained throughout training, without accumulating prototypes separately across domains. The class prototypes $P_c^t$ are used to calculate the prototype alignment loss, and the domain prototypes $P_d^t$ are stored to be incorporated into subsequent training batches for domain-adversarial learning.

## 3. Alignment Strategies

We introduce two alignment strategies: prototype alignment and representation squeezing.

### 3.1 Representation Squeezing

We apply a representation squeezing loss[17] to minimize intra-class distance and maximize inter-class margins. Given a mini-batch of representations and their corresponding labels, we define the sets of positive pairs $p$ and negative pairs $n$ for samples $i$ and $j$ as:

$$p = (i,j) | y_i = y_j \quad (7) \qquad n = (i,j) | y_i \neq y_j. \quad (8)$$

We calculate the intra-class compactness $L_{compact}$ and inter-class separation $L_{separate}$ using cosine similarity:

$$L_{compact} = \frac{1}{|p|} \sum_{(i,j)\in p} \cos(z_i, z_j) \quad (9)$$

$$L_{separate} = \frac{1}{|n|} \sum_{(i,j)\in n} \cos(z_i, z_j). \quad (10)$$

where $z_i$ and $z_j$ represent the target representations. We formulate the final representation squeezing loss as:

$$L_{RS} = (1 - L_{compact}) + \lambda(1 + L_{separate}) \quad (11)$$

where $\lambda$ is a balancing coefficient.

### 3.2 Prototype Alignment

To penalize representation deviations from the previous phase, we formulate a prototype alignment loss between the domain-invariant representation $z_{inv}$ and the class prototype. By minimizing the distance to the prototypes derived from the frozen domain-specific encoder $f_s$, this objective performs prototype-based distillation. Given an input $x$ with label $y$, we define the prototype alignment loss as the cosine distance:

$$L_{Proto} = 1 - \cos(z_{inv}(x), P_y^{t-1}) \quad (12)$$

where $P_y^{t-1}$ is the class prototype for class $y$ at domain $t-1$. This loss is minimized to explicitly update the parameters of the domain-invariant encoder $f_{inv}$. This process distills the stable representations previously captured by the domain-specific encoder $f_s$ into $f_{inv}$, thereby penalizing the drift of current representations $z_{inv}(x)$ from the historically accumulated prototype $P_y^{t-1}$.

## 4. Domain-Adversarial Learning

We apply domain-adversarial learning by routing the domain-invariant representation $z_{inv}$ through a projection layer and a GRL to a domain classifier $D$. We train $D$ to predict the domain label $d$ while simultaneously updating the domain-invariant encoder $f_{inv}$ through the GRL to extract domain-invariant representations. To compute this objective, we incorporate the stored domain prototypes into the current training batch. We formulate the domain classification loss as follows:

$$L_{GRL} = \frac{1}{|B \cup P_d|} \sum_{z_{inv}\in B\cup P_d} CE(D(R(h_{proj}(z_{inv}))), d) \quad (13)$$

where $R(\bullet)$ denotes the GRL, which acts as an identity transform during the forward pass and multiplies the gradients by a negative scalar during the backward pass, $h_{proj}$ denotes the projection

layer, $B$ denotes the current training batch, $P_d$ denotes the set of domain prototypes, $B \cup P_d$ represents their union, and $CE$ denotes the cross-entropy loss function.

### 5. Weight Regularization

We apply weight regularization to constrain the parameter updates by penalizing the L2 distance between the parameters of the current phase and those from previous phase $t-1$. This regularization ensures that the domain-invariant encoder and the class classifier preserve existing knowledge while adapting to new data. The weight regularization loss is formulated as:

$$L_{weight} = \lambda_{bb} \| \theta_{bb} - \theta_{bb}^{t-1} \|_2^2 + \lambda_{fc} \| \theta_{fc} - \theta_{fc}^{t-1} \|_2^2 \quad (14)$$

where $\theta_{bb}$ and $\theta_{fc}$ denote the parameters of the domain-invariant encoder $f_{inv}$ and the class classifier $g$, respectively. The projection head and the domain classifier are not included in this regularization. The two terms are weighted by $\lambda_{bb}$ and $\lambda_{fc}$, respectively.

### 6. Final Objective

We define the overall training objective as a weighted combination of the previously defined loss terms to simultaneously mitigate domain shift and minimize catastrophic forgetting. The total loss, $L_{total}$, is formulated as follows:

$$L_{total} = \lambda_{CE} L_{CE} + \lambda_{GRL} L_{GRL} + \lambda_{\mathrm{Proto}} L_{\mathrm{Proto}} + \lambda_{Weight} L_{Weight} + \lambda_{RS} L_{RS} \quad (15)$$

where $\lambda_{CE}$, $\lambda_{GRL}$, $\lambda_{\mathrm{Proto}}$, $\lambda_{Weight}$, and $\lambda_{RS}$ are weighting coefficients that balance the contribution of each optimization strategy. To strictly enforce the phase-specific optimization we set $\lambda_{RS} = 0$ for $t > 0$.

## IV. Experiments

### 1. Experimental Setup

#### 1.1 Datasets

We evaluate the proposed framework on four widely used domain generalization and DIL benchmarks: Office-Home[25], DomainNet[26], CORe50[2], and PACS[27]. The Office-Home dataset consists of 65 classes across four domains, namely Art, Clipart, Product, and Real-World, with a total of approximately 15,500 images. The DomainNet dataset is a large-scale benchmark containing 345 classes and six domains, including Clipart, Infograph, Painting, Quickdraw, Real, and Sketch, with over 580,000 images in total. The CORe50 dataset contains 50 classes taken under 11 different domains with varying background, pose, and illumination conditions, comprising approximately 164,000 images. The PACS dataset includes 7 classes across four domains, including Art Painting, Cartoon, Photo, and Sketch, with a total of around 9,900 images.

#### 1.2 Comparison methods

We compare the proposed framework with several incremental learning methods, including Finetune, EWC[4], SimpleCIL[8], DUCT[14], and EASE[16]. Finetune sequentially trains the model without any strategy to prevent forgetting. EWC is a representative regularization-based method. SimpleCIL, DUCT, and EASE are recent approaches designed for incremental learning scenarios and provide strong baselines for comparison. Although SimpleCIL and EASE were originally developed for class-incremental learning, we include them as PTM-based baselines, consistent with prior DIL work[14]. Since official domain-incremental implementations are unavailable, we adapt the official class-incremental implementations as follows: each domain is treated as a single task with a fixed, shared label space, SimpleCIL constructs frozen class prototypes and EASE expands a domain-specific subspace. For a fair

Table 1. Comparison with comparison methods on four benchmark datasets. The best performance is shown in **bold** and second-best is underlined.

| Methods | Office-Home | | | | DomainNet | | | | CORe50 | | | | PACS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Last (↑) | Avg (↑) | Forget (↓) | Unseen Avg (↑) | Last (↑) | Avg (↑) | Forget (↓) | Unseen Avg (↑) | Last (↑) | Avg (↑) | Forget (↓) | Unseen Avg (↑) | Last (↑) | Avg (↑) | Forget (↓) | Unseen Avg (↑) |
| Finetune | 83.69 | 83.82 | 6.67 | 80.08 | 43.44 | 50.20 | 32.79 | 34.13 | 95.89 | 96.32 | 4.49 | 92.16 | **96.73** | **97.78** | 2.0 | **70.96** |
| EWC[4] | 84.04 | 83.75 | 5.72 | 79.40 | 47.94 | 54.44 | 27.38 | 38.95 | 95.52 | 97.02 | 4.91 | 91.67 | 96.63 | 97.58 | 1.99 | 68.55 |
| SimpleCIL[8] | 81.29 | 81.47 | 3.62 | 82.06 | 52.63 | 53.69 | 11.60 | 44.84 | 71.44 | 76.73 | **0.00** | 68.03 | 86.66 | 92.87 | 2.51 | 64.56 |
| DUCT[14] | 86.26 | 83.58 | **0.01** | 80.71 | **60.38** | 57.92 | **2.93** | 46.67 | 96.71 | **98.05** | 2.50 | 92.46 | 85.04 | 92.74 | 1.27 | 53.95 |
| EASE[16] | 77.70 | 79.22 | 13.48 | 79.11 | 48.19 | 50.94 | 30.37 | 36.57 | 85.90 | 90.17 | 15.47 | 86.39 | 83.68 | 87.34 | 16.27 | 49.55 |
| OURS | **86.32** | **86.25** | 3.83 | **83.17** | 57.91 | **62.62** | 13.08 | **52.44** | **97.97** | 98.03 | 2.04 | **93.59** | 96.60 | 97.53 | **1.20** | 70.44 |

comparison, all comparison ⊑methods are implemented without memory replay and are trained under the exact same configurations as our proposed method, including the same ViT backbone, number of epochs, batch size, and optimizer budget.

### 1.3 Implementation Details

We adopt a DIL setting in which each domain is treated as a separate incremental task and learned sequentially. For the Office-Home dataset, the training order is Art, Clipart, Product, Real-World. For DomainNet, the domains are learned in the order of Clipart, Infograph, Painting, Quickdraw, Real, Sketch. For CORe50, we use the standard sequential setting with 11 incremental domains. For PACS, the domain order is Art Painting, Cartoon, Photo, Sketch. Following standard protocols in DIL[14], we adopt a single fixed domain order for the experiments.

We use ViT-B/16 model pre-trained on ImageNet-21K as the backbone network. The model is trained using the SGD optimizer with a batch size of 64 for 15 epochs. The initial learning rate is set to 0.1, and the weight decay is 0.0005. A sensitivity analysis on the learning rate is provided in Appendix Table 7. The loss weights are determined via a grid search. A sensitivity analysis on the loss weights, together with their search ranges, is provided in Appendix Table 8. The cross-entropy loss weight $\lambda_{CE}$ is set to 0.3, the domain-adversarial loss weight $\lambda_{GRL}$ is set to 0.3, the prototype alignment loss weight $\lambda_{\mathrm{Proto}}$ is set to 0.1, and the representation squeezing loss weight $\lambda_{RS}$ is set to 0.01. For weight regularization, we use $\lambda_{bb}$ = 0.1 and $\lambda_{fc}$= 0.001. The EMA momentum is set to 0.9. For data partitioning, we follow dataset-specific protocols summarized in Appendix Table 6. All four datasets adopt a 7:3 train-test ratio. DomainNet provides official train-test splits, which we use directly. For Office-Home, CORe50, and PACS, no official split is available, so we construct a 7:3 train-test split with a fixed random seed. In these three cases, the split is stratified by class within each domain. Specifically, the 7:3 ratio is preserved for every class inside each domain, so that both the per-domain class distribution and the domain boundaries are strictly maintained. All experiments are conducted on an NVIDIA A40 GPU. Importantly, at each task $t$, only the training set of the current domain $D^t$ is accessible to the model. Data from future domains $\mathrm{D}^{\mathrm{t+1}}, \cdots, \mathrm{D}^{\mathrm{T}}$ is never used during training, prototype computation or update, domain-adversarial batch construction, or hyperparameter selection.

### 1.4 Evaluation

We evaluate performance using classification accuracy across incremental domains, following standard protocols in incremental learning[7, 28]. Specifically, we report the following metrics. The *Last* metric denotes the final-task accuracy averaged over all seen domains. The *Avg* metric represents the average accuracy over all domains

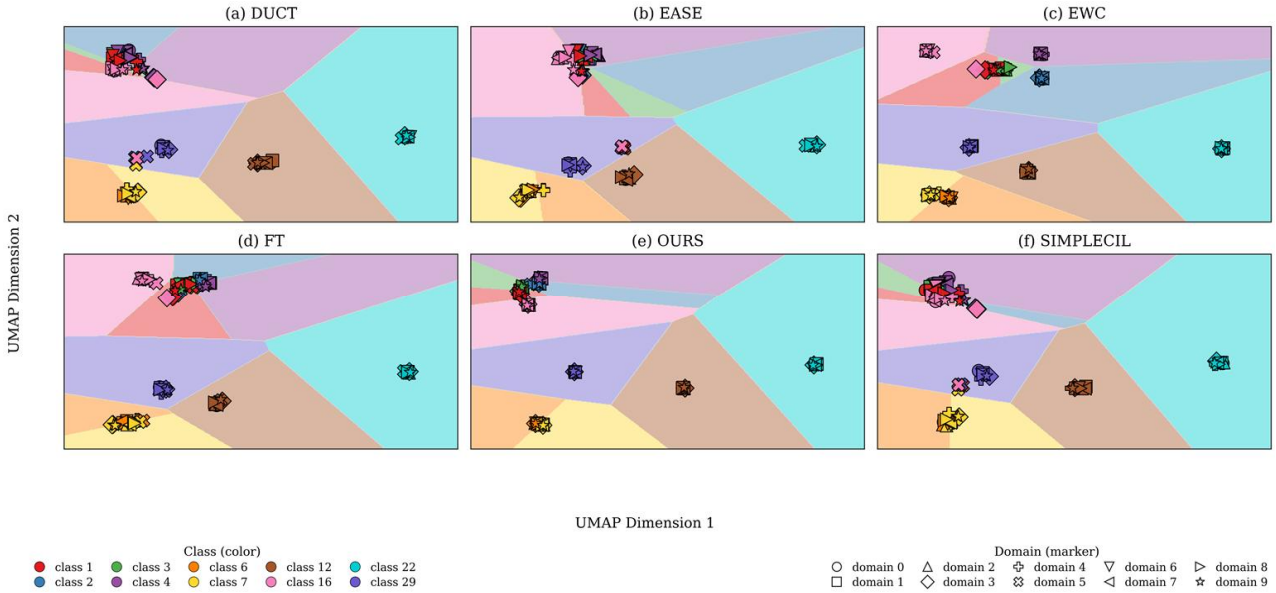



Fig. 2. Visualization of class-domain centroids across different frameworks on the CORe50 dataset.

across all training tasks. The *Forget* metric quantifies the performance degradation from the best accuracy to the final accuracy. In our domain-incremental setting, the domains are learned in a fixed sequence, and the domains that have not yet been encountered at a given task $t$ are referred to as unseen domains. Additionally, we report the *Unseen Avg* metric, which measures the average accuracy on these unseen domains. Specifically, at each task $t$, the model $M_t$ trained on $\mathrm{D}^1, \cdots, \mathrm{D}^t$is evaluated on the test set of every unseen domain $\mathrm{D}^{t+1}, \cdots, \mathrm{D}^T$ in a zero-shot manner, without any further adaptation or parameter update. The per-task unseen accuracy is computed as the mean accuracy over all unseen domains at task $t$, and the final *Unseen Avg* is obtained by averaging across tasks $t = 1, \cdots, T-1$. The final task $t = T$ is excluded since no unseen domain remains. The complete procedure is summarized in Appendix Algorithm 1.

## 2. Experimental results

### 2.1 Overall Performance

Table 1 presents overall performance across all benchmark datasets. The proposed framework outperforms the comparison methods in most cases, particularly achieving the highest results on the Office-Home dataset with 86.32 on *Last* and 86.25 on *Avg.* Detailed visualizations of these performance trends, including forget metrics and the stage-wise mean accuracy on seen domains, are provided in Appendix Figs. 5-8. In these figures, the accuracy at each incremental task is averaged over all domains learned so far. While some comparison methods achieve comparable results on specific datasets, our framework consistently demonstrates the highest *Unseen Avg* on Office-Home, DomainNet, and CORe50, confirming its robust generalization capability. To further examine robustness, we evaluate the sensitivity of each method to domain ordering on Office-Home, as reported in Appendix Table 9 and 10. Across four different domain orders, the proposed framework achieves a mean Last accuracy of 86.11% and a mean Average accuracy of 87.45%. In addition, the proposed framework shows inference cost identical to the comparison methods except for EASE, with a detailed comparison provided in Appendix B.

### 2.2 Generalization to Unseen Domains

To evaluate generalization to unseen domains, we report the *Unseen Avg* performance in Table 1. and Fig. 3. As shown in the figure, the proposed

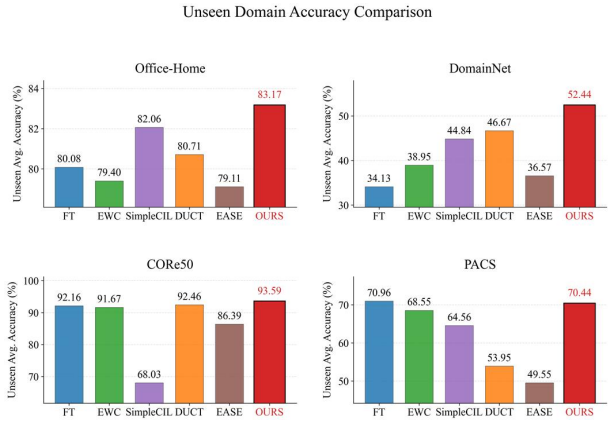



Fig. 3. Unseen Domain Accuracy Comparison.

framework consistently achieves the highest performance across most datasets, including Office-Home, DomainNet, and CORe50. Specifically, the framework improves performance on unseen domains by approximately 1% to 6% compared to comparison methods. While several baselines achieve competitive performance on seen domains, their performance degrades significantly when evaluated on unseen domains. In contrast, the proposed framework maintains stable performance across unseen domains.

### 2.3 Visualization

We visualize the learned representation space by extracting class-domain centroids from the representations, as shown in Fig. 2. Each point represents a centroid corresponding to a specific class and domain, where colors denote class labels and markers indicate domains. Ideally, the representation space should be partitioned into evenly distributed and distinct decision regions, where centroids of the same class are tightly clustered regardless of their domain.

However, as shown in the figure, the comparison methods exhibit significant structural limitations. Specifically, in DUCT (a), EASE (b), and SimpleCIL (f), the centroids for classes 1, 2, 3, and 4 exhibit severe overlap, failing to achieve sufficient spatial separation. Consequently, the representation space is not effectively partitioned, resulting in indistinct decision boundaries.

In contrast, our proposed framework (e) exhibits a more structured representation distribution. The overlap among classes 1, 2, 3, and 4 is reduced, allowing each class to form a distinct decision region with minimal interference. Furthermore, same-class centroids form compact clusters across domains, whereas they are widely scattered in comparison methods. These qualitative observations are quantitatively validated in Appendix Table 4.

Table 2. Ablation Study on Office-Home dataset.

| Methods | Last (↑) | Avg (↑) | Forget (↓) | Unseen Avg (↑) |
|---|---|---|---|---|
| Baseline (w/o all) | 85.61 | 85.57 | 4.53 | 82.54 |
| + GRL | 85.84 | 85.55 | 4.43 | 82.55 |
| + Prototype Alignment | 85.38 | 85.64 | 4.98 | 82.71 |
| + Weight Regularization | 85.88 | 85.90 | 3.98 | 83.10 |
| + Representation Squeezing (FULL) | **86.32** | **86.25** | **3.83** | **83.17** |

### 2.4 Ablation Study

This experiment analyzes the contribution of each component within the proposed framework. As shown in Table 2, the performance progressively improves as each component is integrated. The baseline model obtains a *Last* of 85.61 and an *Unseen Avg* of 82.54. Incorporating GRL leads to a slight improvement, increasing *Last* to 85.84 and *Unseen Avg* to 82.55. The addition of prototype alignment improves *Unseen Avg* to 82.71, indicating the alignment of class-level features across domains. Applying weight regularization further increases *Last* to 85.88 and an *Unseen Avg* to 83.10, while reducing forgetting from 4.93 to 3.98, confirming its role in preserving previously learned knowledge. Finally, integrating representation squeezing achieves the highest performance, reaching 86.32 on *Last*, 83.17 on *Unseen Avg* and reducing *Forget* to 3.83. These results indicate that each component contributes to the overall performance gains and generalization capability.

In addition, we examine the effect of restricting representation squeezing to the base phase. As shown in Table 3, applying representation squeezing at all stages achieves a Last of 84.66, comparable to 84.70 obtained when it is restricted to the base phase, but increases the Forget from 3.71 to 4.16.

Table 3. Effect of restricting representation squeezing to the base phase, evaluated on Office-Home

| Setting | Last | Avg | Forget |
|---|---|---|---|
| t = 0 only | 84.70 | 83.00 | 3.71 |
| all stage | 84.66 | 83.09 | 4.16 |

## V. Conclusion and Future Work

In this paper, we propose a DIL framework that learns domain-invariant and class-discriminative representations to mitigate domain shift. To effectively address domain shifts, the proposed framework explicitly encourages domain-invariant learning through a GRL and prototype-based alignment. This architecture facilitates the distillation of stable knowledge from a domain-specific encoder into a trainable domain-invariant encoder, while utilizing representation squeezing to ensure class-level compactness and discriminability. Extensive experiments on four benchmark datasets demonstrate that the proposed framework achieves competitive performance in DIL while maintaining strong generalization across unseen domains. Furthermore, representation space analysis reveals that our framework produces highly discriminative representation spaces where samples from the same class are consistently grouped across different domains, ensuring superior cross-domain consistency. Future research will focus on extending this framework to real-world applications with more complex domain variations, such as medical imaging, to further validate its practical utility.

## Appendices

### A. Comparison of metrics

#### A.1 Forgetting Measure

Fig. 4 presents the *Forget* metric across different datasets, including Office-Home, DomainNet, CORe50, and PACS. The *Forget* metric quantifies the performance degradation on previously learned domains after sequential training. As shown in the figure, the proposed framework achieves lower forgetting compared to the comparison methods on CORe50 and PACS. On Office-Home and DomainNet, although some comparison methods exhibit slightly lower forgetting, the proposed framework still maintains comparable forgetting values. These results indicate that the proposed framework reduces catastrophic forgetting while sequentially adapting to new domains.

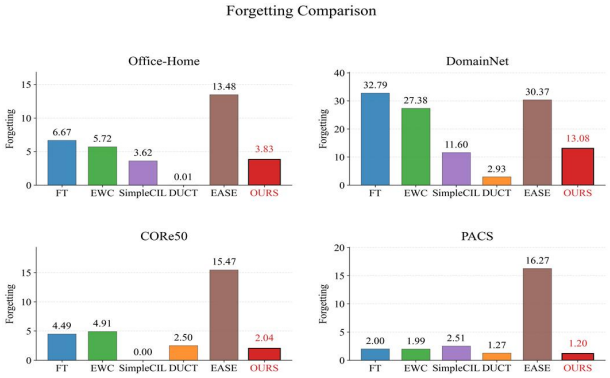



Fig. 4. Comparison of forget metrics.

#### A.2 Representation Metrics and Accuracy Matrix

Table 4 presents the representation metrics across different datasets. Class-pairwise Maximum Mean Discrepancy (MMD) measures the distribution discrepancy between representations for each class across different domains. Intra-class compactness evaluates how closely samples of the same class are clustered, while inter-class separation measures the distance between different classes. Finally, prototype distance quantifies the spatial distance between representations and their corresponding class prototypes.

The proposed framework consistently achieves

Table 4. Comparison of Representation Metrics. Lower values(↓) are better for Class-pairwise MMD, Prototype distance, and Intra-class, while higher(↑) values are better for Inter-class.

| Datasets | Metrics | OURS | FT | EWC[4] | SimpleCIL [8] | DUCT[14] | EASE[16] |
|---|---|---|---|---|---|---|---|
| OfficeHome | Class-Pairwise MMD (↓) | 0.1045 | 0.0798 | 0.0790 | 0.0948 | 0.0889 | 0.0881 |
| | Prototype distance (↓) | 0.3315 | 0.4381 | 0.4299 | 0.4620 | 0.4968 | 0.5151 |
| | Intra-class (Compactness) (↓) | 0.2720 | 0.4862 | 0.4749 | 0.4504 | 0.5398 | 0.5916 |
| | Inter-Class (Separation) (↑) | 0.9029 | 0.9772 | 0.9819 | 1.0213 | 0.8814 | 0.8242 |
| DomainNet | Class-Pairwise MMD (↓) | 0.2435 | 0.1874 | 0.1754 | 0.2828 | 0.2507 | 0.1884 |
| | Prototype distance (↓) | 0.4137 | 0.8092 | 0.7892 | 0.8372 | 0.8512 | 0.8479 |
| | Intra-class (Compactness) (↓) | 0.1860 | 0.5732 | 0.5614 | 0.4909 | 0.5307 | 0.5989 |
| | Inter-Class (Separation) (↑) | 0.5481 | 0.7414 | 0.7635 | 0.5537 | 0.7491 | 0.7165 |
| CORe50 | Class-Pairwise MMD (↓) | 0.3527 | 0.3382 | 0.3436 | 0.3497 | 0.3070 | 0.3378 |
| | Prototype distance (↓) | 0.1815 | 0.6318 | 0.4856 | 0.6600 | 0.6038 | 0.7454 |
| | Intra-class (Compactness) (↓) | 0.0424 | 0.4011 | 0.2738 | 0.3996 | 0.3866 | 0.5177 |
| | Inter-Class (Separation) (↑) | 0.8026 | 1.0087 | 1.1612 | 0.6523 | 0.9524 | 0.8129 |
| PACS | Class-Pairwise MMD (↓) | 0.3428 | 0.3604 | 0.3317 | 0.3540 | 0.4486 | 0.3769 |
| | Prototype distance (↓) | 0.3896 | 0.6338 | 0.6006 | 0.6775 | 0.7173 | 0.7371 |
| | Intra-class (Compactness) (↓) | 0.2190 | 0.4431 | 0.4501 | 0.4407 | 0.2557 | 0.3807 |
| | Inter-Class (Separation) (↑) | 0.7130 | 0.7742 | 0.8406 | 0.7584 | 0.1970 | 0.5364 |

lower intra-class compactness compared to other comparison methods, indicating that samples within the same class establish more compact clusters in the representation space. Such compact clustering across samples from different domains demonstrates that domain-specific variations are effectively reduced within each class. In terms of prototype distance, the proposed framework shows competitive values on several datasets, indicating effective alignment of class-level representations. However, the framework does not always achieve the best performance in class-pairwise MMD and inter-class separation. In particular, inter-class separation tends to be lower compared to some baselines, indicating that the distances between different classes are relatively smaller. Overall, these results indicate that the proposed framework forms compact intra-class clusters across domains while maintaining the reported inter-class distances.

Fig. 5-8 present the stage-wise average accuracy on seen domains for Office-Home, DomainNet, CORe50, and PACS. At each incremental task, the accuracy is averaged over all domains learned so far, illustrating the performance trend of each method.

## B. Computational Cost

We compare the computational cost of all methods on Office-Home. Table 5 reports the number of parameters, FLOPs, inference speed at a batch size of 32, and the cumulative training time over all four domains. All methods use the same backbone. The proposed framework uses only the domain-invariant encoder and the class classifier at inference, so its parameter count, FLOPs, and inference speed are identical to those of comparison methods. EASE, however, additionally expands a subspace for each domain, requiring 4.99 times more FLOPs and 2.46 times longer inference time. In terms of training time, the proposed framework takes 47.0 minutes, which is longer than the comparison methods except for EASE, while the inference cost remains unchanged.

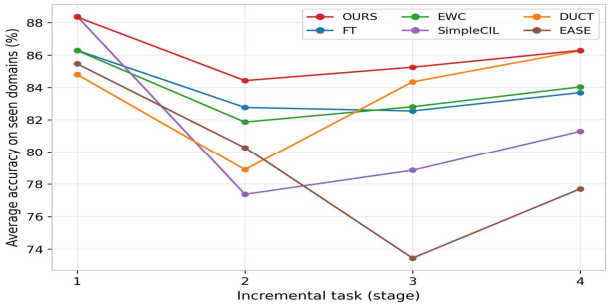



Fig. 5. Stage-wise average accuracy on seen domains for Office-Home.

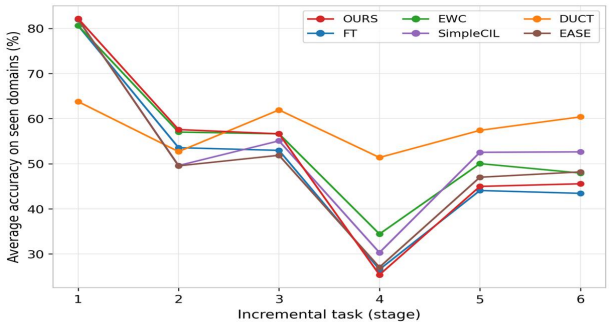



Fig. 6. Stage-wise average accuracy on seen domains for DomainNet.

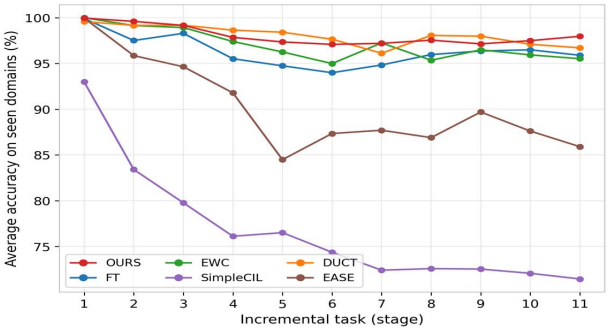



Fig. 7. Stage-wise average accuracy on seen domains for CORe50.

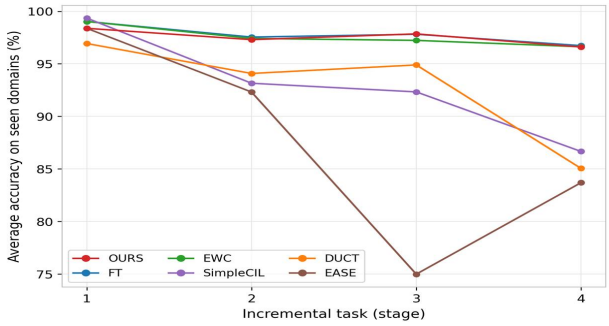



Fig. 8. Stage-wise average accuracy on seen domains for PACS

## Algorithm 1. Unseen Avg Computation

**Algorithm 1:** Unseen Avg Computation

**Input:** Trained models $\{M_1, \ldots, M_{T-1}\}$, domains $\{D^1, \ldots, D^T\}$
**Output:** Unseen Avg
**for** $t = 1$ **to** $T-1$ **do**
  $\mathcal{U}_t \leftarrow \{D^{t+1}, \ldots, D^T\}$ ; // future (unseen) domains at task $t$
  **for** *each* $D^u \in \mathcal{U}_t$ **do**
    $a_{t,u} \leftarrow \text{Accuracy}(M_t, D^u)$ ; // zero-shot evaluation
  $\text{Acc}_{\text{unseen}}(t) \leftarrow \frac{1}{T-t} \sum_{u=t+1}^{T} a_{t,u}$;
Unseen Avg $\leftarrow \frac{1}{T-1} \sum_{t=1}^{T-1} \text{Acc}_{\text{unseen}}(t)$;
**return** Unseen Avg

Table 5. Computational cost on Office-Home.

| Method | Params (M) | FLOPs (G) | Inference (ms/batch) | Training (min) |
|---|---|---|---|---|
| Finetune | 85.65 | 16.86 | 193.56 | 35.0 |
| EWC[4] | 85.65 | 16.86 | 182.66 | 30.6 |
| SimpleCIL[8] | 85.65 | 16.86 | 207.51 | 5.4 |
| DUCT[14] | 85.65 | 16.86 | 180.04 | 32.3 |
| EASE[16] | 87.29 | 84.08 | 436.65 | 50.6 |
| OURS | 85.65 | 16.86 | 178.35 | 47.0 |

Table 6. Data Split Protocol

| Dataset | Official Split | Train-Test Ratio | Stratification | Seed |
|---|---|---|---|---|
| Office-Home | X | 7:3 (custom) | per class within each domain | 42 |
| DomainNet | O | 7:3 (official) | Per official split | - |
| CORe50 | X | 7:3 (custom) | per class within each domain | 42 |
| PACS | X | 7:3 (custom) | per class within each domain | 42 |

Table 7. Performance on Office-Home under different initial learning rates. The selected value (0.1) is shown in bold.

| LR | Last | Avg | Forget |
|---|---|---|---|
| 0.001 | 38.72 | 18.45 | 0.00 |
| 0.01 | 83.04 | 73.06 | 1.79 |
| 0.03 | 84.84 | 82.81 | 3.31 |
| **0.1** | **86.32** | **86.25** | **3.83** |
| 0.3 | 83.50 | 81.65 | 4.11 |
| 1.0 | 78.80 | 73.55 | 2.37 |

Table 8. Loss weight sensitivity on Office-Home. Each cell reports Last accuracy when the given weight is varied with all others fixed. $\lambda_{fc}$ uses a separate range.

| Loss weight | 0.01 | 0.1 | 0.3 | 1.0 | Selected |
|---|---|---|---|---|---|
| $\lambda_{CE}$ | 79.59 | 84.45 | 86.32 | 84.17 | 0.3 |
| $\lambda_{GRL}$ | 86.25 | 85.72 | 86.32 | 85.33 | 0.3 |
| $\lambda_{\text{Proto}}$ | 86.43 | 86.32 | 85.57 | 86.48 | 0.1 |
| $\lambda_{RS}$ | 86.32 | 85.88 | 86.17 | 86.06 | 0.01 |
| $\lambda_{bb}$ | 85.91 | 86.32 | 85.73 | 84.88 | 0.1 |
| Loss weight | 0.0001 | 0.001 | 0.01 | 0.1 | Selected |
| $\lambda_{fc}$ | 85.21 | 86.32 | 86.16 | 85.84 | 0.001 |

Table 9. Last accuracy on Office-Home under four domain orders (A: Art, C: Clipart, P: Product, R: Real-World).

| Methods | ACPR | PACR | RACP | RAPC | mean±std |
|---|---|---|---|---|---|
| Finetune | 83.69 | 84.05 | 83.26 | 83.83 | 83.71±0.33 |
| EWC[4] | 84.04 | 84.95 | 83.51 | 82.84 | 83.84±0.89 |
| SimpleCIL[8] | 81.29 | 79.78 | 78.52 | 77.71 | 79.33±1.56 |
| DUCT[14] | 86.26 | 85.83 | 86.31 | 86.06 | 86.12±0.22 |
| EASE[16] | 77.70 | 77.95 | 76.77 | 80.60 | 78.26±1.64 |
| OURS | 86.32 | 85.87 | 86.30 | 85.97 | 86.11±0.23 |

Table 10. Average accuracy on Office-Home under four domain orders (A: Art, C: Clipart, P: Product, R: Real-World).

| Methods | ACPR | PACR | RACP | RAPC | mean±std |
|---|---|---|---|---|---|
| Finetune | 83.82 | 86.57 | 84.60 | 86.06 | 85.26±1.27 |
| EWC[4] | 83.75 | 87.58 | 84.92 | 85.76 | 85.50±1.61 |
| SimpleCIL[8] | 81.50 | 84.32 | 82.83 | 85.18 | 83.46±1.63 |
| DUCT[14] | 83.58 | 88.25 | 86.75 | 88.41 | 86.75±2.24 |
| EASE[16] | 79.55 | 83.63 | 83.33 | 85.02 | 82.88±2.34 |
| OURS | 86.25 | 88.27 | 87.02 | 88.26 | 87.45±0.99 |

## ACKNOWLEDGEMENT

This research was supported by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (No. P0023675, HRD Program for Industrial Innovation)

## REFERENCES

[1] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains,” European Conference on Computer Vision (ECCV), pp. 213–226, Heraklion, Greece, Sept. 2010. DOI: 10.1007/978-3-642-15561-1_16.

[2] L. Lomonaco and D. Maltoni, “CORe50: a new dataset and benchmark for continuous object recognition,” Conference on Robot Learning (CoRL), pp. 17–26, Mountain View, USA, Nov. 2017. DOI: 10.48550/arXiv.1705.03550.

[3] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” Psychology of Learning and Motivation, vol. 24, pp. 109–165, 1989. DOI: 10.1016/S0079-7421(08)60536-8.

[4] J. Kirkpatrick et al., “Overcoming catastrophic forgetting in neural networks,” Proceedings of the National Academy of Sciences, vol. 114, no. 13, pp. 3521–3526, Mar. 2017. DOI: 10.1073/pnas.1611835114.

[5] R. M. French, “Catastrophic forgetting in connectionist networks,” Trends in Cognitive Sciences, vol. 3, no. 4, pp. 128–135, Apr. 1999. DOI: 10.1016/S1364-6613(99)01294-2.

[6] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” Neural Networks, vol. 113, pp. 54–71, May 2019. DOI: 10.1016/j.neunet.2019.01.012.

[7] M. De Lange et al., “A continual learning survey: Defying forgetting in classification tasks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 7, pp. 3366–3385, Jul. 2021. DOI: 10.1109/TPAMI.2021.3057446.

[8] D.-W. Zhou, Z.-W. Cai, H.-J. Ye, D.-C. Zhan, and Z. Liu, “Revisiting class-incremental learning with pretrained models: Generalizability and adaptivity are all you need,” International Journal of Computer Vision, vol. 133, pp. 1012–1032, 2025. DOI: 10.1007/s11263-025-02045-x.

[9] J. S. Smith et al., “CODA-Prompt: COntinual Decomposed Attention-based Prompting for Rehearsal-Free Continual Learning,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11909–11919, Vancouver, Canada, Jun. 2023. DOI: 10.1109/CVPR52729.2023.01146.

[10] G. Zhang et al., “SLCA: Slow learner with classifier alignment for continual learning on a pre-trained model,” Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 19148–19158, Paris, France, Oct. 2023. DOI: 10.1109/ICCV51070.2023.01757.

[11] Z. Wang et al., “Learning to Prompt for Continual Learning,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11388–11397, New Orleans, USA, Jun. 2022. DOI: 10.1109/CVPR52688.2022.01110.

[12] Z. Wang et al., “DualPrompt: Complementary Prompting for Continual Learning,” European Conference on Computer Vision (ECCV), pp. 631–648, Tel Aviv, Israel, Oct. 2022. DOI: 10.1007/978-3-031-20074-8_37.

[13] Y. Wang, Z. Huang, and X. Hong, “S-prompts learning with pre-trained transformers: An Occam’s razor for domain incremental learning,” Advances in Neural Information Processing Systems, vol. 35, pp. 5682–5695, 2022.

[14] D.-W. Zhou et al., “Dual consolidation for pre-trained model-based domain-incremental learning,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025.

[15] Q. Wang et al., “Non-exemplar domain incremental learning via cross-domain concept integration,” European Conference on Computer Vision (ECCV), 2024.

[16] D.-W. Zhou, H.-L. Sun, H.-J. Ye, and D.-C. Zhan, “Expandable subspace ensemble for pre-trained model-based class-incremental learning,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024.

[17] W. Shi and M. Ye, “Prospective representation learning for non-exemplar class-incremental learning,” Advances in Neural Information Processing Systems, vol. 37, pp. 995–1018, 2024.

[18] Y. Ganin et al., “Domain-adversarial training of neural networks,” Journal of Machine Learning Research, vol. 17, no. 1, pp. 2096–2130, 2016. DOI: 10.5555/2946645.2946704.

[19] S. A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “iCaRL: Incremental classifier and representation learning,”

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2001-2010, Honolulu, USA, Jul. 2017. DOI: 10.1109/CVPR.2017.715

[20] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," Advances in Neural Information Processing Systems, vol. 30, pp. 6372-6382, Long Beach, USA, Dec. 2017.

[21] Z. Li and D. Hoiem, "Learning without forgetting," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 12, pp. 2935-2947, Dec. 2018. DOI: 10.1109/TPAMI.2017.2771331.

[22] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," Advances in Neural Information Processing Systems, vol. 33, pp. 15920-15930, virtual, Dec. 2020.

[23] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," European Conference on Computer Vision (ECCV), pp. 139-154, Munich, Germany, Sep. 2018. DOI: 10.1007/978-3-030-01234-2_9

[24] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," International Conference on Machine Learning (ICML), pp. 3987-3995, Sydney, Australia, Aug. 2017. DOI: 10.48550/arXiv.1703.04200

[25] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5018-5027, Honolulu, USA, Jul. 2017. DOI: 10.1109/CVPR.2017.535

[26] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1406-1415, Seoul, Korea, Oct. 2019. DOI: 10.1109/ICCV.2019.00149

[27] D. Li, Y. Yang, Y. Song, and T. M. Hospedales, "Deeper, broader and artier domain generalization," IEEE International Conference on Computer Vision (ICCV), pp. 5542-5550, Venice, Italy, Oct. 2017. DOI: 10.1109/ICCV.2017.595

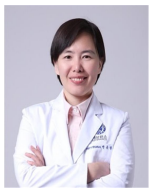
[28] A. Chaudhry, P. K. Dokania, T. Ajanthan, and M. Okatani, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," European Conference on Computer Vision (ECCV), pp. 532-547, Munich, Germany, Sep. 2018. DOI: 10.1007/978-3-030-01216-8_33

## Authors

Min Taek Lim received the B.S. degrees in Software from Sunmoon University, Korea, in 2024. He is currently pursuing the M.S. degree in the Department of Biomedical System Informatics, Yonsei University. His research interests includes continual learning, computer vision, and digital healthcare.

Joo Hyun Lee received the B.S. degree in Nursing from Seoul Women's Colleage of Nursing, Seoul, Korea, in 2013, respectively, and the M.S. degree in Biomedical Systems Informatics from Yonsei University, Seoul, in 2024. She is currently pursing the Ph.D. degree in Biomedical System Informatics at Yonsei University. Her research interests include medical AI, federated learning in healthcare, digital healthcare, and health equity in low- and middle-income countries.

Jinyong Kim received the B.B.A. degree in business from Ajou University, Korea, in 2021, and M.S. degree in digital analytics from Yonsei Univiersity, Korea. He is currently pursuing the Ph.D. degree in artificial intelligence at Yonsei university, Korea. His research interests interest includes the federated learning and video understanding, and AI application for medical field.



Yu Rang Park received the M.S. and Ph.D. degrees in Molecular and Genomic Medicine from Seoul National University, Korea, in 2006 and 2012. She is now a professor at the Department of Biomedical Systems Informatics at Yonsei University, Seoul, Korea. Her research interests focus on Medical informatics, Standardization, Distributed computing.