One Size Fits All for Semantic Shifts: Adaptive Prompt Tuning for Continual Learning

Doyoung Kim [1] **Susik Yoon** [2] **Dongmin Park** [3] **Youngjun Lee** [1] **Hwanjun Song** [1] **Jihwan Bang** [1] **Jae-Gil Lee** [1]

Abstract

In real-world continual learning (CL) scenarios, tasks often exhibit intricate and unpredictable semantic shifts, posing challenges for *fixed* prompt management strategies which are tailored to only handle semantic shifts of *uniform* degree (i.e., uniformly mild or uniformly abrupt). To address this limitation, we propose an *adaptive* prompting approach that effectively accommodates semantic shifts of *varying* degree where mild and abrupt shifts are mixed. AdaPromptCL employs the assign-and-refine semantic grouping mechanism that dynamically manages prompt groups in accordance with the semantic similarity between tasks, enhancing the quality of grouping through continuous refinement. Our experiment results demonstrate that AdaPromptCL outperforms existing prompting methods by up to 21.3%, especially in the benchmark datasets with diverse semantic shifts between tasks.

1. Introduction

Recently, rehersal-free continual learning (CL) methods (Wang et al., 2022c;a;b; Gao et al., 2023) have gained interest because of their superior performance even without using previous samples. Small learnable parameters, known as *prompts*, are added to input data of various tasks for refining a pre-trained model. These methods can be further categorized into *universal* and *specific* prompting methods, depending on how the prompts are managed for sequentially arriving tasks. Universal prompting methods (e.g., LAE (Gao et al., 2023)) train a fixed set of prompts consistently used for all tasks. This strategy captures generalizable knowledge accumulated from all prior tasks which is believed to be beneficial for upcoming similar tasks. On the other hand, specific prompting methods (e.g., S-Prompts (Wang et al., 2022a)) train prompts individually designated for each task, mainly to alleviate catastrophic forgetting which becomes more pronounced with highly divergent tasks.

While these two approaches have proven effective in their respective CL scenarios—similar tasks for universal prompting and distinct tasks for specific prompting, real-world scenarios frequently involve intricate and unpredictable semantic shifts between tasks over time. Let us specify the *degree* of semantic shifts by considering whether a higher-level category of samples undergoes a change. For example, in Figure 1(a), the shift from Task A to Task B is *mild* because both tasks involve product classification in the food and beverage category; on the other hand, the shift from Task B to Task C is *abrupt* because Task C belongs to the apparel and accessories category. This example illustrates a combination of *semantic shifts of varying degrees*.

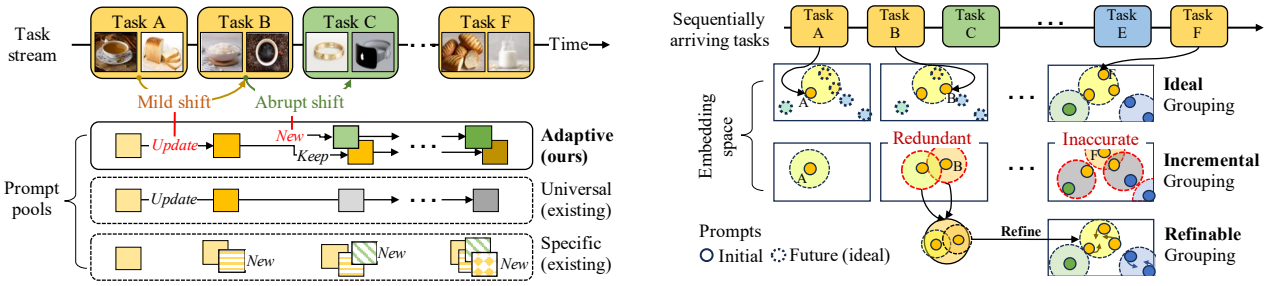
Existing prompting methods, however, are not suitable to accommodate semantic shifts of varying degrees, but can only accommodate those of *uniform* degree. In detail, universal prompting supports uniformly mild shifts, while specific prompting supports uniformly abrupt shifts. On the contrary, for a combination of mild and abrupt shifts in Figure 1(a), universal prompting incurs insufficient (one instead of two) prompts, and specific prompting incurs redundant (four instead of two) prompts. Consequently, the existing methods reach a sub-optimal solution, compromising between forgetting prevention and knowledge transfer.

In this paper, we contend that prompting methods should accommodate semantic shifts of *varying* degrees. As a consequence, we propose an *adaptive* prompting strategy to address the limited effectiveness of the *fixed* prompting strategy—either universal or specific. By carefully considering task semantics, adaptive prompting achieves minimal yet adequate prompts that effectively accommodate varying semantic shifts between tasks. As shown in Figure 1(a), it appropriately updates an existing prompt for mildly shifted tasks (e.g., Task B) and introduces a new prompt for abruptly shifted tasks (e.g., Task C).

The key issue here is grouping semantically similar tasks so as to *share* a prompt among those tasks. This grouping is es-

[1]KAIST, Daejeon, Republic of Korea [2]Korea University, Seoul, Republic of Korea [3]KRAFTON, Seoul, Republic of Korea. Correspondence to: Jae-Gil Lee <jaegil@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.



(a) Comparison of prompting strategies.

(b) Challenge in semantic grouping.

Figure 1. **Key idea of AdaPromptCL**: (a) shows the comparison of our adaptive prompting against existing prompting CL methods on a practical CL scenario; (b) depicts the inherent challenges in implementing adaptive prompting. Colors indicate task semantics.

pecially challenging because the number and nature of tasks are *not* predetermined in a real-world CL setting. It is widely known that the quality of online clustering (e.g., BIRCH) can be influenced by the order of data insertion (Zhang et al., 1996). As shown in Figure 1(b), with prior knowledge on future tasks (see "Ideal Grouping"), a single optimal prompt can be easily derived for the three tasks in yellow. In practice, as tasks arrive sequentially (see "Incremental Grouping"), naïve incremental grouping may inadvertently create *redundant* prompts that separately cover similar tasks; even worse, these redundant prompts result in *inaccurate* prompts that contain the tasks of different semantics.

Our prompt-based CL framework, named AdaPromptCL, incorporates an innovative mechanism of *assign-and-refine* semantic grouping in order to realize adaptive prompting. Because the future tasks are unknown as aforementioned, in the assignment step, each new task is added to one of existing groups or a new group based on its similarity to the existing groups. Importantly, to prepare potential refinement of groups, potentially possible groups are stored as well. In the refinement step, if an incoming task (e.g., Task F in Figure 1(b)) provides a clear clue to a better semantic grouping, the existing groups are refined accordingly, and their prompts are retrieved from those maintained for potential groups (see "Refinable Grouping").

To the best of our knowledge, this is the first work to examine the effectiveness of prompt tuning-based methods in CL across a spectrum of task semantic shifts. In Section 5, a thorough assessment is performed by categorizing CL scenarios into uniform and varying semantic shifts. As briefed in Figure 2, AdaPromptCL demonstrates consistently outstanding performance across both scenarios without regard to the degree of semantic shifts, whereas the fixed prompting methods achieve satisfactory results only within their respective scenarios. In particular, when applied to a realistic CL scenario with varying semantic shifts, AdaPromptCL significantly surpasses the existing methods by up to 13.6% in terms of accuracy.

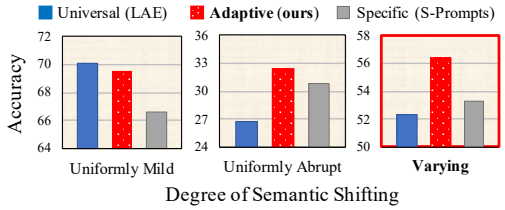


Figure 2. Performance of representative methods for different prompting strategies across various degrees of semantic shifts.

2. Related Work

2.1. Continual Learning

Continual learning (CL) confronts the stability-plasticity dilemma inherent in learning a sequence of tasks. This dilemma describes the challenge of preserving acquired knowledge (stability) while incorporating new information (plasticity). Representative CL approaches can be categorized into *rehearsal-based* methods, which rely on retaining and replaying subsets of past data (Rolnick et al., 2019; Bang et al., 2021; Buzzega et al., 2020; Aljundi et al., 2019a; Shim et al., 2021; Aljundi et al., 2019b; Prabhu et al., 2020; Koh et al., 2022; Yoon et al., 2023; Kim et al., 2024); *regularization* techniques, which constrain updates to preserve past knowledge (Chaudhry et al., 2018; Kirkpatrick et al., 2017); and *architectural* solutions that reconfigure the model structure for new tasks (Mallya & Lazebnik, 2018; Aljundi et al., 2017; Lee et al., 2020). These strategies are detailed in several comprehensive surveys (De Lange et al., 2021; Mai et al., 2022; Mundt et al., 2023).

2.2. Prompt-based Continual Learning

The emergence of *rehearsal-free* methods has offered a novel perspective on CL by exploiting the potential of pre-trained models, such as ViT (Dosovitskiy et al., 2021), which have demonstrated a remarkable ability to grasp general representation. The rehearsal-free methods perform fine-tuning through prompts, i.e., small, learnable weight parameters that refine the model's representation. This ap-

proach brings significant memory and computational efficiency gains since it only modifies a small part of the model's parameters for each new task.

The rehearsal-free strategies are divided into *universal* and *specific* prompting methods, as aforementioned. In universal prompting, VPT (Jia et al., 2022) and L2P (Wang et al., 2022c) respectively optimize a single prompt and a shared pool of prompts; LAE (Gao et al., 2023) advances them by accumulating and ensembling prompts over time. In specific prompting, S-Prompts (Wang et al., 2022a) aims to train prompts to individual tasks to address catastrophic forgetting. Meanwhile, DP (Wang et al., 2022b) employs the simultaneous use of both universal and specific prompts for all tasks. However, most rehearsal-free methods have not adequately considered the semantic relationships between tasks, which could lead to suboptimal outcomes due to the lack of shared learning across semantically related tasks. This paper addresses this gap by integrating task semantics into prompt-tuning strategies.

2.3. Semantic Grouping for Multi-Task Learning

In multi-task learning (MTL), the ability to effectively share knowledge across diverse tasks is crucial for enhancing problem-solving capabilities (Guo et al., 2020; Wu et al., 2020; Liu et al., 2023; Yang et al., 2023). However, arbitrarily combining tasks can result in conflicting task groups in which the tasks interfere with one another, resulting in a degradation in performance, a phenomenon known as negative transfer (Wu et al., 2020). Addressing this issue, *semantic-based task grouping* methods (Zamir et al., 2018; Standley et al., 2020; Fifty et al., 2021; Song et al., 2022; Liu et al., 2023) have been introduced, which emphasize clustering semantically similar tasks to amplify positive transfer and boost performance. An intrinsic need for specific models for distinct task groups has driven the adoption of parameter-efficient techniques such as prompt tuning (Liu et al., 2023). However, their direct adoption poses challenges because they assume simultaneous availability of all tasks, contradicting practical CL environments.

3. Preliminaries

3.1. Continual Learning

We consider a sequence of tasks $\mathcal{T} = \langle \tau^1, \ldots, \tau^{|\mathcal{T}|} \rangle$ for a data stream $\mathcal{D} = \langle D^1, \ldots, D^{|\mathcal{T}|} \rangle$, where each sub-dataset $D_t = \{(x_i^t, y_i^t)\}_i$ for the t-th task is derived from a joint distribution of an input space $\mathcal{X}^t$ and a label space $\mathcal{Y}^t$. The goal of CL is to continuously learn a classifier that maximizes test accuracy for all encountered tasks $\{\tau^1, \tau^2, \ldots, \tau^t\}$, without having access to data streams $D^{t'<t}$ of prior tasks. These data streams may come from diverse environments, and thus the associated tasks exhibit semantic shifts of varying degrees, calling for adaptive CL approaches.

3.2. Prompt-based Continual Learning

Definition of a Prompt. Recent prompt-based CL methods (Wang et al., 2022b;c; Gao et al., 2023) mostly employ a pretrained backbone (e.g., ViT (Dosovitskiy et al., 2021)) with L transformer blocks, where each of these blocks utilizes a multi-head attention module to extract pertinent features. A *prompt* refers to a set of vectors that provide auxiliary context or guide the attention mechanism towards specific patterns in the data for a task,

$$\mathrm{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_p] \in \mathbb{R}^{p \times d}, \tag{1}$$

where p and d are the number and the dimensionality of tokens $\mathbf{p}_i$, respectively. The prompt serves as a prefix to the input for the transformer block's attention module (Wang et al., 2022b; Gao et al., 2023).

Prompt Tuning and Inference. Universal and specific strategies are used for managing and selecting prompts. First, *universal* prompting employs a constant prompt P on all tasks. The optimization objective for a given task τ^t with D^t is to minimize the cross-entropy loss ℓ_{ce},

$$\min_{\mathrm{P},\mathrm{k},\phi} \ell_{ce}(f_\phi(f([x^t; \mathrm{P}])), y^t), \tag{2}$$

where f_ϕ is a classifier, and f is a pre-trained model that receives the concatenation of the input tokens x^t and the prompt P and then returns the [CLS] token for prediction. Second, *specific* prompting employs a collection of task-specific prompts $\{\mathrm{P}^t \mid \tau^t \in \mathcal{T}\}$ and learnable prompt keys $\{\mathrm{k}^t \mid \tau^t \in \mathcal{T}\}$ to determine the prompt to be used. The optimization objective is to minimize the distance between the input and the prompt key, in addition to minimizing the cross-entropy loss,

$$\min_{\mathrm{P}^t,\mathrm{k}^t,\phi} \ell_{ce}(f_\phi(f([x^t; \mathrm{P}^t])), y^t) + \lambda \mathrm{d}(f(x^t), \mathrm{k}^t), \tag{3}$$

where λ denotes a balancing coefficient and $\mathrm{d}(\cdot, \cdot)$ represents a distance function.

During inference, given a testing instance x, the matching prompt $\mathrm{P}^{\hat{t}}$ is chosen to minimize the distance between $f(x^t)$ and its prompt key k^t,

$$\hat{t} = \arg\min_t \mathrm{d}(f(x^t), \mathrm{k}^t). \tag{4}$$

4. AdaPromptCL: Adaptive Prompting via Semantic Grouping

4.1. Problem Statement

Our *adaptive* prompting is in between the two extremes—universal and specific prompting. That is, at the t-th task, while universal and specific prompting strategies maintain *one* and t prompts, respectively, our adaptive prompting strategy manages *from one to* t prompts depending on the

semantic shifts observed so far. Thus, the key challenge is to lie in the management of the best collection of prompts, which is realized by our novel *semantic grouping* process. A prompt is assigned to each semantic group, with a variable number of prompts ranging from *one* to t. Specifically, for a sequence of CL tasks $\mathcal{T}$, this grouping process maintains $\mathcal{G} = \{G_1, \ldots, G_{|\mathcal{G}|}\}$ that minimizes

$$J(\mathcal{G}) = \sum_{G \in \mathcal{G}} \sum_{\tau^i, \tau^j \in G} \mathrm{d}\big(\mathrm{s}(\tau^i), \mathrm{s}(\tau^j)\big) + \alpha|\mathcal{G}|, \quad (5)$$

where G is a group of tasks and $\mathrm{s}(\tau)$ is a semantic representation of a task τ which can be derived by a task-specific prompt P_τ[1]. $\alpha|\mathcal{G}|$ penalizes an excessive number of groups so that they capture the general knowledge from the tasks. A group-wise prompt P_G is assigned to each group G.

Finding optimal semantic groups $\mathcal{G}^*$ for all tasks $\mathcal{T}$ through optimization of Eq. (5) is infeasible in CL, as the data streams of future tasks $D^{t'>t}$ are unknown in practice. A straightforward solution is an incremental greedy approach, where each incoming task is optimally assigned to an existing group or initiated as a new group, depending on its semantic similarity to the current set of groups. However, as demonstrated in Figure 1(b), this method may result in a local-optimal grouping. *Refining* this local-optimal grouping whenever possible is essential to attain the global-optimal grouping for all tasks.

4.2. Overview of AdaPromptCL

Figure 3 illustrates the *assign-and-refine* semantic grouping framework of AdaPromptCL. First, the *assignment* step adds an incoming task τ^t to an appropriate semantic group and prepares potential refinement by proactively reserving *prospective* semantic groups and their prompts. Second, the *refinement* step improves the quality of semantic grouping if possible and reflects a refinement by retrieving the prospective prompts. Figure 4 as well as Appendix A detail each step.

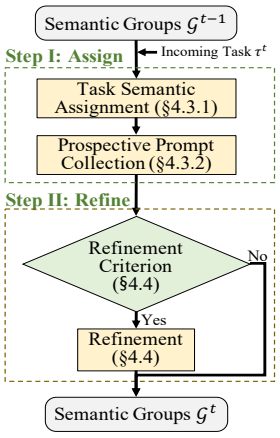


Figure 3. Overall flow.

4.3. Step I: Semantic Assignment

4.3.1. Task Semantic Assignment

For an incoming task τ^t, we form a set of *semantic groups* $\mathcal{G}^t = \{G_1^t, \cdots, G_{|\mathcal{G}^t|}^t\}$ in Definition 1.

[1]Task-specific prompts are known to capture the inherent patterns within the distributions of task data. See Definition 2 for details.

Definition 1. (Semantic Group) A *semantic group* G_i^t, given all seen tasks $\mathcal{T}^t = \langle \tau^1, \cdots, \tau^t \rangle$, is a subset of tasks that are within R from the centroid of the group in the semantic representation space. Formally,

$$G_i^t = \{\tau \in \mathcal{T}^t \mid \delta(G_i^t) \leq R\}, \quad (6)$$

where $\delta(G_i^t)$ is the average distance (Zhang et al., 1996) from a semantic representation $s(\tau)$ ($\tau \in G_i^t$) to the group centroid, calculated as

$$\delta(G_i^t) = \sqrt{\frac{1}{|G_i^t|} \sum_{\tau \in G_i^t} \big(s(\tau) - \mathrm{Centroid}(G_i^t)\big)^2}, \quad (7)$$

where $\mathrm{Centroid}(G_i^t)$ is the mean vector of the semantic representations of the tasks in G_i^t.

Task Semantic Extraction. Given a task τ, we obtain a warm-up prompt $\hat{\mathrm{P}}^\tau$ by the objective goal in Eq. (3). Then, the *semantic representation* of a given task is extracted as specified in Definition 2.

Definition 2. (Task Semantic Representation). Given a warm-up prompt $\hat{\mathrm{P}}^\tau$ for a task τ, the *task semantic representation* $\mathrm{s}(\tau)$ is computed as

$$\mathrm{s}(\tau) = \mathrm{Normalize}\left(\mathrm{AvgPool}(\hat{\mathrm{P}}^\tau)\right) \in \mathbb{R}^d, \quad (8)$$

where AvgPool$(\cdot)$ pools the prompt averaged over all p tokens and Normalize$(\cdot)$ performs l_2-normalization.

Task Assignment. Given the previous semantic groups $\mathcal{G}^{t-1}$ and the current task τ^t, we obtain new groups $\mathcal{G}^t$ by using an assignment function $\mathcal{A}_R$ with a threshold R, i.e., $\mathcal{G}^t = \mathcal{A}_R(\tau^t; \mathcal{G}^{t-1})$. If *neither* of the previous groups $\mathcal{G}^{t-1}$ is semantically relevant to the task τ^t, a new group is initialized for τ^t; otherwise, it is assigned to the closest group (e.g., the assignment of τ^t to G_2^t in Figure 4). Formally, $\mathcal{A}_R(\tau^t; \mathcal{G}^{t-1})$ is defined as

$$\begin{cases} \mathcal{G}^{t-1} \cup \{\tau^t\} & \text{if } \delta(G_i \cup \{\tau^t\}) > R \ \ \forall G_i \in \mathcal{G}^{t-1} \\ \mathcal{G}^{t-1} \backslash G_*^{t-1} \cup \{G_*^{t-1} \cup \{\tau^t\}\} & \text{if } \delta(G_*^{t-1} \cup \{\tau^t\}) \leq R, \end{cases} \quad (9)$$

where $G_*^{t-1} = \arg\min_{G_i^{t-1} \in \mathcal{G}^{t-1}} \delta(G_i^{t-1} \cup \{\tau^t\})$.

4.3.2. Prospective Prompt Collection

To avoid accessing historical tasks, we need to keep a repository of *prospective prompts* so that a refined semantic group is equipped with the prompt that has been adapted to all tasks within the refined semantic group. Thus, this process requires finding *prospective semantic groups* that likely appear in the future by semantic refinement. We note that the prompts for the prospective semantic groups should be learned and reserved for each incoming task because accessing previous tasks is restricted in CL. For example, in

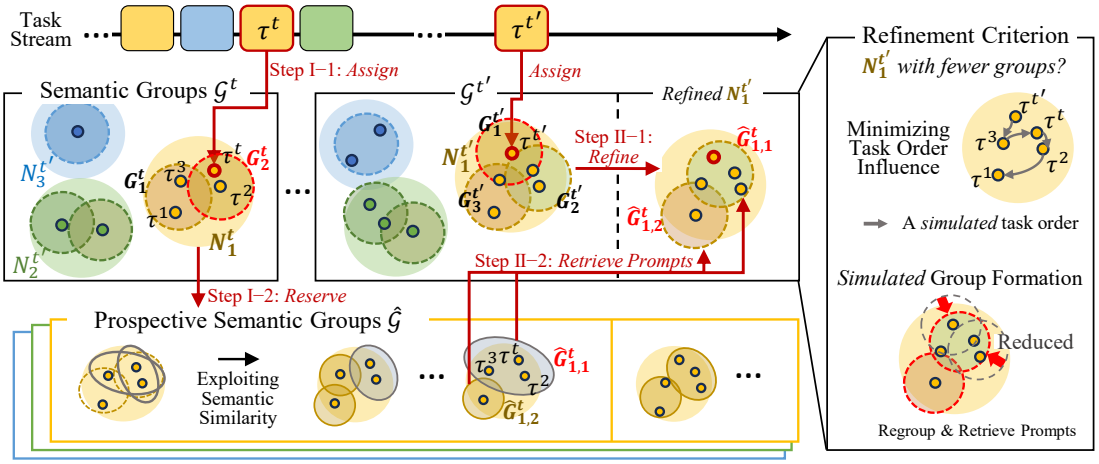


Figure 4. Detailed view of the assign-and-refine semantic grouping process. **I**. The assignment step adds the task τ^t to the group G_2^t and reserves a prospective group $\hat{G}_{1,1}^t$ with its prompt. **II**. The refinement step, assuming that a refinement is needed when the task $\tau^{t'}$ is received, three groups $G_1^{t'}$, $G_2^{t'}$, and $G_3^{t'}$ are reduced to fewer prospective groups $\hat{G}_{1,1}^t$ and $\hat{G}_{1,2}^t$, with their prompts retrieved.

Figure 4, the prompt for the semantic group G_2^t has been adapted to the tasks τ^2 and τ^t; however, the prompt that has been adapted to τ^2, τ^3 and τ^t is also proactively reserved because these three tasks can belong to a certain prospective semantic group $\hat{G}_{1,1}^t$ through future refinement.

To efficiently derive prospective semantic groups, we divide the entire set of tasks into a set of *neighboring task sets* $\mathcal{N}^t = \{N_1^t, \cdots, N_{|\mathcal{N}^t|}^t\}$, where each neighboring task set contains the tasks close to one another (e.g., τ^1, τ^2, τ^3, and τ^t in N_1^t in Figure 4). The neighboring task sets are identified using the same assignment function $\mathcal{A}_{\gamma R}(\tau_t; \mathcal{N}^{t-1})$ with a larger threshold γR where $\gamma > 1$.

Prospective Semantic Groups. The prospective semantic groups are the groups of similar tasks, which do *not* exist in the current set $\mathcal{G}^t$ of semantic groups. Thus, in order to acquire different groupings, we apply a different clustering method such as k-means to each of the neighboring task sets (Bradley & Fayyad, 1998). This clustering procedure is very efficient because the scope of clustering is locally confined to each neighboring task set.

Specifically, the k-means clustering method is executed multiple times for each neighboring task set, and the resulting clusters are added to the set $\hat{\mathcal{G}}$ of prospective semantic groups. That is, a set of prospective semantic groups from a neighboring task set N_i^t is defined by

$$\{\hat{G}_{i,1}^t, \cdots, \hat{G}_{i,|N_i^t|}^t\} = k\text{-means}(N_i^t). \tag{10}$$

For example, in Figure 4, $\{\hat{G}_{1,1}^t, \hat{G}_{1,2}^t\}$ is derived from N_1^t. While executing the k-means clustering method, the number k of clusters is determined to maximize the silhouette score (Rousseeuw, 1987), which is a widely-used clustering quality indicator, as done by Kaufman & Rousseeuw 2009.

Last, each group $\hat{G}_{i,j}^t$ is equipped with a *prospective prompt* that is trained on participating tasks and further adapted to a new task. Again in Figure 4, the prompt for $\hat{G}_{1,1}^t$ is proactively adapted to τ^2, τ^3 and τ^t and reserved, even though it is not yet used for CL. Please see Appendix B for the details and complexity analysis.

4.4. Step II: Semantic Refinement

AdaPromptCL evaluates whether group refinement is necessary and proceeds with the refinement when it is beneficial.

Refinement Criterion. Intuitively, we examine the contribution of the new task to the reduction of semantic groups. Since each semantic group has the same coverage (i.e., the threshold R), a set of fewer semantic groups implies superior generalization of the underlying semantic contexts, thereby optimizing our objective goal in Eq. (5). That is, the goal of refinement is to achieve $|\mathcal{G}^t| \leq |\mathcal{G}^{t-1}|$. An example of this refinement is shown in Figure 4, where the three semantic groups $G_1^{t'}$, $G_2^{t'}$, and $G_3^{t'}$ are reduced to the two semantic groups $\hat{G}_{1,1}^t$ and $\hat{G}_{1,2}^t$.

Here, it is important to mitigate the influence of task order inherent to the CL environment, since the order of data insertion influences online grouping outcomes (Zhang et al., 1996; Bradley & Fayyad, 1998). Accordingly, we implement simulation-based clustering techniques (Celebi et al., 2013; Gama et al., 2014) that permute task orders within each set N_i^t during grouping to identify the one with the fewest semantic groups. Formally, the optimal task order, yielding the fewest groups, is determined by

$$\bar{N}_i^t = \arg\min_{\bar{N}_i^t} |\bar{\mathcal{A}}_R(\bar{N}_i^t)| \;\; s.t. \;\; \bar{\mathcal{A}}_R(\bar{N}_i^t) \subseteq \hat{\mathcal{G}}, \tag{11}$$

where $\bar{N}_i^t$ is a set of the arbitrarily ordered tasks from

N_i^t and $\bar{\mathcal{A}}_R(N_i^t)$ conducts grouping by sequentially applying the assignment function $\mathcal{A}_R$, i.e., $\bar{\mathcal{A}}_R(N_i^t) = \mathcal{A}_R(\tau^t; \mathcal{A}_R(\tau^{t-1}; \dots \mathcal{A}_R(\tau^1; N_i^0) \dots))$. In addition, we add the constraint to ensure that the resulting groups are matched with existing prospective semantic groups $\hat{\mathcal{G}}$ to successfully retrieve their corresponding prompts.

For example, in Figure 4, the tasks have been originally received in the order of τ^1, τ^2, τ^3, τ^t, and $\tau^{t'}$, resulting in the three semantic groups in $N_1^{t'}$ at t'; however, by the simulation of other orders, the order of $\tau^{t'}$, τ^3, τ^t, τ^2, and τ^1 leads to only two semantic groups, which triggers the refinement of these three semantic groups.

Refinement. If permuted tasks $\bar{N}_i^t$, derived from Eq. (11), results in group reduction, i.e., $|\bar{\mathcal{A}}_R(\bar{N}_i^t)| < |\bar{\mathcal{A}}_R(\bar{N}_i^{t-1})|$, then the tasks of N_i^t are refined as $\bar{\mathcal{A}}_R(\bar{N}_i^t)$; otherwise, the refinement is not performed. Following the refinement, their prompts are retrieved from the prospective semantic groups $\hat{\mathcal{G}}$ by further adapting to the current task τ^t.

4.5. Prompt Tuning and Inference

Semantic Group-based Training. AdaPromptCL proactively trains the prompts and keys for semantic groups in prospective semantic groups $\hat{\mathcal{G}}$ including a new task τ^t, to prepare forthcoming refinements. Let θ^t be a set of prompt-key pairs for the groups containing the task τ^t,

$$\theta^t = \{(\mathrm{P}_{\hat{G}_{i,j}^t}, \mathrm{k}_{\hat{G}_{i,j}^t}) \mid \tau^t \in \hat{G}_{i,j}^t \in \hat{\mathcal{G}}\}. \tag{12}$$

Then, the training objective is to optimize θ^t by

$$\min_{\theta^t, \phi} \mathbb{E}_{(\mathrm{P},\mathrm{k})\sim \mathrm{U}(\theta^t)} \left[\ell_{ce}(f_\phi(f([x^t;\mathrm{P}])), y^t) + \lambda \mathrm{d}(f(x^t), \mathrm{k})\right], \tag{13}$$

where (P, k) is sampled uniformly from the set θ^t.

Semantic Group-based Inference. Given a test instance x, we find the matching prompt P by Eq. (4) but from the set $\mathcal{G}^t$ of semantic groups. Specifically, the prompt of i^*-th semantic group is chosen as the best for x if its key is the most similar to $f(x)$,

$$i^* = \arg\min_i \mathrm{d}(f(x), \mathrm{k}_{G_i^t}). \tag{14}$$

5. Evaluation

5.1. Experiment Setting

Dataset Preparation. Our evaluation of AdaPromptCL focuses on its efficacy across a spectrum of task semantic shifts, *uniform* and *varying* shifts. CL scenarios are categorized into uniformly mild, uniformly abrupt, and varying shifts. Visual comparisons between uniformly mild and abrupt scenarios are provided in Appendix C.

Uniformly Mild Scenario: We utilize the *CIFAR-100* (Krizhevsky et al., 2009) and *ImageNet-R* (Hendrycks et al., 2021) datasets to simulate uniformly mild shifts between tasks. By randomly subdividing these datasets into task subsets, we maintain a high degree of similarity across tasks. Following Wang et al. 2022c;b, we split CIFAR-100 and ImageNet-R into 10 tasks, with disjoint class sets resulting in 10 and 20 classes per task, respectively.

Uniformly Abrupt Scenario: We employ the Visual Task Adaptation Benchmark (VTAB) (Zhai et al., 2019) with its 19 distinct datasets to represent uniformly abrupt task shifts. In this benchmark referred to as *VTAB-19T*, each dataset within VTAB is treated as a separate task. Additionally, to simulate severe abrupt shifts, we construct *VTAB-5T*, which comprises the five most semantically distinct datasets.

Varying Scenario: We construct two CL benchmarks, *VTAB-SimS* and *VTAB-RecR*, based on VTAB-19T. They are designed to contain the shifts between tasks that are either semantically similar or dissimilar. For the selection of semantically dissimilar tasks, we draw them from the existing VTAB tasks. For the generation of similar tasks, we employ task merging (Koh et al., 2022; Bang et al., 2021) for VTAB-SimS and task recurrence (Cossu et al., 2022) for VTAB-RecR. Specifically, in VTAB-SimS, tasks overlap in 'S'% of their data instances, whereas, in VTAB-RecR, tasks are repeated 'R' times. We repeat five tasks for scalable evaluation, resulting in five unique semantics within the VTAB-RecR benchmark. Three levels of similarity are tested using the parameters S:{25, 50, 75} and R:{2, 5, 10}, and further details can be found in Appendix C.

Algorithms and Metric. AdaPromptCL is compared against representative rehearsal-free CL methods: L2P (Wang et al., 2022c), VPT (Jia et al., 2022), DP (Wang et al., 2022b), S-Prompts (Wang et al., 2022a), and LAE (Gao et al., 2023). For evaluation, we adopt a widely-used performance metric, the *last accuracy* $A_{last} = \frac{1}{T}\Sigma_{i=1}^{T} A_{T,i}$, where $A_{T,i}$ is the accuracy of the model on i-th task after learning the T-th task sequentially. Given that the last accuracy encompasses both learning adaptability and memory retention, it serves as a comprehensive measure of CL performance (Smith et al., 2023). For reliability, we run every experiment *five* times with different random seeds and report the average value with the standard error.

Implementation Details. All baseline methods are implemented using the publicly accessible CL framework [2], following (Gao et al., 2023). A ViT-B/16 (Dosovitskiy et al., 2021) pre-trained on ImageNet-1k is utilized as the common backbone across all methods, with prompts appended to the initial five transformer layers. Optimization is carried out using the Adam optimizer, setting a batch size of 128 and a learning rate of 0.025. For LAE, as per the original paper, a reduced learning rate of 0.005 is em-

[2] https://github.com/JH-LEE-KR/dualprompt-pytorch

Table 1. Performance (last accuracy) comparison of AdaPromptCL against CL baselines using prompt tuning across uniform and varying shift scenarios. "*Avg. Improv.*" denotes the relative improvement of AdaPromptCL over the average of the five baselines, and "*# Semantics*" the number of semantic groups by AdaPromptCL. The best and second-best values are highlighted in bold and underlined, respectively. Note that AdaPromptCL is even better than or comparable to the baselines for the uniform shifts which are *not* our sweet spots.

Shift Scenarios	CL Datasets	Prompt Tuning CL Algorithms					AdaPromptCL		
		L2P	VPT	LAE	DP	S-Prompts	**AdaPromptCL**	*Avg. Improv.*	*# Semantics*
Varying	VTAB-Sim25	35.77 (±0.40)	36.38 (±0.30)	35.29 (±0.42)	35.77 (±0.34)	<u>36.61</u> (±0.42)	**38.29** (±0.36)	*6.49%*	16.0
	VTAB-Sim50	36.06 (±0.39)	36.15 (±0.32)	35.23 (±0.61)	35.57 (±0.40)	<u>36.25</u> (±0.45)	**37.92** (±0.47)	*5.78%*	13.4
	VTAB-Sim75	35.66 (±0.44)	36.06 (±0.42)	35.66 (±0.60)	<u>36.35</u> (±0.46)	35.76 (±0.53)	**37.59** (±0.37)	*4.72%*	12.8
	VTAB-Rec2	52.18 (±0.13)	51.84 (±0.12)	52.34 (±0.37)	51.66 (±0.15)	<u>53.28</u> (±0.16)	**56.40** (±0.25)	*7.93%*	5.0
	VTAB-Rec5	52.68 (±0.24)	52.76 (±0.11)	<u>54.06</u> (±0.26)	51.66 (±0.16)	52.82 (±0.36)	**56.86** (±0.33)	*7.72%*	5.6
	VTAB-Rec10	52.20 (±0.81)	51.70 (±0.44)	<u>52.34</u> (±0.35)	50.80 (±0.58)	47.72 (±0.20)	**54.22** (±0.52)	*6.54%*	6.2
Uniformly Mild	ImageNet-R	68.05 (±0.11)	69.31 (±0.09)	**70.11** (±0.12)	67.81 (±0.19)	65.89 (±0.10)	<u>69.45</u> (±0.14)	*1.83%*	1.0
	CIFAR100	84.55 (±0.08)	**85.34** (±0.15)	85.16 (±0.11)	84.79 (±0.14)	83.73 (±0.10)	<u>85.31</u> (±0.11)	*0.71%*	1.0
Uniformly Abrupt	VTAB-19T	28.23 (±0.14)	28.11 (±0.20)	26.71 (±0.29)	28.48 (±0.07)	<u>30.83</u> (±0.08)	**32.39** (±0.32)	*14.00%*	18.0
	VTAB-5T	34.31 (±0.04)	34.03 (±0.05)	35.30 (±0.41)	34.17 (±0.04)	<u>38.27</u> (±0.18)	**38.67** (±0.15)	*10.02%*	5.0

ployed. The epoch counts are fixed to 5 and 50 for the universal and specific methods, respectively, reflecting their respective convergence patterns. For AdaPromptCL, the configuration includes 150 warm-up iterations, with the assignment threshold $R = 0.4$ and $\gamma = 3/2$ to maintain consistent relative scaling (Kozawa et al., 2015) across all datasets. All methods are implemented using PyTorch 1.12.1 and Timm 0.8.0 and tested on two NVIDIA RTX 3080 GPUs, and the source code is publicly available at `https://github.com/kaist-dmlab/AdaPromptCL`.

5.2. Main Results

Varying Scenarios. Table 1 presents a comparative analysis of AdaPromptCL against universal and specific prompting methods. In the varying shift scenarios, which we emphasize on, AdaPromptCL outperforms the other baseline methods across all datasets. The empirical results indicate that AdaPromptCL attains an average improvement of 6.35% and 6.91% compared to LAE and S-Prompts across varying scenarios, respectively. Unlike AdaPromptCL, the baseline methods fail to adjust to semantic shifts, resulting in the use of prompts that are either inadequate or excessive, which consequently diminishes their efficacy.

Uniformly Mild and Abrupt Scenarios. In general, AdaPromptCL demonstrates consistently high performance in the uniformly mild and abrupt shift scenarios. Specifically, AdaPromptCL matches the performance of baselines with universal prompts in the uniformly mild scenario and those with specific prompts in the uniformly abrupt scenario, where each is considered the optimal approach. In contrast, the efficacy of the other baselines varies considerably, according to the extent of semantic shifts.

Table 2. Ablation analysis highlighting the impact of semantic refinement and proactive prompt tuning on the last accuracy. The highest values are emphasized in bold.

Shift Scenarios	**No Refine**	**Avg Merge**	**AdaPromptCL**
Uniformly Mild (ImageNet-R)	85.31 (±0.11)	85.31 (±0.11)	85.31 (±0.11)
Uniformly Abrupt (VTAB-19T)	31.25 (±0.20)	30.08 (±0.26)	**32.39** (±0.32)
Varying (VTAB-Rec10)	51.80 (±0.62)	52.92 (±1.06)	**54.22** (±0.52)
Average Degradation.	2.8%	3.4%	-

Interestingly, we observe that AdaPromptCL generates the averages of 1 and 18 prompts, respectively, in the uniformly mild scenario (ImageNet-R) and the uniformly abrupt scenario (VTAB-19T). These values align with the optimal number of prompts employed in each scenario when universal and specific prompting methods are individually applied. This result demonstrates AdaPromptCL's capability to flexibly adapt its prompting architecture to the degree of task semantic shifts, leveraging the benefits of both universal and specific prompts within a single model. Please see Appendix D for the results about the *forgetting* metric.

5.3. In-depth Analysis of Semantic Refinement

Ablation Study. Our analysis focuses on the effectiveness of semantic refinement. Table 2 compares AdaPromptCL with its two variants across three semantic shift scenarios. *No Refine* omits the refinement process, which solely relies on the online assignment in Eq. (9) with the threshold R. *Avg Merge* creates the prompts for the refined semantic groups using simple prompt averages, rather than adapting them to the tasks within these groups; that is, it does not perform proactive tuning of the prompts for prospective semantic groups in Eq. (10).

Table 3. Comparison of grouping quality between AdaPromptCL and *No Refine* variant, with superior values highlighted in bold.

	No Refine	AdaPromptCL	*Reference*
# Semantic Groups	9.60 (±0.540)	**6.20** (±0.090)	5
Adj. Rand Index	0.89 (±0.031)	**0.97** (±0.005)	1
Norm. Mutual Information	0.90 (±0.027)	**0.98** (±0.004)	1

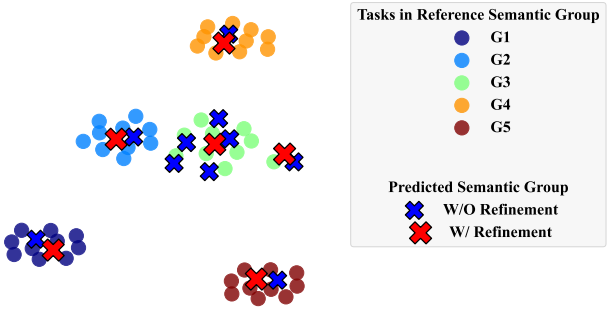


Figure 5. t-SNE visualization on the semantic groups formed by AdaPromptCL across 50 tasks on VTAB-Rec10. The tasks from the same semantic origin are in the same color, denoted by circles ●. The symbols × and × respectively represent the centroids of semantic groups with and without refinement.

Overall, the performance of both variants degrades when compared to AdaPromptCL, with average decreases in accuracy of 3.55% and 5.63% on uniformly abrupt shifts and varying shifts, respectively. Notably, *No Refine* exhibits a greater performance drop in the varying shift scenario, highlighting its susceptibility to inaccurate groupings due to the lack of the refining mechanism. In addition, *Avg Merge* shows vulnerability in the uniformly abrupt scenario, indicating that merely averaging the parameters of prompts is inadequate to blend knowledge acquired individually on different tasks without adaptation to prior tasks.

Correctness of Group Refinement. Table 3 presents the correctness of semantic grouping in the varying shift scenario using widely-used clustering quality metrics, adjusted rand index (Steinley, 2004) and normalized mutual information (Hubert & Arabie, 1985). The true task-to-group assignments are known in the process of generating VTAB-RecR. Details on the metrics are available in Appendix E. Notably, AdaPromptCL has successfully refined the groupings, reducing semantic groups from an average of 9.6 to 6.2. Moreover, the improvements in the metrics, approaching the optimal value of 1.0, further affirm its capability to enhance the precision of semantic grouping.

Visualization of Semantic Grouping. The t-SNE visualization in Figure 5 offers a qualitative insight into the efficacy of the refinement process. Tasks are plotted using their semantic representations $s(\tau)$ from Eq. (8), and semantic groups are indicated by the centroids of the tasks within each group. When the semantic groups are undesirably specified by assigning similar tasks to different semantic groups (see × symbols around the green (G3) tasks), AdaPromptCL successfully reduces the overgeneration of semantic groups (see × symbols). Please see Appendix F for more scenarios.

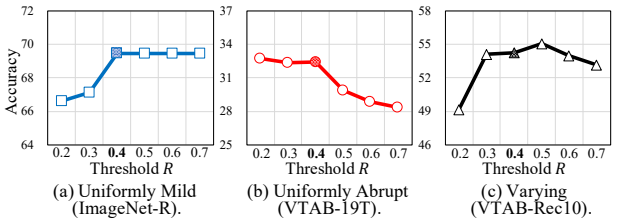


Figure 6. Sensitivity analysis on the assignment threshold R.

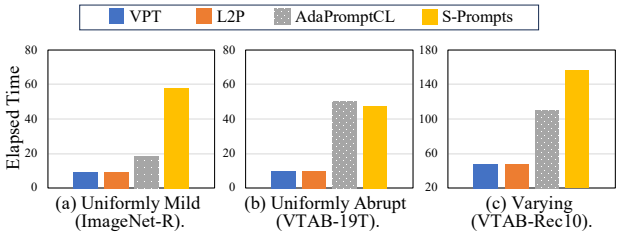


Figure 7. Elapsed GPU runtime (mins) of prompt tuning-based methods across three shift scenarios.

5.4. Parameter Sensitivity Analysis

Figure 6 presents sensitivity analysis on AdaPromptCL's main hyperparameter, the semantic assignment threshold (R) in Eq. (9). Its impact on the last accuracy across the three scenarios—uniformly mild, uniformly abrupt, and varying—is analyzed with fixing γ (for neighboring task sets) to $3/2$. A lower R results in smaller semantic groups, akin to a specific prompting, whereas a higher R leads to larger semantic groups, resembling a universal prompting. In general, a threshold of 0.4 offers the highest adaptability across diverse semantic shifts. Further sensitivity analyses for other hyperparameters are presented in Appendix G.

5.5. Computational Complexity

Figure 7 presents the elapsed GPU runtime for different prompting methods under three shift scenarios. The result indicates that the GPU runtime for AdaPromptCL is contingent upon the severity of the semantic shifts. In scenarios with uniformly mild shifts, the runtime for AdaPromptCL is comparable to that of universal prompting methods whereas in scenarios with abrupt semantic shifts, the runtime aligns with specific prompting methods. AdaPromptCL incurs a slightly increased runtime in comparison to both universal and specific methods, which is attributed to the warm-up phase to extract semantic representations in Eq. (8).

Table 4. Performance (last accuracy) comparison of AdaPromptCL against CL baselines on varying shift scenarios with two new differently scaled pretrained backbones: ViT-S/16 and ViT-T/16. The best values are highlighted in bold.

Backbones	Datasets	Prompt Tuning CL Algorithms L2P	VPT	LAE	DP	S-Prompts	AdaPromptCL
ViT-S/16	VTAB-Rec2	50.30 (±0.46)	50.50 (±0.83)	46.58 (±0.22)	51.43 (±0.17)	48.65 (±0.31)	**54.10** (±0.42)
	VTAB-Rec5	49.90 (±0.25)	52.43 (±0.91)	43.57 (±0.72)	52.67 (±2.24)	48.63 (±0.59)	**58.30** (±3.03)
	VTAB-Rec10	41.65 (±0.67)	51.35 (±3.50)	35.35 (±1.24)	51.10 (±3.32)	47.60 (±0.78)	**57.00** (±3.53)
ViT-T/16	VTAB-Rec2	36.90 (±0.22)	36.34 (±0.40)	33.60 (±0.60)	37.30 (±0.25)	37.68 (±0.78)	**39.62** (±0.41)
	VTAB-Rec5	37.03 (±0.66)	34.77 (±0.71)	34.10 (±0.15)	36.67 (±0.65)	35.90 (±0.22)	**40.13** (±0.52)
	VTAB-Rec10	33.10 (±0.87)	31.37 (±0.77)	29.27 (±0.59)	32.30 (±0.74)	33.87 (±0.43)	**40.80** (±0.43)

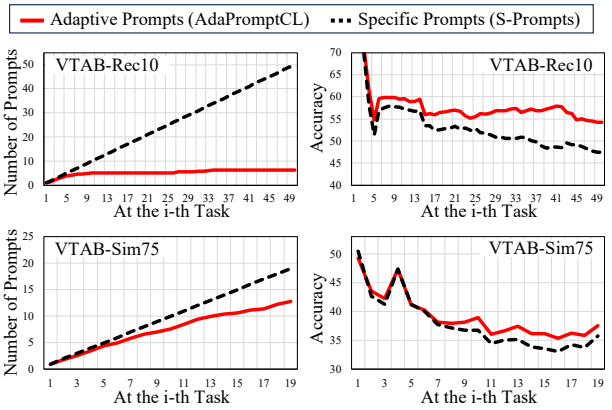
5.6. Detailed Analyses

Generalization across Different Backbone Scales. Table 4 presents a comparison of AdaPromptCL against baseline CL methods on varying semantic shift scenarios, using differently scaled pretrained backbones (Dosovitskiy et al., 2021), specifically ViT-S/16 and ViT-T/16. Overall, this result demonstrates the challenge of adjusting to semantic shifts across variously scaled pretrained backbone architectures, while highlighting the broad generalizability of AdaPromptCL regardless of the backbone architecture. Notably, AdaPromptCL achieves an average performance improvement of 17.58% and 16.06% over the baselines on ViT-S/16 and ViT-T/16, respectively.

Dynamics of Prompting Approaches over CL Streams. Figure 8 further presents a comparative analysis of the evolution in prompt counts and accuracy over task streams, contrasting adaptive prompting by AdaPromptCL and specific prompting such as S-Prompts. In Figure 8(a), the result shows AdaPromptCL's capability to dynamically adjust prompts according to task semantics, unlike specific prompting which consistently adds new prompts for each task. Additionally, Figure 8(b) highlights a widening performance gap between AdaPromptCL and S-Prompts as CL streams advance. This advantage is attributed to the positive transfer among the tasks with high semantic similarities in AdaPromptCL. Consequently, on the VTAB-Rec10, AdaPromptCL realizes a 13.62% performance improvement over S-Prompts while employing merely 12.40% of the number of prompts that S-Prompts employs.

6. Conclusion

We propose AdaPromptCL which exploits *adaptive prompting* to address the diverse and unpredictable semantic shifts. The *assign-and-refine semantic grouping* ensures that prompts are not only minimal but also precisely tailored



(a) Evolution of Prompt Counts in VTAB-Rec10 and VTAB-Sim75.

(b) Evolution of Accuracy in VTAB-Rec10 and VTAB-Sim75.

Figure 8. A comparative analysis of the evolution of prompt counts and accuracy between specific (S-Prompts) and adaptive (AdaPromptCL) prompting methods over CL streams based on VTAB-Rec10 and VTAB-Sim75.

to the evolving semantics of sequential tasks. Our results demonstrate the superior adaptability of AdaPromptCL in managing various semantic shifts. Overall, we believe that our work sheds light on the importance of versatile and adaptable CL models for diverse real-world CL scenarios.

Impact Statement

The deployment of AdaPromptCL, an adaptive prompting approach, is of substantial relevance to continual learning (CL) scenarios in real-world settings, where data and conditions are subject to unpredictable changes, such as those in autonomous systems and dynamic financial markets. AdaPromptCL's effective handling of varying semantic shifts augments the robustness and utility of CL systems, which could lead to AI deployments that are both more reliable and efficient.

From an ethical standpoint, the use of semantic grouping within AdaPromptCL necessitates caution as it could unintentionally reveal hidden biases among separated semantic groups. Consequently, it is important to diligently ensure the avoidance of unintentional prejudice or discrimination across different tasks or datasets, thus maintaining ethical integrity in the application of algorithmic systems.

Acknowledgements

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00862, DB4DL: High-Usability and Performance In-Memory Distributed DBMS for Deep Learning, 50% and No. 2022-0-00157, Robust, Fair, Extensible Data-Centric Continual Learning, 50%).

References

Aljundi, R., Chakravarty, P., and Tuytelaars, T. Expert Gate: Lifelong Learning with a Network of Experts. In *CVPR*, pp. 3366–3375, 2017.

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., and Page-Caccia, L. Online Continual Learning with Maximal Interfered Retrieval. In *NeurIPS*, 2019a.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient Based Sample Selection for Online Continual Learning. In *NeurIPS*, 2019b.

Bang, J., Kim, H., Yoo, Y., Ha, J.-W., and Choi, J. Rainbow Memory: Continual Learning wgith a Memory of Diverse Samples. In *CVPR*, pp. 8218–8227, 2021.

Bradley, P. S. and Fayyad, U. M. Refining Initial Points for k-means Clustering. In *ICML*, volume 98, pp. 91–99, 1998.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark Experience for General Continual Learning: a Strong, Simple Baseline. In *NeurIPS*, pp. 15920–15930, 2020.

Celebi, M. E., Kingravi, H. A., and Vela, P. A. A Comparative Study of Efficient Initialization Methods for the k-means Clustering Algorithm. *Expert systems with applications*, 40(1):200–210, 2013.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In *ECCV*, pp. 532–547, 2018.

Cossu, A., Graffieti, G., Pellegrini, L., Maltoni, D., Bacciu, D., Carta, A., and Lomonaco, V. Is Class-incremental Enough for Continual Learning? *Frontiers in Artificial Intelligence*, 2022.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021.

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. Efficiently Identifying Task Groupings for Multi-Task Learning. In *NeurIPS*, volume 34, pp. 27503–27516, 2021.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. A Survey on Concept Drift Adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

Gao, Q., Zhao, C., Sun, Y., Xi, T., Zhang, G., Ghanem, B., and Zhang, J. A Unified Continual Learning Framework with General Parameter-Efficient Tuning. In *ICCV*, 2023.

Guo, P., Lee, C.-Y., and Ulbricht, D. Learning to Branch for Multi-Task Learning. In *ICML*, pp. 3854–3863, 2020.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al. The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization. In *CVPR*, pp. 8340–8349, 2021.

Hubert, L. and Arabie, P. Comparing Partitions. *Journal of Classification*, 2:193–218, 1985.

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. Visual Prompt Tuning. In *ECCV*, pp. 709–727, 2022.

Kaufman, L. and Rousseeuw, P. J. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 2009.

Kim, D., Park, D., Shin, Y., Bang, J., Song, H., and Lee, J.-G. Adaptive shortcut debiasing for online continual learning. In *AAAI*, pp. 13122–13131, 2024.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Koh, H., Kim, D., Ha, J.-W., and Choi, J. Online Continual Learning on Class Incremental Blurry Task Configuration with Anytime Inference. In *ICLR*, 2022.

Kozawa, Y., Hayashi, F., Amagasa, T., and Kitagawa, H. Parallel Canopy Clustering on Gpus. In *International Conference on Data Management in Cloud, Grid and P2P Systems*, pp. 334–348. Springer, 2015.

Krizhevsky, A., Hinton, G., et al. Learning Multiple Layers of Features from Tiny Images. *Technical Report*, 2009.

Lee, S., Ha, J., Zhang, D., and Kim, G. A Nneural Dirichlet Process Mixture Model for Task-free Continual Learning. In *ICLR*, 2020.

Liu, Y., Lu, Y., Liu, H., An, Y., Xu, Z., Yao, Z., Zhang, B., Xiong, Z., and Gui, C. Hierarchical Prompt Learning for Multi-Task Learning. In *CVPR*, pp. 10888–10898, 2023.

Mai, Z., Li, R., Jeong, J., Quispe, D., Kim, H., and Sanner, S. Online Continual Learning in Image Classification: An Empirical Survey. *Neurocomputing*, 469:28–51, 2022.

Mallya, A. and Lazebnik, S. Packnet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *CVPR*, pp. 7765–7773, 2018.

Mundt, M., Hong, Y., Pliushch, I., and Ramesh, V. A Wholistic View of Continual Learning with Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning. *Neural Networks*, 160:306–336, 2023.

Ng, A., Jordan, M., and Weiss, Y. On Spectral Clustering: Analysis and an Algorithm. *NIPS*, 14, 2001.

Prabhu, A., Torr, P. H., and Dokania, P. K. Gdumb: A Simple Approach that Questions Our Progress in Continual Learning. In *ECCV*, pp. 524–540, 2020.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience Replay for Continual Learning. In *NeurIPS*, 2019.

Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

Shim, D., Mai, Z., Jeong, J., Sanner, S., Kim, H., and Jang, J. Online Class-incremental Continual Learning with Adversarial Shapley Value. In *AAAI*, pp. 9630–9638, 2021.

Smith, J. S., Karlinsky, L., Gutta, V., Cascante-Bonilla, P., Kim, D., Arbelle, A., Panda, R., Feris, R., and Kira, Z. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *CVPR*, pp. 11909–11919, 2023.

Song, X., Zheng, S., Cao, W., Yu, J., and Bian, J. Efficient and Effective Multi-Task Grouping via Meta Learning on Task Combinations. In *NeurIPS*, volume 35, pp. 37647–37659, 2022.

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. Which Tasks Should Be Learned Together in Multi-Task Learning? In *ICML*, pp. 9120–9132, 2020.

Steinley, D. Properties of the Hubert-Arable Adjusted Rand Index. *Psychological Methods*, 9(3):386, 2004.

Wang, Y., Huang, Z., and Hong, X. S-prompts Learning with Pre-trained Transformers: An Occam's Razor for Domain Incremental Learning. In *NeurIPS*, volume 35, pp. 5682–5695, 2022a.

Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.-Y., Ren, X., Su, G., Perot, V., Dy, J., et al. Dualprompt: Complementary Prompting for Rehearsal-free Continual Learning. In *ECCV*, pp. 631–648, 2022b.

Wang, Z., Zhang, Z., Lee, C.-Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., and Pfister, T. Learning to Prompt for Continual Learning. In *CVPR*, pp. 139–149, 2022c.

Wu, S., Zhang, H. R., and Ré, C. Understanding and Improving Information Transfer in Multi-Task Learning. In *ICLR*, 2020.

Yang, E., Pan, J., Wang, X., Yu, H., Shen, L., Chen, X., Xiao, L., Jiang, J., and Guo, G. Adatask: A Task-Aware Adaptive Learning Rate Approach to Multi-Task Learning. In *AAAI*, volume 37, pp. 10745–10753, 2023.

Yoon, S., Meng, Y., Lee, D., and Han, J. SCStory: Self-supervised and continual online story discovery. In *WWW*, pp. 1853–1864, 2023.

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. Taskonomy: Disentangling Task Transfer Learning. In *CVPR*, pp. 3712–3722, 2018.

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., et al. A Large-Scale Study of Representation Learning with the Visual Task Adaptation Benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Zhang, T., Ramakrishnan, R., and Livny, M. Birch: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD*, 25(2):103–114, 1996.

A. Pseudocode of AdaPromptCL

The training procedure of AdaPromptCL is described in Algorithm 1. Our *assign-and-refine* semantic grouping is executed through the semantic assignment (Lines 6–13) and the semantic refinement (Lines 14–19). During the semantic assignment, AdaPromptCL assigns the incoming task to an appropriate semantic group or initiates a new group (Lines 7–9). Concurrently, it updates the neighboring semantic groups and collects the prospective semantic groups using neighboring semantic groups (Lines 9–10) where the detailed procedure of collecting the prospective semantic groups is described in Algorithm 2. In the semantic refinement, AdaPromptCL finds the task order that yields the fewest semantic groups belonging to a neighboring semantic group updated by an incoming task (Line 15). If the semantic groups can be reduced, AdaPromptCL refines the current semantic groups into the reduced groups and retrieves corresponding prompts from prospective semantic groups for these refined groups (Lines 16–18). Lastly, prompts and keys are initialized for each prospective semantic group, optimized with the classifier, and stored for future use (Lines 21–23). The process iterates for each task, progressively refining the semantic groups and tuning prompts.

Algorithm 1 AdaPromptCL: *Assign-and-refine* CL Framework

input a sequence of tasks $\mathcal{T}$, data stream $\{D^1, \ldots, D^{\mathcal{T}}\}$, threshold R, scaling factor γ, classifier ϕ
output semantic groups $\mathcal{G}^{|\mathcal{T}|}$, prompts $\mathcal{P}$, keys $\mathcal{K}$, classifier ϕ
1: $\mathcal{G}^0 \leftarrow \text{InitializeSemanticGroups}()$
2: $\hat{\mathcal{G}} \leftarrow \text{InitializeProspectiveSemanticGroups}()$
3: $\mathcal{N}^0 \leftarrow \text{InitializeNeighboringSemanticGroups}()$
4: $\mathcal{P}, \mathcal{K} \leftarrow \text{PromptsRepository}(), \text{KeysRepository}()$
5: **for** each τ^t **in** T **do**
6: /* SEMANTIC ASSIGNMENT: SECTION 4.3 */
7: $\hat{\text{P}}^{\tau^t} \leftarrow \text{WarmupPrompt}(D^t)$
8: $s(\tau^t) \leftarrow \text{ExtractSemantic}(\hat{\text{P}}^{\tau^t})$
9: $\mathcal{G}^t \leftarrow \text{AssignSemanticGroups}(s(\tau^t), \mathcal{G}^{t-1}, R)$
10: $\mathcal{N}^t \leftarrow \text{AssignNeighboringSemanticGroups}(s(\tau^t), \mathcal{N}^{t-1}, \gamma R)$
11: **for** each N_i^t **in** $\mathcal{N}^t$ **do**
12: **if** $\text{IsUpdated}(N_i^t)$ **then**
13: $\hat{\mathcal{G}} \leftarrow \text{CollectProspectiveSemanticGroups}(N_i^t, \hat{\mathcal{G}}, R)$
14: /* SEMANTIC REFINEMENT: SECTION 4.4 */
15: $\bar{N}^t \leftarrow \text{FindMininumGroupOrder}(N_i^t, R)$
16: **if** $\text{IsReduced}(\bar{N}_i^t, N_i^t)$ **then**
17: $\text{RefineGroups}(\bar{N}_i^t, N_i^t)$
18: $\text{RetrievePrompts}(\bar{N}_i^t, \mathcal{P}, \mathcal{K})$
19: **end if**
20: /* SEMANTIC GROUP BASED PROMPT TUNING: SECTION 4.5 */
21: $\theta_t \leftarrow \text{InitializePromptsKeys}(\tau^t, \hat{\mathcal{G}})$
22: $\theta_t, \phi \leftarrow \text{Train}(\theta_t, \phi, D^t)$
23: $\mathcal{P}, \mathcal{K} \leftarrow \text{UpdatePromptsKeysRepositories}(\theta_t)$
24: **end if**
25: **end for**
26: **end for**
27: **return** $\mathcal{G}^{|\mathcal{T}|}$, $\mathcal{P}$, $\mathcal{K}$, ϕ;

B. Prospective Prompts Collection

Implementation Details. This section provides an in-depth explanation of the *prospective prompts* collection process, which was previously omitted due to space constraints. Prospective prompts are instrumental in ensuring that semantic groups are updated with prompts that are tailored to all the tasks they encompass. Thus, finding prospective prompts requires finding semantic groups that likely appear in the future by semantic refinement. Algorithm 2 outlines the process for collecting prospective semantic groups, given a neighboring semantic group and the preceding prospective semantic groups as indicated in Line 13 of Algorithm 1.

Overall, as suggested in (Kaufman & Rousseeuw, 2009), the first step is to ascertain the ideal number of clusters based on task semantics, using the silhouette score (Rousseeuw, 1987). This is followed by the collection of prospective semantic groups with the *k*-means clustering method. The algorithm commences by initializing a storage for silhouette scores (Line 1). It then iteratively computes silhouette scores for different cluster configurations and records them (Lines 3–8). Subsequently,

Algorithm 2 Prospective Semantic Groups Collection

input neighboring semantic group N_i^t, preceding prospective semantic groups $\hat{\mathcal{G}}$, clustering threshold R, clustering iteration count r, number of representative clusters counts η
output prospective semantic groups $\hat{\mathcal{G}}$

1: $\mathcal{S} \leftarrow \text{InitializeSilhouetteMemory}()$
2: /* IDENTIFICATION OF REPRESENTATIVE CLUSTER COUNTS VIA SILHOUETTE SCORE */
3: **for** each j **from** 1 **to** r **do**
4: $\quad L \leftarrow \text{LabelClusters}(\text{Cluster}(N_i^t, R))$
5: $\quad k \leftarrow \text{CountClusters}(L)$
6: $\quad s \leftarrow \text{MeasureSilhouetteScore}(N_i^t, L)$
7: $\quad \mathcal{S} \leftarrow \text{UpdateSilhouetteMemory}(\mathcal{S}, \{(k, s)\})$
8: **end for**
9: $K \leftarrow \arg\max_{K:\,|K|\leq\eta} \sum_{k\in K} \text{AverageSilhouette}(\mathcal{S}, k)$
10: /* COLLECTION OF PROSPECTIVE SEMANTIC GROUPS VIA REPRESENTATIVE CLUSTER COUNTS */
11: **for** each j **from** 1 **to** r **do**
12: $\quad$ **for** each k **in** K **do**
13: $\quad\quad \hat{\mathcal{G}} \leftarrow \hat{\mathcal{G}} \cup k\text{-means}(N_i^t, k)$
14: $\quad$ **end for**
15: **end for**
16: **return** $\hat{\mathcal{G}}^t$;

it determines the top-η optimal cluster counts that maximize the silhouette score (Line 9). With these optimal counts, the k-means algorithm is applied (Lines 11–15), and the newly identified candidate groups are amalgamated with existing prospective semantic groups (Line 13). For our experiments, we set the clustering iteration count r to 100 and the number of representative clusters counts η to 2. Please see Appendix G for sensitivity analyses regarding these hyperparameters.

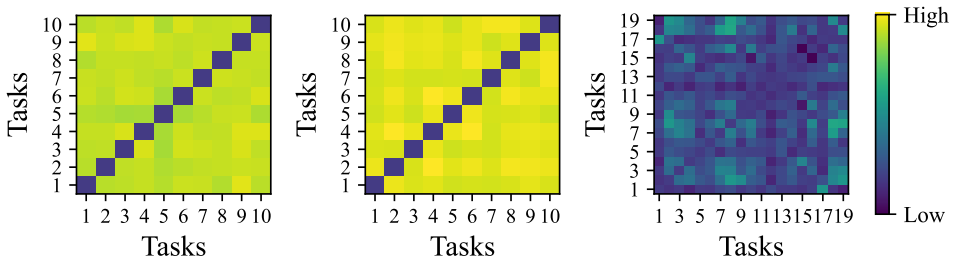
Computational Complexity. Table 5 compares GPU memory usage and runtime between AdaPromptCL configurations with and without the incorporation of prospective semantic groups. The 'No Refine' variant, described in Section 5.3, denotes the configuration that does not include prospective semantic groups, thus lacking the capability for semantic group refinement. In general, computational demand is generally influenced by the extent of semantic shift within the tasks. uniformly mild and abrupt scenarios do not typically benefit from prospective semantic groups due to the unambiguous nature of task relationships. In the varying shifting scenario, however, the varying degrees of semantic shifts between tasks increase the demand for prospective semantic groups. Specifically, an average of 95.2 prospective semantic groups are utilized. Despite this, the increase in total GPU memory usage is marginal, at 149.6MB (1.6%), attributable to the small memory footprint of the prompts. Moreover, this leads to a mere 14 minutes (12.5%) increase in GPU running time, as prospective semantic groups necessitate a short period of fine-tuning for alignment with the current task.

Table 5. Computational overhead incurred by prospective semantic groups of AdaPromptCL across diverse semantic shift scenarios, presenting GPU memory (in MB) and running time (in minutes). All results are obtained utilizing automatic mixed precision to optimize runtime and memory consumption.

Scenarios	**Uniformly Mild** (ImageNet-R)			**Uniformly Abrupt** (VTAB-19T)			**Varying** (VTAB-Rec10)		
Metrics	**#Prospective Prompts**	**GPU Memory**	**Running Time**	**#Prospective Prompts**	**GPU Memory**	**Running Time**	**#Prospective Prompts**	**GPU Memory**	**Running Time**
No Refine	- (-)	8859.0 (±0.0)	18.2 (±0.2)	- (-)	9093.0 (±0.0)	62.4 (±0.2)	- (-)	9107.0 (±0.0)	112.0 (±2.3)
AdaPromptCL	0.0 (±0.0)	8859.0 (±0.0)	18.4 (±0.2)	95.2 (±10.2)	9256.6 (±8.2)	126.0 (±4.1)	0.6 (±0.2)	9103.8 (±4.4)	63.0 (±0.3)

C. VTAB Datasets Preparation

Visual Task Adaptation Benchmark (VTAB) (Zhai et al., 2019) is a dataset in the field of machine learning, specifically designed for evaluating the adaptability and generalization capabilities of visual representation learning methods. VTAB consists of 19 tasks from diverse domains such as natural images, specialized tasks (like medical imaging or satellite imagery), and structured tasks (such as object counting or predicting depth from images). This diversity ensures that models are tested on a wide range of visual understanding capabilities.



(a) CIFAR100. (b) ImageNet-R. (c) VTAB.

Figure 9. Task-to-task semantic similarity matrices for CL datasets: (a) CIFAR100, (b) ImageNet-R, and (c) VTAB, with similarity measured by cosine similarity. Lighter colors indicate higher similarity ('High'), while darker ones correspond to lower similarity ('Low').

Task Semantic Similarity Comparison. Figure 9 shows that the degree of semantic similarity among tasks in VTAB is significantly low when compared to CIFAR100 and ImageNet-R. Semantic similarity is assessed by training task-specific prompts and extracting semantic representations following Eq. (8). The matrices reveal a notable contrast in semantic similarities: CIFAR100 and ImageNet-R show significantly high similarities, reflecting mild semantic shifts, whereas the VTAB shows low similarities, indicating more abrupt semantic shifts between tasks.

Uniformly Abrupt Scenarios with VTAB. By virtue of its wide-ranging domains, VTAB is employed to symbolize abrupt semantic shifts where every dataset is regarded as an individual task, referred to as VTAB-19T. Furthermore, we construct VTAB-5T, a collection of the five datasets that are most distinct in terms of semantics: *(clevr-count*, *resisc45*, *diabetic-retinopathy*, *oxford-flowers102*, *dsprites-loc)*. In order to obtain these distinct datasets, task semantic representations are extracted for each task as in Eq. (8), following the training of prompt for each individual task. Then, using spectral clustering (Ng et al., 2001), we cluster the semantic representations of 19 tasks into five distinct clusters and select a single task from each cluster, for a total of five datasets.

Varying Scenarios with VTAB. For varying semantic shifting scenarios, VTAB-SimS is designed to mimic the overlapping of tasks in VTAB, following the realistic and practical CL setup (Koh et al., 2022). This setup assumes that the model regularly encounters new classes while previously observed classes may reappear in subsequent tasks. Accordingly, VTAB datasets are divided into two categories: dissimilar datasets, introducing entirely new classes to the model, and *similar* datasets, where tasks are comprised of a mix of classes from different datasets. Specifically, each overlapping task in VTAB-SimS includes S% from other datasets and (100-S)% of its dataset. For our experiments, we randomly designate 9 tasks as dissimilar and 10 as similar, experimenting with three different values for S: 25%, 50%, and 75%. A higher S value renders tasks more semantically similar due to a greater proportion of shared classes. Meanwhile, VTAB-RecR provides an additional scenario of varying semantic shifting through the recurrence of tasks where a higher R increases similarities between tasks as semantically similar tasks recur more frequently. In VTAB-RecR, data instances for recurrent tasks are randomly drawn from their respective original tasks.

D. Additional Experiment Results

To supplement the last accuracy A_{last} presented in Table 1, we utilize the CL performance metric *forgetting*, denoted by $F_{last} = \frac{1}{T-1}\Sigma_{j=1}^{T-1} f_{T,j}$. Here, $f_{i,j}$ indicates how much the model forgets about the j-th task after learning the i-th task ($j < i$). It is important to note that A_{last} encompasses the overall model's capability in both acquiring new skills (plasticity) and preserving knowledge of earlier tasks (forgetting) while F_{last} only serves as a supplementary measure to the last accuracy A_{last} while (Smith et al., 2023).

Table 6 reveals that AdaPromptCL achieves superior memory retention, exhibiting an average improvement in forgetting F_{last} by 27.54% and 7.55% compared to the most accurate universal (LAE) and specific (S-Prompts) prompting methods across all scenarios, respectively. The success of AdaPromptCL is primarily due to its adaptive prompt tuning approach, which tailors prompts to semantically similar tasks. This strategy effectively mitigates negative transfer between distinct tasks and curtails the forgetting of previously acquired knowledge during the acquisition of new information.

Table 6. Performance comparison of AdaPromptCL against CL baselines using prompt tuning across uniformly and varying shift scenarios. This table shows both the forgetting for individual datasets and the average forgetting (presented in the final row), highlighting the relative performance degradation of each baseline in comparison to AdaPromptCL.

Shifting Scenarios	CL Datasets	Prompt Tuning CL Algorithms					
		L2P	VPT	LAE	DP	S-Prompts	**AdaPromptCL**
Varying	VTAB-Sim25	5.96 (±0.48)	5.11 (±0.38)	8.09 (±0.54)	5.45 (±0.41)	6.39 (±0.49)	6.16 (±0.55)
	VTAB-Sim50	4.94 (±0.35)	5.04 (±0.42)	6.60 (±0.37)	5.01 (±0.48)	5.43 (±0.30)	4.34 (±0.44)
	VTAB-Sim75	3.99 (±0.26)	3.96 (±0.27)	5.39 (±0.30)	3.49 (±0.26)	3.67 (±0.20)	4.16 (±0.41)
	VTAB-Rec2	6.69 (±0.67)	6.71 (±0.65)	10.59 (±0.56)	6.00 (±0.69)	6.18 (±0.60)	5.63 (±0.57)
	VTAB-Rec5	2.82 (±0.19)	3.63 (±0.06)	4.97 (±0.33)	3.53 (±0.27)	4.27 (±0.16)	2.71 (±0.17)
	VTAB-Rec10	6.33 (±0.37)	8.25 (±0.53)	7.87 (±0.43)	8.07 (±0.46)	8.00 (±0.52)	8.32 (±0.62)
Uniformly Mild	ImageNet-R	5.17 (±0.10)	5.27 (±0.08)	6.05 (±0.21)	4.99 (±0.18)	7.71 (±0.22)	4.94 (±0.08)
	CIFAR100	5.59 (±0.08)	6.19 (±0.07)	6.05 (±0.21)	5.04 (±0.15)	6.41 (±0.13)	6.21 (±0.07)
Uniformly Abrupt	VTAB-19T	6.15 (±0.42)	6.43 (±0.55)	9.48 (±0.54)	5.50 (±0.42)	4.01 (±0.23)	4.30 (±0.69)
	VTAB-5T	15.55 (±1.13)	15.88 (±1.09)	17.72 (±1.29)	15.57 (±1.07)	12.84 (±0.98)	13.25 (±1.00)
Avg. Forgetting (Degrad.)		6.32 (-5.06%)	6.65 (-9.77%)	8.28 (-27.54%)	6.26 (-4.15%)	6.49 (-7.55%)	6.00 (-%)

E. Clustering Metrics

To measure the correctness of semantic grouping by AdaPromptCL, we adopt two widely-used clustering metrics: Adjusted Rand Index (ARI) (Steinley, 2004) and Normalized Mutual Information (NMI) (Hubert & Arabie, 1985). They are commonly used to evaluate the performance of clustering algorithms by comparing their ability to correctly cluster data points to the labels of ground truth clustering. In particular, ARI assesses the degree of agreement between pairs of samples by examining whether pairs from the same or different clusters remain consistent across the two clusterings. On the other hand, NMI evaluates the extent to which one clustering provides insights into the other, thereby determining the degree of mutual dependence between the cluster assignments. Specifically, the score of ARI ranges from -1 to 1 where a score of 1 indicates perfect agreement between two clusterings, 0 suggests no better than chance agreement, and negative values indicate disagreement. Likewise, NMI ranges from 0 to 1 where a score of 1 means perfect correlation (identical clusterings), and 0 indicates no mutual information (independent clusterings).

F. Additional Visualization of Semantic Groups by AdaPromptCL

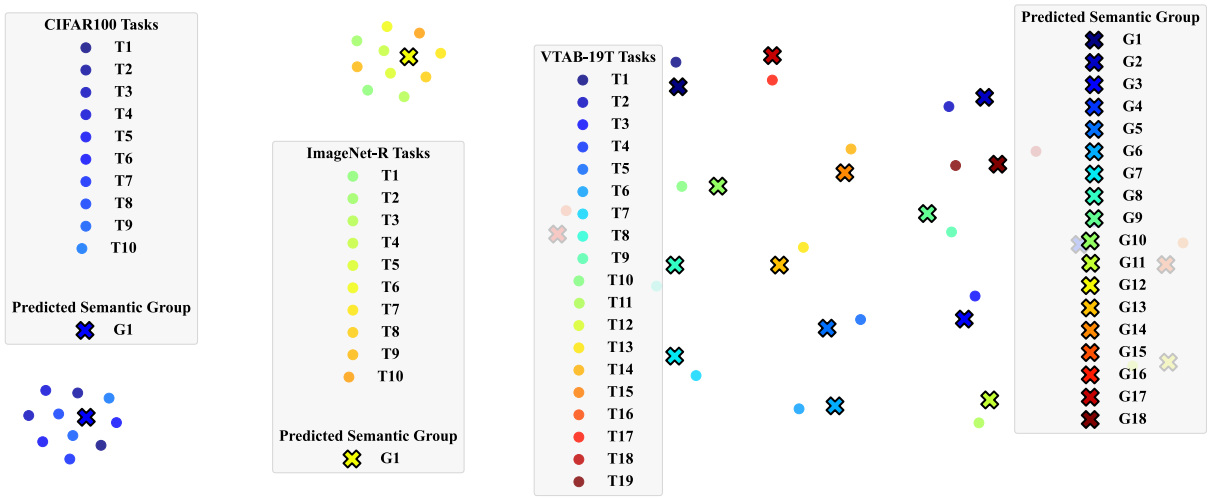
Figure 10 complements Figure 5 by demonstrating the semantic groups formed by AdaPromptCL under uniformly mild and abrupt shifting scenarios. It is evident that the emergence of these semantic groups is dependent on the degree of semantic shift. For cases with mild semantic shifts, such as CIFAR100 and ImageNet-R, all tasks are aggregated into a single semantic group, similar to the approach of universal prompting methods. Conversely, in the scenario with abrupt semantic shifts, the semantic groups tend to correspond nearly on a per-task basis, which is in line with specific prompting methods.

G. Additional Parameter Sensitivity Analysis

Figure 11 presents the sensitivity analysis on the hyperparameters: the clustering iteration count r and and the number of representative clusters η, as applied to the collection of prospective prompts in Eq. (10), and the simulation sampling size κ, utilized for semantic refinement in Eq. (11).

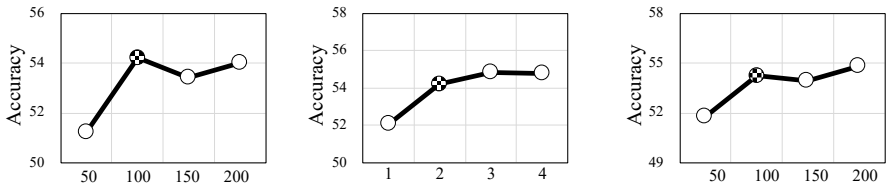
Effect of Clustering Iteration Count. Increasing clustering iteration count facilitates the creation of more comprehensive and diverse prospective semantic groups, which enhances the chances of refining to more generalizable semantic groups. Figure 11(a) illustrates the impact of different clustering iteration counts, specifically $r \in \{50, 100, 150, 200\}$, on test accuracy in varying shifting scenarios where the active refinement is required. Our findings show that there is no significant improvement in performance when the clustering iteration count exceeds 100. This suggests that a count of 100 is adequate for efficiently generating sufficiently comprehensive prospective semantic groups for semantic refinement. Consequently, we adopt $r = 100$ for all experiments.

Effect of Representative Cluster Count. A higher number of representative clusters can prepare more encompassing and varied prospective semantic groups, which in turn supports semantic refinement. Figure 11(b) presents the influence



(a) Uniformly mild shifting scenario. (b) Uniformly abrupt shifting scenario.

Figure 10. t-SNE visualization the semantic groups formed by AdaPromptCL for tasks from the uniformly mild shifting scenarios (CIFAR100, ImageNet-R), and abrupt shifting scenario (VTAB-19T). Tasks are denoted by circles ● and the symbols × represent the semantic groups identified by AdaPromptCL. A single semantic group encompasses all tasks for CIFAR100 and ImageNet-R while distinct semantic groups are formed for nearly every task in the VTAB-19T dataset.



(a) Cluster Iteration Count r. (b) Cluster Count η. (c) Number of Simulations κ.

Figure 11. Parameter sensitivity analysis of clustering iteration count r, number of representative clusters η, and permutation sampling size κ on prospective prompt collection and semantic refinement, using VTAB-Rec10.

of varying representative cluster counts $\eta \in \{1, 2, 3, 4\}$ on test accuracy. We observe that a representative count of 2 or more achieves high accuracies, thereby suggesting that a count of 2 is adequate for the efficient formation of sufficiently comprehensive prospective semantic groups necessary for effective semantic refinement. Thus, for all experiments, we set the representative cluster count η to 2.

Effect of Simulation Sampling Size. The higher number of task order simulations increases the likelihood of identifying more generalizable semantic groups through refinement. Figure 11(c) shows the effect of sampling size $\kappa \in \{50, 100, 150, 200\}$ on the test accuracy in varying shift scenarios where refinement is actively required. Overall, our findings reveal that performance is not significantly affected by the number of simulations, particularly when the sampling size exceeds 100, indicating no meaningful improvement in performance. Accordingly, we establish $\kappa = 100$ as the default sampling size for all experiments.