Enhancing Continual Learning of Vision-Language Models via Dynamic Prefix Weighting

Hyeonseo Jang Hyuk Kwon Kibok Lee
Yonsei University
{jhyeonseo715, kh12043, kibok}@yonsei.ac.kr

Abstract

We investigate recently introduced domain-class incremental learning scenarios for vision-language models (VLMs). Recent works address this challenge using parameter-efficient methods, such as prefix-tuning or adapters, which facilitate model adaptation to downstream tasks by incorporating task-specific information into input tokens through additive vectors. However, previous approaches often normalize the weights of these vectors, disregarding the fact that different input tokens require different degrees of adjustment. To overcome this issue, we propose Dynamic Prefix Weighting (DPW), a framework that dynamically assigns weights to prefixes, complemented by adapters. DPW consists of 1) a gating module that adjusts the weights of each prefix based on the importance of the corresponding input token, and 2) a weighting mechanism that derives adapter output weights as a residual of prefix-tuning weights, ensuring that adapters are utilized only when necessary. Experimental results demonstrate that our method achieves state-of-the-art performance in domain-class incremental learning scenarios for VLMs. The code is available at: https://github.com/YonseiML/dpw.

1. Introduction

Continual learning (CL) enables models to adapt to sequential data streams while mitigating catastrophic forgetting of previously acquired knowledge, offering an alternative to retraining from scratch in ever-changing environments [5, 13, 30]. Recent advances in CL have extended its applicability to multi-modal learning scenarios [44], including vision-language models (VLMs) such as CLIP [28]. Despite their impressive zero-shot capabilities, VLMs often yield suboptimal performance in real-world applications, underscoring the need for CL to improve downstream task adaptation while preserving the foundational knowledge established during pretraining [49]. To address this, previous studies have introduced benchmarks evaluating both the zero-shot and CL performance of VLMs on downstream tasks [18, 49]. However, the large-scale nature of VLMs poses significant challenges in fine-tuning the entire model due to the computational cost [11, 45].

Recent studies have explored parameter-efficient fine-tuning (PEFT) methods to mitigate this challenge. These approaches typically freeze the backbone model and learn task-specific knowledge by introducing lightweight adapters [20, 45] or employing prefix-tuning techniques with learnable prompt vectors [18, 22, 32]. Such methods enable effective adaptation to new tasks while requiring only a small number of additional parameters. In the context of CL, recent PEFT advances primarily focus on adaptively weighting the influence of newly introduced parameters based on input samples, thereby mitigating catastrophic forgetting [32, 45]. However, as illustrated in Fig. 1, existing weighting mechanisms generally operate at the sample level, treating all tokens within a sample uniformly and assigning them the same amount of task-specific information. As prior studies [7, 52] have shown that dynamically adjusting the amount of injected information according to the task relevance of each token can substantially improve performance, the lack of token-level weighting thus represents a fundamental limitation.

We observe that this limitation stems not only from the weighting strategies employed in existing methods but also from inherent properties of the pretrained backbone. For instance, within the prefix-tuning framework, existing methods rely on attention mechanisms to assign prefix weights, yet these mechanisms inherently constrain the ability to produce optimal weighting. Because attention mechanisms tend to capture global contextual relationships within a sample [27, 34], they often project tokens with distinct characteristics closer together in the feature space. This behavior obscures the distinctions between task-relevant and irrelevant tokens, making fine-grained weighting difficult. Moreover, this limitation extends beyond individual samples to the task level, hindering the disentanglement of token embeddings across tasks. As a result, knowledge transfer in CL settings becomes less effective, and task interference increasingly degrades overall performance [9].

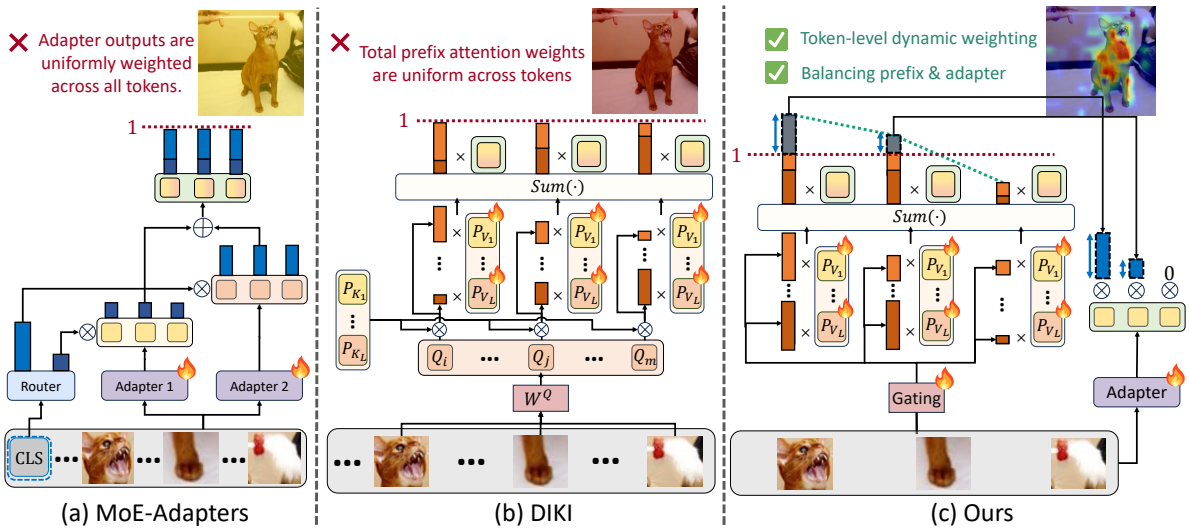

Figure 1. Comparison of the weighting mechanisms of various popular PEFT methods. Light-green boxes denote the output vectors added to the input tokens. (a): MoE-Adapters [45] relies solely on the CLS token for routing and assigns equal weights to the outputs of each adapter, with their sum always fixed at one. (b): DIKI [32] always normalizes the total weight of the prefixes added to each input, forcing them to sum up to one. (c): Our method dynamically adjusts prefixes and adapters weights based on the importance of the input tokens.

In this paper, we investigate how to incorporate an appropriate amount of task-specific information into each input token in CL. To this end, we propose dynamic prefix weighting (**DPW**), a framework that assigns refined weights to both prefixes and adapters based on the task relevance of the corresponding input token. First, we develop a novel gating module that replaces the conventional attention mechanism used in prefix-tuning. Our gating module consists of reparametrized prefix attention (**RePA**) and conditional activation (**CondAct**), which replace the query-key dot product and softmax activation, respectively. Unlike conventional prefix-tuning methods that rely on query-key projection, our RePA computes task relevance scores directly from input tokens using a learnable affine transformation. Subsequently, CondAct replaces the softmax activation with conditional normalization and filtering, ensuring that the total prefix weight is dynamically adjusted within an appropriate range. Second, we introduce a residual weighting mechanism (**RWM**) that assigns weights to adapters based on residual prefix weights generated during conditional normalization. RWM selectively incorporates adapter outputs for input tokens with higher prefix scores, enabling adapters to focus on more informative tokens according to refined prefix scores. As illustrated in Fig. 1(c), DPW provides fine-grained weighting for both prefixes and adapters, enabling precise token-level modulation and improving knowledge retention in CL settings.

2. Related Work

Continual Learning (CL). Existing CL approaches can be categorized into three types: regularization-based, memory-based, and architecture-based methods. *Regularization-based methods* [1, 2, 12, 15, 48] mitigate catastrophic forgetting by introducing penalty terms into the loss function, thereby preserving the prior knowledge acquired from earlier tasks. *Memory-based approaches* [3, 4, 21, 23, 42, 43] store historical data or synthetic samples generated by models, allowing selective replay to retain past knowledge. *Architecture-based methods* [6, 16, 19, 24, 31, 36–38] expand model parameters to incorporate new tasks while retaining knowledge from previous ones. PEFT based approaches, such as adapter and prompt-tuning, are typically categorized as architecture-based methods, as they introduce a small set of task-specific parameters while keeping the backbone frozen. Our method adopts a similar strategy, enabling efficient adaptation to downstream tasks with few additional parameters.

Continual Learning of VLMs. CL of VLMs seeks to enable these models to sequentially adapt to diverse domains and classes while preserving their zero-shot capabilities [18, 49]. One of the pioneering approaches is ZSCL [49], which addresses this challenge by leveraging a reference dataset and employing knowledge distillation from a frozen CLIP. Subsequent work has explored advanced distillation strategies, such as incorporating CLIP models trained on previous tasks into the distillation process with sample-wise adaptive balancing between two teacher models [47], or utilizing diffusion models to generate reference datasets for distillation [40]. However, these approaches require fine-tuning the entire CLIP model and rely on an additional reference dataset, which reduces training efficiency. An alternative line of work leverages PEFT techniques, such as adapters [8, 20, 45], for more efficient adaptation. For example, MoE-Adapters [45] introduce two adapters per task and

use a router to assign weights within a mixture-of-experts framework, combining their outputs with the input tokens. C-CLIP [20] learns LoRA with a distillation loss and integrates it into the backbone after training each task. Another PEFT branch focuses on prompt-based methods [18, 22, 32], which learn task-specific information through a learnable vector called a prompt or a prefix. CoLeCLIP [18] extends to new tasks using a dedicated vocabulary and task-specific prompts. DIKI [32] mitigates prefix interference during CL by using cross-attention to separately assign weights to prefixes, with a batch-wise distribution-aware calibration factor. However, as widely reported in the prompt-based learning literature, these methods often suffer from performance degradation when applied to domains for which the prompts were not trained [36, 46, 51]. Unlike existing approaches, our method dynamically balances the use of prefixes and adapters to leverage the strengths of both.

3. Preliminaries

3.1. Continual Learning Protocol VLMs

Following [32, 45, 49], we consider a collection of T tasks, denoted by $\{\mathcal{T}^t\}_{t=1}^{T}$. Each task $\mathcal{T}^t$ consists of a dataset D_t and a set of class names C_t. The class names in C_t are paired with manually crafted prompts and fed into the text encoder. The dataset D_t, consisting of N_t samples, is represented as $D_t = \{(I_{tn}, y_{tn})\}_{n=1}^{N_t}$, where I_{tn} is the input image and y_{tn} is its corresponding one-hot label from C_t. In CL, models incrementally learn by sequentially processing each task $\mathcal{T}^t = \{D_t, C_t\}$. The primary objective of CL in VLMs is to maintain strong performance on both learned and unseen tasks, with the latter evaluated for zero-shot capabilities after each task to measure a new type of forgetting that arises in the context of CL for VLMs, known as forward forgetting. We consider two relevant benchmarks: Multi-domain Task Incremental Learning (MTIL) [49] and Open-Domain Continual Learning (ODCL-CIL) [18], where both domain and class distributions change across tasks. In MTIL, the model has access to the task ID during training, whereas ODCL-CIL requires classification over all learned classes without task ID information.

3.2. Prompt-Based Continual Learning

Prompt-based methods are widely used in CL to adapt transformer-based pretrained models that process data samples in token units [10, 31, 36–38]. These methods adjust the model by introducing prompt tokens of length L, consisting of learnable vectors of dimension d, while keeping the model's backbone frozen. For each task t, a unique set of vectors $P_t \in \mathbb{R}^{L\times d}$ is learned, constructing a pool $\{P_1, P_2, \ldots, P_T\}$ that spans T tasks. At inference time, the prompt P_t is selected from the pool and attached to the pretrained model to restore task-specific knowledge [32], allowing input tokens $X \in \mathbb{R}^{m\times d}$ to absorb task-specific information through the attention mechanism by modifying the inputs of multi-head self-attention layers [37, 41]. Among prompt-based methods, prefix-tuning [17] learns a pair of task-specific prompts $P_K, P_V \in \mathbb{R}^{L\times d}$ instead of a single P, and separately projects them to the key and value in the attention mechanism.

Specifically, for the i-th head, input tokens and prefix are first transformed into query Q_i, key K_i, and value V_i by corresponding projection matrices W_i^Q, W_i^K, W_i^V and their respective biases. With the input token X, this can be formulated as follows:

$$
\begin{aligned}
Q_i &= XW_i^Q + \mathbf{1}_n\, b_i^Q \in \mathbb{R}^{n\times \frac{d}{h}},\\
K_i &= \begin{bmatrix} K_{X_i} \\ K_{P_i} \end{bmatrix} = \begin{bmatrix} XW_i^K + \mathbf{1}_n\, b_i^K \\ P_K W_i^K + \mathbf{1}_L\, b_i^K \end{bmatrix} \in \mathbb{R}^{(n+L)\times \frac{d}{h}},\\
V_i &= \begin{bmatrix} V_{X_i} \\ V_{P_i} \end{bmatrix} = \begin{bmatrix} XW_i^V + \mathbf{1}_n\, b_i^V \\ P_V W_i^V + \mathbf{1}_L\, b_i^V \end{bmatrix} \in \mathbb{R}^{(n+L)\times \frac{d}{h}}.
\end{aligned}
\tag{1}
$$

The self-attention computes the attention scores S_i using the inner product of query and key $Q_i(K_i)^\top$:

$$
S_i = \left[S_{XX}^{(i)},\, S_{XP}^{(i)}\right] = \left[Q_i\,(K_{X_i})^\top,\; Q_i\,(K_{P_i})^\top\right]. \tag{2}
$$

Here, $S_{XX}^{(i)}$ represents the attention scores among the input tokens, while $S_{XP}^{(i)}$ represents the attention scores between the input tokens and the prefixes. Next, these scores are normalized to attention weights using a Softmax function with temperature τ, which are then used to weight the corresponding value matrix V_i:

$$
h_i = \text{softmax}\left(\frac{S_i}{\sqrt{d/h}}\right) V_i \in \mathbb{R}^{m\times \frac{d}{h}}. \tag{3}
$$

Finally, the outputs from all heads are concatenated and projected through the projection matrix W_o.

4. Methods

The main goals of our method are two-fold. First, we aim to ensure that each input token receives a proper amount of information from the prefixes and adapters throughout CL tasks. Second, we aim to establish an effective synergy between the prefixes and adapters. Our method addresses both motivations within a unified framework. Specifically, we first introduce a gating module that assigns refined weights to the prefixes. Then, guided by these weights, tokens that require additional adaptation provided the information from adapter. Fig. 2 provides an overview of the proposed framework.

4.1. Reparameterized Prefix Attention

In conventional prefix-tuning for CL, the learnable prefix key P_K is appended for each task and projected onto the

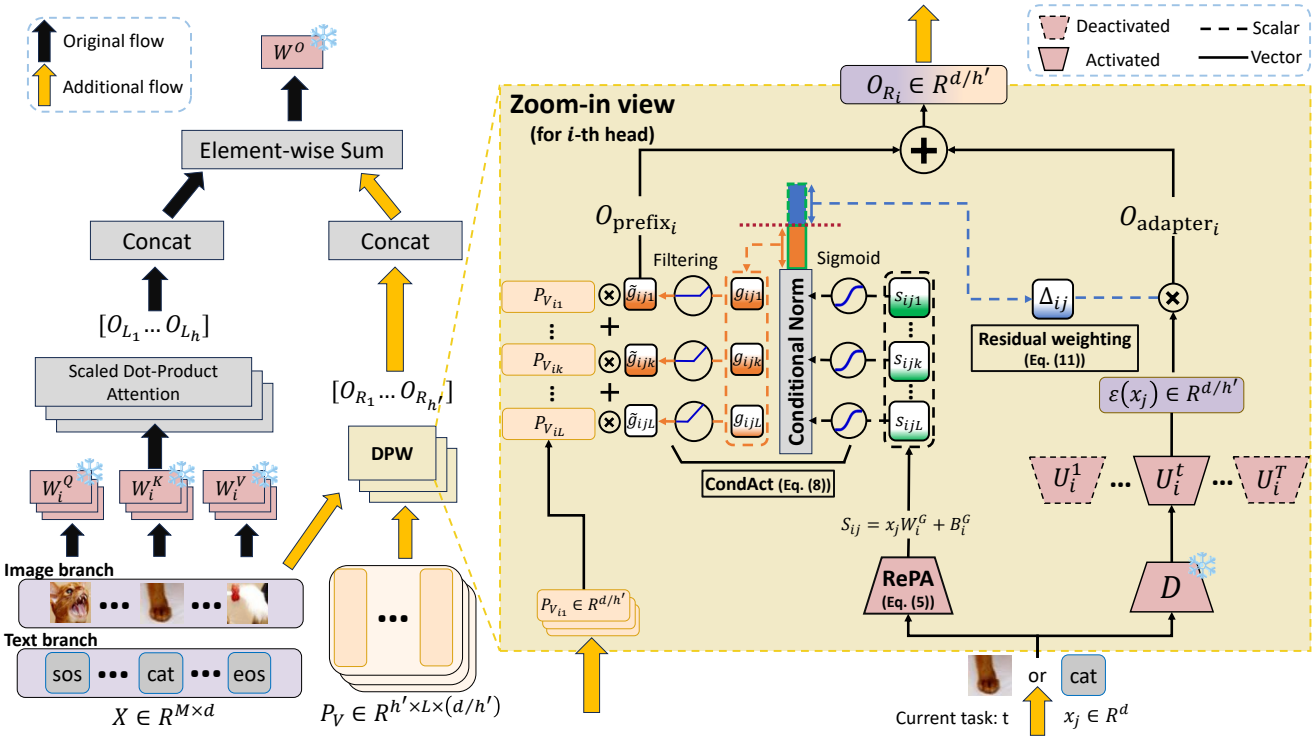

Figure 2. Overall framework of the proposed method. The right side of the figure shows how each input token assigns weights to both the prefixes and the adapter. Specifically, we compute prefix scores for each input token using a RePA module, and then convert those scores into weights with the CondAct module. We also set an upper limit on the total weight that can be assigned to the prefixes. Any weight beyond this limit is applied to the adapter's output. This design enables token-level weighting for the adapter without the need for a router.

pretrained key distribution through a key projection matrix. The P_K is then optimized to align with the pretrained query distribution of the corresponding task, determining the attention scores that weight the prefix value P_V. However, because the pretrained model typically lacks sufficient task-specific knowledge for new tasks [49], these attention scores often fail to accurately capture the task relevance of input tokens, as illustrated in Fig. 3. Consequently, the model fails to effectively regulate the amount of information injected per token, which has been shown to impair downstream task performance [7, 52].

A straightforward way to address this issue is to learn a separate query-key projection matrix for each task, dedicated exclusively to computing the attention scores of the prefix. Although this approach improves performance in practice, the large number of additional parameters, together with the limited data in downstream tasks, often leads to suboptimal results. Instead, we propose a reparameterized prefix attention **(RePA)**, which simplifies the query-key dot product process into a single affine transformation to compute the prefix scores. Specifically, for the i-th attention head, the original attention module computes the prefix score through $S_{XP}^{(i)} = Q_i(K_{P_i})^\top = (XW_i^Q + \mathbf{1}_m b_Q^\top)(P_K W_i^K + \mathbf{1}_L b_K^\top)^\top$. This expression can be rewritten so that the parts involving the input tokens X are isolated from the parts that remain constant regardless of X, yielding the following equation:

$$S_{XP}^{(i)} = X\Big[W_i^Q(P_K W_i^K)^\top + W_i^Q b_K\,\mathbf{1}_L^\top\Big] + \Big[\mathbf{1}_m\, b_Q^\top(P_K W_i^K)^\top + (b_Q^\top b_K)\,\mathbf{1}_{m\times L}\Big]. \tag{4}$$

We can group the terms into two matrices, arriving at RePA:

$$S_{XP}^{(i)} = X\,W_i^G + B_i^G. \tag{5}$$

Note that training W_i^G and B_i^G effectively unifies the original attention projection parameters and the learnable prefix key P_K into a single reparameterized form [26]:

$$W_i^G \approx W_i^Q\,(P_K W_i^K)^\top + W_i^Q\,b_K\,\mathbf{1}_L^\top, \tag{6}$$

$$B_i^G \approx \mathbf{1}_m\,b_Q^\top\,(P_K W_i^K)^\top + (b_Q^\top b_K)\,\mathbf{1}_{m\times L}. \tag{7}$$

By training these composite parameters, W_i^G and B_i^G, we can better capture task relevance by learning task-specific projections applied exclusively to the prefixes, while preserving pretrained knowledge by leaving the original projection matrices unchanged. Consequently, as illustrated in Fig. 3, our RePA assigns higher attention scores to tokens that are directly relevant to each task, thereby facilitating more effective adaptation to downstream tasks [7, 52].

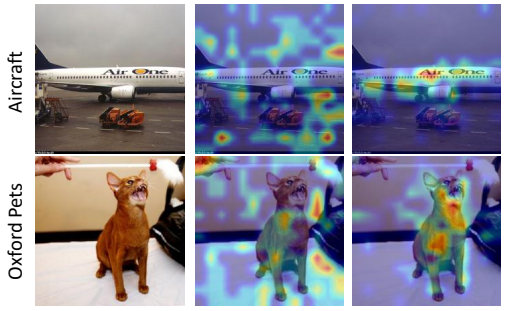

Figure 3. Comparison of prefix score maps between attention and our method on the Aircraft and Oxford Pets images. Our approach produces more task-relevant scores.

4.2. Conditional Activation

Applying the above method, we obtain an enhanced score matrix $S_{XP}^{(i)}$. However, prior work [32] has demonstrated that concatenating $S_{XP}^{(i)}$ with the original attention score $S_{XX}^{(i)}$ before applying the softmax function disrupts the pretrained knowledge encoded in $S_{XX}^{(i)}$, leading to significant degradation in zero-shot performance. Alternatively, DIKI [32] proposes applying a separate softmax to $S_{XP}^{(i)}$. While this approach avoids interference with the pretrained scores, it imposes a fixed total weight across all input tokens, hindering dynamic adjustment of the additional information for each token, thereby limiting token-wise modulation.

To overcome these limitations, we introduce Conditional Activation (**CondAct**), a novel alternative to the conventional softmax activation. CondAct processes the score matrix S_{XP} through two conditional operations, conditional normalization and conditional filtering, applied after a sigmoid function $\sigma(\cdot)$. Let s_{ijk} denote the score between the j-th input token and the k-th prefix, corresponding to the (j,k)-th element of the score matrix $S_{XP}^{(i)}$. The corresponding prefix weight $\widetilde{g}_{ijk}$ is computed via the following procedure:
(Step 1: Conditional Norm)

$$g_{ijk} = \begin{cases} \dfrac{\sigma(s_{ijk})}{\sum_{k=1}^{L}\sigma(s_{ijk})}, & \text{if } \sum_{k=1}^{L}\sigma(s_{ijk}) \geq 1, \\ \sigma(s_{ijk}), & \text{otherwise.} \end{cases} \tag{8a}$$

(Step 2: Conditional Filtering)

$$\widetilde{g}_{ijk} = g_{ijk} \cdot \mathbb{I}\big(g_{ijk} \geq \text{cutoff}\big). \tag{8b}$$

Here, $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the condition is met and 0 otherwise.

The conditional normalization step in Eq. (8a) enables the total sum of prefix weights, $\sum_{k=1}^{L} g_{ijk}$, to flexibly range up to one—allowing the model to dynamically adjust the influence of each input token. This upper bound prevents task-specific information from becoming overly dominant, a common issue when using a sigmoid function alone [29]. Consequently, CondAct mitigates forward forgetting in VLMs and enhances zero-shot generalization in CL tasks.

Next, the conditional filtering step in Eq. (8b) removes prefixes that are less relevant to the current task. Empirically, we observe that the attention scores between the [CLS] token and each prefix follow a Gaussian distribution. Leveraging this property, we determine the cutoff value dynamically for each sample. Specifically, for each prefix, we compute the mean and variance of its attention scores with the [CLS] token across the training data of task t and model its distribution as $\mathcal{N}(\mu_t, \sigma_t^2)$. Given a test sample, we evaluate the log probability density of the observed attention score $s_{i,\text{cls},k}$ under this distribution and transform it via a sigmoid function to obtain a likelihood score, similar as [32]. The cutoff for the k-th prefix is defined as one minus this score:

$$\text{cutoff}_{ijk} = 1 - \sigma\Big(\log \varphi(s_{i,\text{cls},k}; \mu_t, \sigma_t^2)\Big). \tag{9}$$

Since each prefix exhibits distinct activation patterns across input tokens, this prefix-wise adaptive cutoff enables token-wise dynamic filtering conditioned on task-specific likelihoods, allowing the model to selectively suppress irrelevant prefixes. For the i-th head, the additional information derived from the prefixes is computed as follows:

$$O_{\text{prefix}_i} = \widetilde{G}_i\, P_{V_i} \quad \text{with} \quad \widetilde{G}_i = (\widetilde{g}_{ijk})_{j,k} \in \mathbb{R}^{m \times L}. \tag{10}$$

4.3. Residual Weighting Mechanism

Our proposed CondAct dynamically adjusts the total weight of the prefixes for each token, constraining it within an upper limit of one to preserve zero-shot performance. However, strictly enforcing this limit may restrict further improvements, as tokens with higher task relevance often require stronger adaptation. To address this, we introduce the Residual Weighting Mechanism **(RWM)**, which selectively applies the adapter output to tokens that demand additional adaptation. For parameter efficiency, we employ LoRA as the adapter [45] and follow the structure proposed in [33], where a shared down-projection matrix D is used across tasks and a task-specific up-projection U_t is assigned to each task t. To further adapt this design for CL, we freeze the shared projection, initializing it with the top-k left singular vectors of the value projection matrix W_i^V as [25, 35]. This constrain that the adapter is fine-tuned within the row space of D [19], effectively leveraging pretrained knowledge to complement the learned prefixes across tasks. The adapter's output, denoted as $\mathcal{E}_i^t(X)$, is then weighted using RWM:

$$\begin{aligned} O_{\text{adapter}_i} &= \Delta_i \odot \mathcal{E}_i^t(X), \\ \Delta_i &= \text{concat}\big[\Delta_{ij}\big]_{j=1}^{m}, \\ \Delta_{ij} &= \max\Big(0,\, \textstyle\sum_{k=1}^{L}\sigma(s_{ijk}) - 1\Big), \end{aligned} \tag{11}$$

where $\odot$ denotes element-wise multiplication.

Method	Extra data	Params.	Trans.	Avg.	Last	Mean
Zero-shot	-	–	69.4	65.3	65.3	66.7
ZSCL [49]	✓	149.6 M	68.1	75.4	83.6	75.7
DIKI [32]	×	1.8 M	68.7	76.3	85.1	76.7
MoE-Adapter [45]	×	59.6 M	68.9	76.7	85.0	76.9
GIFT [40]	✓	149.6 M	69.3	77.3	86.0	77.5
Ours†	×	4.6 M	<u>70.0</u>	<u>78.6</u>	<u>87.6</u>	<u>78.7</u>
Ours	×	30.8 M	**70.4**	**79.3**	**88.3**	**79.3**

Table 1. Comparison of various SOTA methods on MTIL Order I benchmark in terms of "Transfer", "Average", and "Last" scores (%). Best and second-best results are highlighted in **bold** and <u>underline</u>, respectively.

Method	Trans.	Avg.	Last	Mean
ZSCL [49]	68.0	71.8	77.6	72.5
MoE-Adapter [45]	69.1	66.2	66.9	67.4
CoLeCLIP [18]	68.8	73.7	79.7	74.1
DPeCLIP [22]	69.1	76.1	84.6	76.6
Ours†	<u>70.0</u>	<u>77.9</u>	<u>85.8</u>	<u>77.9</u>
Ours	**70.4**	**78.6**	**86.6**	**78.5**

Table 2. Comparison of various SOTA methods on ODCL-CIL benchmark in terms of "Transfer," "Average," and "Last" scores (%).

RePA	CondAct	RWM	Trans.	Avg.	Last
-	-	-	68.1	76.6	85.9
✓	-	-	68.0	76.8	86.4
-	✓	-	69.5	77.8	86.8
✓	✓	-	69.9	78.9	87.9
✓	✓	✓	**70.4**	**79.3**	**88.3**

Table 3. Ablation study on the MTIL benchmark evaluating each module's contribution.

If the sum of prefix weights, $\sum_{k=1}^{L} \sigma(s_{ijk})$, does not exceed one, the corresponding element of Δ_i becomes zero, meaning that the adapter does not contribute to that token. In this way, RWM leverages the residual prefix weights as adaptive scaling factors, enabling each token to receive additional information proportional to its remaining adaptation demand. Through this combination of prefixes and adapters, we achieve a balance between the aggressive modification introduced by the prefix and the conservative update performed by the adapter. For each head i, the output of the proposed DPW module, denoted as O_{R_i} is computed as follows:

$$O_{R_i} = O_{\text{prefix}_i} + O_{\text{adapter}_i}. \tag{12}$$

5. Experiments

5.1. Experimental Setting

Benchmarks. We evaluate our method in two domain-class incremental settings: Multi-domain Task Incremental Learning (MTIL) [49] and Open-Domain Continual Learning (ODCL-CIL) [18]. MTIL uses task IDs for task-specific classification, while ODCL-CIL classifies across all seen classes without task IDs. Both benchmarks include 11 datasets from various domains, covering 1201 classes.

Metrics. We evaluate our methods in both MTIL and ODCL-CIL using the metrics proposed in [18, 49]: *Transfer*, *Avg.*, and *Last*. The *Transfer* metric measures a model's ability to generalize to unseen data by evaluating its zero-shot performance, and is used to quantify forward forgetting in the context of CL for VLMs [49]. Both benchmarks, MTIL and ODCL-CIL, achieve identical performance on this metric as they rely on task IDs when measuring zero-shot capabilities. The *Last* metric represents the average performance in all tasks after CL, and the *Avg.* metric measures the average accuracy on all datasets and stages.

Compared Methods. We compare our method against various state-of-the-art (SOTA) approaches, including prompt-based, adapter-based, and full fine-tuning methods. For prompt-based learning, we compare CoLeCLIP [18], DIKI [32] and DPeCLIP [22]. For adapter-based learning, we compare MoE-Adapters [45]. For full fine-tuning, we compare ZSCL [49] and GIFT [40]. In addition to our default model (denoted Ours), we also introduce a parameter-efficient variant, Ours†, which reduces the number of trainable parameters. Ours†computes the prefix score for each head using only its corresponding sub-dimension (i.e., d/h') instead of the full embedding, reducing the number of parameters by a factor of h'. We further apply LoRA with a lower rank for additional efficiency. All experiments follow the baseline setup [32], with details provided in the Appendix.

5.2. Main Results.

Performance on MTIL. Tab 1 presents a comparison between the proposed method and several SOTA approaches on the MTIL benchmark. For the sequence of training tasks, we follow the predefined Order I configuration described in [49], where tasks are arranged alphabetically. Detailed results are provided in Appendix, including Order II. Overall, the results show that both Ours and Ours† outperform competing methods across all evaluation metrics. In particular, they surpass the second-best approach, GIFT [40], which performs full fine-tuning of the entire CLIP model using additional data generated by an auxiliary diffusion model. Compared to PEFT methods, such as MoE-Adapter [45] and DIKI [32], our approach consistently achieves superior performance while maintaining a similarly low number of trainable parameters. The strong performance on both *Transfer* and *Last* scores indicates that our method not only adapts effectively to new tasks but also preserves previously learned knowledge, underscoring its suitability for CL.

Performance on ODCL-CIL. We evaluate our method on class-incremental setting using the ODCL-CIL benchmark, as presented in Tab. 2. Compared with the MTIL benchmark, the ODCL-CIL scenario is more challenging because task IDs are unavailable during inference. Following the baseline

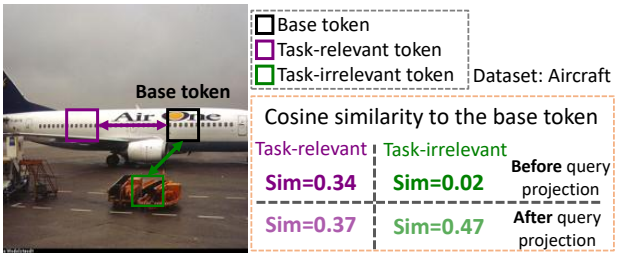

Figure 4. Cosine similarities comparison between the **base token** and both the **task-relevant** and **task-irrelevant** tokens, shown before and after query projection.

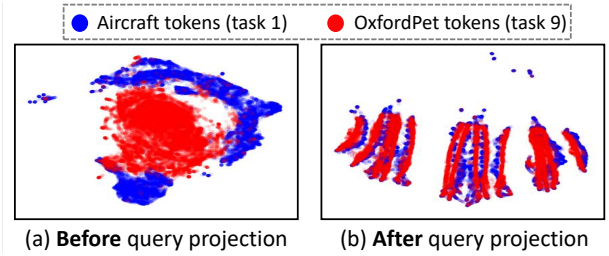

Figure 5. UMAP visualization of token embeddings from the **Aircraft** and **OxfordPet** datasets, illustrating their distributions before and after applying query projection.

Method	Params.	Trans.	Avg.	Last
$QK^\top$ (freeze W^Q, W^K)	2.7 M	68.7	76.4	85.2
$QK^\top$ (train W^Q, W^K)	228.0 M	68.9	76.8	85.9
$QW_G + B_G$	30.8 M	69.7	78.6	87.6
$XW_G + B_G$ (Ours)	30.8 M	**70.4**	**79.3**	**88.3**

Table 4. Comparison between RePA and the traditional attention mechanism.

Sigmoid	CondNorm	Filtering	Trans.	Avg.	Last
-	-	-	68.0	76.8	86.4
✓	-	-	68.6	78.3	88.2
✓	✓	-	69.9	79.0	88.2
✓	✓	✓	**70.4**	**79.3**	**88.3**

Table 5. Ablation study evaluating the impact of different components within the CondAct.

Method	Params.	Trans.	Avg.	Last
Baseline	19.8 M	69.9	78.9	87.9
Prefix length $\times 2$	39.6 M	69.5	78.4	87.6
Learnable router	32.7 M	69.9	79.0	88.1
RWM (Ours)	30.8 M	**70.4**	**79.3**	**88.3**

Table 6. Comparison between the RWM and traditional routing mechanism.

method [32], we adopt its task-identification strategy, which estimates the likelihood of the current image feature under each task distribution and selects the parameters associated with the highest-likelihood task. To avoid potential advantages from batch voting [50], we further evaluate our method using both the default batch size (e.g., 256) and a batch size of one, reporting the lower score. Across all evaluation metrics, our method consistently achieves SOTA performance in this class-incremental setting.

5.3. Analysis

Ablation Study. Our framework consists of three components: **RePA** to compute prefix scores (Eq. (5)), **CondAct** to transform the prefix scores into weights (Eq. (8)), and **RWM** to incorporate the adapter (Eq. (11)). We evaluate each module on the MTIL benchmark in Tab. 3. First, by replacing the traditional query-key dot product with the RePA, we obtain a refined prefix score matrix S_{XP}, leading to an improvement in the *Last* score. However, this component alone cannot overcome the inherent limitation of softmax, which forces scores to sum to one and thus restricts the optimal assignment of weight per token. As shown in the third row, replacing softmax with our CondAct improves overall performance by dynamically controlling the amount of information added to each token, and combining it with the refined scores from RePA yields additional gains by better capturing task-specific characteristics in the scoring process. This token-wise modulation not only improves performance on the trained tasks, but also facilitates effective knowledge transfer across tasks during CL, and mitigating forward forgetting, as evidenced by the improved *Transfer* scores. Moreover, by leveraging the weights assigned to each prefix, our RWM enables adapters to add information only to tokens that are not sufficiently adapted by the prefixes, thereby effectively complementing the prefixes and leading to notable improvements in both the *Last* and *Transfer* scores.

Limitations of Pretrained Attention Projections in CL. To further investigate the limitations of existing attention mechanisms in prefix-tuning for CL, we conduct a comparative analysis of token embeddings before and after the query projection. As shown in Fig. 4, we extract tokens from a single image and label those from object regions (e.g., the body of an airplane) as task-relevant and those from background regions as task-irrelevant, then compute cosine similarities (i) among task-relevant tokens (purple arrow) and (ii) between task-relevant and task-irrelevant tokens (green arrow). Specifically, we find that before the query projection, the similarity between task-relevant and task-irrelevant tokens is merely 0.02. After the projection, however, this similarity increases substantially to 0.47, exceeding even the similarity among task-relevant tokens. This indicates that the query projection disrupts the model's ability to maintain distinctions between task-relevant and task-irrelevant features. Consequently, as shown in Fig. 3, prefix struggles to route attention to the appropriate tokens. Notably, this effect is not limited to individual samples; it also appears at the task level in our CL setting. Fig. 5 illustrates UMAP visualizations of tokens from multiple samples across two tasks. Before the projection, tokens from different tasks are clearly separated. However, after projection, they collapse into overlapping regions, revealing diminished task separability. Such reduced separability increases task interference and weakens effective knowledge transfer across CL tasks [9]. Overall, these findings highlight inherent limitations in approaches that rely on pretrained attention projections [22, 32, 37, 38].

Analysis of RePA. To evaluate whether RePA can effectively replace the conventional attention mechanism, we conduct experiments summarized in Tab. 4. As shown in the second row, re-training the attention projection matrices leads to improved CL performance. However, this approach substantially increases the number of learnable parameters to

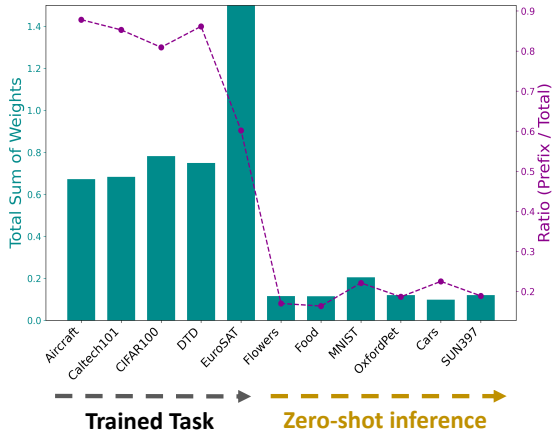

Figure 6. Weight distribution after training on five tasks. **Green bars** show the sum of prefixes and adapters weights, while **purple lines** indicate the ratio of prefix weights to the total.

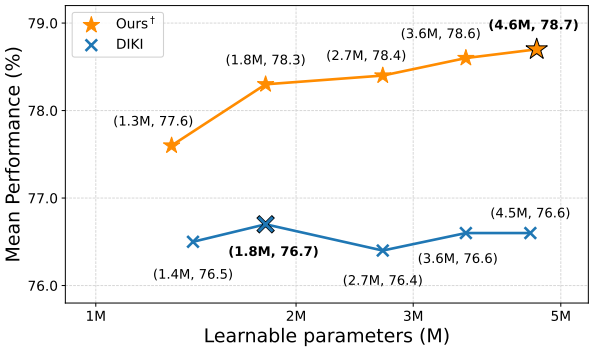

Figure 7. Comparison of mean performance and the number of learnable parameters between Ours† and DIKI [32] under varying prefix lengths. **Bold** indicates configurations reported in Tab. 1. Our method consistently outperforms DIKI across all settings.

228.0M, while the performance gain remains significantly lower than that achieved by RePA. In the third row, we preserve the original query projection process of the attention mechanism and apply RePA on top of it. Similar to the previous case, the performance remains inferior to that obtained when RePA is directly applied to the input tokens. These results highlight the intrinsic limitations of the attention mechanism and demonstrate that RePA serves as an effective and efficient alternative.

Analysis of CondAct. To assess the contribution of each component in the proposed CondAct mechanism, we conduct an ablation study summarized in Tab. 5. Consistent with recent findings [14], which report improved performance when combining softmax and sigmoid activations in prefix tuning, we observe that incorporating a sigmoid activation into RePA also yields notable gains. However, the sigmoid function tends to produce overly large total weights for input tokens [29], potentially distorting their pretrained representations and degrading *Transfer* performance. Our conditional normalization module (8a) mitigates this issue by maintaining the total prefix information within a stable range, leading to a substantial improvement in *Transfer* accuracy. Furthermore, the conditional filtering mechanism (8b), which leverages the distribution of prefix scores to assign prefix-wise cutoffs, enables dynamic and token-wise selection of informative prefixes. This adaptive filtering further enhances the *Transfer* score by refining how prefix information is selectively utilized across tokens.

Analysis of RWM. To assess the effectiveness of our RWM, we conduct experiments summarized in Tab. 6. We first observe that simply increasing the number of learnable parameters does not yield better performance. As shown in the second block, doubling the prefix length degrades performance, highlighting the limitation of performance gains achievable through prefix alone. The third block introduces an adapter with a learnable router, but the resulting improvements remain marginal, suggesting that the adapter is ineffective without an appropriate weighting mechanism. In contrast, our proposed RWM module enables more effective weighting, leading to consistent improvements in both the *Transfer* and *Last* scores. Empirically, we find that the adapter outputs become more orthogonal to the prefixes when trained with RWM, indicating that the adapter complements the prefix by extending beyond the subspace spanned by a fixed number of prefix vectors.

Analysis of Weight Distribution. Fig. 6 presents an analysis of the weights assigned to the prefixes and the adapters. We compute the average weights for each and visualize their total sums and relative ratios. This suggests that the model can distinguish between learned and unseen tasks, highlighting the suitability of our method for CL scenarios for VLMs. Moreover, we observe that trained tasks rely more heavily on prefix information, whereas zero-shot tasks depend primarily on the adapter. This adaptive weighting mechanism represents a core property of our framework: when the model encounters distributions that deviate from the training data, it decreases reliance on the prefixes and activates the adapter, which is designed to preserve generalizable knowledge.

Parameter Efficiency. Fig. 7 shows how performance scales with the number of learnable parameters for both our method and the baseline DIKI [32]. We control the parameter count by varying the number of prefixes L in each method. The results demonstrate that our approach consistently outperforms DIKI, even when using fewer learnable parameters.

6. Conclusion

In this work, we propose a method for CL of VLMs that overcomes the limitations of attention-based weighting by learning task-aware weights for prefixes and adapters. Specifically, we introduce a novel mechanism that evaluates the task relevance of each token and assigns corresponding weights accordingly. As prefix-tuning and adapters have recently gained popularity, we believe our method can be extended to improve their effectiveness beyond the CL setting.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018. 2

[2] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11254–11263, 2019. 2

[3] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8218–8227, 2021. 2

[4] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018. 2

[5] Pramit Dhar, Rajeev Ranjan Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5138–5146, 2019. 1

[6] A. Douillard, A. Ramé, G. Couairon, and M. Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9285–9295, 2022. 2

[7] Chin-Lun Fu, Zih-Ching Chen, Yun-Ru Lee, and Hung-Yi Lee. Adapterbias: Parameter-efficient token-dependent representation shift for adapters in nlp tasks. In *Findings of the Association for Computational Linguistics*, pages 2608–2621, 2022. 1, 4

[8] Haoyuan Gao, Zicong Zhang, Yuqi Wei, Linglan Zhao, Guilin Li, Yexin Li, Linghe Kong, and Weiran Huang. Enhanced continual learning of vision-language models with model fusion. In *ICLR 2025 Workshop*. ICLR, 2025. 2

[9] Naoki Hiratani. Disentangling and mitigating the impact of task similarity for continual learning. *arXiv preprint*, 2024. 1, 7

[10] D. Jung, D. Han, J. Bang, and H. Song. Generating instance-level prompts for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11813–11823, 2023. 3

[11] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19113–19122, 2023. 1

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, 2017. 2

[13] Matthias De Lange, Rahaf Aljundi, Mateusz Masana, Sophie Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(7):3366–3385, 2021. 1

[14] Minh Le, An Nguyen The, Huy Nguyen, Thien Trang Nguyen Vu, Huyen Trang Pham, Linh Ngo Van, and Nhat Ho. Mixture of experts meets prompt-based continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 8

[15] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017. 2

[16] Xilai Li, Yuezhou Zhou, Tianjun Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3925–3934. PMLR, 2019. 2

[17] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00119*, 2021. 3

[18] Yukun Li, Guansong Pang, Wei Suo, Chenchen Jing, Yuling Xi, Lingqiao Liu, Hao Chen, Guoqiang Liang, and Peng Wang. Coleclip: Open-domain continual learning via joint task prompt and vocabulary learning. *arXiv preprint*, 2024. 1, 2, 3, 6, 7

[19] Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23638–23647, 2024. 2, 5

[20] Wenzhuo Liu, Fei Zhu, Longhui Wei, and Qi Tian. C-clip: Multimodal continual learning for vision-language model. In *International Conference on Learning Representations (ICLR)*, 2025. 1, 2, 3

[21] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12245–12254, 2020. 2, 6

[22] Yadong Lu, Shitian Zhao, Boxiang Yun, Dongsheng Jiang, Yin Li, Qingli Li, and Yan Wang. Boosting open-domain continual learning via leveraging intra-domain category-aware prototype. *arXiv preprint*, 2024. 1, 3, 6, 7

[23] Zilin Luo, Yaoyao Liu, Bernt Schiele, and Qianru Sun. Class-incremental exemplar compression for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11371–11380, 2023. 2

[24] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7765–7773, 2018. 2

[25] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, 2024. 5

[26] Samet Oymak, Ankit Singh Rawat, Mahdi Soltanolkotabi, and Christos Thrampoulidis. On the role of attention in prompt-tuning. *arXiv preprint arXiv:2306.03435*, 2023. 4

[27] Xu Pan, Aaron Philip, Ziqian Xie, and Odelia Schwartz. Dissecting query-key interaction in vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. Spotlight Presentation. 1

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint*, 2021. 1, 2

[29] Jason Ramapuram, Federico Danieli, Eeshan Dhekane, Floris Weers, Dan Busbridge, Pierre Ablin, Tatiana Likhomanenko, Jagrit Digani, Zijin Gu, Amitis Shidani, and Russ Webb. Theory, analysis, and best practices for sigmoid self-attention. *arXiv preprint*, 2025. 5, 8

[30] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2001–2010, 2017. 1, 6

[31] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. Accepted. 2, 3

[32] Longxiang Tang, Zhuotao Tian, Kai Li, Chunming He, Hantao Zhou, Hengshuang Zhao, Xiu Li, and Jiaya Jia. Mind the interference: Retaining pre-trained knowledge in parameter efficient continual learning of vision-language models. In *European Conference on Computer Vision (ECCV)*, pages 346–365. Springer, 2024. 1, 2, 3, 5, 6, 7, 8, 4

[33] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng zhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. In *Advances in Neural Information Processing Systems*, 2024. 5, 2

[34] Feng Wang, Jieru Mei, and Alan Yuille. Sclip: Rethinking self-attention for dense vision-language inference. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 1

[35] Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. *arXiv preprint arXiv:2406.09044*, 2024. 5

[36] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 3

[37] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoyu Sun, Haohan Zhang, Ching-Yao Lee, Xinlei Ren, Guodong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision (ECCV)*, pages 631–648. Springer, 2022. 3, 7

[38] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 139–149, 2022. 2, 3, 7

[39] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo-Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6

[40] Bin Wu, Wuxuan Shi, Jinqiao Wang, and Mang Ye. Synthetic data is an elegant gift for continual vision-language models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2, 6, 4, 7

[41] Yinghui Xing, Qirui Wu, De Cheng, Shizhou Zhang, Guoqiang Liang, Peng Wang, and Yanning Zhang. Dual modality prompt tuning for vision-language pre-trained model. *arXiv preprint*, 2022. 3

[42] Qingsen Yan, Dong Gong, Yuhang Liu, Anton van den Hengel, and Javen Qinfeng Shi. Learning bayesian sparse networks with full experience replay for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 109–118, 2022. 2

[43] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3014–3023, 2021. 2

[44] Dianzhi Yu, Xinni Zhang, Yankai Chen, Aiwei Liu, Yifei Zhang, Philip S. Yu, and Irwin King. Recent advances of multimodal continual learning: A comprehensive survey. *arXiv preprint*, 2024. 1

[45] Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23219–23230, 2024. 1, 2, 3, 5, 6, 4, 7

[46] Tao Yu, Zhihe Lu, Xin Jin, Zhibo Chen, and Xinchao Wang. Task residual for tuning vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[47] Yu-Chu Yu, Chi-Pin Huang, Jr-Jen Chen, Kai-Po Chang, Yung-Hsuan Lai, Fu-En Yang, and Yu-Chiang Frank Wang. Select and distill: Selective dual-teacher knowledge transfer for continual learning on vision-language models. In *European Conference on Computer Vision (ECCV)*. Springer, 2024. 2

[48] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3987–3995. PMLR, 2017. 2

[49] Zangwei Zheng, Mingyuan Ma, Kai Wang, Ziheng Qin, Xiangyu Yue, and Yang You. Preventing zero-shot transfer

degradation in continual learning of vision-language models. *arXiv preprint*, 2023. 1, 2, 3, 4, 6, 7

[50] Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2024. 7

[51] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16816–16825, 2022. 3

[52] Nan Zhou, Jiaxin Chen, and Di Huang. ivpt: Improving task-relevant information sharing in visual prompt tuning by cross-layer dynamic connection. *arXiv preprint*, 2024. 1, 4

Enhancing Continual Learning of Vision-Language Models via Dynamic Prefix Weighting

Supplementary Material

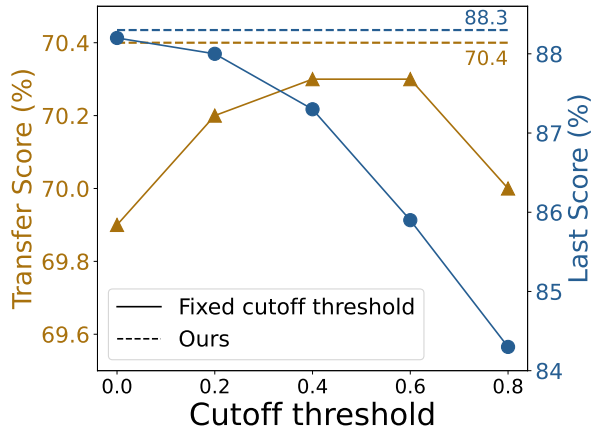

Figure A.1. Comparison between fixed cutoff and the proposed dynamic cutoff strategy.

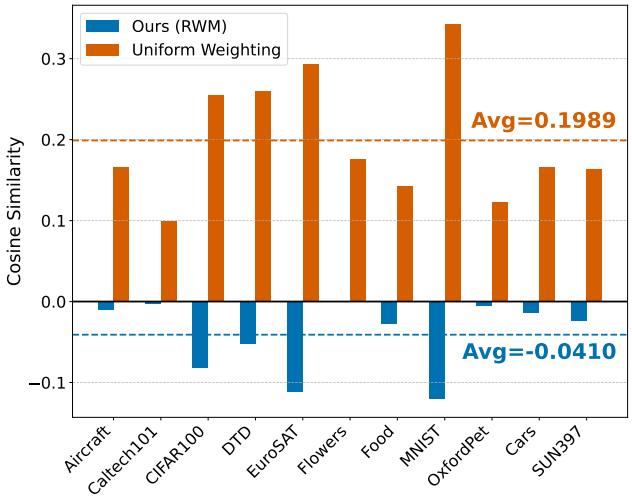

Figure A.2. Comparison between fixed cutoff and the proposed dynamic cutoff strategy.

A. Additional Experimental Results

A.1. Analysis of Conditional Filtering

Our CondAct module refines the prefix weight g_{ijk} through a conditional filtering process, rather than directly applying the outputs of conditional normalization. This additional step selectively suppresses the influence of less relevant prefixes by zeroing out weights that fall below a certain cutoff. Importantly, instead of using a fixed threshold, our method determines this cutoff dynamically, based on the likelihood of each prefix score under the trained task distributions. In Fig. A.1, we compare our dynamic cutoff strategy against several fixed thresholds. In the figure, dashed lines denote the results using our dynamic approach. Our method consistently outperforms fixed thresholds in both *Transfer* and *Last* scores, clearly demonstrating the effectiveness of the proposed filtering mechanism.

A.2. Relationship Between Prefixes and the Adapter

To further understand the relationship between prefixes and the adapter, we analyze the cosine similarity between O_{prefix_i}, which is generated by a linear combination of prefix vectors, and the adapter output O_{adapter_i}. As shown in Fig. A.2, with our residual weighting mechanism (RWM), orthogonality emerges between the prefixes and the adapter. This shows that the adapter complements the prefixes by generating outputs that go beyond the subspace defined by the fixed prefix vectors, thus capturing information that linear combinations of those prefixes alone cannot express. However, as the figure shows, such orthogonality does not emerge when the adapter output is uniformly weighted to all tokens. This highlights that the proposed weighting mechanism plays a crucial role in enabling the complementary synergy between the adapter and the prefixes.

Weighting Method	Transfer	Avg.	Last
$\sum_k g_{ijk}$	69.8	79.1	88.2
$\sum_k \widetilde{g}_{ijk}$	70.1	79.1	88.0
RWM (Ours)	**70.4**	**79.3**	**88.3**

Table A.1. Comparison of weighting strategies for adapter. $\sum_k g_{ijk}$ and $\sum_k \widetilde{g}_{ijk}$ use normalized and filtered prefix weights, respectively.

A.3. Comparison with Alternative Weighting Mechanisms

Our weighting mechanism (RWM) applies token-wise weighting to the adapter's output using the total prefix score $\sum_k \sigma(s_{ijk})$ assigned to each input token. Notably, our method does not utilize the normalized attention weights $\sum_k \widetilde{g}_{ijk}$, which are bounded within the interval $[0, 1]$, but instead relies directly on the raw prefix scores prior to normalization. Rather than adopting an approach that directly transfers the weights assigned to the prefix onto the adapter, our method enables selective focusing by subtracting the prefix contribution from the overall activation score and using only the resulting values that remain non-negative. This design allows the adapter to selectively address tokens that are insufficiently adapted by the prefix alone. As illustrated in

Params.	Single D	Frozen D	SVD Init.	Transfer	Avg.	Last
21.6 M	–	–	–	69.7	79.0	88.1
11.0 M	✓	–	–	58.4	50.9	25.7
11.0 M	✓	✓	–	69.7	78.8	88.0
11.0 M	✓	✓	✓	**70.4**	**79.3**	**88.3**

Table A.2. Ablation study of adapter designs. D denotes the down projection matrix. The first row uses standard LoRA, and the second row applies HydraLoRA [33] with a shared, continually updated D across tasks. Params. represents the number of additional parameters.

Tab. A.1, we compare our RWM with alternative weighting strategies that employ either the sum of normalized weights $\sum_k g_{ijk}$ or filtered weights $\sum_k \widetilde{g}_{ijk}$, obtained through an additional filtering process. The results confirm the effectiveness of our approach, which uses the residual of the raw prefix scores.

A.4. Analysis of Adapter Architecture

In this paper, we adopt the HydraLoRA [33] architecture as our LoRA-based adapter, which achieves parameter efficiency through a single down projection matrix D. To better suit the CL scenarios, we modify this architecture by initializing D using the top-k left singular vectors of the value projection matrix from the CLIP model, and freeze this matrix throughout the CL process. To evaluate the effectiveness of our proposed adapter architecture, we conduct an ablation study on various adapter designs, as summarized in Tab. A.2. The first row in the table corresponds to a baseline using the standard LoRA architecture, which exhibits lower performance compared to our modified adapter. The second row shows results obtained by directly applying the original HydraLoRA [33] architecture, where the D matrix is continually updated for each task. This continual update introduces catastrophic forgetting, resulting in significant performance degradation in the CL scenario. The third row corresponds to another variant, where the down projection matrix D is randomly initialized and then frozen. In this case, the row space of D may lie within a minor subspace, and restricting fine-tuning to this subspace [19] fails to improve performance. Taken together, our proposed adapter design enables fine-tuning directly on the principal subspace of the value projection matrix, effectively leveraging the pretrained knowledge from CLIP. This approach further supports prefixes containing task-specific knowledge and facilitates additional adaptation, thereby improving performance in CL scenarios.

A.5. Computational Costs.

In Tab. A.3, we compare the inference time of our method with other PEFT approaches on the MTIL benchmark. Our method consistently achieves both performance improvements and lower inference time. This efficiency arises from the simplicity of our design: (i) replacing the quadratic query–key dot-product operations with affine transformations, and (ii) removing router operations by directly mapping the residual weights from CondAct to the adapter weights. Together, these design choices significantly reduce computational overhead, resulting in lower inference time.

Method	Inference Time	Mean (%)
MoE-Adapters	35m 47s	76.9
DIKI	3m 46s	76.7
Ours†	**3m 8s**	78.7
Ours	3m 16s	**79.3**

Table A.3. Comparison of methods in terms of throughput. Mean is computed as the average of Transfer, Avg., and Last.

RePA	CondAct	RWM	Inference Time	Mean (%)
-	-	-	3m 44s	76.9
✓	-	-	2m 56s	77.1
-	✓	-	3m 29s	78.0
✓	✓	-	2m 53s	78.9
✓	✓	✓	3m 8s	79.3

Table A.4. Analysis of how each component of our method contributes to the increase in inference time.

In Tab. A.4, we further analyze how each component of our method affects inference time. Starting from the baseline, incorporating RePA substantially reduces inference time by replacing attention with lightweight affine transformations. Using CondAct alone also provides a modest speedup by suppressing less relevant prefixes. When combined, RePA and CondAct further reduce inference time, making the model even faster than RePA alone. This improvement likely arises because RePA produces prefix scores that better capture task relevance, allowing CondAct to suppress more irrelevant prefixes and zero out additional prefix weights, thereby reducing effective computation. Finally, adding RWM slightly increases inference time due to the additional adapter branch, but it achieves the best overall performance while still remaining faster than DIKI.

B. Implementation Details

B.1. Experiments Setup

We follow the baseline setup of DIKI [32] for all our experiments. Specifically, for each task, we use the same handcrafted text prompts as DIKI, combining them with class names to construct the input to the text encoder. Following the baseline, we train the model for 10 epochs using a cosine learning rate scheduler and select the one with the highest validation accuracy, using a consistent data split throughout the process. To ensure a fair comparison, we adopt the CLIP ViT-B/16 [28] model, which is commonly used by other methods on the same benchmark.

Head Adjustment	Transfer (%)	Avg (%)	Last (%)	Params.
Original Head Count (×1)	69.8	78.9	88.2	29.6 M
Reduced Head Count (×0.5)	70.1	79.1	88.1	21.0 M
Expanded Head Count (×2)	69.8	78.9	88.2	46.8 M
Adaptive Head Count (Ours)	**70.4**	**79.3**	**88.3**	30.8 M

Table B.1. Comparison of various head counts and our adaptive head-count selection method.

Building upon this configuration, we minimize cross-entropy loss between model predictions and ground truth labels during training. The learning rate is set to 1.25, and a batch size of 32 is used. The prefix length L is set to 8. The row rank dimension of the LoRA adapter is set to 64 in our default setting (Ours) and reduced to 4 in the parameter-efficient variant (Ours†). Both prefix and adapter modules are integrated into all 12 layers of the visual and text encoders. All experiments are conducted using a single NVIDIA 4090 GPU. For RePA, the bias matrix B_i^G is initialized to -4 for the visual branch and -2 for the text branch. The weight matrix W_i^G and the prefix vector P_V are initialized as orthogonal vectors across all tasks.

B.2. Adaptive Head Count

In this paper, we propose a novel adaptive method for determining the number of heads h' for each task, where h' is a separate set of heads used exclusively in the prefix and adapter modules, distinct from the original number of heads h in CLIP. Specifically, we compute a task-specific scaling factor that captures the discriminability of the feature representations of each task and use it to scale the original number of heads to derive h'. To obtain this factor, we first measure the inter-class and intra-class variances of the features for each task. For image features, let $\mathbf{v}_{i,j}$ denote the normalized feature vector of the j-th sample in class i, and let N_i be the number of samples in class i. We compute the normalized class mean for each class i as:

$$\hat{\mu}_i = \frac{\mu_i}{\|\mu_i\|}, \quad \text{with} \quad \mu_i = \frac{1}{N_i}\sum_{j=1}^{N_i} \mathbf{v}_{i,j}.$$

The inter-class variance for image features is defined as one minus the average cosine similarity between different class means:

$$\text{inter\_variance}_{\text{image}} = 1 - \frac{1}{K(K-1)}\sum_{i\neq j} \hat{\mu}_i \cdot \hat{\mu}_j,$$

where K is the total number of classes. The intra-class variance for image features is computed based on the average pairwise cosine similarity within each class:

$$\text{intra\_variance}_{\text{image}} = 1 - \frac{1}{K}\sum_{i=1}^{K}\left(\frac{1}{N_i(N_i-1)}\sum_{j\neq k}\mathbf{v}_{i,j}\cdot\mathbf{v}_{i,k}\right).$$

For text features, assuming that each class i is represented by a normalized embedding $\mathbf{t}_i$, the inter-class variance is similarly defined as:

$$\text{inter\_variance}_{\text{text}} = 1 - \frac{1}{K(K-1)}\sum_{i\neq j}\mathbf{t}_i \cdot \mathbf{t}_j.$$

Next, we combine these variances into an *easiness score*, which quantifies task difficulty by jointly considering class inter-separability and within-class consistency, with an additional correction for the number of classes. The proposed *easiness score* is computed as follows:

$$\text{easiness} = \frac{\frac{1}{2}\left(\text{inter\_variance}_{\text{image}} + \text{inter\_variance}_{\text{text}}\right)}{\text{intra\_variance}_{\text{image}} + \left(\alpha / n_{\text{cls}}\right)},$$

where n_{cls} is the number of classes, and α (set to 10) is used as a scaling factor to balance the units of the class count term. A higher score indicates well-separated and consistent class features, implying an easier classification task. This score is then divided by the model's zero-shot accuracy (zs_accuracy) to yield the composite scaling factor F for each task:

$$F = \frac{\text{easiness}}{\text{zs\_accuracy}}.$$

Dividing the easiness score by the zero-shot performance allows the model to assess how well its pretrained knowledge aligns with each task. When a task has a high easiness score but the model shows low zero-shot accuracy, this indicates that the task is considered to require more adaptation capacity, as the pretrained features are not sufficiently informative. The original number of heads h is scaled by F and adjusted to the nearest power of two to maintain architectural consistency:

$$h' = \begin{cases} 1, & \text{if } h \times F = 0, \\ 2^{\text{round}(\log_2(h\times F))}, & \text{otherwise.} \end{cases}$$

As shown in Tab. B.1, this adaptive adjustment yields superior performance compared to simply increasing the number of parameters uniformly, highlighting the effectiveness of our adaptive approach.

C. Details about Benchmark

C.1. Datasets

Both MTIL and ODCL-CIL benchmarks were evaluated using two distinct dataset orders, followed by [49]. MTIL is designed for the task-incremental setting, while ODCL-CIL, also referred to as MCIL, targets the class-incremental setting. In Order-I, the datasets are arranged alphabetically as follows: Aircraft, Caltech101, CIFAR100, DTD, EuroSAT, Flowers, Food, MNIST, OxfordPet, StanfordCars, and SUN397. In contrast, Order-II lists the datasets in a random sequence: StanfordCars, Food, MNIST, OxfordPet, Flowers, SUN397, Aircraft, Caltech101, DTD, EuroSAT, and CIFAR100.

Method	Trans	Avg	Last	Mean
DIKI [32]	68.7	72.5	78.6	73.3
MoE-Adapter [45]	69.3	75.3	82.3	75.6
Ours	**69.6**	**75.9**	**83.7**	**76.4** (+0.8)

Table C.1. Comparison with SOTA methods on 16-shot MTIL-FS benchmark. Results for DIKI and MoE-Adapter are based on our replications.

C.2. Metrics

We further formulate the *Transfer*, *Avg*, and *Last* metrics, which were originally introduced in [49]. Let $p_j^{(i)}$ denote the accuracy achieved on task j after the model has been trained on task i. Given a total of T tasks, these metrics are computed as follows.

The **Transfer** metric, which measures the model's forward forgetting by evaluating its zero-shot performance after completing training on task j is defined as:

$$\text{Transfer}_j = \frac{1}{j-1}\sum_{i=1}^{j-1} p_j^{(i)}, \quad j = 2, 3, \ldots, T.$$

The **Avg** metric, representing the average performance across all tasks, is defined as:

$$\text{Avg}_j = \frac{1}{T}\sum_{i=1}^{T} p_j^{(i)}, \quad j = 1, 2, \ldots, T.$$

Finally, the **Last** metric, which reflects the final performance after training on all tasks, is given by:

$$\text{Last}_j = p_j^{(T)}, \quad j = 1, 2, \ldots, T.$$

C.3. Few-Shot Setting

In contrast to the baseline few-shot setup used in DIKI [32], we conduct experiments across all datasets included in the MTIL benchmark. Specifically, DIKI excluded certain datasets, arguing that prompt-based CL methods fail to capture sufficient information from few-shot samples in some tasks. However, our method effectively addresses this limitation through a conditional filtering process that suppresses the influence of noisy prefixes while simultaneously leveraging adapters to supplement additional required information. For a fair comparison, we exclude GIFT [40], which utilizes additional synthetic data generated via diffusion models. Instead, we replicate MoE-Adapter [45], another PEFT-based method, alongside DIKI [32], adapting both to our experimental setting for comparison. A summary of the results is provided in Tab.C.1, with the complete table in Tab.E.1. In the few-shot setting, our method outperforms other SOTA methods, demonstrating non-trivial improvements. These results indicate that our approach performs well even in data-scarce environments.

C.4. Detailed Results

Tab. E.2 shows the performance comparison with SOTA methods on the second-order setting of the MTIL benchmark. Our method consistently outperforms other approaches in the second-order setting as well. The complete results for both orders can be found in Tab. E.3 and Tab .E.4, respectively. The complete results for the ODCL-CIL setting are included in Tab. E.5.

D. Theoretical Analysis of CondAct for Forward Forgetting

In this section, we provide a theoretical analysis demonstrating how our proposed CondAct mechanism mitigates forward forgetting by bounding the prefix-induced drift in token-level representations.

In the pretrained model, the token-level representation of the j-th input token at the i-th attention head is computed as

$$h_{ij}^{orig} = \sum_{u=1}^{m} \alpha_{iju} V_{iu}, \quad \text{where} \sum_{k=1}^{m} \alpha_{ijk} = 1.$$

Here, α_{ijk} denotes the attention weights, and V_{iu} denotes the corresponding projected value vectors. We define the prefix-induced representation as:

$$p_{ij} = \sum_{k=1}^{L} w_{ijk} P_{V_k}^{(i)},$$

where $P_{V_k}^{(i)}$ is the value vector associated with the k-th prefix token, and w_{ijk} represents the attention weight between the j-th input token and k-th prefix token. These weights are obtained by applying a sigmoid gate to the attention score. Specifically, let s_{ijk} denote the attention score between the i-th input token and the j-th prefix token; then we consider two cases:

$$w_{ijk} = \begin{cases} \sigma(s_{ijk}), & \text{without CondAct, Sigmoid-only.} \\ g_{ijk} \text{ or } \tilde{g}_{ijk}, & \text{with CondAct, Eq. (8).} \end{cases}$$

Assumption (Bounded prefix values). *There exists a constant $c_V > 0$ such that*

$$\left\| P_{V_k}^{(i)} \right\|_2 \le c_V \quad \text{for all prefix tokens } k.$$

Proposition (Prefix-induced drift bound). *For any token j at head i, the norm of the prefix-induced representation p_{ij} satisfies*

$$\left\| p_{ij} \right\|_2 \le L c_V \quad \text{(without CondAct, Sigmoid-only),}$$
$$\left\| p_{ij} \right\|_2 \le c_V \quad \text{(with CondAct, Eq. (8)).}$$

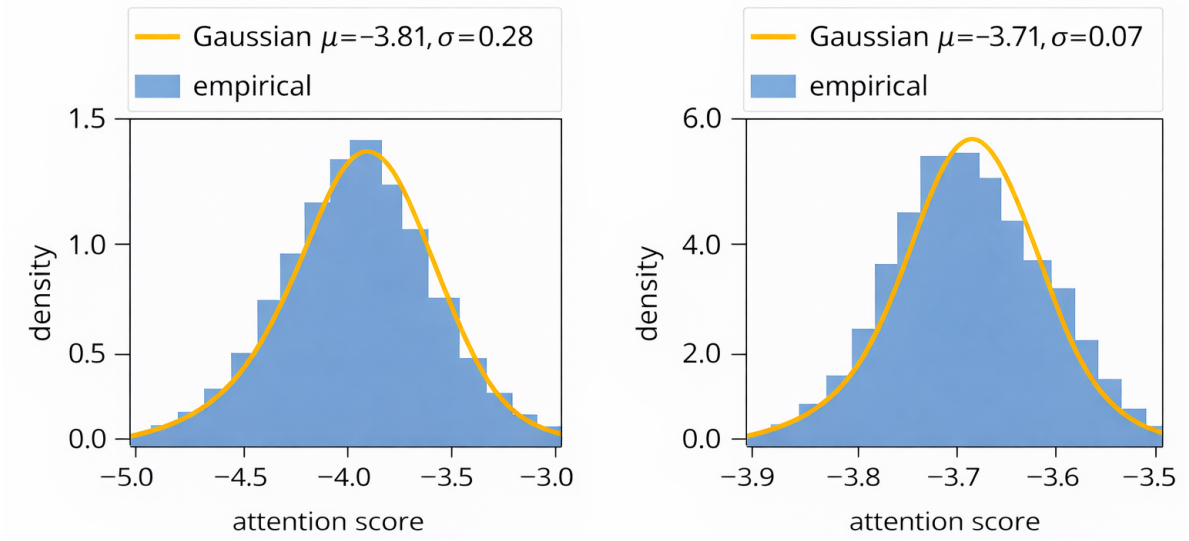

Figure C.1. Mean and variance of attention scores between prefix tokens and the [CLS] token measured on the Aircraft dataset. The left plot shows results from Layer 3, and the right plot shows results from Layer 9. The observed distributions closely match their Gaussian counterparts, supporting the validity of the proposed cutoff design.

In particular, CondAct reduces the worst-case prefix-induced drift per token from Lc_V to c_V.

Proof. By the definition of prefix-induced representation,

$$\|p_{ij}\|_2 = \Big\|\sum_{k=1}^{L} w_{ijk}\, P_{V_k}^{(i)}\Big\|_2 \le \sum_{k=1}^{L} |w_{ijk}|\, \|P_{V_k}^{(i)}\|_2,$$

where we used the triangle inequality. In the Sigmoid-only setting, $w_{ijk} = \sigma(s_{ijk}) \in [0,1]$, so $|w_{ijk}| \le 1$ for all k. Together with the previous assumption, this yields

$$\|p_{ij}\|_2 \le \sum_{k=1}^{L} \|P_{V_k}^{(i)}\|_2 \le \sum_{k=1}^{L} c_V = Lc_V,$$

We now consider the CondAct case. By construction (Eq. (8a)), CondAct applies conditional normalization such that the total prefix weight per token is explicitly bounded,

$$\sum_{k=1}^{L} g_{ijk} \le 1, \qquad g_{ijk} \ge 0 \quad \text{for all } i, j, k.$$

In this case as well, under the same assumption as above, we obtain

$$\|p_{ij}\|_2 \le \sum_{k=1}^{L} g_{ijk}\, \|P_{V_k}^{(i)}\|_2 \le c_V \sum_{k=1}^{L} g_{ijk} \le c_V,$$

□

Corollary (Relative prefix-induced drift). *Suppose the pretrained representation satisfies*

$$\|h_{ij}^{\text{orig}}\|_2 \ge c_h > 0$$

for some constant c_h. Then the relative drift satisfies

$$\frac{\|p_{ij}\|_2}{\|h_{ij}^{\text{orig}}\|_2} \le \begin{cases} L \cdot \dfrac{c_V}{c_h}, & \text{without CondAct,} \\ \dfrac{c_V}{c_h}, & \text{with CondAct.} \end{cases}$$

In summary, CondAct reduces the worst-case relative drift by a factor of up to L.

E. Prefix Score Distribution

In Eq. (9), we leverage the empirical observation that the attention scores between prefix tokens and the [CLS] token follow a Gaussian distribution, which enables us to determine the distribution used for computing the cutoff. To validate this assumption, we measure the prefix scores on the Aircraft dataset at Layers 3 and 9, which are arbitrarily selected layers. As shown in Fig. C.1, the empirical distributions of the prefix scores at both layers closely align with the corresponding Gaussian fits, indicating that the Gaussian assumption provides a reasonable approximation. This observation supports the stability of our cutoff design.

Task	Method	Aircraft	Caltech101	CIFAR100	DTD	EuroSAT	Flowers	Food	MNIST	OxfordPet	Cars	SUN397	Average
MTIL-FS	**Transfer**												
	DIKI [32]	-	92.7	68.6	44.1	47.8	69.9	86.1	58.4	89.0	65.1	65.2	68.7
	MoE-Adapters [45]	-	88.4	68.2	42.9	53.5	70.1	88.5	63.3	89.0	64.7	65.1	69.3
	Ours	-	**92.9**	68.9	45.2	**53.7**	**71.1**	86.4	60.4	89.7	66.0	65.8	**69.6** (+0.3)
	Avg.												
	DIKI [32]	39.7	95.6	76.7	61.4	61.2	83.3	86.3	69.3	90.2	67.6	66.1	72.5
	MoE-Adapters [45]	43.3	92.1	77.2	66.7	80.0	84.8	88.3	74.4	89.0	67.1	65.9	75.3
	Ours	**51.3**	**95.7**	**77.5**	**65.9**	**74.1**	**85.3**	86.7	**71.2**	91.0	68.8	67.1	**75.9** (+0.6)
	Last												
	DIKI [32]	39.6	95.8	78.5	68.0	67.0	94.4	86.4	88.3	93.5	78.6	74.3	78.6
	MoE-Adapters [45]	43.3	92.6	79.2	75.6	95.1	97.1	88.1	93.8	89.1	78.0	73.7	82.3
	Ours	**51.4**	**96.0**	**79.4**	**73.8**	**90.2**	**97.4**	87.0	**91.8**	**93.9**	83.4	76.6	**83.7** (+1.4)

Table E.1. Comparison with SOTA methods on MTIL-FS benchmark (Order I) in terms of "Transfer", "Average", and "Last" scores (%). Best results are highlighted in **bold**. Ours† indicates our reduced-parameter variant.

Method	Extra data	Train Params.	Transfer	Avg.	Last	Mean
Zero-shot	–	–	65.4	65.3	65.3	65.3
LwF [21]	✓	149.6 M	53.2	62.2	71.9	62.4
iCaRL [30]	✓	149.6 M	50.9	56.9	71.6	59.8
WiSE-FT [39]	✓	149.6 M	51.0	61.5	72.2	61.6
ZSCL [49]	✓	149.6 M	64.2	74.5	83.4	74.0
DIKI [32]	×	1.8 M	64.4	74.5	85.5	74.8
MoE-Adapter [45]	×	59.6 M	64.3	74.7	84.1	74.4
GIFT [40]	✓	149.6 M	**65.9**	75.7	85.3	75.6
Ours†	×	4.6 M	65.3	76.0	87.6	76.2 **(+0.6)**
Ours	×	30.8 M	65.7	**76.4**	**88.1**	**76.7** **(+1.1)**

Table E.2. Comparison of SOTA methods on MTIL Order II.

Task	Method	Aircraft	Caltech101	CIFAR100	DTD	EuroSAT	Flowers	Food	MNIST	OxfordPet	Cars	SUN397	Average
MTIL	**Transfer**												
	ZSCL [49]	-	86.0	67.4	45.4	50.4	69.1	87.6	61.8	86.8	60.1	**66.8**	68.1
	DIKI [32]	-	**92.9**	69.0	43.2	48.2	67.4	85.2	63.0	87.9	63.8	66.2	68.7
	MoE-Adapters [45]	-	87.9	68.2	44.4	49.9	70.7	**88.7**	59.7	89.1	64.5	65.5	68.9
	GIFT [40]	-	88.5	**69.8**	**46.0**	49.4	68.5	87.1	**69.9**	88.9	57.7	67.7	69.3
	Ours†	-	**92.9**	68.9	45.2	53.7	**71.1**	86.4	60.4	**89.7**	**66.0**	65.8	70.0 (+0.7)
	Ours	-	**92.9**	69.0	45.3	**54.2**	**71.1**	86.2	63.8	89.3	65.7	66.5	**70.4** (+1.1)
	Avg.												
	ZSCL [49]	45.1	92.0	80.1	64.3	79.5	81.6	**89.6**	75.2	88.9	64.7	**68.0**	75.4
	DIKI [32]	45.1	95.5	83.1	64.8	79.9	83.5	87.0	76.2	89.6	67.0	67.1	76.3
	MoE-Adapters [45]	50.2	91.9	83.1	69.4	78.9	84.0	89.1	73.7	89.3	67.7	66.9	76.7
	GIFT [40]	51.9	93.9	81.4	67.7	80.3	82.8	89.3	80.6	90.3	63.1	68.9	77.3
	Ours†	57.8	**96.2**	83.9	69.2	82.1	**85.8**	87.8	74.6	**91.2**	**69.6**	67.0	78.6 (+1.3)
	Ours	**60.4**	**96.2**	**84.5**	**70.8**	**82.4**	**85.8**	87.7	**76.8**	90.9	69.4	67.6	**79.3** (+2.0)
	Last												
	ZSCL [49]	40.6	92.2	81.3	70.5	94.8	90.5	**91.9**	98.7	93.9	85.3	80.2	83.6
	DIKI [32]	45.2	95.7	86.3	72.9	98.0	97.0	89.2	99.4	94.2	81.6	76.6	85.1
	MoE-Adapters [45]	49.8	92.2	86.1	78.1	95.7	94.3	89.5	98.1	89.9	81.6	80.0	85.0
	GIFT [40]	47.9	95.6	82.8	75.1	97.3	94.2	91.7	99.2	94.2	**87.0**	**80.9**	86.0
	Ours†	57.8	96.5	87.3	78.2	98.3	**98.1**	89.5	99.5	94.9	85.6	78.1	87.6 (+1.6)
	Ours	**60.5**	**96.6**	**87.9**	**80.4**	**98.5**	**98.1**	89.4	**99.6**	**95.1**	86.3	78.8	**88.3** (+2.3)

Table E.3. Comparison with SOTA methods on MTIL benchmark (Order I) in terms of "Transfer", "Average", and "Last" scores (%). Best results are highlighted in **bold**. Ours† indicates our reduced-parameter variant.

Task	Method	Cars	Food	MNIST	OxfordPet	Flowers	SUN397	Aircraft	Caltech101	DTD	EuroSAT	CIFAR100	Average
MTIL	**Transfer**												
	ZSCL [49]	-	88.3	57.5	84.7	68.1	64.8	21.1	88.2	**45.3**	**55.2**	68.2	64.2
	DIKI [32]	-	85.8	59.8	89.1	71.8	62.6	24.3	93.3	42.7	46.8	67.8	64.4
	MoE-Adapters [45]	-	**88.8**	59.5	89.1	69.9	64.4	18.1	86.9	43.7	54.6	68.2	64.3
	GIFT [40]		88.3	**63.4**	88.1	70.8	**67.7**	22.8	90.4	**46.7**	51.8	68.8	**65.9**
	Ours†	-	86.9	61.6	89.3	71.3	63.2	24.3	93.3	44.8	**50.4**	68.6	65.3 (-0.6)
	Ours	-	85.9	63.3	**89.4**	**72.2**	64.2	**24.6**	**94.0**	44.6	49.4	**69.1**	65.7 (-0.2)
	Avg.												
	ZSCL [49]	81.7	**91.3**	91.1	91.0	82.9	72.5	33.6	89.7	53.3	**62.8**	69.9	74.5
	DIKI [32]	81.9	88.9	92.1	92.8	87.7	70.3	34.3	94.2	51.5	56.1	69.5	74.5
	MoE-Adapters [45]	84.9	89.9	89.3	91.4	86.2	72.2	33.4	89.4	53.3	61.4	69.9	74.7
	GIFT [40]	83.2	90.8	92.6	92.8	85.8	**74.1**	36.0	92.1	**54.7**	60.0	70.4	75.7
	Ours†	85.7	89.1	92.4	93.2	88.4	71.3	39.1	94.5	53.2	59.1	70.3	76.0 **(+0.3)**
	Ours	**86.6**	88.9	**92.8**	**93.6**	**88.5**	71.5	**40.4**	**94.9**	54.2	58.3	**70.9**	**76.4 (+0.7)**
	Last												
	ZSCL [49]	78.2	**91.1**	97.6	92.5	87.4	78.2	45.0	92.3	72.7	96.2	86.3	83.4
	DIKI [32]	81.9	89.2	99.4	94.3	96.8	76.7	46.3	95.9	74.8	98.3	86.6	85.5
	MoE-Adapters [45]	84.1	88.5	94.0	91.8	94.1	77.8	50.4	93.3	77.1	87.7	86.6	84.1
	GIFT [40]	81.0	90.2	98.6	94.0	91.5	**78.6**	51.7	94.6	75.6	95.4	86.6	85.3
	Ours†	85.7	89.4	99.5	94.7	**98.1**	78.1	56.9	**96.6**	78.4	**98.4**	87.3	87.6 **(+2.3)**
	Ours	**86.6**	89.2	**99.6**	**95.1**	97.9	78.4	**59.4**	96.4	**79.8**	**98.4**	**87.9**	**88.1 (+2.8)**

Table E.4. Comparison with SOTA methods on MTIL benchmark (Order II) in terms of "Transfer", "Average", and "Last" scores (%). Best results are highlighted in **bold**. Ours† indicates our reduced-parameter variant.

Task	Method	Aircraft	Caltech101	CIFAR100	DTD	EuroSAT	Flowers	Food	MNIST	OxfordPet	Cars	SUN397	Average
ODCL-CIL (MCIL)	**Transfer**												
	ZSCL [49]	-	84.6	67.5	44.8	51.5	69.0	87.6	62.3	87.1	59.7	66.4	68.0
	MoE-Adapters [45]	-	88.2	66.8	44.7	54.1	70.6	88.4	59.5	89.0	64.7	65.0	69.1
	CoLeCLIP [18]	-	88.2	65.1	44.7	54.1	68.8	**88.5**	59.5	89.0	64.7	65.1	68.8
	DPeCLIP [22]	-	88.2	67.2	44.7	54.0	70.6	88.2	59.5	89.0	64.7	64.8	69.1
	Ours†	-	**92.9**	68.9	45.2	53.7	**71.1**	86.4	60.4	**89.7**	**66.0**	65.8	70.0 **(+0.9)**
	Ours	-	**92.9**	**69.0**	**45.3**	**54.2**	**71.1**	86.2	**63.8**	89.3	65.7	**66.5**	**70.4 (+1.3)**
	Avg.												
	ZSCL [49]	46.3	68.3	74.3	56.3	79.1	81.4	89.5	74.0	89.0	64.4	67.5	71.8
	MoE-Adapters [45]	37.2	65.3	79.5	67.6	19.7	83.1	80.5	74.0	88.5	67.5	65.3	66.2
	CoLeCLIP [18]	48.2	77.8	71.7	65.7	76.8	83.8	89.6	72.2	90.3	68.0	66.4	73.7
	DPeCLIP [22]	49.9	85.3	81.5	65.3	81.6	84.3	**89.9**	74.0	90.4	68.3	66.2	76.1
	Ours†	57.3	92.8	83.2	67.5	81.7	**85.0**	87.7	74.5	**90.6**	**69.5**	66.9	77.9 **(+1.8)**
	Ours	**60.2**	**93.5**	**83.9**	**68.9**	**82.0**	84.8	87.6	**76.7**	90.4	69.4	**67.6**	**78.6 (+2.5)**
	Last												
	ZSCL [49]	42.5	64.4	67.2	54.8	89.7	90.4	91.7	95.8	93.4	85.2	78.3	77.6
	MoE-Adapters [45]	34.1	47.6	80.9	75.5	0.00	93.0	70.8	**99.4**	86.4	79.8	68.9	66.9
	CoLeCLIP [18]	48.1	73.1	65.2	69.6	84.0	96.2	90.9	94.6	93.5	82.6	79.3	79.7
	DPeCLIP [22]	49.9	84.2	83.2	71.1	97.0	95.8	**92.0**	**99.4**	**93.9**	84.5	**80.2**	84.6
	Ours†	57.2	89.9	85.2	74.1	97.7	**96.5**	89.1	99.2	92.7	85.1	77.4	85.8 **(+1.2)**
	Ours	**60.1**	**91.1**	**85.9**	**75.7**	**97.9**	96.2	89.1	99.2	93.0	**86.0**	78.3	**86.6 (+2.0)**

Table E.5. Comparison with SOTA methods on ODCL-CIL benchmarks in terms of "Transfer," "Average," and "Last" scores (%). Best results are highlighted in **bold**. Ours† indicates our reduced-parameter variant.