MoE-Adapters++: Toward More Efficient Continual Learning of Vision-Language Models Via Dynamic Mixture-of-Experts Adapters

Jiazuo Yu, Zichen Huang, Yunzhi Zhuge, Lu Zhang, *Member, IEEE*, Ping Hu, *Member, IEEE*, Dong Wang, Huchuan Lu, *Fellow, IEEE*, and You He

***Abstract*—In this paper, we first propose MoE-Adapters, a parameter-efficient training framework to alleviate long-term forgetting issues in incremental learning with Vision-Language Models (VLM). Our MoE-Adapters leverages incrementally added routers to activate and integrate exclusive expert adapters from a pre-defined static expert set, enabling the pre-trained CLIP to efficiently adapt to new tasks. To preserve the zero-shot capability of VLM, a Distribution Discriminative Auto-Selector (DDAS) is introduced that automatically routes in-distribution and out-of-distribution inputs to the MoE-Adapters and the original CLIP, respectively. However, relying on a static expert set and a separate distribution selector can lead to parameter redundancy and increased training complexity. In response, we further extend an MoE-Adapters++ framework by introducing dynamic MoE-adapters, which allows experts to be adaptively involved during the continual learning process. Additionally, a Latent Embedding Auto-Selector (LEAS) is proposed that incorporates distribution selection within CLIP to create a more unified architecture. Extensive experiments across diverse settings demonstrate that the proposed method consistently surpasses previous state-of-the-art approaches while concurrently improving training efficiency.**

***Index Terms*—Continual learning, mixture-of-experts, parameter efficient fine-tuning, vision-language models.**

I. Introduction

ARTIFICIAL Intelligence (AI), particularly in the realm of large-scale foundation models, has made significant strides in understanding the open world, as evidenced by recent advancements [1], [2], [3], [4], [5]. An ideal AI, akin to human cognition, should be able to continuously assimilate new knowledge from the dynamic environment. Traditional fully-supervised training paradigms struggle to adapt to this scenario due to the high computational costs of integrating new data with historical datasets. In contrast, Continual Learning (CL) offers an efficient solution by incrementally acquiring new knowledge from incoming data. However, CL faces a significant challenge of "catastrophic forgetting", where a model loses previously acquired knowledge upon learning new tasks [6], [7], [8], [9], [10], [11].

To remedy this issue, one of the popular solutions in current CL methods [17], [18], [19], [20] is to develop dynamic expansion frameworks by incrementally adding task-specific components to a shared base model (see Fig. 1(a)). Although these methods show promise in memorization and scalability, they cannot distinguish unseen data, thereby overlooking zero-shot transfer capability. Recent advancements like ZSCL [12] have brought the zero-shot transfer ability into continual learning by leveraging a pre-trained Vision Language Model (VLM). As illustrated in Fig. 1(b), this method relies on knowledge distillation to integrate zero-shot generalization ability from the frozen CLIP [3] and uses parameter regularization to prevent knowledge degradation in continual learning. However, these designs often entail large computational burdens and exhibit limitations in long-term memorization.

Recently, Parameter-Efficient Fine-Tuning (PEFT) methods [21], [22], [23], [24], [25], [26], [27] have demonstrated that large-scale models can quickly adapt to downstream tasks via fine-tuning only a small subset of parameters. This has inspired subsequent approaches [13], [14], [15] to apply prompt-tuning in pre-trained models, enabling efficienct generalization and mitigating forgetting in CL. Recently, DIKI [16] leverages PEFT techniques to form a distribution-aware interference-free knowledge integration framework, where a fully residual mechanism is used to infuse new knowledge to a pre-trained VLM. Nevertheless, these methods are still limited by the accuracy performance of transfer learning or lacking collaboration between tasks.

To overcome the outlined challenges, in [28], we leverage recent advancements of Mixture-of-Experts (MoE) [29], [30] and propose a parameter-efficient continual learning framework. As illustrated in Fig. 1(c), we build an expansible architecture on

Received 21 November 2024; revised 7 August 2025; accepted 9 August 2025. Date of publication 11 August 2025; date of current version 5 November 2025. This work was supported by the National Natural Science Foundation of China under Grant 62206039, Grant 62406053, Grant 62476048, and Grant 62293544, in part by the Fundamental Research Funds for the Central Universities under Grant DUT24RC(3)025, in part by Basic Scientific Research Funding of the Central Universities of China under Grant DUTZD25225, and in part by Liaoning Provincial Science and Technology Joint Program Project under Grant 2024011188-JH2/1026 and Grant 2024-0011 (ZX20240867). Recommended for acceptance by J. Dai. *(Jiazuo Yu and Zichen Huang contributed equally to this work.) (Corresponding author: Lu Zhang.)*

Jiazuo Yu, Zichen Huang, Yunzhi Zhuge, Lu Zhang, Dong Wang, and Huchuan Lu are with the School of Information and Communication Engineering, Dalian University of Technology, Dalian 116024, China (e-mail: yujiazuo@mail.dlut.edu.cn; huangzichen@mail.dlut.edu.cn; zgyz@dlut.edu.cn; zhangluu@dlut.edu.cn; wdice@dlut.edu.cn; lhchuan@dlut.edu.cn).

Ping Hu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: chinahuping@gmail.com).

You He is with Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China (e-mail: heyou@tsinghua.edu.cn).

The code and models are located at https://github.com/JiazuoYu/MoE-Adapters4CL.

Digital Object Identifier 10.1109/TPAMI.2025.3597942

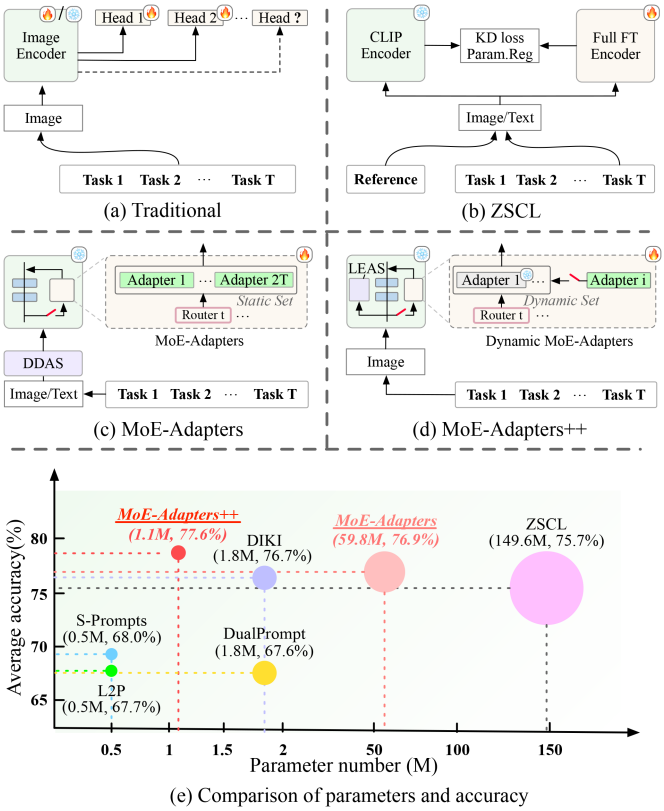


Fig. 1. (a) Traditional dynamic expansion-based CL cannot distinguish unseen data. (b) Zero-shot CL [12] suffers from significant computational burdens. (c) The proposed MoE-Adapters and DDAS collaborate to form a parameter-efficient zero-shot CL. (d) The extended MoE-Adapters++ introduces dynamic MoE-Adapters and a Latent Embedding Auto-Selector (LEAS) to form a more flexible and unified framework for improving training efficiency. (e) Comparison between our models and other methods in terms of accuracy and training parameters, where the "Average accuracy" represents the mean of three metrics: Last, Average, and Transfer. The compared methods include ZSCL [12], S-Prompts [13], L2P [14], DualPrompt [15], and DIKI [16].

a frozen CLIP model [3], dubbed as incremental MoE-Adapters. Specifically, we take adapters as sparse experts and utilize incrementally incorporated task-specific routers to activate the corresponding $Top\text{-}k$ experts from pre-defined expert set. In the continual learning process, we further employ an activate-freeze strategy to facilitate intra-task knowledge acquisition and encourage inter-task collaboration. Additionally, a Distribution Discriminative Auto-Selector module (DDAS) is proposed to automatically allocate the testing data to MoE-Adapters or the pre-trained CLIP, enabling effective predictions on seen data and zero-shot for unseen data within a unified system.

Despite achieving state-of-the-art performance in Multi-domain Task Incremental Learning (MTIL) [12], certain limitations persist in the architectural design of MoE-Adapters [28]. First, the number of experts is typically determined by the total number of tasks. This not only results in parameter redundancy when addressing earlier tasks but also limits flexibility in adapting to new tasks beyond predefined benchmarks. Furthermore, the parallel integration of the DDAS module with CLIP increases training complexity and hinders the unification of the entire network. To improve training efficiency, we extend our previous work [28] as MoE-Adapters++ by introducing a dynamic MoE-Adapters and a Latent Embedding Auto-Selector (LEAS), as illustrated in Fig. 1(d). Specifically, we implement the Dynamic MoE-Adapters in the frozen CLIP model, where both routers and experts are incrementally incorporated to adapt to new tasks. Instead of relying on a static expert set, we design a Dynamic Expert-expansion Controller (DEeC) strategy to adaptively determine whether to add a new trainable expert to the expert set by controlling an expansion signal. Additionally, we propose a Latent Embedding Auto-Selector (LEAS) that integrates distribution selection into the early layers of CLIP to form a unified network architecture. The LEAS eliminates the need for an additional feature extractor, effectively reducing trainable parameters and improving computational efficiency.

Extensive experiments across various settings demonstrate the effectiveness of the MoE-Adapters and MoE-Adapters++ in mitigating the forgetting issue. Furthermore, the extended version reduces parameter burdens by 98% and GPU memory requirements by 70% compared with the original MoE-Adapters, achieving the optimal trade-off between efficiency and performance (see Fig. 1(e)). Overall, our contributions can be summarized as follows:

- We introduce a parameter-efficient training framework for vision-language models in continual learning, which employs incremental MoE-Adapters for enhanced adaptability and efficiency.
- We develop an incremental activate-freeze strategy in the MoE framework, enabling experts to simultaneously acquire intra-task knowledge and engage in inter-task collaboration.
- We design a Distribution Discriminative Auto-Selector (DDAS) network for automated substream assignment, effectively merging anti-forgetting and zero-shot transfer capabilities.
- We propose an MoE-Adapters++ framework that further enhances training efficiency and architectural flexibility. Experiments demonstrate that the proposed method consistently performs well in both accuracy and efficiency.

II. Related Work

Continual Learning: Depending on the domain variations of incremental data, existing continual learning methods mainly focus on addressing *i.e.,* Class Incremental Learning (CIL) [31], [32], [33], [34], [35], [36], [37], [38] and Task Incremental Learning (TIL) [12], [39], [40]. Existing efforts in this area have been made by developing various architectures [41], including memory-based, regularization-based, and dynamic-based models. Memory-based methods [42], [43], [44], [45], [46], [47], [48] retain the historical knowledge by storing them in a memory bank, which will be accessed and updated in incremental learning. However, the continuously increasing learned data usually poses a burden on the memory bank, resulting in limited lifelong learning ability. Regularization-based methods add explicit regularization terms on weights [49], [50], [51], [52] or data [20], [53], [54], [55] to balance between the older and new tasks. They are usually used as an auxiliary trick in

memory-based or dynamic models to alleviate the forgetting issue. Dynamic methods [17], [34], [56], [57], [58], [59], [60], [61] address continual learning by incrementally adding new parameters on the baseline, such as neurons, branches or prediction heads. Dynamic methods usually perform favorably against the other two pipelines. However, like memory-based methods, the dynamic architecture often incurs large-scale model sizes, limiting the models' efficiency. Despite the promising performance of the approaches aforementioned, addressing the crucial capability of AI agents, namely zero-shot transfer to unseen knowledge, remains challenging and complex to integrate into existing popular pipelines. In this paper, we propose incorporating the dynamic architecture on vision-language models to boost their memorization of historical knowledge and alleviate the degradation of zero-shot transfer ability. The highly related work is ZSCL [12], which uses parameter regularization in the continual learning of large-scale models. In contrast to the fully fine-tuning strategy in ZSCL, our method propose an incremental MoE-Adapters and a dynamic MoE-Adapters++ to reduce the number of tuned parameters and enhance collaboration between previously learned adapters and newly added ones.

Parameter Efficient Fine-Tuning: In the realm of Natural Language Processing (NLP), fine-tuning large-scale models (*e.g.,* 175B GPT-3 [62]) imposes significant burdens in both parameter complexity and time consumption. Thus, several parameter-efficient fine-tuning methods [25], [27], [63], [64], [65], [66], [67] have been explored, which only set a few trainable parameters and fine-tune them for efficiency. The success of efficient tuning strategies in NLP promotes their applications on vision-language models [21], [22], [23], [68], [69] like CLIP [21]. Recently, some methods utilize a small set of parameters to continually adapt a pre-trained model to sequential tasks, such as prompt-based [13], [14], [15] and adapter-based [35] methods. These methods are primarily applied to CIL, and they fail to retain the pre-trained knowledge, leading to a degradation in zero-shot performance. Building on this, DIKI [16] proposes a distribution-aware interference-free knowledge integration framework to maintain the pre-trained knowledge and enhance the generalization capabilities of VLMs in TIL. However, its task-specific tuning paradigm limits the collaboration of knowledge across tasks. In this paper, we propose a novel PEFT approach with inter-task collaboration to boost both the anti-forgetting and zero-shot abilities in continual learning. Our model can flexibly adapt to CIL and TIL, achieving promising results even when trained with limited data.

Mixture of Experts: The MoE [29] contains multiple experts and a routing network. It aggregates the expert outputs via a weighted strategy by the routing network. Based on the sparse architecture of MoE [30], some methods [70], [71], [72], [73] are proposed to decrease computational costs and improve model capacity. This technique is also introduced to continual learning to mitigate the forgetting issue. For example, Aljundi et al. [17] propose to train multiple backbones as experts and automatically feed the test samples to a relevant expert. Chen et al. [74] utilize the pre-trained experts and gates to store previous knowledge. These methods have demonstrated MoE's promising performance in continual learning. We propose incremental MoE-Adapters and dynamic MoE-Adapters++ for continual learning. We use adapters as experts to increase the adaption speed and introduce an incremental expert interaction strategy to facilitate the collaboration of experts during continual learning.

III. MoE-Adapters

A. Continual Learning

Given a set of T tasks $\{\mathcal{T}^t\}_{t=1}^{T}$, continual learning works by sequentially accessing and learning on each task $\mathcal{T}^t = \{\mathcal{D}^t, \mathcal{C}^t\}$. Here, $\mathcal{D}^t = \{I_i^t, y_i^t\}_{i=1}^{N^t}$ represents the data of t^{th} task $\mathcal{T}^t$, where I_i^t is the input image, $y_i^t \in \mathcal{C}^t$ is the corresponding class label, and N^t is the size of data. The category set $\mathcal{C}^t = \{c_j^t\}_{j=1}^{M^t}$ encompasses the class names within $\mathcal{T}^t$, with a total of M^t classes. CL aims to achieve good performance across all tasks and can be broadly categorized into Task Incremental Learning (TIL) and Class Incremental Learning (CIL). In TIL, the model generates predictions within a task-specific set $\mathcal{C}^t$, which is determined by the current task identity t. In CIL, the challenge involves distinguishing between all the previously classes $\cup_{i=1}^{t}\mathcal{C}^i$.

B. Framework Overview

We first present a parameter-efficient framework, achieving robust historical knowledge memorization without sacrificing the zero-shot generalization abilities. Our method is built upon the CLIP [21] model, which contains parallel encoders $(\mathcal{F}_I, \mathcal{F}_T)$ to extract features of input images and texts, respectively. The overall framework of our method is shown in Fig. 2. Specifically, we introduce an MoE structure onto a frozen CLIP to consolidate all the downstream tasks within a unified model, in which the task-specific routers are sequentially added to modulate the experts for each task. Adapter modules, such as LoRA [25], function as the experts in the MoE setup, enhancing adaptation speed during training. To relieve MoE's reliance on task identities, we further propose a Distribution Discriminative Auto-Selector (DDAS). The DDAS automatically infers the task context by analyzing the variations of the target image distributions. As a result, in-distribution data will be allocated to the corresponding routers within MoE, while the out-of-distribution inputs will be identified and directed to the original CLIP to perform zero-shot recognition.

C. Incremental Mixture-of-Experts Adapters

We leverage MoE [30] to build an expansible architecture to alleviate the "catastrophic forgetting" issue in the continual learning of CLIP. The MoE is composed of several experts $\{\mathcal{E}_i\}_{i=1}^{N_E}$ and routers, where N_E is the number of pre-defined experts. For current task $\mathcal{T}^t$, only a task-dependent router $\mathcal{R}^t, t \in [1, T]$ is added to the system.

Adapters as Experts: MoE in vision-language models usually incorporate the experts inside networks, which can be MLPs or attention heads [30], [75], [76]. However, inserting the MoE inside VLM might bring significant computational burdens due to the full-parameter tuning. Inspired by the adapter-based methods [21], [35], we use the effective adapter LoRA [25], which works by decoupling the original heavy and frozen parameters into low-rank trainable space, as the experts to speed up continual learning with CLIP.

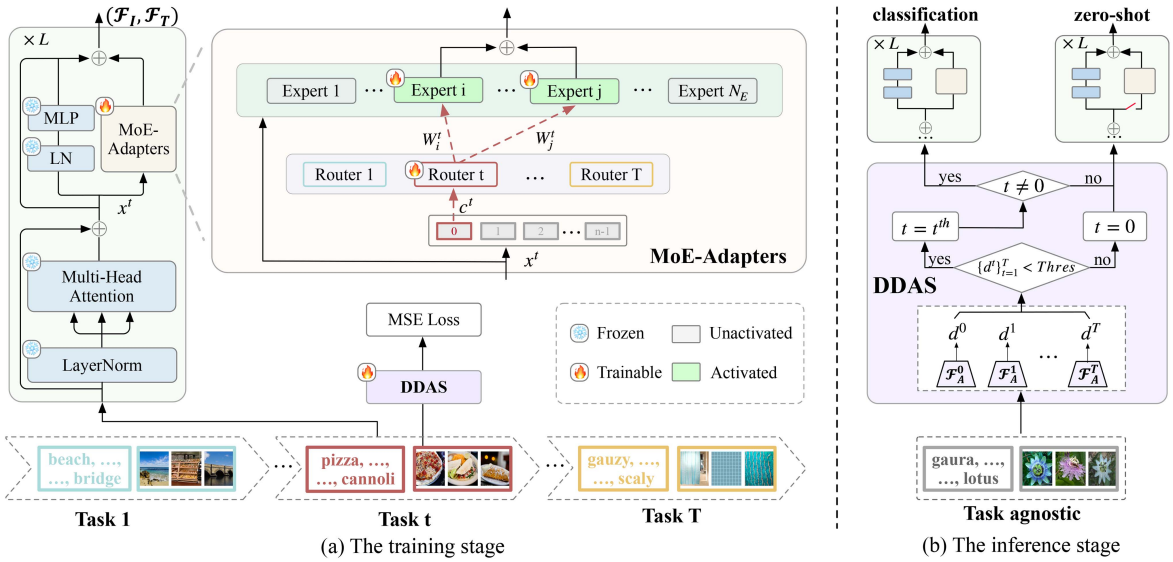


Fig. 2. Overall framework of MoE-Adapters. (a) At the training stage, CLIP's image and text encoders $(\mathcal{F}_I, \mathcal{F}_T)$ take input samples from **Task** t. In each of transformer blocks, there is a MoE-Adapters, whose input is the tokens $\boldsymbol{x}^t$. The router takes the task-specific [CLS] token $\boldsymbol{c}^t$ as input and produces experts' weights W_i^t and W_j^t to combine the expert's output. DDAS is trained using only images via the MSE loss defined by (3). (b) At the inference stage, the proposed DDAS determines the data distribution by comparing the distribution $\{d^t\}_{t=1}^T$ in each Auto-Encoder of the **task-agnostic** images. It can automatically assign the testing data into MoE-Adapters or original CLIP to predict with either seen or unseen data.

Our MoE-Adapters is implemented in all the Transformer blocks of the parallel encoders $(\mathcal{F}_I, \mathcal{F}_T)$, as shown in Fig. 2. To be more specific, in each Transformer block, the feature tokens $x^t \in \mathbb{R}^{n \times d}$ after the multi-head attention output are passed to all the experts in the MoE. Then, the task-specific router $\mathcal{R}^t$ is applied to fuse the experts' outputs via gated summation.

Incremental Mixture of Experts: In our MoE framework, task-specific routers $\mathcal{R}^t$ determine the activation of experts $\mathcal{E}_i$ to produce outcomes tailored to each task t. The combined output for a task, $\boldsymbol{y}^t$, is computed as:

$$\boldsymbol{y}^t = \sum_{i=1}^{N_E} W_i^t \mathcal{E}_i(\boldsymbol{x}^t), \tag{1}$$

where $W^t = \{W_i^t\}_{i=1}^{N_E}$ represents the gating weights assigned by $\mathcal{R}^t$, dictating each expert's contribution. $\boldsymbol{x}^t$ denotes the tokens processed for task t, and $\boldsymbol{y}^t$ is the corresponding output from the MoE-Adapters, matching the shape of $\boldsymbol{x}^t$. We refine the MoE-Adapters for continual learning with two key modifications. We utilize the initial token, known as the [CLS] token ($\boldsymbol{c}^t \in \mathbb{R}^{1 \times d}$), to enhance processing efficiency. The gating weights are computed as:

$$W^t = Softmax(Topk(\mathcal{R}^t(\boldsymbol{c}^t))), \tag{2}$$

where $\mathcal{R}^t$ projects $\boldsymbol{c}^t$ to a 1-D vector indicating each expert's likelihood of activation. The $Topk(\cdot)$ function selects the k most relevant experts, while setting the rest to be $-\infty$. The $Softmax(\cdot)$ function normalizes these weights to emphasize the selected experts' contribution.

Training MoE-Adapters: We train the MoE-Adapters through simple back-propagation, orchestrated by an incremental activate-freeze strategy. The objective is to augment experts with intra-task knowledge and inter-task collaboration. Specifically, after training on an older task, we count the distribution of its router's outputs. The Top-k most activated experts are then kept frozen during subsequent task training to preserve task-specific knowledge. In this manner, when faced with a new task, the respective router can access frozen experts for leveraging shareable knowledge from historical tasks, and optimize unfrozen experts to acquire specific information for the new task. As illustrated in Fig. 3, during training, the router can activate (a) only the untapped experts, (b) both untapped and previously learned experts, and (c) only the learned experts from previous tasks. As a result, this strategy allows experts to consolidate their knowledge collaboratively, resembling the human brain's mechanism of reinforcing and connecting new information with existing memories.

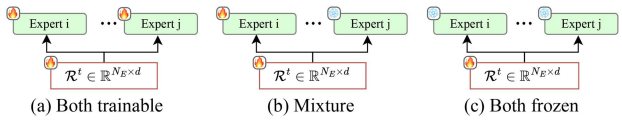


Fig. 3. The three distinct combinations among activated experts: (a) both trained, (b) trainable and frozen, (c) both experts are frozen, and only the router is trainable.

D. Distribution Discriminative Auto-Selector

The task-specific nature of the routers in our MoE-Adapters necessitates manual task identity to activate the appropriate router. Such manual intervention is not aligned with the automated and practical nature of task incremental learning and class incremental learning, and restricts the inherent zero-shot generalization capability of the CLIP model. To address this limitation, we develop the Distribution Discriminative Auto-Selector (DDAS), which automatically selects the proper router

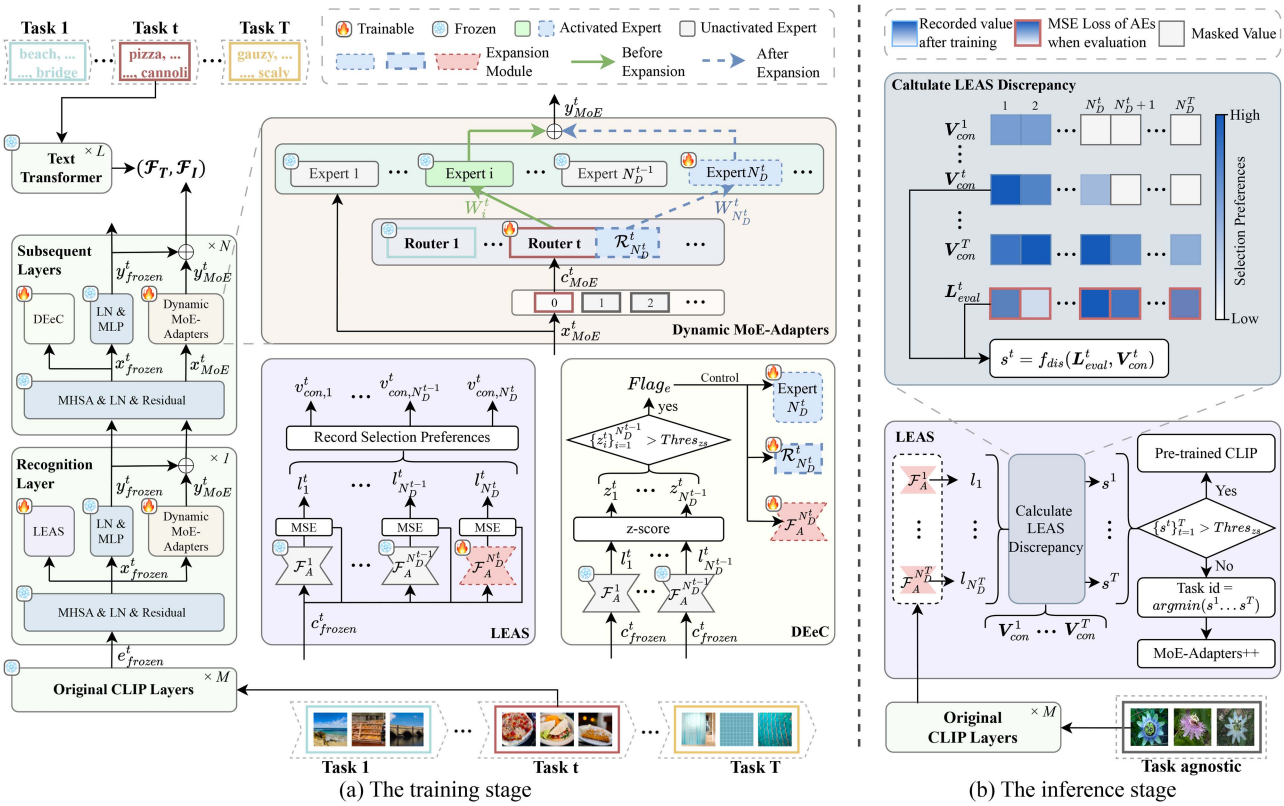


Fig. 4. Overall framework of MoE-Adapters++. (a) At the training stage for the **Task *t***, in Dynamic MoE-Adapters, the router takes the task-specific [CLS] token c^t_{MoE} as input and produces experts' weights W^t_i and $W^t_{N^t_D}$ to combine the expert's output. The LEAS and DEeC take c^t_{frozen} from frozen CLIP layers as input, both of them are trained by MSE loss defined by (5). DEeC monitors the z-score defined by (6) to control the expansion signal $Flag_e$ of for a dynamic expert set. And the LEAS records the selection preference $v^t_{con,i}$ of each Auto-Encoder (AE), ultimately forming the selection preference vector $\boldsymbol{V}^t_{con}$. (b) At the inference stage, the proposed LEAS determines the data distribution by comparing the discrepancy $\{s^t\}^T_{t=1}$ between MSE loss $\boldsymbol{L}^t_{eval}$ of the test **task-agnostic** image embedding and $\boldsymbol{V}^t_{con}$ of each training task.

with the task context inferred by analyzing the variation in the distribution of input. DDAS extends the incremental MoE framework by introducing a series of task-specific Auto-Encoders (AEs), $\{\mathcal{F}^t_A\}^T_{t=1}$, which are trained to independently capture the distribution characteristics for the tasks $\{\mathcal{T}^t\}^T_{t=1}$. The loss function employed for training the Auto-Encoders is the Mean Squared Error (MSE) loss, defined as:

$$d^t = ||\boldsymbol{f}^t_i - \boldsymbol{f}^t_o||^2, \tag{3}$$

where $\boldsymbol{f}^t_i$ is the intermediate feature extracted from the input image, and $\boldsymbol{f}^t_o = \mathcal{F}^t_A(\boldsymbol{f}^t_i)$ is the reconstructed feature representation by the Auto-Encoder of task t. Since the Auto-Encoder $\mathcal{F}^t_A$ is individually learned on the data of task t, the resulting reconstruction score d^t reflects the likelihood that an input image pertains to the task, with a lower score suggesting a higher probability.

Moreover, to preserve CLIP's zero-shot transfer ability during continual learning, we include an additional Auto-Encoder, $\mathcal{F}^0_A$, trained on a reference dataset to identify out-of-distribution data. Upon completion of the learning process, DDAS computes a set of distribution scores $\{d^t\}^T_{t=1}$ for each input image. Should all scores surpass a specific threshold, $Thres$, the system classifies the input as "unseen data" and redirects it to the frozen CLIP for zero-shot transfer. Conversely, inputs below this threshold are routed to the corresponding router with the lowest distribution score, ensuring efficient and accurate task identification.

IV. MoE-Adapters++

A. Framework Overview

We address the parameter redundancy and training complexity in MoE-Adapters by extending MoE-Adapters++ with Dynamic MoE-Adapters and a Latent Embedding Auto-Selector (LEAS). The overall framework of MoE-Adapters++ is illustrated in Fig. 4. Specifically, we implement dynamic MoE on a frozen CLIP model, allowing for the incremental incorporation of both task-specific routers and experts. To eliminate reliance on a static expert set, we design a Dynamic Expert-expansion Controller (DEeC) to determine the inclusion of a new expert. Additionally, we introduce a latent embedding auto-selector, which utilizes visual embeddings from CLIP for precise distribution selection, enabling the unification of the entire architecture.

B. Dynamic Mixture-of-Experts Adapters

We divide the L image encoder blocks into M Frozen Layers, 1 Recognition Layer, and N Subsequent Layers, such that $L = M + N + 1$, and implement Dynamic MoE-Adapters in

the Recognition and Subsequent Layers. Specifically, for current task $\mathcal{T}^t$, the dynamic expert set, composed of N_D^t experts, can be represented as $\{\mathcal{E}_i\}_{i=1}^{N_D^t}$. Then, the task-specific router $\mathcal{R}^t$ integrates the experts' outputs via:

$$\boldsymbol{y}_{MoE}^t = \sum_{i=1}^{N_D^t} W_i^t \mathcal{E}_i(\boldsymbol{x}_{MoE}^t), \tag{4}$$

where $W^t = \{W_i^t\}_{i=1}^{N_D^t}$ represents the gating weights assigned by $\mathcal{R}^t$, which dictates each expert's contribution. $\boldsymbol{x}_{MoE}^t$ and $\boldsymbol{y}_{MoE}^t$ denote the input and combined output embeddings of the Dynamic MoE-Adapters, respectively. Similar to (2), the `[CLS]` token $\boldsymbol{c}_{MoE}^t$ of the inputs $\boldsymbol{x}_{MoE}^t$ is utilized to calculate the weights W^t of the $Top\text{-}k$ experts.

Training Strategy: We train the Dynamic MoE-Adapters using the same loss functions as MoE-Adapters, while also introducing an expert expansion strategy and an improved frozen strategy. Specifically, the expansion strategy allows for the construction of a dynamic expert set, where experts are gradually added throughout the continual learning process. For the first task $\mathcal{T}^1$, we initialize a trainable router $\mathcal{R}^1$ and k experts that will be frozen in the subsequent tasks. Given a new task, the Dynamic MoE-Adapters can adaptively decide whether to add a new expert based on an expansion signal $Flag_e$ released by the Dynamic Expert-expansion Controller. In addition, we simplify the original activate-freeze strategy by training only the newly added router and expert, eliminating the need to record the selection frequency for each expert, thereby streamlining the whole training process.

C. Dynamic Expert-Expansion Controller

We propose a Dynamic Expert-expansion Controller module (DEeC), which utilizes an expansion signal $Flag_e$ to determine the necessity of expert expansion for the current task. As shown in Fig. 4, the DEeC module is implemented across N Subsequent Layers, which adopts a series of Auto-Encoders to estimate the distribution variations between the current and previous tasks. Specifically, for task $\mathcal{T}^t$, it takes the input as the `[CLS]` token $\boldsymbol{c}_{frozen}^t$ from the MHSA's output $\boldsymbol{x}_{frozen}^t$ in the frozen Transformer blocks. We train each Auto-Encoder using MSE loss, which can be defined as:

$$l_i^t = ||\boldsymbol{c}_{frozen}^t - \boldsymbol{f}_o^t||^2, \tag{5}$$

where the $\boldsymbol{f}_o^t = \mathcal{F}_A^i(\boldsymbol{c}_{frozen}^t)$ is the reconstructed latent embedding representation of Auto-Encoder i. For each task $(t > 1)$, the expert set is not expanded at the early training stage with N_D^{t-1} Auto-Encoders represented as $\{\mathcal{F}_A^i\}_{i=1}^{N_D^{t-1}}$, as shown in Fig. 5(a). We keep the previous experts frozen and only train task-specific router $\mathcal{R}^t$ to leverage historical knowledge without adding parameters. Inspired by SEMA [77], we compute the z-score of each reconstruction loss for the current training batch in each existing AE:

$$z_i^t = (l_i^t - \mu_i^t)/\sigma_i^t, \tag{6}$$

where μ_i^t and σ_i^t are the mean and the standard deviation of the reconstruction loss set from all the training batches for the current task $\mathcal{T}^t$. If all z_i^t for $i \in [1, N_D^{t-1}]$ are larger than an expansion threshold $Thres_e$, it indicates that the existing frozen experts can no longer adapt to the current task. Leveraging the property of (6), we empirically select the standard deviation of z-score ($Thres_e = 1.0$) as the guidance expansion threshold. As shown in Fig. 5(b), the expansion signal is triggered (*i.e.,* $Flag_e = 1$), leading to an increase in the number of experts to $N_D^t = N_D^{t-1} + 1$. Accordingly, the router is added with a new column of parameters to match the expanded expert set. Then, we use the expanded MoE to calculate the output of $\boldsymbol{x}_{MoE}^t$ as in (4), while keeping both the router and new expert trainable. Meanwhile, a new Auto-Encoder is added in DEeC and trained using (5) until the end of task $\mathcal{T}^t$, which will be used for expansion decisions in the next tasks.

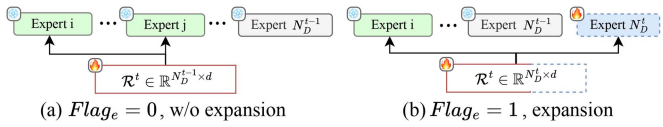


Fig. 5. Different trained combinations of expert expansion: (a) without expansion, only the router is trainable, (b) with expansion, where both the new expert and the router with extended dimensions are trainable.

Notably, the difference from SEMA [77] is that we keep the task-specific router trainable throughout the process, utilizing an enhanced frozen strategy for training and extending to a broader range of incremental tasks.

D. Latent Embedding Auto-Selector

We replace the separate DDAS with a Latent Embedding Auto-Selector (LEAS) embedded in CLIP's Recognition Layer, which infers task context from frozen CLIP embeddings and sends the task ID to all Dynamic MoE-Adapters. As shown in Fig. 4, LEAS contains a series of trainable, expert-specific Auto-Encoders $\{\mathcal{F}_A^i\}_{i=1}^{N_D^t}$, which are unshared with the Auto-Encoders in DEeC. During training, we record the selection preference of each Auto-Encoder for each task. During inference, we infer the task ID by comparing reconstruction loss discrepancies of test samples, then use it to activate the corresponding routers and experts in the Dynamic MoE-Adapters.

Recording Selection Preferences: Similarly, given the class token $\boldsymbol{c}_{frozen}^t$ of the frozen CLIP as input, we first calculate a reconstruction loss l_i^t for each Auto-Encoder using (5). Then, we record the convergence values of these reconstruction losses as the selection preferences, which can be defined as:

$$v_{con,i}^t = f_{con}(l_i^t), \tag{7}$$

where f_{con} represents the sliding window method for calculating the convergence value, and we utilize the mean value of the last window as the final result. The $v_{con,i}^t$ of all Auto-Encoders form the selection preference vector $\boldsymbol{V}_{con}^t = [v_{con,1}^t, v_{con,2}^t, \ldots, v_{con,N_D^t}^t]$. In the Recognition Layer, we remove the control of DEeC, allowing experts and Auto-Encoders to incrementally expand with each new task to enhance the discrimination of task distributions.

Inferring Task ID: At testing, LEAS infers the distribution of the task-agnostic samples by comparing their reconstruction losses across all Auto-Encoders with recorded selection preferences of each task. Specifically, the reconstruction loss vector $\boldsymbol{L}_{eval}^{T} = [l_1, l_2, \ldots, l_{N_D^T}]$ of the test samples is calculated by (5). Then, we calculate the discrepancy s^t between $\boldsymbol{L}_{eval}^{t}$ and $\boldsymbol{V}_{con}^{t}$ by:

$$s^t = \left\| \frac{\boldsymbol{L}_{eval}^{t} - \boldsymbol{V}_{con}^{t}}{\max(\boldsymbol{L}_{eval}^{t}, \boldsymbol{V}_{con}^{t})} \cdot \boldsymbol{\alpha} \right\|_2 / \beta, \tag{8}$$

where $\|\cdot\|_2$ and $\max(\cdot,\cdot)$ represent the L2 norm and element-wise maximum of two vectors, $\boldsymbol{\alpha}$ is a weighted vector for different Auto-Encoders, β is the scaling factor used to balance the effects of different number of Auto-Encoders corresponding to each task. $\boldsymbol{L}_{eval}^{t} = [l_1, l_2, \ldots, l_{N_D^t}]$ contains N_D^t elements to align with $\boldsymbol{V}_{con}^{t}$. The s^t reflects the relevance of the task-agnostic images to task t, with a lower s^t indicating stronger relevance. Finally, we set a zero-shot threshold $Thres_{zs}$ to determine whether the samples are seen or unseen and use $argmin(s^1, \ldots, s^T)$ as the task id.

When calculating the discrepancy s^t between the test samples and each task t, the item with the lower selection preferences should be highlighted to maximize the correct Auto-Encoder's perception for task-agnostic samples. Therefore, we use $1/\boldsymbol{V}_{con}^{t}$ as the weighted vector $\boldsymbol{\alpha}$. To normalize the vectors $\boldsymbol{L}_{eval}^{t}$ and $\boldsymbol{V}_{con}^{t}$ with different lengths for each task, we choose $\sqrt{N_D^t}$ as the scaling factor β, which is positively correlated to the L2 norm.

The selection of $Thres_{zs}$: We leverage training discrepancy computation to estimate test discrepancy upper bounds. This allows $Thres_{zs}$ to be automatically derived during training, whose process consists of three key steps: ❶ Monitor the peak reconstruction loss of each task-specific Auto-Encoder during training, which reflects the worst-case reconstruction performance. Both the peak loss $\boldsymbol{L}_{\max_{training}}^{t}$ and final converged loss $\boldsymbol{V}_{con}^{t}$ are recorded as reference indicators. ❷ Compute Current Predicted Discrepancy $s_{training}^{t}$ via (9) between stable loss and peak to bound test discrepancy:

$$s_{training}^{t} = \left\| \frac{\boldsymbol{L}_{\max_{training}}^{t} - \boldsymbol{V}_{con}^{t}}{\max(\boldsymbol{L}_{\max_{training}}^{t}, \boldsymbol{V}_{con}^{t})} \cdot \boldsymbol{\alpha} \right\|_2 / \beta, \tag{9}$$

where $\boldsymbol{\alpha}$ and β are set to $1/\boldsymbol{L}_{\max_{training}}^{t}$ and $\sqrt{n} * n_{trainable}$. Here, n is the total number of experts, and $n_{\text{trainable}}$ denotes the number of trainable experts in the recognition layer, introduced to balance the impact of varying trainable parameter counts on training stability. ❸ After training on each task, the maximum value among all $s_{training}^{t}$ is selected as $s_{training}^{\max}$ and used as a reference for setting the threshold.

V. Experiments

A. Experimental Setting

Datasets: We evaluate our method across three tasks: Multi-domain TIL (MTIL) [12], DomainNet TIL [98], and CIL. For MTIL, we follow the two-order training protocol proposed in [12]. For DomainNet TIL, we follow the domain order: ClipArt, InfoGraph, Painting, QuickDraw, Real, and Sketch. For CIL, we follow [56] to conduct experiments on CIFAR100 [56], TinyImageNet [34] and ImageNet-1k [95]. The 100 classes of CIFAR100 are divided into $\{10, 20, 50\}$ subsets, the 100 classes from TinyImageNet are divided into $\{5, 10, 20\}$ subsets, and the 1000 classes from ImageNet-1k are divided into 100 subsets.

Metrics: To evaluate our method on the MTIL, we utilize metrics proposed by [12], namely "Transfer", "Average", and "Last". In CIL, following [56], we calculate the average accuracy over all subsets ("Average") and specifically for the last subset ("Last").

Implementation Details: We use the CLIP model with ViT-B/16 [99] as our backbone for all the experiments. The router is a single MLP that mixes the experts with Top-2 gating scores. In DDAS, the reference data is TinyImageNet [34], and the threshold is 0.065 and 0.06 for full-shot and few-shot. The Auto-Encoder is built upon a pretrained AlexNet [100] with MLP and a non-linear layer. We use AdamW [101] optimizer and a label smoothing [102] technique for a better result. For TIL, we train 1 k iterations on full-shot and 500 iterations on few-shot for each task. For DDAS, we train 1 k and 300 iterations for reference datasets and incremental tasks, respectively. The learning rates are searched within $[10^{-3}, 10^{-4}]$. Label smoothing can substitute the regularization of weight decay and achieve better performance. The label smoothing strength is searched between $\{0.1, 0.2\}$. For CIL, we set weight decay as 0 and label smoothing as 0.0.

In MoE-Adapters++, we keep the first six layers of the image encoder frozen, designate the seventh layer as the Recognition Layer, and configure layers 8 to 12 as Subsequent Layers. Each Auto-Encoder is composed of two MLPs and a non-linear layer. We empirically set the threshold $Thres_e$ of DEeC as 1.0, and the learning rates are searched within $[10^{-2}, 10^{-4}]$. During the inference stage, the zero-shot thresholds $Thres_{zs}$ for LEAS are searched within $[2.0, 3.0]$, based on the reference value obtained by Eq.9. Other settings remain consistent with MoE-Adapters [28].

B. Comparison With State-of-The-Art Methods

Multi-domain Task Incremental Learning: Table I showcases a comparison between our proposed method and alternative approaches in the MTIL task. Our approach utilizes the pre-defined Order-I from [12], where datasets are trained and tested sequentially in a left-to-right order. The Table I displays the outcomes of applying CLIP independently on each task through zero-shot inference, full parameter fine-tuning, and parameter-efficient fine-tuning. The "zero-shot" represents the optimal results of CLIP's zero-shot transfer on each task, while the other two rows indicate the highest possible outcomes achieved by fine-tuning CLIP in each respective task. The extended method outperforms the second-best approach on most tasks, leading to overall improvement of by 0.3%, 1.2%, and 1.1% in terms of "Transfer", "Average" and "Last", respectively. Furthermore, our model achieves less degradation than DIKI when compared with the upper bound methods, demonstrating superior performance in anti-forgetting and zero-shot transfer.

TABLE I
COMPARISON WITH STATE-OF-THE-ART METHODS ON MTIL BENCHMARK IN TERMS OF "TRANSFER", "AVERAGE", AND "LAST" SCORES (%)

	Method	Aircraft [78]	Caltech101 [79]	CIFAR100 [80]	DTD [81]	EuroSAT [82]	Flowers [83]	Food [84]	MNIST [85]	OxfordPet [86]	Cars [87]	SUN397 [88]	*Average*
CLIP	Zero-shot	24.3	88.4	68.2	44.6	54.9	71.0	88.5	59.4	89.0	64.7	65.2	65.3
	Full Fine-tune	62.0	95.1	89.6	79.5	98.9	97.5	92.7	99.6	94.7	89.6	81.8	89.2
	Fine-tune Adapter	56.8	92.6	89.4	79.0	98.4	97.0	92.9	99.2	94.1	89.1	82.7	88.3
Transfer	Continual-FT		67.1	46.0	32.1	35.6	35.0	57.7	44.1	60.8	20.5	46.6	44.6
	LwF [20]		74.5	56.9	39.1	51.1	52.6	72.8	60.6	75.1	30.3	55.9	58.9
	iCaRL [47]		56.6	44.6	32.7	39.3	46.6	68.0	46.0	77.4	31.9	60.5	50.4
	LwF-VR [89]		77.1	61.0	40.5	45.3	54.4	74.6	47.9	76.7	36.3	58.6	57.2
	WiSE-FT [90]		73.5	55.6	35.6	41.5	47.0	68.3	53.9	69.3	26.8	51.9	52.3
	ZSCL [12]		86.0	67.4	**45.4**	50.4	69.1	87.6	61.8	86.8	60.1	**66.8**	68.1
	L2P [14]		65.6	50.9	30.4	41.4	49.3	71.8	36.3	77.5	55.3	53.4	53.2
	DualPrompt [15]		56.7	51.4	28.7	33.7	45.6	70.9	59.5	77.7	49.5	50.4	52.4
	S-Prompts [13]		67.3	49.4	26.4	39.7	47.1	70.2	34.3	78.9	56.7	52.2	52.2
	DIKI [16]		92.9	69.0	43.2	48.2	67.4	85.2	**63.0**	87.9	63.8	66.2	68.7
	MoE-Adapters [28]		**87.9**	**68.2**	44.4	49.9	70.7	88.7	59.7	**89.1**	**64.5**	65.5	68.9(+0.2)
	MoE-Adapters++		**87.9**	**68.2**	45.1	**54.6**	**71.2**	**88.8**	59.5	88.6	63.3	63.1	**69.0**(+0.3)
Average	Continual-FT	25.5	81.5	59.1	53.2	64.7	51.8	63.2	64.3	69.7	31.8	49.7	55.9
	LwF [20]	36.3	86.9	72.0	59.0	73.7	60.0	73.6	74.8	80.0	37.3	58.1	64.7
	iCaRL [47]	35.5	89.2	72.2	60.6	68.8	70.0	78.2	62.3	81.8	41.2	62.5	65.7
	LwF-VR [89]	29.6	87.7	74.4	59.5	72.4	63.6	77.0	66.7	81.2	43.7	60.7	65.1
	WiSE-FT [90]	26.7	86.5	64.3	57.1	65.7	58.7	71.1	70.5	75.8	36.9	54.6	60.7
	ZSCL [12]	45.1	92.0	80.1	64.3	79.5	81.6	89.6	75.2	88.9	64.7	**68.0**	75.4
	L2P [14]	38.0	85.2	78.2	61.3	72.9	74.9	79.7	59.1	82.0	59.7	55.4	67.9
	DualPrompt [15]	37.8	84.3	78.6	60.1	71.1	73.2	79.1	73.9	82.3	55.1	52.8	68.0
	S-Prompts [13]	37.5	92.5	77.5	58.2	76.4	74.1	78.8	57.9	83.0	60.8	54.4	68.3
	DIKI [16]	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 [28]	50.2	91.9	**83.1**	69.4	78.9	**84.0**	89.1	73.7	89.3	**67.7**	66.9	76.7(+0.4)
	MoE-Adapters++	**55.8**	94.6	81.4	**70.4**	**82.4**	83.6	**90.0**	73.5	**90.1**	66.0	64.2	**77.5**(+1.2)
Last	Continual-FT	31.0	89.3	65.8	67.3	88.9	71.1	85.6	**99.6**	92.9	77.3	81.1	77.3
	LwF [20]	26.3	87.5	71.9	66.6	79.9	66.9	83.8	**99.6**	92.1	66.1	80.4	74.6
	iCaRL [47]	35.8	93.0	77.0	70.2	83.3	88.5	90.4	86.7	93.2	81.2	**81.9**	80.1
	LwF-VR [89]	20.5	89.8	72.3	67.6	85.5	73.8	85.7	**99.6**	93.1	73.3	80.9	76.6
	WiSE-FT [90]	27.2	90.8	68.0	68.9	86.9	74.0	87.6	**99.6**	92.6	77.8	81.3	77.7
	ZSCL [12]	40.6	92.2	81.3	70.5	94.8	90.5	**91.9**	98.7	93.9	**85.3**	80.2	83.6
	L2P [14]	38.0	87.1	84.2	72.9	86.0	96.1	89.2	99.0	94.1	79.6	76.0	82.0
	DualPrompt [15]	37.8	87.1	84.6	71.8	89.2	96.3	89.1	99.1	**94.5**	79.9	76.5	82.3
	S-Prompts [13]	37.5	95.1	83.7	70.2	97.5	96.5	89.0	99.1	94.0	79.5	75.8	83.4
	DIKI [16]	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 [28]	49.8	92.2	86.1	78.1	95.7	94.3	89.5	98.1	89.9	81.6	80.0	85.0(-0.1)
	MoE-Adapters++	**55.8**	95.2	84.3	**79.9**	**98.2**	96.3	91.4	98.1	94.2	78.3	76.2	**86.2**(+1.1)

We label the best and second-best methods with bold and underline styles. The top block indicates the upper-bound solutions to adapt the CLIP on each task.

Few-shot Multi-Domain Task Incremental Learning: We further evaluate on few-shot MTIL (Table II), where the extended version surpasses the second-best method by 4.0%, 7.3%, and 4.4% in "Transfer", "Average", and "Last". These results validate the effectiveness of our incremental and dynamic MoE-Adapters in mitigating forgetting with limited samples, while DDAS and LEAS efficiently learn data distribution discrimination under few-shot settings.

Class Incremental Learning: We conduct experiments on class incremental learning (CIL). Unlike MTIL, the task ID of the input image is unknown in CIL. Notably, our MoE-Adapters employ a single router with two experts for all subsets, which is sufficient for single-domain incremental learning, eliminating the need for additional dynamic strategies or module selection. The comparison results between our method and state-of-the-art approaches on CIFAR100 and TinyImageNet are shown in Tables III and IV, respectively. The proposed method consistently outperforms the other competitors, including dynamic expansion and CLIP-based approaches. We further conduct experiments on the ImageNet-1k [95] dataset, which contains over 1.2 million images across 1,000 categories. As shown in Table V, our method also achieves the best performance on both top-1 and top-5 accuracy.

More Comparison Results on MTIL: Tables VII and VIII show the results of the full-shot and few-shot MTIL benchmarks in Order-II. As we can see, the extended method MoE-Adapters++ performs favorably against the original method and other state-of-the-art approaches in terms of three metrics in both settings. Notably, the zero-shot transfer ability of the proposed method closely reaches the upper bound of the pre-trained CLIP. It demonstrates that our improved LEAS effectively identifies whether data is seen or unseen.

Generalization to Other Datasets and Backbones: As shown in Table VI, we conduct cross-domain incremental learning (DomainNet Dataset [98]) on ViT-B to highlight the versatility of our method under the dynamic and distribution-shifting dataset. DomainNet is a large-scale benchmark comprising approximately 0.6 million images across 345 categories and six diverse domains. The results show that our extended method further surpasses the SOTA (0.39%, 1.23%, 2.37%) across all these metrics.

In addition, to further assess the generalizability of our approach across diverse model architectures and scales, we conduct further experiments with SigLIP and ViT-L on the MTIL and domain TIL benchmarks. As shown in Table IX(a) & (b), our method achieves state-of-the-art performance with ViT-L and SigLIP on MTIL, improving over ZSCL by 1.03% to 5.82% across different metrics, demonstrating that our method generalizes effectively to larger and more powerful architectures. As shown in Table IX(c) & (d), our method achieves state-of-the-art

TABLE II
COMPARISON WITH STATE-OF-THE-ART METHODS ON FEW-SHOT MTIL BENCHMARK IN TERMS OF "TRANSFER", "AVERAGE", AND "LAST" SCORES (%)

	Method	Aircraft [78]	Caltech101 [79]	CIFAR100 [80]	DTD [81]	EuroSAT [82]	Flowers [83]	Food [84]	MNIST [85]	OxfordPet [86]	Cars [87]	SUN397 [88]	*Average*
CLIP	Zero-shot	24.3	88.4	68.2	44.6	54.9	71.0	88.5	59.4	89.0	64.7	65.2	65.3
	5-shot Full Fine-tune	30.6	93.5	76.8	65.1	91.7	92.9	83.3	96.6	84.9	65.4	71.3	77.5
	5-shot Fine-tune Adapter	29.7	90.0	75.3	63.9	81.1	94.2	87.8	90.4	89.0	68.2	72.5	76.6
Transfer	Continual-FT		72.8	53.0	36.4	35.4	43.3	68.4	47.4	72.6	30.0	52.7	51.2
	LwF [20]		72.1	49.2	35.9	44.5	41.1	66.6	50.5	69.0	19.0	51.7	50.0
	LwF-VR [89]		82.2	62.5	40.1	40.1	56.3	80.0	60.9	77.6	40.5	60.8	60.1
	WiSE-FT [90]		77.6	60.0	41.3	39.4	53.0	76.6	58.1	75.5	37.3	58.2	57.7
	ZSCL [12]		84.0	68.1	44.8	46.8	63.6	84.9	61.4	81.4	55.5	62.2	65.3
	L2P* [14]		66.7	54.3	30.6	-	47.3	71.5	-	-	54.6	52.4	-
	DualPrompt* [15]		78.8	64.4	32.0	-	51.7	77.5	-	-	49.4	51.3	-
	S-Prompts* [13]		70.3	52.7	31.5	-	54.8	74.0	-	-	55.4	50.0	-
	DIKI* [16]		92.7	68.8	44.1	-	70.0	86.2	-	-	65.1	65.5	-
	MoE-Adapters [28]		**87.9**	**68.2**	44.1	48.1	64.7	**88.8**	**69.0**	**89.1**	**64.5**	**65.1**	68.9(+3.6)
	MoE-Adapters++		**87.9**	**68.2**	**45.1**	**54.6**	**71.3**	88.7	59.5	**89.1**	63.9	64.9	**69.3**(+4.0)
Average	Continual-FT	28.1	86.4	59.1	52.8	55.8	62.0	70.2	64.7	75.5	35.0	54.0	58.5
	LwF [20]	23.5	77.4	43.5	41.7	43.5	52.2	54.6	63.4	68.0	21.3	52.6	49.2
	LwF-VR [89]	24.9	89.1	64.2	53.4	54.3	70.8	79.2	66.5	79.2	44.1	61.6	62.5
	WiSE-FT [90]	32.0	87.7	61.0	55.8	68.1	69.3	76.8	71.5	77.6	42.0	59.3	63.7
	ZSCL [12]	28.2	88.6	66.5	53.5	56.3	73.4	83.1	56.4	82.4	57.5	62.9	64.4
	L2P* [14]	30.2	84.5	70.1	51.9	-	69.6	77.1	-	-	60.0	55.2	-
	DualPrompt* [15]	36.5	89.5	72.5	52.7	-	72.3	80.8	-	-	56.1	54.2	-
	S-Prompts* [13]	30.6	86.8	70.0	51.7	-	74.3	78.5	-	-	60.7	53.0	-
	DIKI* [16]	41.3	95.3	76.5	58.5	-	82.2	86.4	-	-	68.2	66.6	-
	MoE-Adapters [28]	30.0	89.6	**73.9**	58.7	69.3	79.3	88.1	**76.5**	**89.1**	**65.3**	**65.8**	71.4(+7.0)
	MoE-Adapters++	**32.7**	**93.9**	70.8	**59.0**	**73.9**	**80.1**	**88.3**	72.3	89.0	63.9	65.2	**71.7**(+7.3)
Last	Continual-FT	27.8	86.9	60.1	58.4	56.6	75.7	73.8	93.1	82.5	57.0	66.8	67.1
	LwF [20]	22.1	58.2	17.9	32.1	28.1	66.7	46.0	84.3	64.1	31.5	60.1	46.5
	LwF-VR [89]	22.9	89.8	59.3	57.1	57.6	79.2	78.3	77.7	83.6	60.1	69.8	66.9
	WiSE-FT [90]	30.8	88.9	59.6	60.3	80.9	81.7	77.1	**94.9**	83.2	62.8	70.0	71.9
	ZSCL [12]	26.8	88.5	63.7	55.7	60.2	82.1	82.6	58.6	85.9	66.7	70.4	67.4
	L2P* [14]	30.2	87.1	75.4	64.7	-	91.9	86.4	-	-	76.1	74.7	-
	DualPrompt* [15]	36.5	91.0	75.1	65.1	-	92.9	86.2	-	-	76.2	74.2	-
	S-Prompts* [13]	30.6	89.2	75.8	63.8	-	93.9	86.2	-	-	76.7	73.9	-
	DIKI* [16]	41.3	95.6	79.0	67.3	-	94.4	86.8	-	-	77.6	74.4	-
	MoE-Adapters [28]	30.1	89.3	**74.9**	64.0	82.3	**89.4**	87.1	89.0	**89.1**	**69.5**	**72.5**	76.1(+4.2)
	MoE-Adapters++	**32.7**	**94.5**	71.3	**65.2**	**84.9**	87.7	**87.8**	94.8	89.0	63.3	68.0	**76.3**(+4.4)

Ours converges in 500 iterations on few-shot. The top block indicates the upper-bound solutions to adapt the CLIP on each task. "*" means the implementation from DIKI [16], using the 16-shot setup in 8 tasks of MTIL, other setting is 5-shot in all 11 tasks of MTIL.

TABLE III
COMPARISON OF DIFFERENT METHODS ON CIFAR100 IN CIL

Method	10 step		20 step		50 step	
	Avg.	Last	Avg.	Last	Avg.	Last
UCIR [55]	58.66	43.39	58.17	40.63	56.86	37.09
Bic [91]	68.80	53.54	66.48	47.02	62.09	41.04
PODNet [54]	58.03	41.05	53.97	35.02	51.19	32.99
DER [34]	74.64	64.35	73.98	62.55	72.05	59.76
DyTox+ [56]	74.10	62.34	71.62	57.43	68.90	51.09
DNE [58]	74.86	70.04	-	-	-	-
CLIP Zero-shot	74.47	65.92	75.20	65.74	75.67	65.94
Fine-tune	65.46	53.23	59.69	43.13	39.23	18.89
LwF [20]	65.86	48.04	60.64	40.56	47.69	32.90
iCaRL [47]	79.35	70.97	73.32	64.55	71.28	59.07
LwF-VR [89]	78.81	70.75	74.54	63.54	71.02	59.45
ZSCL [12]	82.15	73.65	80.39	69.58	79.92	67.36
MoE-Adapters(++)	**85.21**	**77.52**	**83.72**	**76.20**	**83.60**	**75.24**

We label the best and second-best methods with bold and underline styles.

TABLE IV
COMPARISON OF DIFFERENT METHODS ON TINYIMAGENET DATASET IN CIL

Method	5 step		10 step		20 step	
	Avg.	Last	Avg.	Last	Avg.	Last
EWC [50]	19.01	6.00	15.82	3.79	12.35	4.73
EEIL [92]	47.17	35.12	45.03	34.64	40.41	29.72
UCIR [55]	50.30	39.42	48.58	37.29	42.84	30.85
MUC [93]	32.23	19.20	26.67	15.33	21.89	10.32
PASS [94]	49.54	41.64	47.19	39.27	42.01	32.93
DyTox [56]	55.58	47.23	52.26	42.79	46.18	36.21
CLIP Zero-shot	69.62	65.30	69.55	65.59	69.49	65.30
Fine-tune	61.54	46.66	57.05	41.54	54.62	44.55
LwF [20]	60.97	48.77	57.60	44.00	54.79	42.26
iCaRL [47]	77.02	70.39	73.48	65.97	69.65	64.68
LwF-VR [89]	77.56	70.89	74.12	67.05	69.94	63.89
ZSCL [12]	80.27	73.57	78.61	71.62	77.18	68.30
MoE-Adapters(++)	**81.12**	**76.81**	**80.23**	**76.35**	**79.96**	**75.77**

We label the best and second-best methods with bold and underline styles.

results across all metrics except the Transfer score with ViT-L. This is due to the improved LEAS accuracy in MoE-Adapters++, which prefers high-confidence original branches and thus limits Transfer performance, especially on domain-consistent yet semantically similar benchmarks like DomainNet.

Computational Cost: We further compare the computational cost of the proposed methods with others. The results in Table X show that MoE-Adapters is superior to the method ZSCL, with a reduction of approximately 60%, 15%, and 60% in terms of training parameters, GPU burdens, and training times, respectively. Further, the extended MoE-Adapters++ demonstrates greater training efficiency compared to the MoE-Adapters [28], with a reduction of approximately 98%, 70%, and 56% . For the inference stage, MoE-Adapters++ achieves a 19% reduction in inference latency compared to the MoE-Adapters. The average inference latency of our extended method is 0.77 seconds per iteration (batch size 64), which is only 0.004 seconds per sample slower than competitive SOTA baselines. We further evaluate

TABLE V
COMPARISON OF DIFFERENT METHODS ON IMAGENET-1K [95] UNDER A CLASS-INCREMENTAL LEARNING SETTING WITH 10 INCREMENTAL STEPS, EACH INTRODUCING 100 NEW CLASSES

Method	top-1		top-5	
	Average	Last	Average	Last
E2E [92]	–	–	72.09	52.29
Simple-DER [66]	66.63	59.24	85.62	80.76
ICaRL [47]	38.40	22.70	63.70	44.00
WA [96]	65.67	55.60	86.60	81.10
DER [34]	68.84	60.16	88.17	82.86
DyTox [56]	73.21	64.56	91.09	87.07
DPFormer [97]	76.13	66.08	92.39	88.19
CLIP Zero-shot	74.86	66.98	94.03	90.81
LwF [20]	79.63	70.12	95.84	94.12
LwF-VR [89]	79.87	70.56	96.51	94.33
ZSCL [12]	80.65	71.53	96.83	94.47
MoE-Adapters(++)	**81.78**	**73.38**	**97.10**	**94.81**

TABLE VI
COMPARISON WITH STATE-OF-THE-ART METHODS ON THE DOMAINNET DATASET [98] USING VIT-B IN TERMS OF "LAST", "AVERAGE", AND "TRANSFER" SCORES (%)

Method	Transfer	Average	Last
Zero-shot	57.86	60.79	60.79
Full Fine-tune	51.25	59.03	56.09
LwF [20]	52.54	63.14	66.13
LwF-VR [89]	53.57	64.03	66.95
WiSE-FT [90]	54.02	64.57	67.95
ZSCL [12]	55.74	65.13	68.73
MoE-Adapters [28]	56.03(+0.29)	66.22(+1.09)	70.50(+1.77)
MoE-Adapters++	**56.13**(+0.39)	**66.36**(+1.23)	**71.10**(+2.37)

The incremental learning follows the domain order: ClipArt, InfoGraph, Painting, QuickDraw, Real, and Sketch.

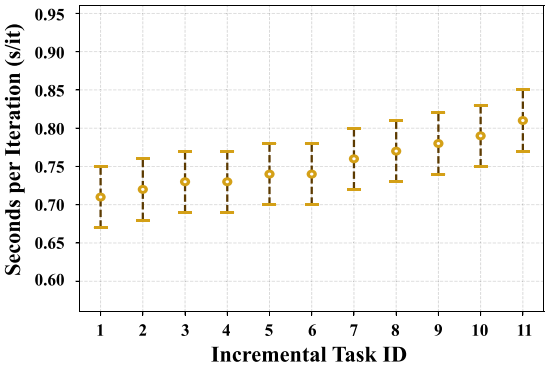


Fig. 6. Inference latency of MoE-Adapters++ as the number of tasks increases, evaluated on the MTIL following Order-I.

the inference latency from adaptively adding routers and Auto-Encoders. As shown in Fig. 6, the per-iteration inference latency increases by only around 0.01 seconds as Auto-Encoders are added over time. Across all 11 tasks, the total increase remains below 0.002 seconds per sample. This level of latency increase can demonstrate the practical feasibility and scalability of our method.

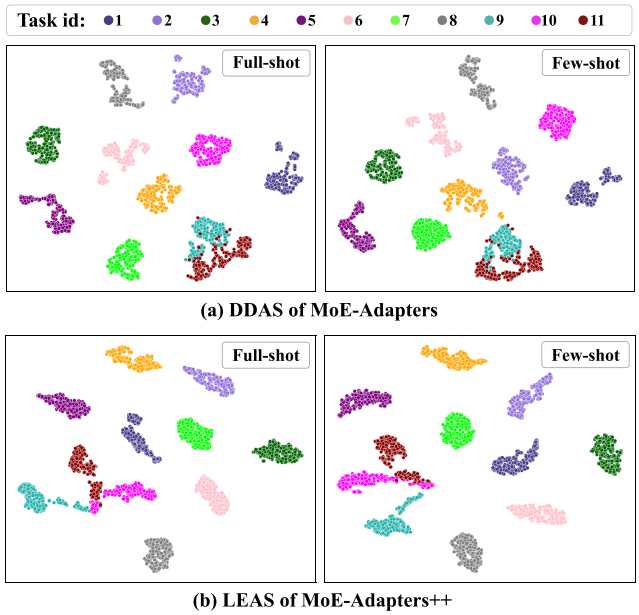


Fig. 7. The figures (a) and (b) show the t-SNE on DDAS's and LEAS's output of each task on MTIL, respectively. The corresponding task from $id = 1 - 11$ matches with the datasets listed from left to right in Table I.

C. Ablation Study

In this section, we mainly analyze the efficacy of the proposed MoE-Adapters and extended MoE-Adapters++. All the experiments are conducted in MTIL setting.

Analysis of MoE-Adapters: We conduct detailed ablation studies on MoE-Adapters (as shown in Table XI). Compared with zero-shot CLIP and the one-adapter fine-tuned version, our method effectively mitigates forgetting while retaining zero-shot transfer on unseen data. In addition, the results show that task-specific routers contribute more to anti-forgetting and transfer ability than simply increasing experts. We further introduce an incremental activate-freeze strategy to enable collaboration between learned and inactive experts, as evidenced by the performance gap between "MoE-Adapters" and "+22E/11R w/o F."

Analysis of DDAS: We employ DDAS with task-specific Auto-Encoders to distinguish inputs from seen or unseen data by learning distribution variations. To validate its effectiveness, we analyze feature-space discrimination using reconstructed features $\boldsymbol{f}_o^t$ after continual learning, as shown in Fig. 7(a). Results show that DDAS effectively captures discriminative distributions for learned tasks in both full-shot and few-shot MTIL. In addition, we conduct ablation experiments on various loss functions for the Auto-Encoders of DDAS, and the results are shown in Table XII. It can be seen that our method achieves the best performance when utilizing the MSE loss.

Analysis of MoE-Adapters++: We conduct comprehensive ablation studies on MoE-Adapters++ settings, as shown in Table XIII. The "Single Router" refers to the use of just one router that continuously expands the parameter dimensions to accommodate the dynamic number of experts. Compared with

TABLE VII
COMPARISON WITH STATE-OF-THE-ART METHODS ON MTIL BENCHMARK (ORDER II) IN TERMS OF "TRANSFER", "AVERAGE", AND "LAST" SCORES (%)

	Method	Cars [87]	Food [84]	MNIST [85]	OxfordPet [86]	Flowers [83]	SUN397 [88]	Aircraft [78]	Caltech101 [79]	DTD [81]	EuroSAT [82]	CIFAR100 [80]	*Average*
CLIP	Zero-shot	64.7	88.5	59.4	89.0	71.0	65.2	24.3	88.4	44.6	54.9	68.2	65.3
	Full Fine-tune	89.6	92.7	99.6	94.7	97.5	81.8	62.0	95.1	79.5	98.9	89.6	89.2
	Fine-tune Adapter	89.1	92.9	99.2	94.1	97.0	82.7	56.8	92.6	79.0	98.4	89.4	88.3
Transfer	Continual-FT		85.9	59.6	57.9	40.0	46.7	11.1	70.0	30.5	26.6	37.7	46.6
	LwF [20]		87.8	58.5	71.9	46.6	57.3	12.8	81.4	34.5	34.5	46.8	53.2
	iCaRL [47]		86.1	51.8	67.6	50.4	57.9	11.0	72.3	31.2	32.7	48.1	50.9
	LwF-VR [89]		88.2	57.0	71.4	50.0	58.0	13.0	82.0	34.4	29.3	47.6	53.1
	WiSE-FT [90]		87.2	57.6	67.0	45.0	54.0	12.9	78.6	35.5	28.4	44.3	51.1
	ZSCL [12]		88.3	57.5	84.7	68.1	64.8	21.1	88.2	**45.3**	**55.2**	**68.2**	64.1
	L2P [14]		70.6	30.7	78.3	42.8	38.3	17.4	75.3	27.4	23.1	20.7	42.5
	DualPrompt [15]		79.9	46.9	85.2	51.3	45.1	9.3	82.7	29.9	42.9	47.2	52.1
	S-Prompts [13]		59.8	46.2	67.7	47.5	43.8	13.5	76.8	31.4	22.6	43.5	45.3
	DIKI [16]		85.8	**59.8**	**89.1**	**71.8**	62.6	**24.3**	**93.3**	42.7	46.8	67.8	64.4
	MoE-Adapters [28]		**88.8**	59.5	**89.1**	69.9	64.4	18.1	86.9	43.7	54.6	**68.2**	64.3(-0.1)
	MoE-Adapters++		**88.8**	59.5	**89.1**	71.3	**65.2**	18.2	87.9	45.1	54.6	**68.2**	**64.8**(+0.4)
Average	Continual-FT	42.1	70.5	**92.2**	80.1	54.5	59.1	19.8	78.3	41.0	38.1	42.3	56.2
	LwF [20]	49.0	77.0	92.1	85.9	66.5	67.2	20.9	84.7	44.6	45.5	50.5	62.2
	iCaRL [47]	52.0	75.9	77.4	74.6	58.4	59.3	11.7	79.6	42.1	43.2	51.7	56.9
	LwF-VR [89]	44.9	75.8	91.8	85.3	63.5	67.6	16.9	84.9	44.0	40.6	51.3	60.6
	WiSE-FT [90]	52.6	79.3	91.9	83.9	63.4	65.2	23.3	83.7	45.4	40.0	48.2	61.5
	ZSCL [12]	81.7	**91.3**	91.1	91.0	82.9	**72.5**	33.6	89.7	53.3	**62.8**	**69.9**	74.5
	L2P [14]	80.1	87.4	86.7	89.6	76.8	59.1	27.7	79.5	39.9	34.6	26.5	62.5
	DualPrompt [15]	78.6	88.4	89.7	91.7	80.0	62.4	23.2	85.0	41.3	51.6	50.7	67.5
	S-Prompts [13]	79.2	86.5	89.5	87.0	78.2	61.5	25.5	83.6	41.9	36.3	47.2	65.1
	DIKI [16]	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 [28]	84.9	89.9	89.3	91.4	86.2	72.2	33.4	89.4	53.3	61.4	**69.9**	74.7(+0.2)
	MoE-Adapters++	**86.8**	91.0	91.1	92.7	86.4	70.6	30.0	90.8	**54.3**	62.6	69.5	**75.1**(+0.6)
Last	Continual-FT	24.0	67.3	99.1	87.4	44.3	67.0	29.5	92.3	61.3	81.0	**88.1**	67.4
	LwF [20]	34.6	69.6	99.3	88.7	61.1	72.5	32.5	88.1	65.6	90.9	87.9	71.9
	iCaRL [47]	46.0	81.5	91.3	82.8	66.5	72.2	16.3	91.6	68.1	83.2	87.8	71.6
	LwF-VR [89]	27.4	61.2	99.4	86.3	60.6	70.7	23.4	88.0	61.3	84.3	**88.1**	68.2
	WiSE-FT [90]	35.6	76.9	**99.5**	89.1	62.1	71.8	27.8	90.8	67.0	85.6	87.6	72.2
	ZSCL [12]	78.2	91.1	97.6	92.5	87.4	**78.2**	45.0	92.3	72.7	96.2	86.3	83.4
	L2P [14]	80.1	89.1	99.1	93.8	96.2	76.5	40.1	86.9	73.5	86.3	84.2	82.3
	DualPrompt [15]	78.6	89.3	99.2	94.1	96.5	76.8	39.8	89.0	71.6	90.7	84.9	82.8
	S-Prompts [13]	79.2	89.1	99.1	**94.3**	95.8	76.3	39.9	95.5	70.1	97.6	84.4	83.8
	DIKI [16]	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 [28]	84.1	88.5	94.0	91.8	94.1	77.8	**50.4**	93.3	77.1	87.7	86.6	84.1(-1.4)
	MoE-Adapters++	**86.8**	**91.3**	98.1	**94.3**	**96.9**	75.2	44.0	95.8	**78.9**	**98.7**	81.9	**85.6**(+0.1)

We label the best and second-best methods with bold and underline styles. The top block indicates the upper-bound solutions to adapt the CLIP on each task.

TABLE VIII
COMPARISON WITH STATE-OF-THE-ART METHODS ON FEW-SHOT MTIL BENCHMARK (ORDER II) IN TERMS OF "TRANSFER", "AVERAGE", AND "LAST" SCORES (%)

	Method	Cars [87]	Food [84]	MNIST [85]	OxfordPet [86]	Flowers [83]	SUN397 [88]	Aircraft [78]	Caltech101 [79]	DTD [81]	EuroSAT [82]	CIFAR100 [80]	*Average*
CLIP	Zero-shot	64.7	88.5	59.4	89.0	71.0	65.2	24.3	88.4	44.6	54.9	68.2	65.3
	5-shot Full Fine-tune	65.4	83.3	96.6	84.9	92.9	71.3	30.6	93.5	65.1	91.7	76.8	77.5
	5-shot Fine-tune Adapter	68.2	87.8	90.4	89.0	94.2	72.5	29.7	90.0	63.9	81.1	75.3	76.6
Transfer	Continual-FT		76.0	64.6	67.1	49.7	53.7	8.3	77.9	33.9	23.9	37.1	49.2
	LwF [20]		64.2	59.1	68.1	38.4	54.9	6.7	78.0	35.5	33.5	47.4	48.6
	LwF-VR [89]		80.1	55.4	77.7	50.4	61.4	9.1	83.5	40.1	31.5	54.8	54.4
	WiSE-FT [90]		77.3	60.0	76.9	54.2	58.0	11.1	81.8	37.6	31.7	48.1	53.7
	ZSCL [12]		87.3	**64.8**	85.3	67.9	64.5	18.9	86.6	43.6	43.2	65.7	62.8
	MoE-Adapters [28]		**88.8**	59.5	**89.1**	**71.2**	**65.3**	18.2	**87.9**	44.2	**54.6**	**68.2**	64.7(+1.9)
	MoE-Adapters++		**88.8**	59.5	**89.1**	**71.3**	64.0	**23.4**	**87.9**	**45.1**	**54.6**	**68.2**	**65.2**(+2.4)
Average	Continual-FT	50.1	56.9	73.5	64.5	45.9	51.2	8.2	81.8	37.9	29.9	38.6	49.0
	LwF [20]	64.1	55.0	79.5	69.2	55.7	58.3	10.8	81.7	41.3	39.2	47.4	54.7
	LwF-VR [89]	63.3	76.9	71.4	79.1	68.9	65.0	13.4	86.0	45.7	36.3	55.3	60.1
	WiSE-FT [90]	59.3	64.7	77.4	70.3	51.3	58.6	10.8	84.2	42.0	38.6	49.1	55.1
	ZSCL [12]	**70.0**	85.0	79.8	86.1	79.4	68.3	21.8	88.8	48.8	49.3	66.5	67.6
	MoE-Adapters [28]	61.2	87.0	**87.3**	89.1	79.3	**68.5**	23.4	89.4	49.9	60.8	**68.8**	69.5(+1.9)
	MoE-Adapters++	67.0	**88.1**	86.5	**90.0**	**80.6**	67.7	**25.4**	**89.6**	**50.0**	**60.9**	68.6	**70.4**(+2.8)
Last	Continual-FT	35.2	28.6	58.3	51.2	14.0	46.1	5.3	89.5	47.0	52.9	53.6	42.8
	LwF [20]	57.1	40.1	84.1	58.1	50.5	57.6	14.3	87.9	54.7	64.0	47.0	56.8
	LwF-VR [89]	57.3	70.1	72.1	74.6	71.9	65.8	17.4	89.5	60.0	56.0	60.2	63.5
	WiSE-FT [90]	48.1	47.7	66.9	59.8	25.0	56.1	7.4	88.5	52.2	66.8	59.4	51.8
	ZSCL [12]	**67.4**	82.7	78.7	85.7	81.3	71.2	25.0	**92.5**	62.0	72.2	74.4	71.8
	MoE-Adapters [28]	59.4	87.0	91.8	89.0	84.1	**71.9**	**29.4**	91.4	**64.2**	88.8	**75.0**	75.7(+3.9)
	MoE-Adapters++	67.0	**88.0**	**92.5**	**90.4**	**85.6**	70.7	27.8	**92.5**	63.0	**89.4**	72.2	**76.3**(+4.5)

Ours converges in 500 iterations on few-shot. The top block indicates the upper-bound solutions to adapt the CLIP on each task.

TABLE IX
COMPARISON WITH STATE-OF-THE-ART METHODS ON THE MTIL AND DOMAINNET TIL BENCHMARKS IN TERMS OF "TRANSFER", "AVERAGE", AND "LAST" SCORES (%)

(a) MTIL (ViT-L)

Method	Transfer	Average	Last
Zero-shot	77.55	73.52	73.52
Full Fine-tune	52.28	64.41	81.72
LwF [20]	60.37	68.71	82.62
LwF-VR [89]	62.79	70.24	85.03
WiSE-FT [90]	61.55	69.79	85.59
ZSCL [12]	75.03	76.61	87.16
MoE-Adapters [28]	76.76(+1.73)	82.21(+5.60)	88.19(+1.03)
MoE-Adapters++	**76.92**(+1.89)	**82.43**(+5.82)	**88.45**(+1.29)

(b) MTIL (SigLIP)

Method	Transfer	Average	Last
Zero-shot	78.77	75.59	75.59
Full Fine-tune	70.68	77.28	84.90
LwF [20]	71.78	77.14	84.63
LwF-VR [89]	71.82	77.31	85.03
WiSE-FT [90]	73.96	78.97	85.75
ZSCL [12]	76.76	82.78	88.11
MoE-Adapters [28]	77.22(+0.46)	84.48(+1.70)	90.56(+2.45)
MoE-Adapters++	**78.29**(+1.53)	**85.19** (+1.23)	**90.76**(+2.41)

(c) DomainNet TIL (ViT-L)

Method	Transfer	Average	Last
Zero-shot	62.55	65.48	65.48
Full Fine-tune	57.34	63.60	59.36
LwF [20]	58.73	67.97	71.05
LwF-VR [89]	58.68	68.52	71.69
WiSE-FT [90]	58.99	68.71	72.21
ZSCL [12]	64.29	71.37	74.02
MoE-Adapters [28]	**64.44**(+0.15)	72.05(+0.68)	74.79(+0.77)
MoE-Adapters++	62.45(-1.84)	**72.53**(+1.16)	**76.97**(+2.95)

(d) DomainNet TIL (SigLIP)

Method	Transfer	Average	Last
Zero-shot	64.13	67.54	67.54
Full Fine-tune	58.69	69.52	72.95
LwF [20]	58.70	69.57	73.06
LwF-VR [89]	58.39	69.60	73.05
WiSE-FT [90]	61.87	71.52	74.43
ZSCL [12]	63.86	72.00	74.91
MoE-Adapters [28]	63.91(+0.05)	72.22(+0.22)	75.30(+0.39)
MoE-Adapters++	**64.27**(+0.41)	**72.27**(0.27)	**75.94**(+1.03)

The DomainNet TIL follows the domain order: ClipArt, InfoGraph, Painting, QuickDraw, Real, and Sketch, while MTIL follows the Order-I.

TABLE X
COMPARISON OF COMPUTATIONAL COST DURING TRAINING BETWEEN OUR METHODS AND OTHERS IN TERMS OF TRAINING PARAMETERS, GPU BURDEN, TRAINING TIME, AND INFERENCE TIME OF EACH ITERATION

Method	Train Params ↓	GPU ↓	Train Times ↓	Test Times ↓
LWF [20]	149.6M	32172MiB	1.54s/it	**0.50s/it**
LWF-VR [89]	149.6M	32236MiB	1.51s/it	**0.50s/it**
ZSCL [12]	149.6M	26290MiB	3.94s/it	0.52s/it
MoE-Adapters [28]	59.8M	22358MiB	1.58s/it	0.96s/it
MoE-Adapters++	**1.1M**	**6529MiB**	**0.68s/it**	0.77s/it
Δ	-98.16%	-70.80%	-56.92%	-19.79%

The Δ is the improvement relative to the original MoE-Adapters [28].

TABLE XI
ABLATION STUDIES ON INCREMENTAL MOE-ADAPTERS

Method	Transfer	Δ	Avg.	Δ	Last	Δ
CLIP Zero-shot	69.4	+0.5	65.3	-11.4	65.3	-19.7
+Adapter	45.0	-23.9	57.0	-19.7	71.5	-13.5
+2E/1R	45.1	-23.8	56.3	-20.4	71.1	-13.9
+2E/11R	68.1	-0.8	72.6	-4.1	77.9	-7.1
+22E/1R	44.1	-24.8	56.0	-20.7	66.2	-18.8
+22E/11R w/o F	68.6	-0.3	75.1	-1.6	82.0	-3.0
MoE-Adapters [28]	**68.9**	0.0	**76.7**	0.0	**85.0**	0.0

"*m*E/*n*R" indicates MoE with *m* experts and *n* routers, respectively. "F" represents the incremental activate-freeze strategy.

TABLE XII
ABLATION STUDY OF DIFFERENT LOSSES IN DDAS

Method	full-shot			5-shot		
	Trans.	Avg.	Last	Trans.	Avg.	Last
ZSCL [12]	68.1	75.4	83.6	65.3	64.4	67.4
MAE	68.4	73.8	77.8	68.5	68.5	70.5
Smooth L1	68.3	**76.9**	84.9	**69.0**	69.5	72.6
MSE	**68.9**	76.7	**85.0**	68.9	**71.7**	**76.1**

TABLE XIII
ABLATION STUDIES ABOUT DIFFERENT MODULES ON DYNAMIC MOE-ADAPTERS++

Method	Transfer	Δ	Avg.	Δ	Last	Δ
CLIP Zero-shot	69.4	+0.4	65.3	-12.2	65.3	-20.9
+Adapter	49.0	-20.0	58.9	-18.6	72.5	-13.7
+Adapter(only Image)	48.5	-20.5	59.2	-19.4	73.5	-12.7
+Single Router w/o LEAS	52.5	-16.5	62.5	-12.5	64.6	-21.6
+Single Router	68.9	-0.1	70.6	-6.9	71.4	-14.8
MoE-Adapters++	**69.0**	0.0	**77.5**	0.0	**86.2**	0.0

TABLE XIV
ABLATION STUDIES ON TRAINING STRATEGY OF MOE-ADAPTERS++

Method	$Thres_e$	Transfer	Δ	Avg.	Δ	Last	Δ
Same Input	0.0	68.5	-0.5	76.6	-0.9	85.1	-1.1
	1.0	68.1	-0.9	76.7	-0.8	85.8	-0.4
	2.0	68.5	-0.5	76.6	-0.9	85.0	-1.2
DEeC in R.L.	0.0	68.5	-0.5	76.9	-0.6	85.8	-0.4
	1.0	68.4	-0.6	76.8	-0.7	84.9	-1.3
	2.0	68.5	-0.5	76.0	-1.5	82.5	-3.7
MoE-Adapters++	0.0	68.5	-0.5	76.9	-0.6	85.8	-0.4
	1.0	**69.0**	0.0	**77.5**	0.0	**86.2**	0.0
	2.0	68.7	-0.3	76.7	-0.8	85.2	-1.0

The $Thres_e$ represents the expansion threshold.

the zero-shot CLIP model and the fine-tuned adapter, our MoE-Adapters++ effectively mitigates the "catastrophic forgetting" while preserving zero-shot transfer ability. Compared with the "Single Router w/o LEAS" setting, the proposed LEAS enhances the model's ability to perceive data distributions, allocating the test data to the most appropriate experts. And it also preserves CLIP's zero-shot capability for unseen data, improving both anti-forgetting and "Transfer" abilities. In Dynamic MoE-Adapters, task-specific routers enhance prediction accuracy on seen data more effectively than a single router by better retaining task-specific knowledge. The comparison between "MoE-Adapters++" and "Single Router" demonstrates the effectiveness of task-specific routers.

Analysis of Training Strategy for MoE-Adapters++: As shown in Table XIV, we show ablation studies with different training strategy settings. In "Same Input" setting, DEeC and the router use the same inputs (*i.e.,* $\boldsymbol{c}^t_{MoE}$). In "DEeC in R.L." setting, we use the same DEeC module in the Recognition Layer to automatically control the expansion of experts, and DEeC and LEAS share the parameters of the Auto-Encoders. "$Thres_e = 0.0$" means the force expansion strategy. "MoE-Adapters++" and "DEeC in R.L." are equivalent when $Thres_e = 0.0$, since both employ the force expansion strategy in all layers. Compared with "Same Input" setting, using different inputs (*i.e.,* $\boldsymbol{c}^t_{frozen}$

TABLE XV
ABLATION STUDY ON DIFFERENT WEIGHTED VECTORS OF DISCREPANCY FUNCTION IN LEAS

Vector $\boldsymbol{\alpha}$	Full-shot			Acc. of LEAS	
	Trans.	Avg.	Last	Unseen	Seen
None	67.9	76.5	85.6	53.8	98.6
Norm $1/\boldsymbol{V}_{std}$	68.4	76.4	82.6	90.2	88.4
$1/\boldsymbol{V}_{std}$	67.5	76.1	84.6	90.4	93.4
Norm $1/\boldsymbol{V}_{con}$	67.8	76.2	83.0	78.1	87.5
$1/\boldsymbol{V}_{con}$	**69.0**	**77.5**	**86.2**	**96.5**	**99.9**

"None" indicates that the α weighting is not being used.

TABLE XVI
ABLATION STUDY OF THE NUMBER OF LAYERS FOR MOE-ADAPTERS++'S DEPLOYMENT

Layer	Full-shot			Acc. of LEAS	
	Trans.	Avg.	Last	Unseen	Seen
1R/9S	59.9	67.1	76.4	53.9	67.4
1R/7S	**69.3**	76.6	85.7	**99.5**	96.4
1R/5S	69.0	**77.5**	**86.2**	96.5	**99.9**
1R/3S	67.9	75.1	82.9	85.0	93.5
1R/1S	65.4	73.9	83.2	54.8	95.7
3R/5S	**69.3**	75.5	83.9	98.7	95.0

"mR/nS" indicates transformer with m Recognition Layers and n Subsequent Layers.

for DEeC and $\boldsymbol{c}^t_{MoE}$ for router) avoids the mutual interference of gradient back-propagation between MoE and DEeC during training and achieves better performance. The comparison of "DEeC in R.L." and "MoE-Adapters++" demonstrates the effectiveness of adopting the incremental expansion in LEAS, which ensures all Auto-Encoders trained from the first batch to enhance the perception of each Auto-Encoder.

Analysis of LEAS: Table XV reports the ablation on discrepancy functions, demonstrating the effectiveness of the weighted vector in (8). "Unseen" is the accuracy of routing unseen data to frozen CLIP, and "Seen" is routing seen data to the correct router. The "None" means $\boldsymbol{\alpha} = 1$, and "Norm" represents the process of normalizing. As shown in Table XV, the weighted vector improves the ability of LEAS to perceive data distribution compared with "None" setting. We also conduct ablation on the standard deviation ($\boldsymbol{V}_{std}$) of reconstruction losses to explore the impact of the Auto-Encoder's stability for data reconstruction on assessing data distribution. "$1/\boldsymbol{V}_{con}$" achieves better performance by being more directly expressive of the adaptation between task data and Auto-Encoders. In addition, the settings of "$1/\boldsymbol{V}_{con}$" and "$1/\boldsymbol{V}_{std}$" show better performance in recognition accuracy of LEAS compared to the use of normalized vectors, as normalization diminishes each task-specific Auto-Encoder's response to task-agnostic data distribution. To further verify the effectiveness of the designed LEAS, as shown in Fig. 7(b), we apply the weighted vector $1/\boldsymbol{V}_{con}$ to visualize the output of the distribution in feature space. Compared to the MoE-Adapters, our extended version achieves better differentiation of data distribution with reduced computational overhead.

Analysis of Layers Deployed in MoE-Adapters++: Table XVI illustrates the impact of the number of deployment layers on the performance of MoE-Adapters++. When the Recognition Layer is deployed in middle position like "1R/5S" and "1R/7S" settings, LEAS exhibits the best distribution discrimination ability, along with better performance across all metrics. For the shallower Recognition Layer like "1R/9S" setting, it limits

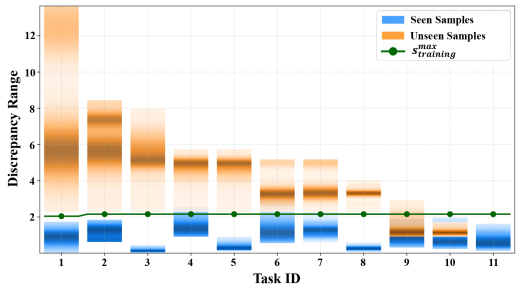


Fig. 8. The discrepancy distributions across tasks during training and inference stages. The blue and yellow bars represent the discrepancy values s^t of seen and unseen testing samples, respectively, after completing training on the current task ID. The green curve denotes the maximum $s^t_{training}$ ($s^{\max}_{training}$) among the first t tasks, which serves as the reference $Thres_{zs}$ after task t training.

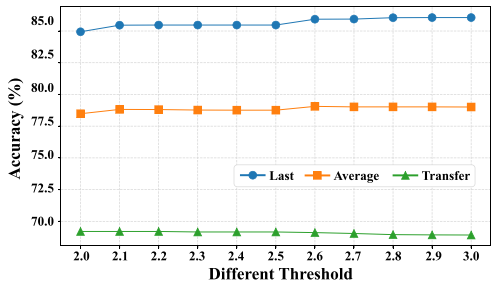


Fig. 9. Metrics across varying Zero-shot Thresholds $Thres_{zs}$ of MoE-Adapters++, evaluated on the MTIL benchmark following the Order-I.

the representational capacity of the latent feature embeddings fed to LEAS, reducing its ability to distinguish distributions and increasing the risk of incorrect routing. For the deeper Recognition Layer like "1R/3S" and "1R/1S" settings, the recognition ability of LEAS actually decreases. This may be due to the domain-specific information in the deeper latent feature embeddings, which limits recognition across different domains. In the "3R/5S" setting, we deploy multi-layer LEAS, integrating multiple layers of latent feature embeddings to infer task context. This enhances the "Transfer" and "Unseen" capability but reduces the "Last" and "Seen" capability.

Analysis of $Thres_{zs}$: To intuitively demonstrate the practicality of generating thresholds based on training data, we visualize the discrepancy distributions of seen and unseen samples during inference, along with the threshold curve obtained during training, as shown in Fig. 8. The visualization results indicate that the predicted threshold $s^{\max}_{training}$ effectively distinguishes between seen and unseen regions. Furthermore, to verify the robustness and reproducibility of the proposed thresholding strategy, we conduct additional ablation studies by varying the threshold values around the predicted point. As shown in Fig. 9, the results demonstrate that all evaluation metrics remain relatively stable across a wide range of $Thres_{zs}$ values, highlighting the strong robustness of the LEAS framework.

Visualization of Layer-wise Expansion Behavior: We further conduct an ablation study to analyze DEeC's layer-wise expert expansion behavior, focusing on all subsequent layers of the ViT-B backbone on the MTIL benchmark. As shown in

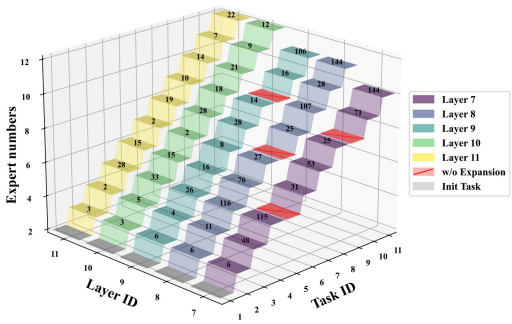


Fig. 10. The visualization of layer-wise expansion behavior of MoE-Adapters++. The number above each square indicates the iteration count of expansion signal triggering for the corresponding task and layer. The "w/o Expansion" represents layers that never activated expansion signals during the training process.

TABLE XVII
ABLATION STUDY ON THE NUMBER OF EXPERTS ACROSS DIFFERENT SIZE OF DATASET

N_E/N_D	4-task			8-task			11-task		
	Trans.	Avg.	Last	Trans.	Avg.	Last	Trans.	Avg.	Last
$N_E = 4$	64.9	67.5	77.1	65.0	71.2	77.9	66.5	71.1	74.1
$N_E = 8$	65.1	68.3	78.3	65.4	73.7	84.9	67.4	75.7	82.4
$N_E = 16$	65.3	67.7	77.9	65.5	73.9	84.9	68.0	76.4	84.6
$N_E = 20$	65.5	67.4	77.0	66.6	74.6	85.8	67.6	76.0	84.2
$N_E = 22$	66.9	68.0	76.9	67.1	74.5	85.8	68.9	76.7	85.0
N_D	**67.1**	**69.8**	**78.8**	**67.9**	**75.9**	**87.1**	**69.0**	**77.5**	**86.2**

The number of experts in MoE-Adapters and MoE-Adapters++ is denoted by N_E and N_D, respectively.

Fig. 10, at the beginning of each task, expansion is typically triggered within just a few training iterations across all layers. As the task sequence progresses and more knowledge becomes available, expansion in the shallower layers slows down noticeably, showing a higher degree of knowledge reusability as the number of experts increases. In contrast, deeper layers require more to require expert expansion, due to a greater need for task-specific, discriminative knowledge. This observation aligns with the general understanding of the transformer architecture, where shallower layers tend to capture generalizable features while deeper layers encode more specialized representations.

Impact of Dataset Size on Expert Number: We conduct ablation experiments to determine the optimal number of experts for different task sizes (Table XVII) in the full-shot setting. For MoE-Adapters, larger task counts generally require more experts, but simply adding experts may not improve accuracy, leading to parameter redundancy in early continual learning stages. In contrast, MoE-Adapters++ adopts a dynamic expansion strategy with variable N_D experts, incrementally extending knowledge capacity while consistently maintaining high performance.

Effectiveness of Router Selection: We visualize expert selection frequencies for each incremental task in Fig. 11(a), recorded in the 6th and 8th visual transformer blocks of CLIP with 22 experts per block and Top-2. The results reveal sparse activations and cooperation between specialized and shared experts, but also show that many pre-defined experts are rarely used, causing parameter redundancy and higher GPU memory usage. For MoE-Adapters++ (Fig. 11(b)), the expert set grows dynamically with tasks to avoid redundancy, with task-specific

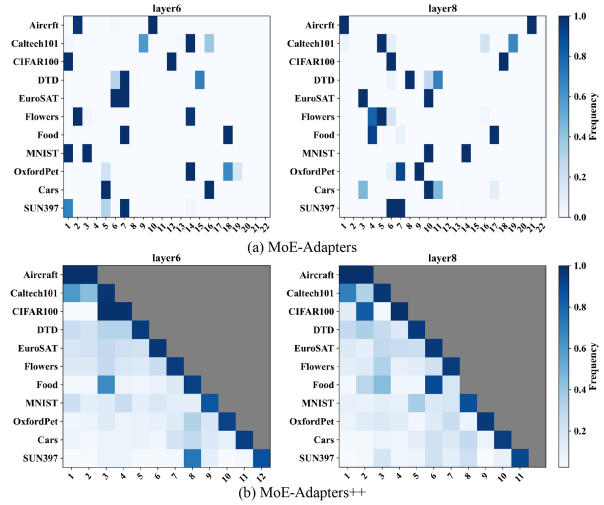


Fig. 11. Visualization of the frequency that experts are selected for each task. (a) The activation frequencies of MoE-Adapters' experts, with 22 experts and Top-k as 2. (b) The activation frequencies of MoE-Adapters++'s experts, with dynamic experts and Top-k as 2.

routers adapting to the changing number of experts and enabling effective collaboration between new and existing experts. Notably, in layer 8 of the Food task, some tasks are trained solely by reusing existing experts, further improving training efficiency.

VI. CONCLUSION

In this work, we propose a parameter-efficient training framework MoE-Adapters, and further extend an MoE-Adapters++ to reduce the parameter redundancy and training burdens. Based on MoE-Adapters, we design a dynamic MoE-Adapters to address the redundancy of pre-defined expert set by enabling the flexible expansion of routers and experts with incremental tasks. In addition, we improve the Distribution Discriminative Auto-Selector (DDAS) to Latent Embedding Auto-Selector (LEAS) for unifying the training pipeline and reducing the training overhead, which can directly utilize low-dimensional features from frozen CLIP's image encoder for distribution selection. Extensive experiments across various settings demonstrate the superiority of the MoE-Adapters and MoE-Adapters++ over previous arts in terms of classification accuracy and training efficiency.

REFERENCES

[1] J. Achiam et al., "Gpt-4 technical report," 2023, *arxiv:2303.08774*.
[2] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.
[3] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.
[4] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 34892–34916.
[5] S. Liu et al., "Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2024, pp. 38–55.
[6] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, New York, NY, USA: Elsevier, 1989, vol. 24, pp. 109–165.

[7] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–10.

[8] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 8, pp. 5362–5383, Aug. 2024.

[9] Q. Pham, C. Liu, and S. C. H. Hoi, "Continual learning, fast and slow," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 1, pp. 134–149, Jan. 2024.

[10] P. Mazumder, P. Singh, P. Rai, and V. P. Namboodiri, "Rectification-based knowledge retention for task incremental learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 3, pp. 1561–1575, Mar. 2024.

[11] K. Han, S.-A. Rebuffi, S. Ehrhardt, A. Vedaldi, and A. Zisserman, "Autonovel: Automatically discovering and learning novel visual categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6767–6781, Oct. 2022.

[12] Z. Zheng, M. Ma, K. Wang, Z. Qin, X. Yue, and Y. You, "Preventing zero-shot transfer degradation in continual learning of vision-language models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 19125–19136.

[13] Y. Wang, Z. Huang, and X. Hong, "S-prompts learning with pre-trained transformers: An Occam's Razor for domain incremental learning," in *Proc. Conf. Neural Inf. Process. Syst.*, 2022, pp. 5682–5695.

[14] Z. Wang et al., "Learning to prompt for continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 139–149.

[15] Z. Wang et al., "DualPrompt: Complementary prompting for rehearsal-free continual learning," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 631–648.

[16] L. Tang et al., "Mind the interference: Retaining pre-trained knowledge in parameter efficient continual learning of vision-language models," in *Proc. Eur. Conf. Comput. Vis.*, 2024, pp. 346–365.

[17] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert Gate: Lifelong learning with a network of experts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3366–3375.

[18] J. Donahue et al., "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 647–655.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.

[20] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[21] P. Gao et al., "Clip-adapter: Better vision-language models with feature adapters," *Int. J. Comput. Vis.*, vol. 132, no. 2, pp. 581–595, 2023.

[22] Y.-L. Sung, J. Cho, and M. Bansal, "VL-Adapter: Parameter-efficient transfer learning for vision-and-language tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5227–5237.

[23] R. Zhang et al., "Tip-Adapter: Training-free clip-adapter for better vision-language modeling," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 493–510.

[24] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *Proc. Annu. Meet. Assoc. Comput. Linguistics*, 2022, pp. 1–9.

[25] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–26.

[26] Z.-Y. Hu, Y. Li, M. R. Lyu, and L. Wang, "VL-Pet: Vision-and-language parameter-efficient tuning via granularity control," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 3010–3020.

[27] M. Jia et al., "Visual prompt tuning," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 709–727.

[28] J. Yu et al., "Boosting continual learning of vision-language models via mixture-of-experts adapters," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 23219–23230.

[29] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 1991.

[30] N. Shazeer et al., "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–19.

[31] T. De Min, M. Mancini, K. Alahari, X. Alameda-Pineda, and E. Ricci, "On the effectiveness of layernorm tuning for continual learning in vision transformers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2023, pp. 3577–3586.

[32] Q. Jodelet, X. Liu, Y. J. Phua, and T. Murata, "Class-incremental learning using diffusion model for distillation and replay," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 3425–3433.

[33] E. Belouadah and A. Popescu, "IL2M: Class incremental learning with dual memory," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 583–592.

[34] S. Yan, J. Xie, and X. He, "DER: Dynamically expandable representation for class incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3014–3023.

[35] X. Liu, X. Cao, H. Lu, J. wen Xiao, A. D. Bagdanov, and M.-M. Cheng, "Class incremental learning with pre-trained vision-language models," 2023, *arxiv:2310.20348*.

[36] M. Boschini, L. Bonicelli, P. Buzzega, A. Porrello, and S. Calderara, "Class-incremental continual learning into the extended der-verse," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5497–5512, May 2023.

[37] D.-W. Zhou, Q.-W. Wang, Z.-H. Qi, H.-J. Ye, D.-C. Zhan, and Z. Liu, "Class-incremental learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 9851–9873, Dec. 2024.

[38] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: Survey and performance evaluation on image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5513–5533, May 2023.

[39] G. Oren and L. Wolf, "In defense of the learning without forgetting for task incremental learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 2209–2218.

[40] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7765–7773.

[41] M. De Lange et al., "A continual learning survey: Defying forgetting in classification tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3366–3385, Jul. 2022.

[42] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 2994–3003.

[43] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 6470–6479.

[44] A. Prabhu, P. H. Torr, and P. K. Dokania, "GDumb: A simple approach that questions our progress in continual learning," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 524–540.

[45] F. Lavda, J. Ramapuram, M. Gregorova, and A. Kalousis, "Continual classification learning using generative models," in *Proc. Adv. Neural Inf. Process. Syst. Worksh.*, 2018, pp. 1–5.

[46] O. Ostapenko et al., "Continual learning with foundation models: An empirical study of latent replay," in *Proc. Conf. Lifelong Learn. Agents*, 2022, pp. 60–91.

[47] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2001–2010.

[48] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 3302–3309.

[49] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (NOT) to forget," in *Eur. Conf. Comput. Vis.*, 2018, pp. 139–154.

[50] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017.

[51] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 4655–4665.

[52] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3987–3995.

[53] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5138–5146.

[54] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle, "PODNet: Pooled outputs distillation for small-tasks incremental learning," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 86–102.

[55] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 831–839.

[56] A. Douillard, A. Ramé, G. Couairon, and M. Cord, "DyTox: Transformers for continual learning with dynamic token expansion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9285–9295.

[57] C. Zhang et al., "Reusable architecture growth for continual stereo matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 9, pp. 6167–6184, Sep. 2024.

[58] Z. Hu, Y. Li, J. Lyu, D. Gao, and N. Vasconcelos, "Dense network expansion for class incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 11858–11867.
[59] F. Ye and A. G. Bors, "Self-evolved dynamic expansion model for task-free continual learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 22102–22112.
[60] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–11.
[61] J. Xu, J. Ma, X. Gao, and Z. Zhu, "Adaptive progressive continual learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6715–6728, Oct. 2022.
[62] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 1877–1901.
[63] J. O. Zhang, A. Sax, A. Zamir, L. Guibas, and J. Malik, "Side-tuning: A baseline for network adaptation via additive side networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 698–714.
[64] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.
[65] R. Karimi Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 1022–1035.
[66] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proc. Annu. Meet. Assoc. Comput. Linguistics Int. Joint Conf. Nat. Lang. Process.*, 2021, pp. 1–15.
[67] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2021, pp. 3045–3059.
[68] C. Ju, T. Han, K. Zheng, Y. Zhang, and W. Xie, "Prompting visual-language models for efficient video understanding," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 105–124.
[69] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Conditional prompt learning for vision-language models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16816–16825.
[70] C. Riquelme et al., "Scaling vision with sparse mixture of experts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 8583–8595.
[71] B. Mustafa, C. Riquelme, J. Puigcerver, R. Jenatton, and N. Houlsby, "Multimodal contrastive learning with LIMoE: The language-image mixture of experts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 9564–9576.
[72] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, no. 1, pp. 5232–5270, 2022.
[73] E. Daxberger et al., "Mobile V-MoEs: Scaling down vision transformers via sparse mixture-of-experts," 2023, *arXiv:2309.04354*.
[74] W. Chen et al., "Lifelong language pretraining with distribution-specialized experts," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 5383–5395.
[75] X. Zhang, Y. Shen, Z. Huang, J. Zhou, W. Rong, and Z. Xiong, "Mixture of attention heads: Selecting attention heads per token," in *Proc. Conf. Empirical Methods Nat. Lang. Process.,* 2022, pp. 4150–4162.
[76] Z. Chen et al., "Mod-Squad: Designing mixtures of experts as modular multi-task learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 11828–11837.
[77] H. Wang, H. Lu, L. Yao, and D. Gong, "Self-expansion of pre-trained models with mixture of adapters for continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2024, pp. 10087–10098.
[78] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," 2013, *arXiv:1306.5151*.
[79] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshop*, 2004, pp. 178–178.
[80] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Accessed: Aug. 4, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744
[81] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3606–3613.
[82] P. Helber, B. Bischke, A. Dengel, and D. Borth, "EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 7, pp. 2217–2226, Jul. 2019.
[83] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th indian Conf. Comput. Vis., Graph. Image Process.*, 2008, pp. 722–729.
[84] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 446–461.
[85] L. Deng, "The MNIST database of handwritten digit images for machine learning research [Best of the Web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[86] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar, "Cats and dogs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3498–3505.
[87] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2013, pp. 554–561.
[88] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 3485–3492.
[89] Y. Ding, L. Liu, C. Tian, J. Yang, and H. Ding, "Don't stop learning: Towards continual learning for the clip model," 2022, *arXiv:2207.09248*.
[90] M. Wortsman et al., "Robust fine-tuning of zero-shot models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 7959–7971.
[91] Y. Wu et al., "Large scale incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 374–382.
[92] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 233–248.
[93] Y. Liu, S. Parisot, G. Slabaugh, X. Jia, A. Leonardis, and T. Tuytelaars, "More classifiers, less forgetting: A generic multi-classifier paradigm for incremental learning," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 699–716.
[94] F. Zhu, X.-Y. Zhang, C. Wang, F. Yin, and C.-L. Liu, "Prototype augmentation and self-supervision for incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5871–5880.
[95] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[96] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia, "Maintaining discrimination and fairness in class incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13208–13217.
[97] S.-K. Huang, J.-F. Chang, and C.-R. Huang, "DPFormer: Dynamic prompt transformer for continual learning," 2025, *arXiv:2506.07414*.
[98] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1406–1415.
[99] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–11.
[100] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, pp. 1097–1105.
[101] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–11.
[102] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 4697–4703.

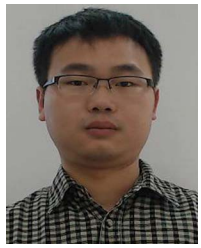
Jiazuo Yu received the BS degree from the Chengdu University of Technology, in 2020. He is currently working toward the PhD degree with the School of Information and Communication Engineering, the Dalian University of Technology. His research interests include continual learning, vision language model, multi-modal large language model.

Zichen Huang received the BE degree in electronic information engineering from the Dalian University of Technology, Dalian, China, in 2024, where he is currently working toward the master's degree in engineering. His research interest includes deep learning, continual learning, and vision language model.

Yunzhi Zhuge received the BE degree in electrical engineering from the Shandong University of Science and Technology, Qingdao, China, in 2017, the MS degree in biomedical engineering from the Dalian University of Technology, Dalian, China, in 2019, and the PhD degree in computer science from The University of Adelaide, Adelaide, SA, Australia, in 2023. He is currently a postdoctoral researcher with Dalian University of Technology.

Lu Zhang (Member, IEEE) received the bachelor and PhD degrees with the Dalian University of Technology from 2010 to 2021. She is currently an associate professor with the School of Electronic and Information Engineering, the Dalian University of Technology. Her research interest includes deep learning, video understanding, and object segmentation.

Ping Hu (Member, IEEE) received the BE degree from Sichuan University, Chengdu, China, in 2013, the ME degree from the University of Chinese Academy of Sciences, Beijing, China, in 2016, and the PhD degree from Boston University, Boston, MA, USA, in 2023. His research interests include machine learning and computer vision. He was a Senior Program Committee Member for IJCAI 2023 and an Area Chair for CVPR 2024.



Dong Wang received the BE degree in electronic information engineering and the PhD degree in signal and information processing from the Dalian University of Technology (DUT), Dalian, China, in 2008 and 2013, respectively. He is currently a full professor with the School of Information and Communication Engineering, DUT. His current research interests include object detection and tracking.

Huchuan Lu (Fellow, IEEE) received the MS degree in signal and information processing and the PhD degree in system engineering from the Dalian University of Technology (DUT), Dalian, China, in 1998 and 2008, respectively. In 1988, he joined the faculty of the School of Information and Communication Engineering, DUT, where he is currently a full professor. He is also the Dean of the School of Future Technology, DUT. His current research interests include computer vision and artificial intelligence with a focus on visual tracking, saliency detection, and multimodal large model.

You He received the PhD degree from Tsinghua University, Beijing, China, in 1997. In 2013, he was selected as an Academician with the Chinese Academy of Engineering, Beijing. He is currently a professor with the Department of Electronic Engineering, Tsinghua University, and the Director of the Institute of Information Fusion, Naval Aeronautical University, Yantai, China, and the Key Laboratory of Information Perception and Fusion Technology, Yantai. His main research interests include information fusion, computer vision, and Big Data technology. He is an IET Fellow. He was the recipient of the four second prizes of national science and technology progress and seven first prizes of military science and technology progress.