

Domain Generalizable Continual Learning

Hongwei Yan^{1,3}, Guanglong Sun^{1,3}, Zhiqi Kang⁴, Yi Zhong^{1,3},
Liyuan Wang^{2*}

¹School of Life Sciences, IDG/McGovern Institute for Brain Research,
Tsinghua University, Beijing, China.

²Department of Psychological and Cognitive Sciences, Tsinghua
University, Beijing, China.

³Tsinghua-Peking Center for Life Sciences, Beijing, China.

⁴Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, Grenoble,
France.

*Corresponding author(s). E-mail(s): wly19@tsinghua.org.cn;
Contributing authors: yanhw22@mails.tsinghua.edu.cn;
sgl23@mails.tsinghua.edu.cn; zhiqi.kang@inria.fr;
zhongyithu@tsinghua.edu.cn;

Abstract

To adapt effectively to dynamic real-world environments, intelligent systems must continually acquire new skills while generalizing them to diverse, unseen scenarios. Here, we introduce a novel and realistic setting named domain generalizable continual learning (DGCL): a model learns sequential tasks with each involving a single domain, aiming to perform well across all encountered tasks and domains. This setting poses unique challenges in acquiring, retaining, and leveraging both semantic- and domain-relevant information for robust generalization. Although state-of-the-art continual learning (CL) methods have employed pre-trained models (PTMs) to enhance task-specific generalization, they typically assume identical training and testing domains for each task and therefore perform poorly in DGCL. To this end, we propose adaptive Domain Transformation (DoT), an innovative PTMs-based approach tailored to DGCL. Inspired by the distributed-plus-hub theory of the human brain, DoT disentangles semantic- and domain-relevant information in representation learning, and adaptively transforms task representations across various domains for output alignment, ensuring balanced and generalized predictions. DoT serves as a plug-in strategy that greatly facilitates state-of-the-art CL baselines under both full parameter tuning and parameter-efficient tuning paradigms in DGCL, validated by extensive

experiments. Also, DoT is shown to accumulate domain-generalizable knowledge from DGCL, and ensure resource efficiency with a lightweight implementation.

Keywords: domain generalization, continual learning, catastrophic forgetting, knowledge transfer, pre-trained models

1 Introduction

Human learning is characterized by the ability to learn only one task at a time in a single scenario, yet remember the underlying knowledge and generalize it well to other scenarios [1, 2]. This remarkable adaptability underlies human intelligence in dynamic environments. Naturally, we expect artificial intelligence (AI) to adapt in a similar way. For example, a housekeeping robot must continuously keep up with emerging skill requirements, while adapting to varied user preferences and room layouts. Likewise, a healthcare diagnostic AI must continually incorporate medical imaging data from individual hospitals to improve disease identification, while generalizing to images from other hospitals with varying equipment and patient demographics. In this regard, we introduce a novel and realistic setting named domain generalizable continual learning (DGCL): a model learns sequential tasks with each involving a single domain, and is required to perform well across all encountered tasks and domains over time.

The most relevant fields to DGCL are domain generalization (DG) [3, 4] and continual learning (CL) [5, 6], which have been traditionally studied in isolation and are not equipped to handle their integrated challenges. DG aims to generalize from seen domains to unseen domains of individual tasks, assuming that the training data for all tasks is provided simultaneously. CL focuses on mitigating catastrophic forgetting when learning sequential tasks, assuming that the training and testing domains for each task are identical. In contrast, DGCL confronts both the continuity of learning tasks and the variability of testing domains, which poses unique challenges of acquiring, retaining, and leveraging semantic- and domain-relevant information for robust generalization. Although recent advances in CL have leveraged powerful pre-trained models (PTMs) to enhance task-specific generalization [7, 8, 9, 10], these methods are optimized for conventional CL settings and suffer significant performance degradation when applied to DGCL (see Sec. 4). In particular, many more advanced methods underperform relatively simple methods (i.e., L2P [7] for parameter-efficient tuning and SLCA [11] for full parameter tuning) in DGCL, exposing severe limitations in both representation learning and output alignment.

To address these challenges, we draw inspiration from the human brain, which achieves robust DGCL-like capabilities through its organizing principles and memory consolidation mechanisms. Specifically, task-dependent experiences are consolidated into task-independent generalized knowledge after learning and are subsequently reconstructed in the light of such prior knowledge, as reflected in the neural representations with corresponding memory traces [12, 13, 14]. This process, encapsulated in the distributed-plus-hub theory, underpins cross-task and cross-domain generalization [2]

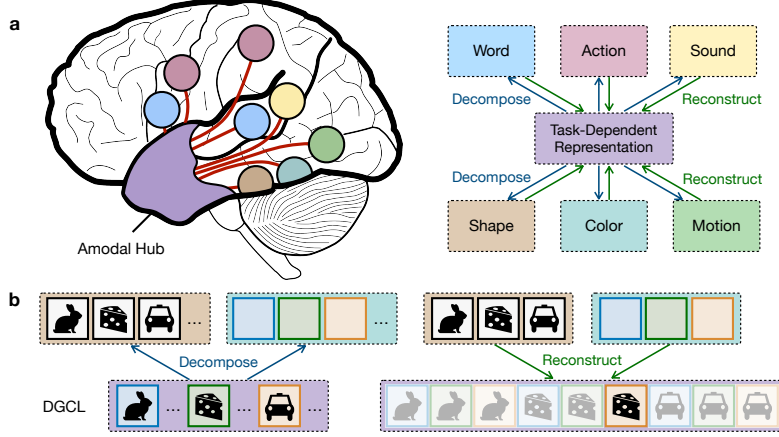


Fig. 1: Motivation of DGCL and DoT. **a**, The human brain naturally supports DGCL through distributed-plus-hub of memory attributes to reconstruct task-dependent representations [1, 2]. **b**, Inspired by the human brain, we propose to address DGCL by coordinating semantic- and domain-relevant information in task-dependent representations. Best viewed in color.

(see Fig. 1a). With the neural inspiration, we propose adaptive Domain Transformation (DoT), an innovative PTMs-based approach tailored to DGCL. In representation learning, DoT obtains decoupled semantic- and domain-relevant information with the innate differentiation of layer-wise features in PTMs. The feature distributions corresponding to both types of information are effectively preserved, and combined flexibly with an attention-based transformation strategy. The transformed features encompass all encountered tasks and domains, collectively producing balanced and generalized predictions (see Fig. 1b).

We rigorously construct evaluation benchmarks for DGCL and demonstrate DoT’s superiority with extensive experiments. DoT serves as a plug-in strategy that greatly facilitates state-of-the-art CL methods under both full parameter tuning and parameter-efficient tuning paradigms, allowing for significant advancements in DGCL. In addition, DoT is shown to accumulate domain-generalizable knowledge from DGCL, providing a more pronounced advantage of performing all seen tasks in completely unseen domains. DoT further ensures resource efficiency through a lightweight implementation, making it a practical choice for real-world applications.

Overall, our main contributions can be summarized as follows:

- We introduce DGCL, a novel and realistic setting that integrates the dual challenges of DG and CL in adapting to real-world complexity.
- We conduct an extensive empirical investigation of DGCL, revealing the severe limitations of cutting-edge advances in continual representation learning and continual output alignment.
- We develop an innovative approach with reference to the human brain, which transforms semantic- and domain-relevant information in continually learned representations to achieve robust generalization.

- Extensive experiments demonstrate the effectiveness of our approach, which significantly enhances state-of-the-art CL methods under both full parameter tuning and parameter-efficient tuning paradigms in the DGCL setting.

2 Related Work

Domain Generalization (DG) aims to develop models capable of generalizing from one or more source domains to unseen target domains, without access to the latter during training [3, 4]. Based on the composition of domain and label spaces, DG has been studied under several key settings. Multi-source DG leverages data from multiple source domains to learn domain-invariant representations [15], while single-source DG confronts the challenge of limited data diversity, often aligning closely with out-of-distribution (OOD) robustness research [16]. Homogeneous DG assumes shared label spaces between source and target domains, whereas heterogeneous DG considers different label spaces between them [17]. Open-set DG [18] adds previously unseen classes to seen classes in unseen target domains. Despite significant progress in capturing spatial-scale domain invariance (e.g., style normalization [19]), most existing methods lack the ability to encode semantic- and domain-relevant information across temporal scales. This limitation becomes critical when sequential tasks are tied to distinct source domains that evolve over time, rendering conventional DG methods inapplicable to the DGCL setting.

Several related settings extend the paradigm of DG, such as domain adaptation (DA) [20], test-time adaptation (TTA) [21], and continual test-time adaptation (CTTA) [22], which leverage training or test data from target domains to enhance generalization. Specifically, DA employs sparsely labeled or unlabeled training data of target domains, TTA adapts to target domains using single or mini-batch test data, while CTTA extends TTA by accommodating dynamic target domains that evolve over time. In contrast, DGCL operates under the constraints of continual learning, where neither training nor test data from multiple tasks are accessible simultaneously. These realistic considerations make DGCL fundamentally distinct and necessitate novel approaches to address its unique challenges.

Continual Learning (CL) aims to develop models to learn sequential tasks while mitigating catastrophic forgetting of previous knowledge [5, 6, 23, 24]. Based on the composition of domain and label spaces, CL is typically categorized into task-incremental learning (TIL), class-incremental learning (CIL), and domain-incremental learning (DIL) [25]. TIL and CIL involve disjoint label spaces, with TIL requiring task identity during testing, while DIL assumes a shared label space across tasks but varying domains.¹ Recent efforts have introduced more flexible paradigms, such as task-blurry incremental learning (TBIL) [26, 27], which allows overlapping label spaces in CIL, and versatile incremental learning (VIL) [28], which involves either new classes or new domains of old classes in each task. However, these settings inherently assume that the training and test data of each task share the same data distribution, failing to account for the broader challenges posed by DG (see Fig. 2).

¹For naming consistency, we refer to each training phase with a distinct data distribution as a “task”.

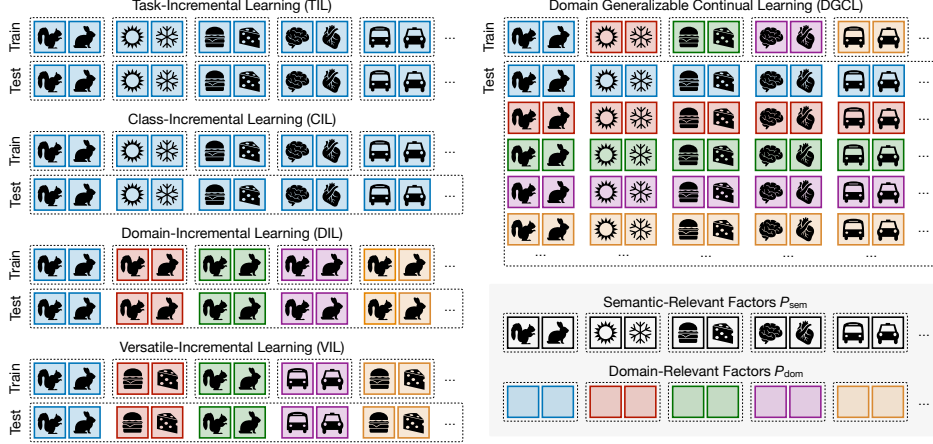


Fig. 2: Comparison of DGCL with representative CL settings, such as TIL, CIL, DIL, and VIL. DGCL is the only setting with training and test sets of each task belonging to different domains.

To address catastrophic forgetting, conventional CL methods are often categorized into regularization-based [29, 30], replay-based [31, 32], and architecture-based ones [33, 34]. Initially designed for training from scratch, these methods have shown limited effectiveness when applied to complex scenarios. Recent state-of-the-art CL methods transfer knowledge from PTMs to enhance generalization [11]. To avoid overwriting pre-trained parameters, these methods usually keep the pre-trained backbone frozen and adopt parameter-efficient tuning (PET) techniques for representation learning [7, 8, 9, 10, 35]. When handling incremental tasks with distinct data distributions, they often employ task-shared or task-specific PET architectures, where the former requires gradual updates to mitigate catastrophic forgetting while the latter requires an additional query function to predict task identities [36]. However, these methods are constrained by conventional CL settings, leaving their capabilities to address unseen domains of incremental tasks largely unexplored.

3 Preliminaries

In this section, we first describe the problem formulation of our proposed DGCL setting, and then evaluate state-of-the-art CL methods under this new paradigm.

3.1 Problem Formulation

DGCL requires a model to learn a sequence of tasks from their individual training sets, each associated with a potentially distinct domain (i.e., the domains may be identical or different from task to task), while generalizing to all encountered tasks and domains at test time. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the global input and output spaces, respectively. For each task $t \in \{1, \dots, T\}$, the input space $\mathcal{X}_t \subseteq \mathcal{X}$ comprises both semantic- and domain-relevant information. Specifically, the input data is drawn from

a joint distribution $P_X^{(t)} = P_{\text{dom}}^{(t)} \circ P_{\text{sem}}^{(t)}$, where $P_{\text{dom}}^{(t)}$ denotes the domain-relevant factor and $P_{\text{sem}}^{(t)}$ denotes the semantic-relevant factor (see examples in Fig. 2). Each task is further characterized by a semantic distribution $P_Y^{(t)}$ over its output space $\mathcal{Y}_t \subseteq \mathcal{Y}$.

The training set for each task t is defined as $\mathcal{D}_t = \{(\mathbf{x}_{t,n}, y_{t,n})\}_{n=1}^{N_t}$, where $\mathbf{x}_{t,n} \sim P_X^{(t)} = P_{\text{dom}}^{(t)} \circ P_{\text{sem}}^{(t)}$, $y_{t,n} \sim P_Y^{(t)}$, and N_t is the total number of training samples. The input data of task t originates from a particular source domain d_t with domain-relevant factor $P_{\text{dom}}^{(t)}$, and its semantic-relevant factor $P_{\text{sem}}^{(t)}$ aligns with the task-specific semantic distribution $P_Y^{(t)}$. Notably, the input domains may differ across tasks, i.e., $\exists i \neq j, P_{\text{dom}}^{(i)} \neq P_{\text{dom}}^{(j)}$ and $d_i \neq d_j$; while the semantic-relevant factors remain disjoint as in regular CL settings, i.e., $\forall i \neq j, P_{\text{sem}}^{(i)} \neq P_{\text{sem}}^{(j)}$ and $P_Y^{(i)} \neq P_Y^{(j)}$. At the testing phase of T tasks, the domain-relevant factors of all tasks $1, \dots, T$ are combined into a unified one $\bigcup_{i=1}^T P_{\text{dom}}^{(i)}$ from all domains d_i , while the semantic distribution of each task t remains aligned with its original $P_{\text{sem}}^{(t)}$ and $P_Y^{(t)}$. Consequently, the test data for task t is sampled as $\mathbf{x} \sim \bigcup_{i=1}^T P_{\text{dom}}^{(i)} \circ P_{\text{sem}}^{(t)}$ while its ground-truth label $y \sim P_Y^{(t)}$ preserves the original distribution.

Consider a neural network model comprising a backbone $f_\theta(\cdot)$ with parameters θ and an output layer $h_\psi(\cdot)$ with parameters ψ . The objective is to learn a projection from $\bigcup_{t=1}^T \mathcal{X}_t$ to $\bigcup_{t=1}^T \mathcal{Y}_t$ by minimizing empirical errors across sequentially observed tasks and domains, so that the model can correctly predict the label $\hat{y} = h_\psi(f_\theta(\mathbf{x}))$ for unseen test data $\mathbf{x}$. However, learning such a desirable projection is highly nontrivial in DGCL: while the training data for each task is confined to a single source domain, the test data for ultimate evaluation spans all encountered tasks and domains. Distinct from TIL, CIL, DIL, and VIL paradigms, DGCL uniquely combines CL with task-specific DG, addressing the integrated challenges of mitigating catastrophic forgetting, managing disjoint label spaces, and generalizing across different domains (see Fig. 2). These challenges are reinforced by the inherent CL constraint of task-specific data isolation - historical training samples stay strictly *inaccessible* during the incremental learning process.

3.2 CL with PTMs

State-of-the-art CL methods leverage advanced PTMs to enhance task-specific generalization. The pre-trained backbone $f_\theta(\cdot)$ often employs transformer-based architectures [37], such as the Vision Transformer (ViT) [38]. ViT processes input data via consecutive multi-head self-attention layers (assuming a total of L layers), producing sequence-like token embeddings $\mathbf{h}^{(L)} \in \mathbb{R}^{s \times m}$, where s is the sequence length and m is the embedding dimension. For each layer $l \leq L$, relationships within the input $\mathbf{h}^{(l-1)} \in \mathbb{R}^{s \times m}$ are modeled by computing attention scores for queries $\mathbf{h}_Q^{(l)}$, keys $\mathbf{h}_K^{(l)}$, and values $\mathbf{h}_V^{(l)}$ using learnable projection matrices $\mathbf{W}_Q^{(l)}$, $\mathbf{W}_K^{(l)}$, and $\mathbf{W}_V^{(l)}$, respectively, resulting in the layer-wise output $\mathbf{h}^{(l)}$ that captures both global and local context.

When updating the entire parameter set θ , i.e., full parameter tuning, $f_\theta(\cdot)$ can effectively adapt its output representations to specific tasks, but this often leads to severe forgetting of the pre-trained knowledge. A common alternative is to keep $f_\theta(\cdot)$

frozen and employ PET techniques for lightweight modifications. For example, Prompt Tuning (ProT) and Prefix Tuning (PreT) both add a set of learnable prompts to the layer-wise input: ProT prepends prompts to keys, queries, and values, whereas PreT divides them into task-specific prefixes for keys and values. Adapter Tuning (AdaT) [39] inserts lightweight neural modules between transformer layers, while Low-Rank Adaptation (LoRA) [40] approximates the updates to specific projection matrices (e.g., $\mathbf{W}_Q^{(l)}$ and $\mathbf{W}_V^{(l)}$) via low-rank decomposition for efficient adaptation.

In general, recent efforts in CL with PTMs focus on two aspects for sequential tasks: continual representation learning for $f_\theta(\cdot)$ and continual output alignment for $h_\psi(\cdot)$, as summarized below. Their effectiveness in DGCL will be examined in Sec. 4.

Representation Learning: Existing methods have developed a variety of PET architectures, including task-specific parameters, which avoid inter-task interference but require additional inference of task identities; task-shared parameters, which avoid additional inference of task identities but need to address catastrophic forgetting during parameter updates; or a combination of both. As a pioneering work, L2P [7] constructs a pool of prompts, each associated with a learnable key, and searches the most relevant prompts based on the cosine similarity between uninstructed representations and these keys. DualPrompt [8] incorporates either task-specific or task-shared prompts in different layers, while S-Prompt [41] and HiDe [10, 36] employ only task-specific prompts. CODA-Prompt [9] performs a weighted summation over an expandable prompt pool, implementing orthogonal regularization to reduce inter-task interference. LAE [42] and RanPAC [43] adopt task-shared parameters, which are updated at a reduced pace or even frozen during CL.

Output Alignment: Many recent methods approximate and replay pre-trained representations of previous tasks to rectify potential bias in $h_\psi(\cdot)$. For example, SLCA [11, 44] performs full parameter tuning with a reduced learning rate, preserving pre-trained representations of each class with dedicated mean and covariance matrices. RanPAC [43] expands pre-trained representations into a high-dimensional space, maintaining the inherent structure with a shared covariance matrix. HiDe [10, 36] evaluates various strategies for preserving pre-trained representations, such as using randomly selected prototypes, multiple centroids, mean-variance, and mean-covariance. ICON [28] performs adaptive classifier expansion and cluster-based regularization to accommodate either new tasks or new domains of old tasks.

4 Empirical Investigation of DGCL

We perform an extensive empirical investigation to analyze the particular challenges of DGCL. We construct sequential tasks with well-established class recognition datasets composed of distinct domains, including Office-Home [45], DigitsDG [17], CRe50 [46], and DomainNet [47]. Experiments are conducted in both CIL and DGCL settings. Specifically, we first split class labels into disjoint task-specific subsets. In CIL, data from all domains is available for each task. Nevertheless, in DGCL, only one randomly assigned domain is available for each task, restricting its training data to that source domain exclusively (see Sec. 6.1 for implementation details). We consider a wide range

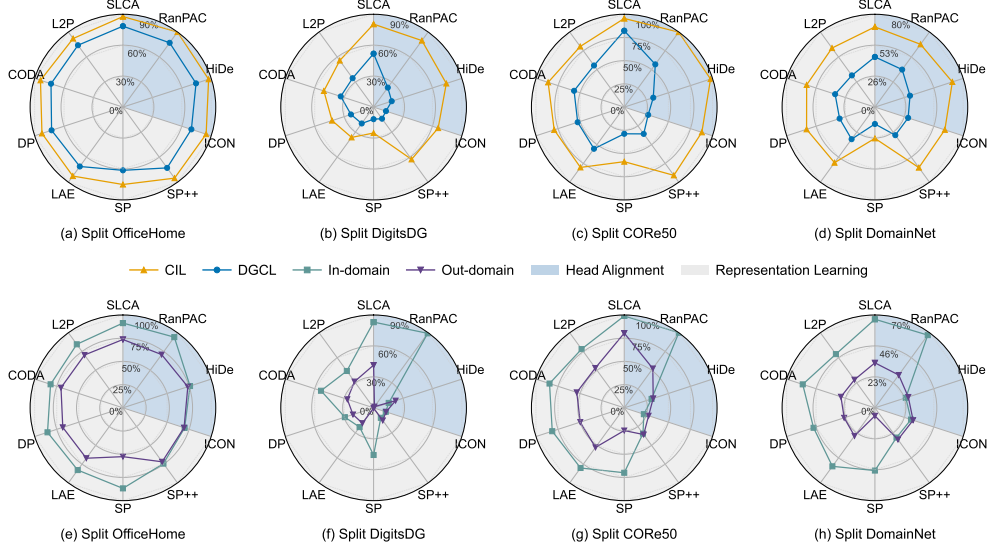


Fig. 3: Empirical investigation of DGCL with state-of-the-art CL baselines. **a-d**, We report the average accuracy of all encountered tasks and domains under CIL and DGCL settings (i.e., average all-domain accuracy A_{all}). **e-h**, We further report the average accuracy of tasks with seen domains (i.e., average in-domain accuracy A_{in}) and tasks with unseen domains (i.e., average out-domain accuracy A_{out}). Please refer to Sec. 6.1 for implementation details. Methods in light gray focus on instructing $f_{\theta}(\cdot)$, while those covered in light blue focus on rectifying $h_{\psi}(\cdot)$ with or without instructing $f_{\theta}(\cdot)$. DP, DualPrompt. SP, S-Prompt. CODA, CODA-Prompt.

of state-of-the-art baselines described in Sec. 3.2. Most of them are designed for CIL, except S-Prompt for DIL and ICON for VIL.

Overall Performance: We compare the average accuracy of classes across all encountered tasks and domains under both CIL and DGCL settings, as shown in Fig. 3 **a-d**. State-of-the-art baselines exhibit significant performance degradation in DGCL, primarily due to the limited availability of domain-relevant information underlying the associated semantic-relevant information in sequential tasks. Notably, L2P, the first PET-based CL method, is inferior to other more advanced methods in CIL but achieves almost the highest overall performance in DGCL. SLCA, a simple baseline that performs full parameter tuning with a reduced learning rate and performs representation recovery for output alignment, achieves overwhelmingly better performance than all PET-based CL methods in DGCL. S-Prompt and ICON, despite their specific design for incremental domains, cannot outperform other CIL methods. These results underscore that cutting-edge advances in CL are less effective in resolving DGCL.

In- vs Out-Domain: We further compare the average accuracy of classes in tasks with seen domains versus those with unseen domains² in DGCL (see Fig. 3 e-h). Evaluation on seen domains is essentially equivalent to the CIL setting (while using only a small amount of training samples within one domain) as both the training and test sets of each task belong to the same domain, and all baselines show similar trends of comparably strong performance. However, the average accuracy on unseen domains deteriorates significantly for most baselines, which is the main cause of their performance degradation in the DGCL setting. This highlights the limited generalization capacity and the difficulty of transferring domain-relevant information across tasks. We then provide a more detailed analysis regarding different focuses of the evaluated methods, as described below.

Representation Learning: For continual representation learning of $f_{\theta}(\cdot)$, different PET architectures show mixed performance in CIL, without a clear optimal choice. However, in DGCL, using a prompt pool (i.e., L2P and CODA-Prompt) is generally more effective than either task-specific prompts (i.e., DualPrompt, S-Prompt and S-Prompt++) or task-shared prompts (i.e., DualPrompt and LAE). This is because task-specific prompts hinder the transfer of domain-relevant information across tasks, whereas task-shared prompts have limited capacity to accommodate interference over a range of distinct tasks and domains. In contrast, the prompt pool often has a larger capacity of available tuning parameters than either task-specific prompts or task-shared prompts. The prompt ensembling process implicitly disentangles both information and combines them in a weighted manner, therefore facilitating transfer and reducing interference. SLCA benefits from a similar mechanism by possessing adequate capacity (i.e., full parameter tuning) to ensemble representations across tasks and domains effectively.

Output Alignment: Regarding continual output alignment of $h_{\psi}(\cdot)$, PET-based CL methods with representation recovery generally outperform those without in CIL, as the ensemble of semantic-relevant information rectifies potential biases in predictions. However, this is not necessarily the case in DGCL (e.g., HiDe underperforms L2P in many cases), which underscores the challenges in approximating and recovering domain-relevant information. Notably, methods that incorporate representations with higher complexity tend to perform better, with performance ranking as: ICON (prototypes) < HiDe (multiple centroids) < RanPAC (high-dimensional covariance matrix) < SLCA (multiple covariance matrices). While RanPAC achieves near-perfect accuracy on seen domains, its performance on unseen domains drops drastically. For example, its average in-domain accuracy is 89.25% whereas average out-domain accuracy is only 1.44% on Split DigitsDG. This suggests that representation recovery strategies with higher complexity (e.g., second-order statistics) may effectively capture domain-relevant information, but struggle to transfer or even interfere with generalization to other domains.

In summary, state-of-the-art baselines face significant limitations in both representation learning and output alignment. Relatively simple methods instead perform better, since many more advanced designs in conventional CL settings fail to adapt

²Unless otherwise specified, we use “unseen domain” to denote a domain that has been unseen for a certain task but seen for other tasks. We also examine a “completely unseen domain” for all tasks, and present the results in Table 4.

to DGCL. For representation learning, an ensemble of semantic- and domain-relevant information with adequate capacity in tuning parameters tends to be a desirable choice, such as the prompt pool and full parameter tuning. For output alignment, an ensemble of semantic- and domain-relevant information is also important to obtain balanced and generalized predictions, which requires accurately approximating and recovering feature distributions across tasks and domains. These empirical insights align with how the human brain processes incremental information in diverse forms and generalizes the underlying knowledge to different scenarios (Fig. 1). We therefore sought a more effective solution to DGCL based on the current progress, which will be presented in the next section.

5 Method

In this section, we draw inspiration from the distributed-plus-hub theory [2] of the human brain, and propose adaptive Domain Transformation (DoT) to address DGCL (see Fig. 5). The key innovation is to disentangle semantic- and domain-relevant information in representation learning, and adaptively transform previously learned representations across encountered tasks and domains for output alignment. We first describe an interesting observation that PTMs can naturally disentangle these two types of information in layer-wise features, and then present our approach that strategically exploits this property for domain transformation.

5.1 Layer-Wise Domain Disentanglement

To formally describe the limitation of existing methods and to propose our solution, we begin with revisiting the basic (continual) learning objective. Given the training set $\mathcal{D}_t = \{(\mathbf{x}_{t,n}, y_{t,n})\}_{n=1}^{N_t}$ of task t , the neural network model $h_\psi(f_\theta(\cdot))$ is often optimized with the following loss function:

$$\mathcal{L}_t(\theta, \psi) = \frac{1}{N_t} \sum_{n=1}^{N_t} \mathcal{L}_{\text{task}}(h_\psi(f_\theta(\mathbf{x}_{t,n})), y_i), \quad (1)$$

where the task-specific loss $\mathcal{L}_{\text{task}}$ can be further specified as the cross-entropy loss $\mathcal{L}_{\text{ce}}$ for classification tasks. This loss function is to learn the mapping function from $P_X^{(t)} = P_{\text{dom}}^{(t)} \circ P_{\text{sem}}^{(t)}$ to $P_Y^{(t)}$. Even if the mapping functions of individual tasks are completely preserved from catastrophic forgetting (e.g., by using state-of-the-art CL methods), it is still difficult to address the objective of DGCL, i.e., learning the mapping function from $\bigcup_{i=1}^T P_{\text{dom}}^{(i)} \circ P_{\text{sem}}^{(t)}$ to $P_Y^{(t)}$. Since the domain-relevant factor $P_{\text{dom}}^{(i)}$ varies in DGCL while the semantic-relevant factor $P_{\text{sem}}^{(t)}$ keeps aligned with $P_Y^{(t)}$, a promising solution is to disentangle $P_{\text{dom}}^{(i)}$ from $P_{\text{sem}}^{(t)}$, approximate and preserve both distributions, and transform them across encountered tasks and domains.

However, it is highly nontrivial to explicitly decouple these two factors in DGCL, because the training set for each task and its associated domain is provided sequentially, which is a fundamental difference from multi-source DG. To address this, we

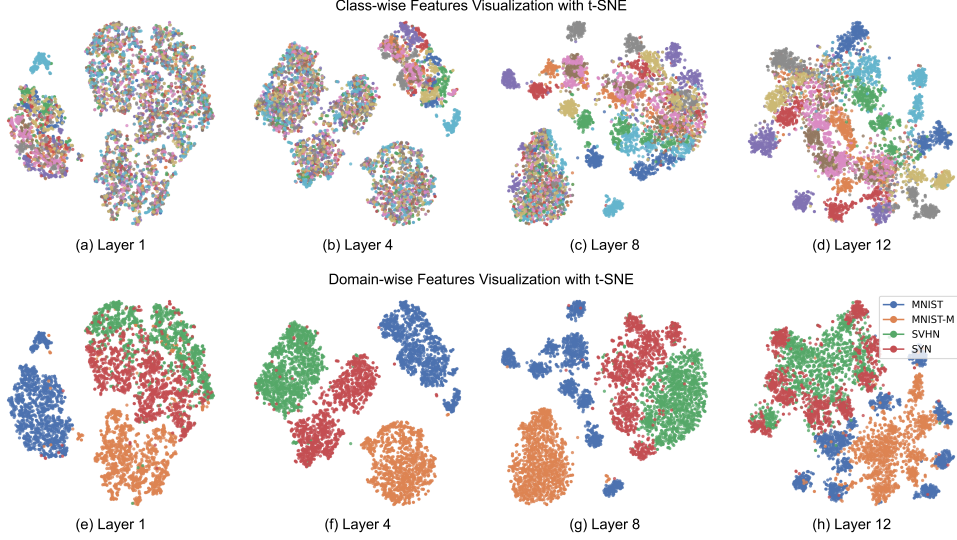


Fig. 4: Layer-wise feature visualization using t-SNE. We extract the layer-wise features $\mathbf{r}^{(l)}$ of a ViT-B/16 backbone on the test set of Split DigitsDG, which has been trained using SLCA for all incremental tasks. **a-d**, Points are colored with their class labels. **e-h**, Points are colored with their domain identities. Best viewed in color.

propose to implicitly decouple these two factors by exploiting the innate differentiation of layer-wise features in PTMs. Specifically, the pre-trained backbone $f_{\theta}(\cdot)$ often consists of multiple consecutive layers to encode input data into well-distributed representations. This process can be expressed as $f_{\theta}(\cdot) = (f_{\theta_1} \circ f_{\theta_2} \circ \dots \circ f_{\theta_L})(\mathbf{x})$ of L consecutive layers, generating layer-wise token embeddings $\mathbf{h}^{(l)}$. The feature representations $\mathbf{r}^{(l)} \in \mathbb{R}^m$ are often extracted as the global average vector or the CLS token of $\mathbf{h}^{(l)}$, denoted as $\mathbf{r}^{(l)} = g(\mathbf{h}^{(l)})$. We further denote all the intermediate features for $1 : L$ layers as $\mathbf{R} = [\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \dots, \mathbf{r}^{(L)}]^{\top} \in \mathbb{R}^{L \times m}$.³

Intuitively, the layer-wise features $\mathbf{r}^{(l)}$ closer to $h_{\psi}(\cdot)$ tend to carry more semantic-relevant information, with the final-layer features $\mathbf{r}^{(L)}$ carrying the most. The intermediate features in $\mathbf{R}$, on the other hand, carry more domain-relevant information to varying degrees. This trend is validated by our empirical investigation of CL with PTMs under the DGCL setting. As shown in Fig. 4, we perform t-SNE visualization of $\mathbf{r}^{(l)}$ for each backbone layer $l \leq L$. The intermediate layers (e.g., $l = 4, 8$) provide domain-discriminative features that can effectively separate different domains, while the deeper layers ($l = 12$) allow for semantic discrimination after task-specific fine-tuning. This suggests that the deeper layers capture more semantic-relevant information while the intermediate layers encode more domain-relevant information, aligning with the distributed-plus-hub theory [2] of neural representations (Fig. 1).

³For clarity, the layer identity l of $\mathbf{r}$ is placed in superscript. Its underlying task identity t , domain identity d_t , class identity c , and instance identity n are placed in subscript, while omitted if not necessary.

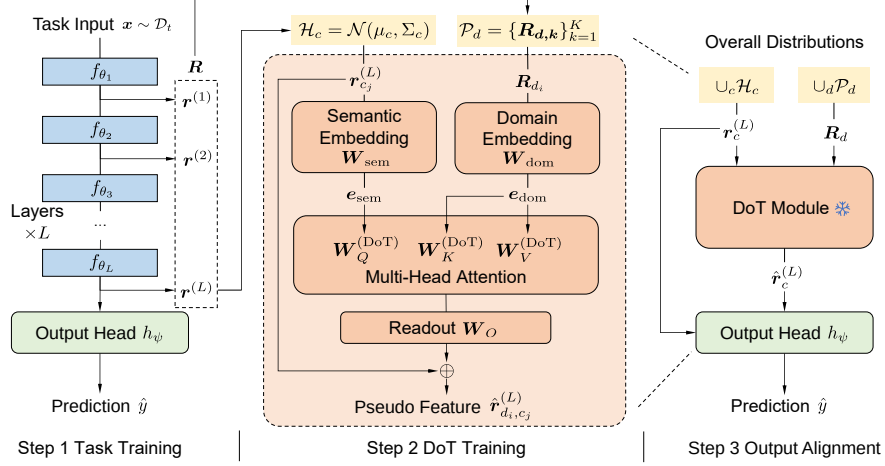


Fig. 5: Illustration of the proposed framework for DGCL. In step 1, the entire model is trained with incremental tasks, where the feature distributions $\mathcal{H}_c$ and $\mathcal{P}_d$ are preserved. In step 2, the DoT module is trained to generate pseudo feature based on sampled features. In step 3, the output layer h_ψ is aligned with both sampled features and pseudo features produced by the DoT module.

This innate capability of domain disentanglement is then used to capture $P_{\text{dom}}^{(t)}$ and $P_{\text{sem}}^{(t)}$ for domain transformation, as detailed below.

5.2 Attention-Based Domain Transformation

With the layer-wise differentiation of semantic- and domain-relevant information, we need to approximate and preserve them for subsequent combination. Considering the distinct properties of their distributions, we adopt adaptive strategies to ensure the effectiveness:

1. **Semantic-Relevant Information.** The final-layer features $\mathbf{r}^{(L)}$ collected from $f_\theta(\cdot)$ tend to distribute aligning with the semantic labels. For classification tasks, $\mathbf{r}_{c_t}^{(L)}$ belonging to the same class $c_t \in \mathcal{Y}_t$ tend to have a single-peaked distribution [11, 10], which can be naturally approximated as a Gaussian $\mathcal{H}_{c_t} = \mathcal{N}(\boldsymbol{\mu}_{c_t}, \boldsymbol{\Sigma}_{c_t})$ with mean vector $\boldsymbol{\mu}_{c_t} \in \mathbb{R}^m$ and covariance matrix $\boldsymbol{\Sigma}_{c_t} \in \mathbb{R}^{m \times m}$. The covariance matrix can be further simplified into the variance vector by taking the diagonal $\text{Diag}(\boldsymbol{\Sigma}_{c_t}) \in \mathbb{R}^m$, thus reducing the additional parameter cost to a negligible value.
2. **Domain-Relevant Information.** For each task t and its associated domain d_t , the intermediate features in $\mathbf{R}_{d_t} = [\mathbf{r}_{d_t}^{(1)}, \mathbf{r}_{d_t}^{(2)}, \dots, \mathbf{r}_{d_t}^{(L)}]^\top$ carry domain-relevant information of d_t to varying degrees, which are difficult to fit with a specific probability distribution. Therefore, we simply approximate their distributions $\mathcal{P}_{d_t}$ by selecting K prototypes $\{\mathbf{R}_{d_t, k}\}_{k=1}^K$ (through random sampling or K -nearest neighbors).

Next, we propose an attention-based strategy to aggregate semantic- and domain-relevant information from these preserved feature distributions. After sampling the

final-layer features $\mathbf{r}_{c_j}^{(L)} \sim \mathcal{H}_{c_j}$ of encountered class $c_j \in \mathcal{Y}_j$ and intermediate features $\mathbf{R}_{d_i} \sim \mathcal{P}_{d_i}$ of encountered domain d_i , we first project them into different latent spaces with learnable semantic embedding matrix $\mathbf{W}_{\text{sem}} \in \mathbb{R}^{m \times m}$ and domain embedding matrix $\mathbf{W}_{\text{dom}} \in \mathbb{R}^{m \times m}$, respectively:

$$\mathbf{e}_{\text{sem}} = \mathbf{r}_{c_j}^{(L)} \mathbf{W}_{\text{sem}} \in \mathbb{R}^m, \quad \mathbf{e}_{\text{dom}} = \mathbf{R}_{d_i} \mathbf{W}_{\text{dom}} \in \mathbb{R}^{L \times m}. \quad (2)$$

This embedding step further enforces the separation of semantic- and domain-relevant information carried by the original features $\mathbf{r}_{c_j}^{(L)}$ and $\mathbf{R}_{d_i}$. Thus, we regard $\mathbf{e}_{\text{sem}}$ as the query vector and $\mathbf{e}_{\text{dom}}$ as both the key and the value vector to perform the attention calculation, so that the extracted information can be flexibly reassembled:

$$\mathbf{a} = \text{Softmax} \left(\mathbf{e}_{\text{sem}} \mathbf{W}_Q^{(\text{DoT})} (\mathbf{e}_{\text{dom}} \mathbf{W}_K^{(\text{DoT})})^\top / \sqrt{m} \right) \mathbf{e}_{\text{dom}} \mathbf{W}_V^{(\text{DoT})}, \quad (3)$$

where $\mathbf{W}_Q^{(\text{DoT})}$, $\mathbf{W}_K^{(\text{DoT})}$ and $\mathbf{W}_V^{(\text{DoT})} \in \mathbb{R}^{m \times m}$ are learnable projection matrices in the DoT attention module. In practice, we employ a multi-head version by splitting the embedding dimension m into parallel subspaces. Each head independently computes Eq.(3) within its subspace, and the results are concatenated as the output (the head identity is omitted for clarity).

Finally, the pseudo features $\hat{\mathbf{r}}_{d_i, c_j}^{(L)}$ are obtained through a fully-connected readout layer $\mathbf{W}_O \in \mathbb{R}^{m \times m}$ with a residual connection and non-linear activation operation σ :

$$\hat{\mathbf{r}}_{d_i, c_j}^{(L)} = \sigma \left(\mathbf{r}_{c_j}^{(L)} + \mathbf{a} \mathbf{W}_O \right). \quad (4)$$

This transformation step incorporates the domain-relevant information of d_i into the pseudo features $\hat{\mathbf{r}}_{d_i, c_j}^{(L)}$, while aligning them with the semantic anchors $\mathbf{r}_{c_j}^{(L)}$. Together, we integrate Eq.(2-4) for the entire DoT process:

$$\hat{\mathbf{r}}_{d_i, c_j}^{(L)} = \text{DoT}(\mathbf{r}_{c_j}^{(L)}, \mathbf{R}_{d_i}). \quad (5)$$

To ensure the generated pseudo features $\hat{\mathbf{r}}_{d_i, c_j}^{(L)}$ positioned appropriately in semantic- and domain-specific latent spaces, we introduce two contrastive losses $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{dom}}$ to optimize the linear projection heads p_{cls} and p_{dom} , respectively:

$$\mathcal{L}_{\text{cls}}(\hat{\mathbf{r}}_{d_i, c_j}^{(L)}) = - \sum_{\mathbf{r}^{(L)} \sim \mathcal{H}_{c_j}} \log \left[\exp \left(p_{\text{cls}}(\hat{\mathbf{r}}_{d_i, c_j}^{(L)}) \cdot p_{\text{cls}}(\mathbf{r}^{(L)}) / \tau \right) / s_{\text{cls}} \right], \quad (6)$$

$$\mathcal{L}_{\text{dom}}(\hat{\mathbf{r}}_{d_i, c_j}^{(L)}) = - \sum_{\mathbf{r}_{d_i}^{(L)} \sim \bigcup_{c \in \mathcal{Y}_i} \mathcal{H}_c} \log \left[\exp \left(p_{\text{dom}}(\hat{\mathbf{r}}_{d_i, c_j}^{(L)}) \cdot p_{\text{dom}}(\mathbf{r}_{d_i}^{(L)}) / \tau \right) / s_{\text{dom}} \right], \quad (7)$$

where s_{cls} and s_{dom} are normalizing factors representing the sum of latent distances between $\hat{\mathbf{r}}_{d_i, c_j}^{(L)}$ and all other sampled final-layer features $\mathbf{r}^{(L)} \sim \mathcal{H}_c$ for $\forall c \in \mathcal{Y}_{1:t}$, and τ is a temperature hyperparameter. $\mathcal{L}_{\text{cls}}$ optimizes the semantic similarity between $\hat{\mathbf{r}}_{d_i, c_j}^{(L)}$

and $\mathbf{r}^{(L)}$ of the same class c_j ; while $\mathcal{L}_{\text{dom}}$ optimizes the domain similarity between $\hat{\mathbf{r}}_{d_i, c_j}^{(L)}$ and $\mathbf{r}_{d_i}^{(L)}$ of the same domain d_i .

The overall loss function for training the DoT process consists of both $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{dom}}$, controlled by a loss-weight hyperparameter $\lambda \in (0, 1)$:

$$\mathcal{L}_{\text{DoT}} = (1 - \lambda)\mathcal{L}_{\text{cls}} + \lambda\mathcal{L}_{\text{dom}}. \quad (8)$$

By transforming feature representations across all encountered tasks and domains, we perform output alignment with these pseudo features to obtain balanced and generalized predictions. Specifically, we synthesize $\hat{\mathbf{r}}_{d_i, c_j}^{(L)}$ from both $\mathbf{r}_{c_j}^{(L)} \sim \mathcal{H}_{c_j}, \forall c_j \in \mathcal{Y}_j$ and $\mathbf{R}_{d_i}$ for all possible (i, j) pairs. Therefore, $h_\psi(\cdot)$ is enforced to learn the sampled features $\mathbf{r}^{(L)}$ and generated pseudo features $\hat{\mathbf{r}}^{(L)}$ with cross-entropy loss:

$$\mathcal{L}_{\text{OA}} = \sum_{\mathbf{r}^{(L)} \sim \bigcup_c \mathcal{H}_c} \sum_{\mathbf{R} \sim \bigcup_d \mathcal{P}_d} \left[\mathcal{L}_{\text{ce}}(h_\psi(\mathbf{r}^{(L)}), c) + \mathcal{L}_{\text{ce}}(h_\psi(\text{DoT}(\mathbf{r}^{(L)}, \mathbf{R})), c) \right]. \quad (9)$$

After rectifying the output layer with Eq. 9, the DoT parameters (i.e., $\mathbf{W}_{\text{sem}}, \mathbf{W}_{\text{dom}}, \mathbf{W}_Q^{(\text{DoT})}, \mathbf{W}_K^{(\text{DoT})}, \mathbf{W}_V^{(\text{DoT})}, \mathbf{W}_O, p_{\text{cls}}$ and p_{dom}) can be simply discarded to avoid additional parameter overhead. Once a new task is introduced, these parameters are reinitialized and trained with updated $\mathcal{H}_{c_t}$ and $\mathcal{P}_{d_t}$, so as to accommodate the incoming classes and domain. The overall training pipeline is summarized in Alg. 1.

6 Experiment

In this section, we first describe the experimental setups of DGCL, and then present the experimental results with an extensive analysis.

6.1 Experimental Setup

Benchmark. We construct evaluation benchmarks of DGCL with four representative datasets widely used in both DG and CL, including Office-Home [45] of 4 domains (art, clip art, product, and real) with 65 classes, DigitsDG [17] of 4 domains (MNIST [48], SVHN [49], MNIST-M [50], and SYN [50]) with 10 classes; CORE50 [46] of 11 domains (changes in lighting, background, and occlusions) with 50 classes, and DomainNet [47] of 6 domains (clip art, infograph, painting, quickdraw, real, and sketch) with 345 classes. The split of training samples and testing samples of each domain and each class follows their original papers. Each dataset is further split into multiple incremental tasks with disjoint classes, where the training samples are restricted to a randomly selected domain while testing samples span all domains. We further ensure that each domain is assigned to the training samples of at least one task, except in the CORE50 dataset where one domain remains unvisited (1 out of 11 domains is excluded for 10 tasks). Detailed statistics of DGCL benchmarks are summarized in Table 1.

Baseline. We consider a variety of recent strong baselines for CL with PTMs, especially for PET-based CL methods. A majority of them are designed for CIL, including L2P [7], DualPrompt [8], S-Prompt++ [10], CODA-Prompt [9], LAE [42],

Table 1: Detailed statistics of DGCL benchmarks, including the total number of images, domains, and classes, as well as the number of incremental tasks and incremental classes in each task.

Dataset	Total-Image	Total-Domain	Total-Class	Incre-Task	Incre-Class
Office-Home [45]	15,588	4	65	5	13
DigitsDG [17]	24,000	4	10	5	2
CORe50 [46]	160,000	11	50	10	5
DomainNet [47]	600,000	6	345	10	35

SLCA [11], RanPAC [51], and HiDe [10, 36]. We also include methods designed for other CL settings, such as S-Prompt [41] for DIL and ICON [28] for VIL. The above methods share some convergent ideas, such as task-specific parameters, task-shared parameters, and representation recovery, as summarized in Sec. 3.2. While LAE [42] and HiDe [10, 36] apply to various PET techniques, a majority of these methods focus on ProT and PreT, which serves as our default choice unless otherwise specified.

Implementation. We adopt commonly used implementations of the comparison baselines, with PILOT [52, 53] as the codebase. Specifically, we employ a ViT-B/16 backbone with supervised pre-training of ImageNet-21K. We train all methods with an Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) of a cosine-decaying learning rate 0.001, except for SLCA using an SGD optimizer of a head learning rate 0.01 and a backbone learning rate $1e-5$. Batch sizes and epoch numbers for each method follow the default setting used in PILOT, and get doubled for the larger dataset DomainNet. We resize the image inputs to 224×224 and normalize them to the range $[0, 1]$. The prompt pools used by L2P and CODA-Prompt follow their default size, prompt length, and location. Other PET-based CL methods employ a similar PET architecture, i.e., PreT with prompt length 5 is inserted at layers 1-5. For DoT’s configuration, we employ a 4-head attention architecture and assign $K = 16$ prototypes per domain distribution $\mathcal{P}_d$. The loss weighting coefficient is set to $\lambda = 0.5$. The training epoch number for output alignment is set to $E_{OA} = 3$, mirroring SLCA’s implementation. The training epoch number for domain transformation is set to $E_{DoT} = 10$.

Evaluation. We define the domain-specific accuracy $a_{t,d}$ for task t evaluated on domain d , and formulate the following metrics to comprehensively evaluate DGCL performance. The average all-domain accuracy $A_{all} = \frac{1}{T} \sum_{t=1}^T \frac{1}{|\mathcal{S}|} \sum_{d \in \mathcal{S}} a_{t,d}$ measures the mean accuracy across all tasks and domains, where T is the total number of tasks and $\mathcal{S}$ is the set of all encountered domains. The average in-domain accuracy $A_{in} = \frac{1}{T} \sum_{t=1}^T a_{t,d_t}$ measures the mean accuracy of all tasks, each evaluated on its training domain d_t . The average out-domain accuracy $A_{out} = \frac{1}{T} \sum_{t=1}^T \frac{1}{|\mathcal{S} \setminus \{d_t\}|} \sum_{d \in \mathcal{S} \setminus \{d_t\}} a_{t,d}$ measures the mean accuracy of all tasks, each evaluated on other domains $d \in \mathcal{S} \setminus \{d_t\}$ out of d_t . The average worst-case accuracy $W_{out} = \frac{1}{T} \sum_{t=1}^T \min_{d \in \mathcal{S} \setminus \{d_t\}} a_{t,d}$ measures the mean accuracy of all tasks, each evaluated on its worst-performing domain out of d_t . The average unseen-domain accuracy $A_{un} = \frac{1}{T} \sum_{t=1}^T a_{t,\bar{d}}$ measures the mean accuracy of all tasks on a completely unseen domain $\bar{d} \notin \mathcal{S}$. The average all-domain forgetting $F_{all} = \frac{1}{T} \sum_{t=1}^T \frac{1}{|\mathcal{S}|} \sum_{d \in \mathcal{S}} (\max_{i \leq T} a_{t,d}^{(i)} - a_{t,d}^{(T)})$, where $a_{t,d}^{(i)}$ and $a_{t,d}^{(T)}$ denote the accuracy of task t on domain d after learning the i -th and the final task T , respectively.

Table 2: Overall performance of DGCL. We report the average all-domain accuracy A_{all} , average in-domain accuracy A_{in} , average out-domain accuracy A_{out} , and average worst-case accuracy W_{out} . We implement the covariance matrix (Cov) and variance vector (Var) for output alignment in DoT and SLCA. All results are averaged over three runs with randomly sampled task sequences. The error bar denotes the standard deviation.

Method	Split Office-Home				Split DigitsDG			
	A_{all} ($\uparrow$)	A_{in} ($\uparrow$)	A_{out} ($\uparrow$)	W_{out} ($\uparrow$)	A_{all} ($\uparrow$)	A_{in} ($\uparrow$)	A_{out} ($\uparrow$)	W_{out} ($\uparrow$)
DualPrompt [8]	72.74 $\pm$ 1.07	85.43 $\pm$ 1.90	68.12 $\pm$ 0.73	56.04 $\pm$ 1.04	23.07 $\pm$ 2.88	29.30 $\pm$ 7.51	21.03 $\pm$ 1.62	13.14 $\pm$ 3.32
S-Prompt [41]	61.29 $\pm$ 1.17	86.77 $\pm$ 2.02	52.60 $\pm$ 1.30	33.39 $\pm$ 2.72	11.70 $\pm$ 0.94	45.72 $\pm$ 3.72	0.36 $\pm$ 0.07	0.00 $\pm$ 0.00
S-Prompt++ [10]	72.84 $\pm$ 1.03	74.27 $\pm$ 1.30	71.86 $\pm$ 0.90	55.32 $\pm$ 3.57	13.89 $\pm$ 0.84	11.94 $\pm$ 4.17	15.16 $\pm$ 0.93	3.64 $\pm$ 1.80
CODA-Prompt [9]	73.31 $\pm$ 0.34	81.95 $\pm$ 0.87	70.15 $\pm$ 0.90	58.42 $\pm$ 1.10	33.49 $\pm$ 4.58	53.61 $\pm$ 5.76	26.79 $\pm$ 4.24	15.31 $\pm$ 1.61
LAE-PreT [42]	71.15 $\pm$ 1.37	82.88 $\pm$ 1.01	67.10 $\pm$ 1.75	53.92 $\pm$ 2.18	19.62 $\pm$ 1.07	23.11 $\pm$ 4.18	18.45 $\pm$ 2.73	8.78 $\pm$ 2.65
RanPAC [51]	77.06 $\pm$ 1.52	94.11 $\pm$ 1.54	70.91 $\pm$ 1.34	56.05 $\pm$ 0.68	23.40 $\pm$ 1.38	89.25 $\pm$ 6.06	1.44 $\pm$ 0.70	0.06 $\pm$ 0.05
HiDe-PreT [10]	74.51 $\pm$ 0.82	75.99 $\pm$ 0.90	73.48 $\pm$ 1.33	56.14 $\pm$ 3.80	18.47 $\pm$ 1.79	15.42 $\pm$ 4.33	22.44 $\pm$ 3.63	3.86 $\pm$ 3.35
ICON [28]	69.78 $\pm$ 1.67	69.87 $\pm$ 3.59	69.22 $\pm$ 2.17	55.48 $\pm$ 2.80	12.61 $\pm$ 0.93	12.39 $\pm$ 1.38	12.69 $\pm$ 0.84	0.89 $\pm$ 0.19
L2P [7]	74.13 $\pm$ 1.35	84.57 $\pm$ 3.28	70.36 $\pm$ 1.20	57.53 $\pm$ 1.01	34.62 $\pm$ 4.70	44.14 $\pm$ 4.89	31.44 $\pm$ 7.54	20.69 $\pm$ 9.14
DoT-L2P (Cov)	78.30 $\pm$ 0.82	90.09 $\pm$ 1.79	74.03 $\pm$ 0.44	61.18 $\pm$ 0.68	38.17 $\pm$ 1.04	81.97 $\pm$ 3.35	23.57 $\pm$ 2.11	10.17 $\pm$ 3.48
DoT-L2P (Var)	77.63 $\pm$ 0.90	88.60 $\pm$ 1.71	73.81 $\pm$ 0.62	61.09 $\pm$ 0.74	38.01 $\pm$ 1.11	80.53 $\pm$ 3.32	23.84 $\pm$ 0.94	9.67 $\pm$ 3.22
SLCA (Cov) [11]	78.46 $\pm$ 0.31	91.30 $\pm$ 0.90	73.60 $\pm$ 0.45	62.14 $\pm$ 2.15	51.81 $\pm$ 2.14	83.11 $\pm$ 6.78	41.38 $\pm$ 0.85	21.83 $\pm$ 2.44
SLCA (Var) [11]	77.89 $\pm$ 0.47	89.87 $\pm$ 1.28	73.34 $\pm$ 0.99	62.16 $\pm$ 1.87	56.39 $\pm$ 4.56	82.78 $\pm$ 5.52	47.59 $\pm$ 4.50	28.94 $\pm$ 6.54
DoT-SLCA (Cov)	79.77 $\pm$ 1.01	91.89 $\pm$ 1.56	75.23 $\pm$ 0.91	63.83 $\pm$ 0.66	61.68 $\pm$ 2.83	87.92 $\pm$ 2.10	52.93 $\pm$ 3.13	32.39 $\pm$ 5.76
DoT-SLCA (Var)	79.41 $\pm$ 0.44	90.66 $\pm$ 1.66	75.17 $\pm$ 0.15	64.73 $\pm$ 0.81	62.48 $\pm$ 3.20	87.28 $\pm$ 1.44	54.21 $\pm$ 4.58	35.14 $\pm$ 9.34

Method	Split CORE50				Split DomainNet			
	A_{all} ($\uparrow$)	A_{in} ($\uparrow$)	A_{out} ($\uparrow$)	W_{out} ($\uparrow$)	A_{all} ($\uparrow$)	A_{in} ($\uparrow$)	A_{out} ($\uparrow$)	W_{out} ($\uparrow$)
DualPrompt [8]	52.76 $\pm$ 2.27	81.55 $\pm$ 3.75	49.82 $\pm$ 2.23	35.49 $\pm$ 1.83	32.01 $\pm$ 1.88	48.65 $\pm$ 4.35	24.38 $\pm$ 2.75	4.42 $\pm$ 0.32
S-Prompt [41]	28.75 $\pm$ 4.51	69.93 $\pm$ 10.36	24.48 $\pm$ 4.07	5.06 $\pm$ 2.75	14.53 $\pm$ 0.65	47.30 $\pm$ 2.84	6.02 $\pm$ 1.53	0.04 $\pm$ 0.03
S-Prompt++ [10]	35.66 $\pm$ 1.89	34.31 $\pm$ 2.03	35.59 $\pm$ 1.16	16.77 $\pm$ 1.94	29.92 $\pm$ 0.68	27.71 $\pm$ 0.64	29.57 $\pm$ 1.12	13.50 $\pm$ 0.96
CODA-Prompt [9]	56.83 $\pm$ 2.33	84.64 $\pm$ 3.53	53.80 $\pm$ 2.37	35.96 $\pm$ 3.23	36.07 $\pm$ 1.97	57.23 $\pm$ 1.83	26.94 $\pm$ 3.22	4.66 $\pm$ 0.16
LAE-PreT [42]	55.47 $\pm$ 1.87	80.09 $\pm$ 2.66	52.71 $\pm$ 2.00	39.03 $\pm$ 0.41	34.18 $\pm$ 0.28	54.53 $\pm$ 1.27	26.17 $\pm$ 2.09	4.71 $\pm$ 0.84
RanPAC [51]	56.90 $\pm$ 0.16	100.00 $\pm$ 0.00	52.34 $\pm$ 0.32	18.27 $\pm$ 2.77	39.96 $\pm$ 2.19	67.77 $\pm$ 0.56	30.57 $\pm$ 4.61	2.15 $\pm$ 1.39
HiDe-PreT [10]	32.87 $\pm$ 3.17	30.96 $\pm$ 4.86	32.78 $\pm$ 3.06	14.75 $\pm$ 2.56	31.80 $\pm$ 3.41	24.47 $\pm$ 4.55	26.46 $\pm$ 4.67	12.55 $\pm$ 1.59
ICON [28]	27.07 $\pm$ 4.99	22.19 $\pm$ 6.77	27.56 $\pm$ 4.81	7.93 $\pm$ 5.37	30.09 $\pm$ 0.90	28.61 $\pm$ 0.56	30.27 $\pm$ 1.43	13.36 $\pm$ 0.19
L2P [7]	55.41 $\pm$ 3.45	78.20 $\pm$ 4.33	53.00 $\pm$ 3.20	43.17 $\pm$ 2.29	33.77 $\pm$ 1.42	50.04 $\pm$ 3.55	26.21 $\pm$ 1.48	4.36 $\pm$ 1.00
DoT-L2P (Cov)	73.50 $\pm$ 0.41	98.33 $\pm$ 0.68	70.87 $\pm$ 0.55	57.20 $\pm$ 1.66	38.65 $\pm$ 0.77	56.86 $\pm$ 2.24	30.56 $\pm$ 0.60	4.62 $\pm$ 1.03
DoT-L2P (Var)	71.63 $\pm$ 0.89	96.68 $\pm$ 0.68	68.98 $\pm$ 1.13	56.11 $\pm$ 0.92	38.63 $\pm$ 0.80	56.50 $\pm$ 2.20	30.62 $\pm$ 0.52	4.68 $\pm$ 1.01
SLCA (Cov) [11]	82.29 $\pm$ 2.12	98.94 $\pm$ 0.23	80.48 $\pm$ 2.36	68.09 $\pm$ 3.77	43.25 $\pm$ 0.62	66.79 $\pm$ 1.00	33.91 $\pm$ 1.52	4.74 $\pm$ 0.51
SLCA (Var) [11]	79.40 $\pm$ 1.36	97.46 $\pm$ 0.15	77.45 $\pm$ 1.56	65.88 $\pm$ 0.84	43.19 $\pm$ 0.79	65.96 $\pm$ 0.89	33.36 $\pm$ 1.32	5.69 $\pm$ 0.44
DoT-SLCA (Cov)	84.73 $\pm$ 1.29	99.20 $\pm$ 0.22	83.14 $\pm$ 1.48	73.00 $\pm$ 2.30	43.82 $\pm$ 1.00	66.83 $\pm$ 0.89	34.52 $\pm$ 1.41	5.10 $\pm$ 0.63
DoT-SLCA (Var)	82.05 $\pm$ 2.24	98.21 $\pm$ 0.88	80.28 $\pm$ 2.53	69.99 $\pm$ 3.69	44.46 $\pm$ 0.95	65.96 $\pm$ 1.05	35.64 $\pm$ 1.14	5.88 $\pm$ 0.40

The average unseen-domain forgetting $F_{\text{un}} = \frac{1}{T} \sum_{t=1}^T (\max_{i \leq T} a_{t,\bar{d}}^{(i)} - a_{t,\bar{d}}^{(T)})$, where $a_{t,\bar{d}}^{(i)}$ and $a_{t,\bar{d}}^{(T)}$ denote the accuracy of task t on a completely unseen domain $\bar{d}$ after learning the i -th and the final task T , respectively.

6.2 Experimental Result

Overall Performance: We first present the overall performance of DGCL in Table 2. Consistent with the analysis in Sec. 4, advanced PET-based CL methods exhibit varying degrees of performance degradation in A_{all} ⁴, especially for unseen domains in

⁴We consider A_{all} as the primary evaluation metric, as it accommodates the overall objective of DGCL in both seen domains and unseen domains.

Table 3: Comparison of computational and parameter overhead. We present the results on Split DigitsDG ($\mathcal{H}_c$ and $\mathcal{P}_d$ for other benchmarks will vary with the number of classes and domains, but the trend is consistent). The training time is evaluated with one-card 3090 GPU, AMD EPYC 7402 (2.8G Hz). $^\dagger\mathcal{H}_c$ and $\mathcal{P}_d$ are preserved but **not** trainable in DGCL. ‡ The DoT parameters are trainable but **not** preserved in DGCL.

Method	$A_{\text{all}}(\%)$	$\mathcal{H}_c^\dagger$	$\mathcal{P}_d^\dagger$	DoT ‡	Trainable	Training Time
L2P [7]	34.62	–	–	–	0.14M	9.23min
DoT-L2P (Cov)	38.17	5.63M	1.41M	2.39M	2.53M	18.25min
DoT-L2P (Var)	38.01	0.01M	1.41M	2.39M	2.53M	18.20min
SLCA (Cov) [11]	51.81	5.63M	–	–	81.82M	40.41min
SLCA (Var) [11]	56.39	0.01M	–	–	81.82M	39.88min
DoT-SLCA (Cov)	61.68	5.63M	1.41M	2.39M	84.21M	42.41min
DoT-SLCA (Var)	62.48	0.01M	1.41M	2.39M	84.21M	41.37min

A_{out} compared to seen domains in A_{in} . L2P and CODA-Prompt generally achieve the highest overall performance in this avenue, since they both employ the ensemble of a prompt pool. SLCA significantly outperforms all PET-based CL methods due to the large capacity of full parameter tuning and the representation recovery for output alignment. In comparison, our proposed DoT serves as a plug-in strategy that significantly improves the performance of state-of-the-art CL baselines under both PET (i.e., L2P) and full parameter tuning (i.e., SLCA) paradigms. Notably, DoT-SLCA improves SLCA mainly in A_{out} rather than A_{in} , suggesting that the benefits of domain transformation are specific to DGCL rather than CIL. DoT-L2P improves L2P in both A_{in} and A_{out} due to the additional benefits of output alignment to representation learning, as analyzed in previous work [11, 44].

Since SLCA approximates the distribution of each class as a Gaussian with a dedicated mean vector and covariance matrix, we use it as the default implementation of DoT-SLCA as well as DoT-L2P to ensure comparison clarity and fairness (denoted as “Cov” in Table 2). However, this implementation incurs parameter overhead of a remarkably high complexity $O(m^2)$ [36], where m denotes the embedding dimension. For example, Office-Home has a total of 65 classes, and the additional parameter overhead is 36.56M with $m = 768$ (around 42.51% of the ViT-B/16 backbone). In this regard, we reduce the covariance matrix to a lightweight variance vector of complexity $O(m)$ (denoted as “Var” in Table 2), which achieves comparable or even better performance with our proposed DoT in DGCL. This implementation results in additional parameter overhead of only 0.05M for Office-Home (around 0.06% of the ViT-B/16 backbone), which is essentially negligible. Accordingly, DoT-SLCA (Var) achieves state-of-the-art performance in DGCL and requires much fewer parameters than the original SLCA (Cov) (see Table 3).

We provide a more detailed analysis of computational and parameter overhead in Table 3. Notably, when DoT performs DGCL, the feature distributions are preserved but not trainable, and the DoT parameters are trainable but not preserved. The simplified variance vector further reduces the cost. Using an SLCA-like output alignment incurs significant computational overhead (DoT-L2P versus L2P), while the training

Table 4: Performance of additional unseen domain and forgetting measurement. We report the average all-domain accuracy A_{all} (removing the completely unseen domain), average unseen-domain accuracy A_{un} , average all-domain forgetting F_{all} (removing the completely unseen domain), and average unseen-domain forgetting F_{un} . We implement the covariance matrix (Cov) and variance vector (Var) for output alignment in DoT and SLCA. All results are averaged over three runs with randomly sampled task sequences. The error bar denotes the standard deviation.

Method	Split CORE50				Split DigitsDG-C			
	A_{all} ($\uparrow$)	A_{un} ($\uparrow$)	F_{in} ($\downarrow$)	F_{out} ($\downarrow$)	A_{all} ($\uparrow$)	A_{un} ($\uparrow$)	F_{in} ($\downarrow$)	F_{out} ($\downarrow$)
DualPrompt [8]	52.99 $\pm$ 2.06	49.86 $\pm$ 5.25	17.34 $\pm$ 1.89	18.36 $\pm$ 4.53	26.08 $\pm$ 2.45	17.69 $\pm$ 1.93	20.85 $\pm$ 8.84	16.83 $\pm$ 11.76
S-Prompt [41]	27.55 $\pm$ 4.35	39.20 $\pm$ 8.64	11.76 $\pm$ 5.88	7.34 $\pm$ 3.46	14.71 $\pm$ 1.77	12.64 $\pm$ 1.41	10.85 $\pm$ 0.61	15.83 $\pm$ 4.47
S-Prompt++ [10]	34.79 $\pm$ 1.27	42.22 $\pm$ 1.65	9.89 $\pm$ 0.40	9.91 $\pm$ 0.99	15.99 $\pm$ 1.77	12.17 $\pm$ 2.40	14.66 $\pm$ 2.84	18.36 $\pm$ 8.65
CODA-Prompt [9]	57.41 $\pm$ 2.46	48.58 $\pm$ 2.44	6.45 $\pm$ 1.48	6.17 $\pm$ 1.62	28.54 $\pm$ 3.32	25.50 $\pm$ 7.09	15.04 $\pm$ 3.86	13.72 $\pm$ 2.65
LAE-PreT [42]	55.48 $\pm$ 1.83	52.38 $\pm$ 4.84	17.51 $\pm$ 3.00	19.17 $\pm$ 4.11	21.89 $\pm$ 1.01	20.92 $\pm$ 3.26	20.88 $\pm$ 5.85	17.83 $\pm$ 9.21
RanPAC [51]	55.38 $\pm$ 0.19	69.72 $\pm$ 2.23	18.16 $\pm$ 2.48	11.25 $\pm$ 4.26	28.54 $\pm$ 3.32	25.50 $\pm$ 7.09	15.04 $\pm$ 3.86	13.72 $\pm$ 2.65
HiDe-PreT [10]	32.21 $\pm$ 3.10	36.70 $\pm$ 4.56	4.03 $\pm$ 0.61	3.75 $\pm$ 0.54	26.45 $\pm$ 2.24	17.97 $\pm$ 4.59	2.37 $\pm$ 1.73	2.22 $\pm$ 0.85
ICON [28]	26.24 $\pm$ 5.17	35.37 $\pm$ 4.64	13.37 $\pm$ 1.06	10.92 $\pm$ 3.88	12.31 $\pm$ 2.61	10.83 $\pm$ 0.51	9.71 $\pm$ 7.13	8.86 $\pm$ 1.97
L2P [7]	55.41 $\pm$ 3.63	53.43 $\pm$ 0.49	15.27 $\pm$ 2.10	15.11 $\pm$ 3.23	32.80 $\pm$ 2.98	27.69 $\pm$ 2.17	19.52 $\pm$ 0.32	16.83 $\pm$ 6.69
DoT-L2P (Cov)	73.43 $\pm$ 0.16	72.73 $\pm$ 3.76	9.06 $\pm$ 0.48	8.80 $\pm$ 0.67	38.29 $\pm$ 2.21	31.67 $\pm$ 5.51	20.14 $\pm$ 2.01	21.19 $\pm$ 3.52
DoT-L2P (Var)	71.38 $\pm$ 0.76	69.03 $\pm$ 2.05	10.89 $\pm$ 1.05	11.23 $\pm$ 1.72	38.06 $\pm$ 2.53	29.75 $\pm$ 5.51	19.30 $\pm$ 3.21	22.28 $\pm$ 4.22
SLCA (Cov) [11]	82.23 $\pm$ 2.28	80.48 $\pm$ 3.18	7.27 $\pm$ 1.89	7.46 $\pm$ 1.69	47.62 $\pm$ 3.17	45.11 $\pm$ 13.50	20.42 $\pm$ 5.35	16.11 $\pm$ 7.40
SLCA (Var) [11]	80.55 $\pm$ 0.96	78.11 $\pm$ 2.75	6.60 $\pm$ 0.41	6.82 $\pm$ 2.63	48.57 $\pm$ 6.05	42.28 $\pm$ 14.31	15.31 $\pm$ 0.40	10.03 $\pm$ 6.90
DoT-SLCA (Cov)	84.74 $\pm$ 1.26	83.14 $\pm$ 2.36	5.99 $\pm$ 0.47	5.67 $\pm$ 0.65	58.74 $\pm$ 4.95	54.75 $\pm$ 5.62	20.43 $\pm$ 1.94	17.45 $\pm$ 5.86
DoT-SLCA (Var)	82.14 $\pm$ 2.16	79.58 $\pm$ 4.01	6.07 $\pm$ 0.49	6.22 $\pm$ 1.79	61.72 $\pm$ 6.35	56.69 $\pm$ 12.02	14.88 $\pm$ 3.07	13.81 $\pm$ 4.96

of DoT parameters is clearly lightweight (DoT-SLCA versus SLCA). As a result, our approach requires only a small amount of additional overhead while achieving strong improvements in DGCL. The additional overhead can be even further reduced by using fewer epochs and prototypes, as shown later in the analysis of Fig. 6.

Impact of Unseen Domain. The DGCL setting emphasizes that the model learns each task from a single domain, while generalizing to all encountered domains out of the task-specific training set. This consideration stems from the fact that there are usually many more tasks (or classes) to be learned than their available domains, consistent with the statistics of widely-used datasets in both DG and CL (Table 1). We further evaluate the capability of DG under more stringent conditions, i.e., the model should generalize all previously learned tasks to a completely unseen domain after DGCL. For example, for Split CORE50 that consists of 10 tasks and 11 domains, we learn each task with a distinct domain and additionally evaluate the average accuracy of these 10 tasks under the 11th domain. For Split DigitsDG that consists of 5 tasks and 4 domains, we reconstruct this benchmark as learning the 5 tasks with each belonging to one of the 3 domains (i.e., there are several tasks learned from the same domain), denoted as Split DigitsDG-C (“-C” stands for “crowded”), and additionally evaluate the average accuracy of these 5 tasks under the 4th domain.

In this way, Split CORE50 and Split DigitsDG-C represent two extreme cases of DGCL: domains being overly sparse (i.e., different domains for each task) versus domains being overly dense (i.e., identical domains for many tasks), allowing for a more comprehensive analysis. As shown in Table 4, our approach provides consistently strong improvements to L2P and SLCA in both cases. Interestingly, the improvement

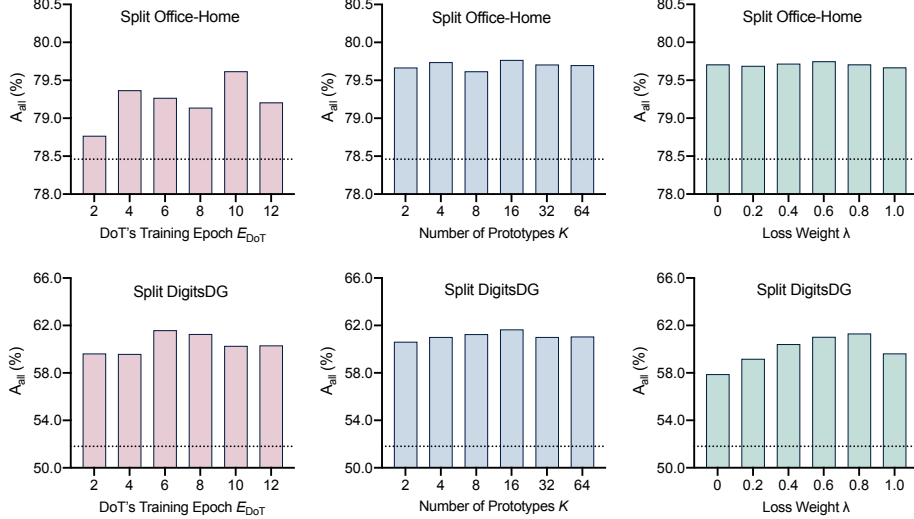


Fig. 6: Impact of hyperparameters. We evaluate the impact of DoT’s training epochs E_{DoT} , number of preserved prototypes K , and loss weight λ with DoT-SLCA (Cov). The dashed line is SLCA (Cov) as the baseline. All results are averaged over three runs with randomly sampled task sequences.

to the average accuracy of unseen domains (i.e., A_{un}) tends to be even more significant than that of all observed domains (i.e., A_{all}), as our cross-task and cross-domain transformation enables PTMs to accumulate domain generalizable knowledge from DGCL. We further evaluate the average all-domain forgetting F_{all} for the maximum decrease of all-domain accuracy and average unseen-domain forgetting F_{un} for the maximum decrease of unseen-domain accuracy in DGCL. In general, our approach achieves comparable F_{all} and F_{un} compared to the corresponding baselines, which further suggests that the improvement to the overall performance is mainly from better generalization to unseen domains. Again, the covariance and variance versions of our approach exhibit comparable performance.

Detailed Analysis of DoT. To provide a more in-depth analysis of our approach, we investigate the impact of three critical hyperparameters, including DoT’s training epochs E_{DoT} , the number of preserved prototypes K , and the loss weight λ . As shown in Fig. 6, these hyperparameters are relatively stable across a wide range of specific values, and deliver consistent improvements over the baseline, i.e., DoT-SLCA (Cov) versus SLCA (Cov). This property ensures the robustness of DoT in response to uncertain changes in real-world applications. The results for all hyperparameters exhibit an overall single-peak pattern, especially in Split DigitsDG that enjoys comparably substantial improvements, suggesting the effectiveness of our designs. In particular, we observe that DoT can already achieve strong improvements by using only 4 epochs and preserving only 2 prototypes (our default setting is 10 epochs and 16 prototypes), which further reduces the computational and parameter overhead.

To further validate the efficacy of our approach, we use t-SNE to visualize the final-layer features of DoT-SLCA in Fig. 7. We compare three types of features: (1)

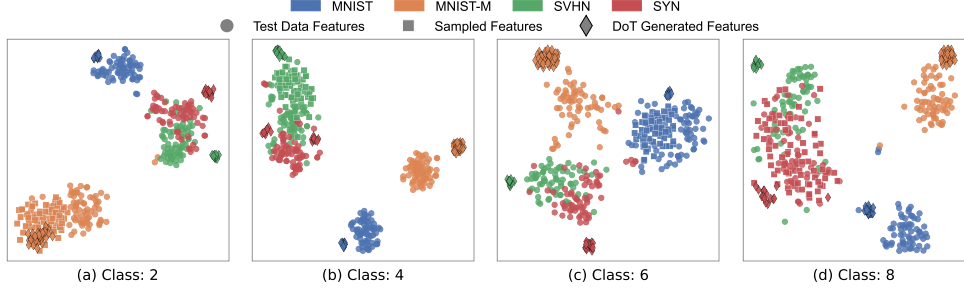


Fig. 7: t-SNE visualization of DoT features. We depict three kinds of final-layer features obtained from DoT-SLCA on Split DigitsDG, including real features extracted from the test set of all domains (circle), in-domain features sampled from the preserved $\mathcal{H}_c$ (square, white edge), and DoT’s generated features of all domains (diamond, black edge). Points are colored according to domain identities. **a-d**, The features of class 2, 4, 6, and 8 from different tasks. Best viewed in color.

real features extracted from the test set of all domains, (2) in-domain features sampled from the preserved $\mathcal{H}_c$, and (3) DoT’s generated features of all domains. It can be seen that the real features of all domains are naturally divided into multiple clusters, and the sampled in-domain features only align with one of them, suggesting the particular challenges of DGCL. In contrast, the generated features exhibit appropriate alignment with the corresponding real-domain clusters, highlighting that the DoT module can effectively capture the domain-relevant information and combine it with semantic-relevant information, so as to address the DGCL problem.

7 Conclusion

In this work, we introduce domain generalizable continual learning (DGCL), a novel and realistic setting that addresses the unique challenges of domain generalization in continual learning. By performing an extensive empirical investigation of DGCL, we expose the severe limitations of state-of-the-art baselines in continually learning representations and aligning outputs for generalizing to unseen domains. To overcome these challenges, we propose an innovative approach that adaptively transforms semantic- and domain-relevant representations with pre-trained models, drawing inspiration from the robust mechanisms of the human brain. Our approach delivers significant improvements in DGCL, functioning as a plug-in strategy that supports both full parameter tuning and parameter-efficient tuning. These contributions not only advance the state-of-the-art in continual learning to accommodate unseen scenarios, but also set a promising direction for visual domain generalization in adapting to temporal-scale changes, with broad implications for deploying adaptive and robust AI systems in real-world applications where environments are both diverse and dynamic.

8 Data Availability

All the data used for this research is publicly accessible and proper citations are also provided in the article.

9 Statements and Declarations

The authors declare no competing interests.

10 Acknowledgment

This work was supported by STI2030-Major Projects (2022ZD0204900), the NSFC Projects (Nos. 62406160, 32021002), Beijing Natural Science Foundation L247011, and the Tsinghua-Peking Joint Center for Life Sciences.

References

- [1] D. Kudithipudi, M. Aguilar-Simon, J. Babb, M. Bazhenov, D. Blackiston, J. Bon-gard, A. P. Brna, S. Chakravarthi Raja, N. Cheney, J. Clune, *et al.*, “Biological underpinnings for lifelong learning machines,” *Nature Machine Intelligence*, 2022.
- [2] K. Patterson, P. J. Nestor, and T. T. Rogers, “Where do you know what you know? the representation of semantic knowledge in the human brain,” *Nature Reviews Neuroscience*, vol. 8, no. 12, pp. 976–987, 2007.
- [3] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, “Domain generalization: A survey,” *IEEE TPAMI*, vol. 45, no. 4, pp. 4396–4415, 2022.
- [4] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and S. Y. Philip, “Generalizing to unseen domains: A survey on domain generalization,” *IEEE TKDE*, vol. 35, no. 8, pp. 8052–8072, 2022.
- [5] L. Wang, X. Zhang, H. Su, and J. Zhu, “A comprehensive survey of continual learning: Theory, method and application,” *IEEE TPAMI*, 2024.
- [6] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural Networks*, 2019.
- [7] Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister, “Learning to prompt for continual learning,” in *CVPR*, 2022.
- [8] Z. Wang, Z. Zhang, S. Ebrahimi, R. Sun, H. Zhang, C.-Y. Lee, X. Ren, G. Su, V. Perot, J. Dy, *et al.*, “Dualprompt: Complementary prompting for rehearsal-free continual learning,” in *ECCV*, 2022.
- [9] J. S. Smith, L. Karlinsky, V. Gutta, P. Cascante-Bonilla, D. Kim, A. Arbelle, R. Panda, R. Feris, and Z. Kira, “Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning,” in *CVPR*, 2023.
- [10] L. Wang, J. Xie, X. Zhang, M. Huang, H. Su, and J. Zhu, “Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality,” *NeurIPS*, 2023.
- [11] G. Zhang, L. Wang, G. Kang, L. Chen, and Y. Wei, “Slca: Slow learner with classifier alignment for continual learning on a pre-trained model,” in *ICCV*, 2023.

- [12] B. Lei, B. Kang, Y. Hao, H. Yang, Z. Zhong, Z. Zhai, and Y. Zhong, “Reconstructing a new hippocampal engram for systems reconsolidation and remote memory updating,” *Neuron*, 2024.
- [13] P. W. Frankland and B. Bontempi, “The organization of recent and remote memories,” *Nature Review Neuroscience*, 2005.
- [14] D. N. Barry and E. A. Maguire, “Remote memory and the hippocampus: A constructive critique,” *TICS*, vol. 23, no. 2, pp. 128–142, 2019.
- [15] G. Blanchard, G. Lee, and C. Scott, “Generalizing from several related classification tasks to a new unlabeled sample,” *NeurIPS*, vol. 24, 2011.
- [16] D. Hendrycks and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” in *ICLR*, 2018.
- [17] K. Zhou, Y. Yang, T. Hospedales, and T. Xiang, “Learning to generate novel domains for domain generalization,” in *ECCV*, pp. 561–578, Springer, 2020.
- [18] X. Wang, J. Zhang, L. Qi, and Y. Shi, “Generalizable decision boundaries: Dualistic meta-learning for open set domain generalization,” in *ICCV*, pp. 11564–11573, 2023.
- [19] X. Jin, C. Lan, W. Zeng, Z. Chen, and L. Zhang, “Style normalization and restitution for generalizable person re-identification,” in *CVPR*, pp. 3143–3152, 2020.
- [20] M. Wang and W. Deng, “Deep visual domain adaptation: A survey,” *Neurocomputing*, vol. 312, pp. 135–153, 2018.
- [21] J. Liang, R. He, and T. Tan, “A comprehensive survey on test-time adaptation under distribution shifts,” *IJCV*, pp. 1–34, 2024.
- [22] Q. Wang, O. Fink, L. Van Gool, and D. Dai, “Continual test-time domain adaptation,” in *CVPR*, pp. 7201–7211, 2022.
- [23] Y. Wu, L.-K. Huang, R. Wang, D. Meng, and Y. Wei, “Meta continual learning revisited: Implicitly enhancing online hessian approximation via variance reduction,” in *ICLR*, 2024.
- [24] Y. Wu, H. Wang, P. Zhao, Y. Zheng, Y. Wei, and L.-K. Huang, “Mitigating catastrophic forgetting in online continual learning by modeling previous task interrelations via pareto optimization,” in *ICML*, 2024.
- [25] G. M. Van de Ven and A. S. Tolias, “Three scenarios for continual learning,” *arXiv preprint arXiv:1904.07734*, 2019.
- [26] J.-Y. Moon, K.-H. Park, J. U. Kim, and G.-M. Park, “Online class incremental learning on stochastic blurry task boundary via mask and visual prompt tuning,” in *ICCV*, pp. 11731–11741, 2023.
- [27] Z. Kang, L. Wang, X. Zhang, and K. Alahari, “Advancing prompt-based methods for replay-independent general continual learning,” in *ICLR*, 2025.
- [28] M.-Y. Park, J.-H. Lee, and G.-M. Park, “Versatile incremental learning: Towards class and domain-agnostic incremental learning,” in *ECCV*, pp. 271–288, Springer, 2025.
- [29] L. Wang, M. Zhang, Z. Jia, Q. Li, C. Bao, K. Ma, J. Zhu, and Y. Zhong, “Afec: Active forgetting of negative transfer in continual learning,” *NeurIPS*, 2021.
- [30] Y. Lyu, L. Wang, X. Zhang, Z. Sun, H. Su, J. Zhu, and L. Jing, “Overcoming recency bias of normalization statistics in continual learning: Balance and

- adaptation,” *NeurIPS*, vol. 36, pp. 25475–25494, 2023.
- [31] L. Wang, X. Zhang, K. Yang, L. Yu, C. Li, L. Hong, S. Zhang, Z. Li, Y. Zhong, and J. Zhu, “Memory replay with data compression for continual learning,” in *ICLR*, 2021.
 - [32] H. Yan, L. Wang, K. Ma, and Y. Zhong, “Orchestrate latent expertise: Advancing online continual learning with multi-level supervision and reverse self-distillation,” in *CVPR*, pp. 23670–23680, 2024.
 - [33] L. Wang, X. Zhang, Q. Li, J. Zhu, and Y. Zhong, “Coscl: Cooperation of small continual learners is stronger than a big one,” in *ECCV*, 2022.
 - [34] L. Wang, X. Zhang, Q. Li, M. Zhang, H. Su, J. Zhu, and Y. Zhong, “Incorporating neuro-inspired adaptability for continual learning in artificial intelligence,” *Nature Machine Intelligence*, 2023.
 - [35] Y. Wu, H. Piao, L.-K. Huang, R. Wang, W. Li, H. Pfister, D. Meng, K. Ma, and Y. Wei, “Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning,” in *ICLR*, 2025.
 - [36] L. Wang, J. Xie, X. Zhang, H. Su, and J. Zhu, “Hide-pet: Continual learning via hierarchical decomposition of parameter-efficient tuning,” *arXiv preprint arXiv:2407.05229*, 2024.
 - [37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *NeurIPS*, 2017.
 - [38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *ICLR*, 2020.
 - [39] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, “Learning multiple visual domains with residual adapters,” *NeurIPS*, 2017.
 - [40] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.
 - [41] Y. Wang, Z. Huang, and X. Hong, “S-prompts learning with pre-trained transformers: An occam’s razor for domain incremental learning,” *NeurIPS*, 2022.
 - [42] Q. Gao, C. Zhao, Y. Sun, T. Xi, G. Zhang, B. Ghanem, and J. Zhang, “A unified continual learning framework with general parameter-efficient tuning,” in *ICCV*, 2023.
 - [43] M. D. McDonnell, D. Gong, A. Parveneh, E. Abbasnejad, and A. v. d. Hengel, “Ranpac: Random projections and pre-trained models for continual learning,” *NeurIPS*, 2023.
 - [44] G. Zhang, L. Wang, G. Kang, L. Chen, and Y. Wei, “Slca++: Unleash the power of sequential fine-tuning for continual learning with pre-training,” *arXiv preprint arXiv:2408.08295*, 2024.
 - [45] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, “Deep hashing network for unsupervised domain adaptation,” in *CVPR*, pp. 5018–5027, 2017.
 - [46] V. Lomonaco and D. Maltoni, “Core50: a new dataset and benchmark for continuous object recognition,” in *CoRL*, 2017.
 - [47] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, “Moment matching

- for multi-source domain adaptation,” in *ICCV*, pp. 1406–1415, 2019.
- [48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
 - [49] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, *et al.*, “Reading digits in natural images with unsupervised feature learning,” in *NIPS*, vol. 2011, p. 4, 2011.
 - [50] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in *ICML*, pp. 1180–1189, 2015.
 - [51] M. D. McDonnell, D. Gong, A. Parvaneh, E. Abbasnejad, and A. van den Hengel, “Ranpac: Random projections and pre-trained models for continual learning,” *NeurIPS*, vol. 36, 2024.
 - [52] D.-W. Zhou, H.-L. Sun, J. Ning, H.-J. Ye, and D.-C. Zhan, “Continual learning with pre-trained models: A survey,” *arXiv preprint arXiv:2401.16386*, 2024.
 - [53] H.-L. Sun, D.-W. Zhou, H.-J. Ye, and D.-C. Zhan, “Pilot: A pre-trained model-based continual learning toolbox,” *arXiv preprint arXiv:2309.07117*, 2023.

Algorithm 1 Training Algorithm of DoT in DGCL

```
1: Input: Task sequence  $\{\mathcal{D}_1, \dots, \mathcal{D}_T\}$  with domains  $\{d_1, \dots, d_T\}$ 
2: Hyperparameters: DoT's training epochs  $E_{\text{DoT}}$ , OA's training epochs  $E_{\text{OA}}$ ,
   prototype count  $K$ , loss weight  $\lambda$ 
3: Phase 1: Task Training and Distribution Accumulation
4: Initialize semantic memory  $\mathcal{H} \leftarrow \emptyset$ , domain memory  $\mathcal{P} \leftarrow \emptyset$ 
5: for each task  $t = 1$  to  $T$  do
6:    $\triangleright$  Standard CL training on  $\mathcal{D}_t$ 
7:   Update  $f_\theta, h_\psi$  via  $\mathcal{L}_t$  in Eq.(1)
8:    $\triangleright$  Accumulate semantic distributions
9:   for each class  $c \in \mathcal{Y}_t$  do
10:     $\{\mathbf{r}_n^{(L)}\} \leftarrow$  final-layer features of class  $c$ 
11:     $\mathcal{H}_c \leftarrow \mathcal{N}(\text{mean}(\{\mathbf{r}_n^{(L)}\}), \text{cov}(\{\mathbf{r}_n^{(L)}\}))$ 
12:     $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathcal{H}_c\}$ 
13:   end for
14:    $\triangleright$  Accumulate domain prototypes
15:    $\{\mathbf{R}_n\} \leftarrow$  intermediate features from  $\mathcal{D}_t$ 
16:    $\mathcal{P}_{d_t} \leftarrow \text{RandomSample}(\{\mathbf{R}_n\}, K)$ 
17:    $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathcal{P}_{d_t}\}$ 
18: end for
19:
20: Phase 2: Attention-Based Domain Transformation
21: Initialize DoT parameters  $\mathbf{W}_{\text{sem}}, \mathbf{W}_{\text{dom}}, \mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_O$ 
22: for epoch = 1 to  $E_{\text{DoT}}$  do
23:   Sample domain  $d_i \sim \mathcal{P}$ , class  $c_j \sim \mathcal{H}$ 
24:    $\mathbf{r}_{c_j}^{(L)} \sim \mathcal{H}_{c_j}, \mathbf{R}_{d_i} \sim \mathcal{P}_{d_i}$ 
25:    $\triangleright$  DoT feature transformation
26:    $\hat{\mathbf{r}} \leftarrow \text{DoT}(\mathbf{r}_{c_j}^{(L)}, \mathbf{R}_{d_i})$  in Eq.(2-4)
27:    $\triangleright$  Contrastive alignment
28:   Compute  $\mathcal{L}_{\text{cls}}$  in Eq.(6) and  $\mathcal{L}_{\text{dom}}$  in Eq.(7)
29:   Update DoT params via  $\nabla \mathcal{L}_{\text{DoT}} = \nabla[(1 - \lambda)\mathcal{L}_{\text{cls}} + \lambda\mathcal{L}_{\text{dom}}]$  in Eq.(8)
30: end for
31:
32: Phase 3: Output Alignment with Synthesized Features
33: for epoch = 1 to  $E_{\text{OA}}$  do
34:   for each  $(d_i, c_j)$  pair in  $\mathcal{P} \times \mathcal{H}$  do
35:     $\mathbf{r}_{c_j}^{(L)} \sim \mathcal{H}_{c_j}, \mathbf{R}_{d_i} \sim \mathcal{P}_{d_i}$ 
36:     $\hat{\mathbf{r}}^{(L)} \leftarrow \text{DoT}(\mathbf{r}_{c_j}^{(L)}, \mathbf{R}_{d_i})$ 
37:    Update  $h_\psi$  via  $\nabla \mathcal{L}_{\text{OA}} = \nabla[\mathcal{L}_{\text{ce}}(h_\psi(\mathbf{r}_{c_j}^{(L)}), c_j) + \mathcal{L}_{\text{ce}}(h_\psi(\hat{\mathbf{r}}^{(L)}), c_j)]$  in Eq.(9)
38:   end for
39: end for
40: return Frozen  $f_\theta$ , aligned  $h_\psi$ 
```
