Self-Expansion of Pre-trained Models with Mixture of Adapters for Continual Learning

Huiyi Wang[1], Haodong Lu[1], Lina Yao[2,1], and Dong Gong[⋆1]

[1]University of New South Wales, [2]CSIRO's Data61
{huiyi.wang, haodong.lu, dong.gong}@unsw.edu.au,
lina.yao@data61.csiro.au

Abstract. Continual learning aims to learn from a stream of continuously arriving data with minimum forgetting of previously learned knowledge. While previous works have explored the effectiveness of leveraging the generalizable knowledge from pre-trained models in continual learning, existing parameter-efficient fine-tuning approaches focus on the use of a predetermined or task-wise set of adapters or prompts. However, these approaches still suffer from forgetting due to task interference on jointly used parameters or restricted flexibility. The reliance on a static model architecture may lead to the allocation of excessive parameters that are not essential or, conversely, inadequate adaptation for downstream tasks, given that the scale and distribution of incoming data are unpredictable in continual learning. We propose **S**elf-**E**xpansion of pre-trained models with **M**odularized **A**daptation (SEMA), a novel fine-tuning approach which automatically decides to reuse or add adapter modules on demand in continual learning, depending on whether drastic distribution shift that could not be handled by existing modules is detected at different representation levels. We design each adapter module to consist of an adapter and a representation descriptor, specifically, implemented as an autoencoder. The representation descriptor functions as a distributional shift indicator during training and triggers adapter expansion. For better usage of the adapters, an expandable weighting router is learned jointly for mixture of adapter outputs. By comparing with vision-transformer-based continual learning adaptation methods, we demonstrate that the proposed framework outperforms the state-of-the-art without memory rehearsal.

Keywords: Continual learning · fine-tuning · self-expansion

1 Introduction

While deep learning models, such as Vision Transformer (ViT) [16], have achieved magnificent success in many computer vision tasks, they are mostly trained with fixed datasets and might not be able to handle real-world scenarios with dynamic requirements where data distribution and tasks can change over time. To

⋆ D. Gong is the corresponding author.

address the difficulty of maintaining all seen data and repeatedly re-training the model, continual learning (CL) aims to learn incrementally from a continuous data stream [15, 52, 62]. CL faces the notorious challenge of catastrophic forgetting [42], where acquiring new knowledge leads to the erasure of previously learned information. Many CL approaches, such as those utilizing experience replay (ER) [7, 8, 69] and diverse regularization techniques [35, 74], have been investigated and shown considerable results in reducing forgetting.

While most CL methods [6,42,70] root in the "training-from-scratch" paradigm, recent works have started to explore the potential of integrating pre-trained foundation models into CL as robust feature extractors [45, 79], or adapting them to downstream tasks through parameter-efficient fine-tuning with prompts and/or adapters [14, 60, 65, 66, 79, 80]. On the other hand, these methods enable continual fine-tuning of pre-trained models on real-world downstream tasks arriving in a streaming manner. Existing approaches primarily employ a *fixed* set of prompts or adapters shared by all tasks, which provides limited adaptation to downstream tasks and causes foreseeable forgetting due to interference [45, 65, 66, 79]. The model capacity for adaptation is also limited by the fixed size of the prompt/adapter pool [14, 65, 66]. Some methods add task-wise parameters periodically for tackling forgetting [60], which are mainly restricted to prompts and lack of effectiveness and flexibility. While some works investigate the use of sub-network modules to alleviate task interference, they mostly confine themselves to pre-defined per-task modules or rely on later pruning, with limited flexibility [11,30,51,58]. Recent development in dynamic expansion architecture [48, 61] has demonstrated its potential in continual learning to prevent forgetting by growing capacity for novel tasks, which are restricted to simple applications due to the complex model designing.

To address the above issues, we propose SEMA, a continual learning approach based on **S**elf-**E**xpansion of pre-trained models with **M**odularized **A**daptation, which can be integrated into transformer-based pre-trained models. With adapters inserted into transformer blocks at different layers, we efficiently align the pre-trained model with the distribution of downstream tasks. Through the flexible expansion of new adapters at arbitrary layers, our method can effectively accommodate drastic distribution shifts in incoming tasks without overwriting previously learned knowledge and retain performance on old tasks. Unlike previous methods manually adding task-oriented adapters [30,79] or prompts [60], SEMA is proposed to *automatically* decide whether to reuse existing adapters or add new ones, as shown in Fig. 1. It enables the model to perform much better by slightly expanding the parameter size on the demand in moderation. The key challenge arises from two perspectives: i) when and where (*i.e.*, which layer) would an adapter expansion be considered necessary to perform; ii) how to best combine the acquired knowledge from different adapters.

We introduce *modular/modularized adapters* that can be identified and reused to solve new tasks, selectively adding and learning a subset of new adapters for unseen knowledge in these tasks. Specifically, we design the modular adapter as a pair of a functional *adapter* and a *representation descriptor*. In addition to the adapter that functions as an adaptation features generator, the representation descriptor is responsible for capturing the feature distribution relevant to the coupled adapter at the corresponding layer. It serves as an indicator of the distribution shift at the representation level instead of only the input image level, to detect novel patterns in data. For example, cat images and dog images have more shared features than car images; a SEMA model only trained on cat images tends to expand more new adapters when training on car images than dog images. We implement the representation descriptors with autoencoder (AE) [25] and train them to model the distribution of the relevant input feature of the corresponding task. SEMA expands itself by adding a new adapter to supplement the existing model, when detecting significant distribution shifts according to the expansion signal from the representation descriptors. To effectively use and obtain mixture of the added adapters, we design an *expandable weighting router* to mix the adapters, which are simultaneously expanded and learned as the adapter modules themselves are expanded. Learning the mixture through the expandable router also mitigates concerns associated with the indirect learning of mixture patterns, as opposed to weighting the adapters based on the distributional similarity estimated by representation descriptors.

We summarize our contribution as follows:

– We propose a novel continual learning approach via self-expansion of pre-trained models with modularized adapters, *i.e.* SEMA. It learns and reuses modules, while adding new ones at specific layers exclusively for samples with new patterns. It incorporates automated strategies to determine expansion necessity and location and to facilitate the learning of new adapters. SEMA operates without the need for rehearsal.
– To achieve SEMA, our approach involves crafting modular adapters comprising a functional adapter and a representation descriptor. The representation descriptor captures the distribution of pertinent input features, serving as a signal generator for expansion during training. The expandable weighting router is introduced for adapter mixture.
– We performed extensive experiments to verify the performance and scrutinize the behavior of the self-expansion approach we proposed.

2 Related Work

Continual Learning (CL). The mainstream taxonomy classifies continual learning methods into three categories: replay-based methods, regularization-based methods and architecture-based methods [15, 62]. Replay-based methods aim to alleviate catastrophic forgetting by retaining a memory buffer to store the information from old tasks for future replay [7, 9, 43, 52]. With simple intuition and effectiveness in preventing forgetting, these methods are limited by the size of the memory buffer and may also raise privacy concerns. An alternative approach is to implicitly maintaining a generative model for producing pseudo-samples with similar distribution to old classes [12, 34, 53, 54, 59]. Regularization-based methods penalize significant changes to important parameters for seen

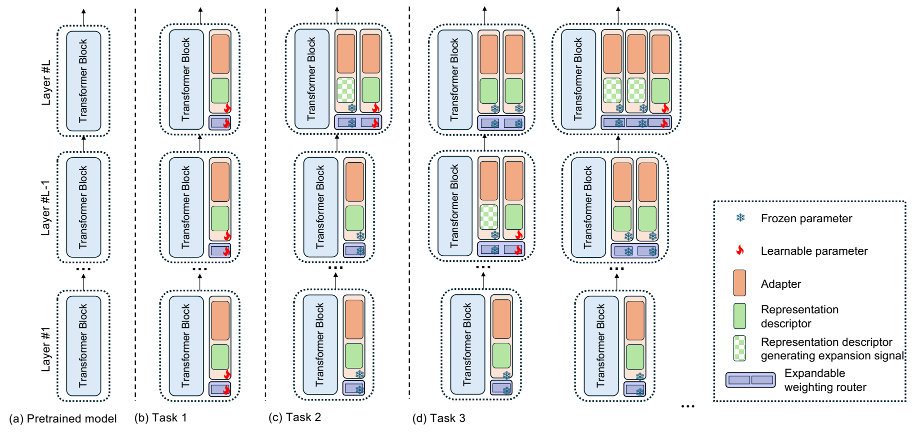


Fig. 1: Illustration of the expansion process. (a) The pretrained model with L transformer blocks is provided for adaptation. (b) At the start of training, each transformer block is equipped with one router and one adapter module, including one functional adapter and its paired representation descriptor. All modules are activated for training. (c) All modules and the router are frozen after the training on Task 1. When Task 2 arrives, the representation descriptor in the L-th block observes a feature distribution shift and generates expansion signal. A new module is added in the L-th block and available for training. The router is expanded. (d) As Task 3 arrives, expansion signal is triggered in the $L-1$-th block. After sufficient training, the newly added module is frozen and detection for distribution shift in later blocks is executed. When both representation descriptors in the L-th block consider the incoming feature as an outlier, expansion signal will again be triggered and a new module is added for training.

tasks [2,3,35,47,74,75], or consolidate the knowledge learnt from previous tasks with knowledge distillation [26,37,42,77]. Instead of using all available parameters for all tasks, architecture-based methods allocate a subset of parameters dedicated to each task, which can be performed with task masking [33,44,57,67] or dynamic architecture [4,29,40,41,48,61,70–73]. These methods tend to achieve optimal performance with less forgetting as isolating the parameters reduces task interference during training.

Parameter-Efficient Fine-Tuning (PEFT). Parameter-efficient fine-tuning methods train a small set of additional parameters rather than the entire pretrained model, which reduce the demands placed upon computational resources. Prompting applies learnable prompts that modifies the inputs to provide the model with more instructions [31,39]. LoRA [28] injects low-rank matrices to approximate weight updates and avoids additional inference latency via reparameterization, which has been further utilized as experts with mixture modeling in recent works [17,21,64,68]. Adapters introduced by [27], along with its variants [10,32], insert lightweight learnable modules into the transformer. To enhance the efficacy of adapter learning, [22] investigates different insertion forms, and [13,50,55] explores the potential of adapter compositions.

CL with Vision Transformer. Recent works adopt ViT as the backbone in the continual learning system to exploit its robust representational ability. Without any tuning, ViT can serve as a feature extractor for prototypes, which can be used for classification with distance measurement [46, 49, 79]. PEFT techniques are also widely used to adapt ViT to CL tasks, including adaptation and prompting. L2P [66], which first applies visual prompt tuning [31] in CL, and DualPrompt [65] uses fixed prompt pool(s) and learn the distribution of new tasks with incremental tuning. The prompt learning process is further improved by [60] with an attention mechanism and input-conditioned weights. Similar to prompting in CL, some works also explore the use of a fixed set of adapters [14, 18] or task-oriented expansion [78] for better transfer of ViT to downstream CL tasks. Furthermore, [20] builds a unified framework which allows incorporation of both prompting and adapter-based methods.

3 Methodology

3.1 Problem Definition

Continual learning constructs a scenario where the model is required to learn from sequentially arriving tasks [15]. Consider a sequence of T tasks $(\mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^T)$, where $\mathcal{D}^t = \{(x_i^t, y_i^t)\}_{i=1}^{n_t}$ is the dataset containing n_t data samples for t-th task. Only samples from $\mathcal{D}_{train}^t$ are accessible during the training of t-th task [63], if without additional experience replay process [9]. In class-incremental learning, the classes in different tasks are non-overlapping, specifically, with the label space of t-th task denoted by Y_t, $Y_t \cap Y_{t'} = \emptyset$ for $t \neq t'$. The goal is to learn a model f_θ that performs well on all seen tasks and classes, with the objective to minimize classification loss:

$$\mathcal{L}_{\text{CE}}^t = \mathbb{E}_{(x,y)\in D^t} \ell(f_\theta(x), y), \tag{1}$$

where $\ell(\cdot, \cdot)$ denotes the cross entropy loss.

3.2 Overview

We propose a modular framework (*i.e.*, SEMA) with a self-expansion mechanism for adding adapters at arbitrary layers for novel distribution in continual learning tasks, as shown in Fig. 1 and 2. To achieve modular learning process towards balanced transfer and forgetting [5, 48], we design the modular adapter as a pair of functional adapter $f(\cdot)$ and a representation descriptor $g(\cdot)$, where the representation descriptor capturing the distribution of the relevant data of the adapter at the corresponding network layers(see Sec. 3.3).

As shown in Fig. 1, SEMA expands the adapters only if a distribution shift that existing adapters cannot handle is observed (indicated by the expansion signal from the representation descriptor), which is processed at each layer. This active self-expansion process enables adding new adapters on-demand, and avoids addition of too many adapters with later pruning. Learned adapters are frozen and reusable, alleviating forgetting and engaging knowledge transfer. The newly added modular adapter $(f(\cdot), g(\cdot))$ is trained on the task that triggered the expansion. Activation weights of each adapter in the transformer block at the same layer is determined by a weighting router that can dynamically grow while encountering adapter expansion. A simple linear head is used for classification trained with cross entropy loss.

3.3 Formulation of Modular Adapter

Each adapter module consists of a *functional adapter* $f(\cdot)$ (as part of the forward process) coupled with a *representation descriptor* $g(\cdot)$ that captures the distribution of the task which triggers addition of its corresponding adapter (as a detector for novel knowledge and only used during training time). An adapter can be added at different layers based on the specification distribution shifts appearing on the representations.

The adapter $f(\cdot)$ is used to close the representation gap between the pre-trained model and the downstream task. We implement $f(\cdot)$ as a lightweight adapter [10] contains a down-projection layer with parameters $W_{\text{down}} \in \mathbb{R}^{d\times r}$, a non-linear activation ReLU [1], and a up-projection layer with parameters $W_{\text{up}} \in \mathbb{R}^{r\times d}$.

We insert the adapter in parallel to the MLP module inside each transformer block in ViT. Given the input of MLP module as x^{ℓ} (with a slight abuse of notation) in the Transformer block at ℓ-th layer, the output of one adapter module is formulated as:

$$f(x^{\ell}) = \text{ReLU}(x^{\ell} \cdot W_{\text{down}}) \cdot W_{\text{up}}. \tag{2}$$

In [79], a similar adapter formulation is used to conduct adaptive fine-tuning only on the first task data. The controlled adaptation can perform well for other tasks with similar distribution, relying on the strong generalization ability of the pre-trained models. Unlike it, we explore how to enable the model to learn new tasks continually.

Representation descriptor. In this work, we implement the representation descriptors as AEs [25] for modeling the characteristics of the data encountered by the adapter. The AE model $g_t^{\ell}(\cdot)$ added at layer ℓ triggered by task t consists of an encoder $p_{\theta_t^{\ell}}$ and a decoder $q_{\phi_t^{\ell}}$. It is trained on the input data corresponding to the task that triggered adapter expansion. Training of the representation descriptor at layer ℓ added at task t can be achieved by minimizing the reconstruction loss:

$$\mathcal{L}_{\text{RD}}(x^{\ell}, x^{\ell}_{\text{recon}}) = ||x^{\ell} - x^{\ell}_{\text{recon}}||^2. \tag{3}$$

The trained representation descriptor is used as an indicator of the data distribution that the corresponding adapter can handle. If new tasks/data contain significant distribution shifts, the features cannot be well-reconstructed by already trained descriptor, indicating new adapters are required. It can also identify the reusable adapters for the new data where the learned representation can be directly generalized on. We thus use the reconstruction error to generate the expansion signals.

3.4 Expandable Weighting Router for Mixture Usage of Adapters

For effectively identifying the proper adapters to use as well as mixing and reusing all learned adapters, we propose to learn an *expandable weighting router* jointly with the automatically added modules. Specifically, we employ a simple linear layer (similar to [17]) as the expandable weighting router to dynamically determine the weighting of outputs generated by each adapter. For the ℓ-th layer of the model (at arbitrary training step) with M^ℓ added modules/adapters, we define the routing function as $h^l(\cdot; W_\mathrm{r}) : \mathbb{R}^d \to \mathbb{R}^{M^\ell}$, where $W_\mathrm{r} \in \mathbb{R}^{d\times M^\ell}$ denotes the weight matrix of the router function. Given an input of adapter module (also the input of MLP module) as x^ℓ, the mixture weight of i-th adapter is computed with:

$$w_i^\ell = \mathrm{softmax}(h^\ell(x^\ell))_i. \tag{4}$$

The output x_o^ℓ of the transformer block at ℓ-th layer with M^ℓ adapter modules is obtained with:

$$x_o^\ell = \mathrm{MLP}(x^\ell) + \textstyle\sum_{i=1}^{M^\ell} w_i^\ell \cdot f_i(x^\ell), \tag{5}$$

where $f_i(x^\ell)$ is the output of i-th adapter. Note that the weight matrix of router W_r will be expanded simultaneously when expansion signal is triggered and new adapter is added to the network. To prevent the weighting router from catastrophic forgetting, we freeze the weight matrix and make the router only partially trainable with the newly added weight matrix, which corresponds to the computation of contribution of the new adapter. Although module-wise representation descriptors are used to indicate the representations encountered by the specific adapter, they are learned separately from the adapters and independently from each other. Thus, the responses of AE-based representation descriptors cannot be used easily for the mixture of adapters or to produce better results.

3.5 Self-Expansion Strategy

To avoid rewriting and forgetting the learned knowledge in a modularized network, the most straightforward approach is to keep the old modules frozen and add new modules to learn new knowledge (if required).

Task-oriented expansion. Considering parameter efficiency, we use the task identification available in training as prior knowledge in self-expansion. In the task-oriented expansion, at most one adapter will be added in each task (per layer) if the data triggers an expansion signal. Each layer of the ViT is equipped with one adapter at the initial step for the first task in CL. Then, automatic self-expansion is adopted in the subsequent tasks.

Multi-layer expansion. While each transformer block is initialized with one adapter module, tuning at different stages (layers) leads to the accommodation of different types of distribution shifts, and less tuning on some layers promotes inter-class knowledge sharing [19,38]. We thus let the model automatically decide whether to expand at arbitrary layers separately. In class-incremental learning, considering the distribution shifts mainly happen at the semantic level, we let the expansion focus on the last few transformer blocks.

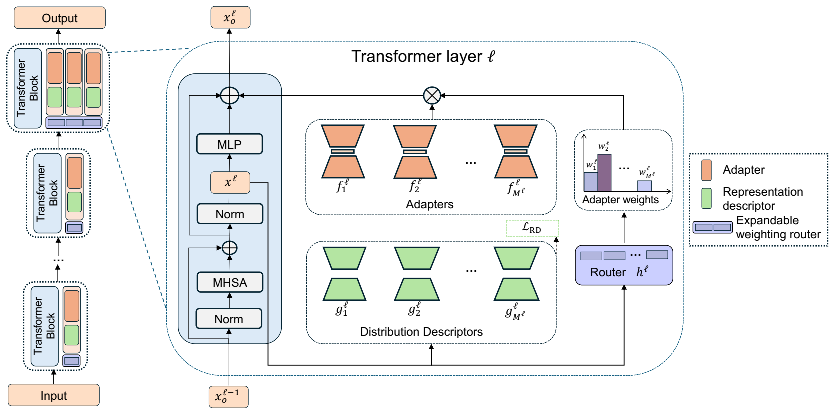


Fig. 2: Overview of the adaptation process. Representation descriptors estimate the distribution similarity between incoming features and previous task, and trigger expansion signals. The representation descriptors are trained to fit the feature distribution of the corresponding task via only $\mathcal{L}_{\text{RD}}$, without being influenced by the gradient back-propagated from the classification loss. MHSA denotes the multi-head self-attention module in the transformer block.

***z*-score based expansion signal.** Instead of inserting new adapters for every new task, we add adapters only when the distribution of incoming tasks is determined to be dissimilar from all past tasks to achieve parameter efficiency. For each representation descriptor, the running statistics of mean μ_{recon} and standard deviation σ_{recon} of reconstruction error is computed. The expansion signal is triggered if the statistical z-score of mean reconstruction error e_{recon} of current batch

$$z = \frac{e_{\text{recon}} - \mu_{\text{recon}}}{\sigma_{\text{recon}}} \tag{6}$$

is greater than the predefined expansion threshold z'. Once the expansion signal is received, a new adapter module will be inserted into the transformer block and start training. By default, we prioritize the addition of modules to transformer blocks closer to the input and only consider expansion at later blocks after the training of all newly inserted adapters is finished.

Training added adapter and expanded router. Once module expansion is performed, training of the newly added modules is executed. We decouple the training of the new adapter and representation descriptor with two separate stages, where the new adapter is trained with old frozen adapters, with weighting computed by the router, via optimizing classification loss and the representation descriptor is trained by optimizing $\mathcal{L}_{\text{RD}}$. After the training is finished, the new adapter, router and representation descriptors are frozen and the model starts checking for the expansion signal for the next transformer block. In this phase, no parameter updates are allowed. We provide the pseudo-code of the training phase in Algorithm 1.

Algorithm 1 Training of our model

Given Components: Pre-trained ViT f with B transformer blocks, sets of trained adapters $f = \{f^1, ..., f^B\}$, expandable weighting router h, number of training epochs M
Input: train data $\mathcal{D}^t_{train}$
Output: Updated model

```
 1: function TrainModule($\mathcal{D}^t_{train}, f_{new}, g_{new}, h, M$)
 2:     for e=1,...,$M$ do
 3:         for mini-batch $b$ in $\mathcal{D}^t_{train}$ do
 4:             Update $f_{new}, h$ with cross-entropy loss
 5:     for e=1,...,$M$ do
 6:         for mini-batch $b$ in $\mathcal{D}^t_{train}$ do
 7:             Update $g_{new}$ with Eq. (3)
 8:
 9: for sample batch in $\mathcal{D}^t_{train}$ do
10:     for $l$=0,...,$B$ do
11:         for each adapter in $l$-th transformer block do
12:             Compute $z$-score $z$ of reconstruction error using Eq. (6)
13:         if all $z$-scores exceeds expansion threshold then
14:             Add new adapter $f_{new}$ and representation descriptor $g_{new}$
15:             TrainModule($\mathcal{D}^t_{train}, f_{new}, g_{new}, h, M$)
16:             Freeze adapters, representation descriptors and router weight matrix
```

4 Experiments

4.1 Setting and Implementation Details

Datasets. The experiments are conducted on common datasets used for adaptation of pre-trained ViT in the conventional class-incremental learning, including CIFAR100 [36], ImageNet-R [23], ImageNet-A [24] and VTAB [76].
Baselines. We compare our framework against representative ViT-based fine-tuning methods, including fully fine-tuning of the adapter, L2P [66], Dual-Prompt [65], CODA-P [60], SimpleCIL [79] and ADAM with Adapter [79]. All baselines are originally designed for rehearsal-free continual learning and experimented without any memory buffer in this paper.
Training details. We adopt ViT-B/16 model [16] pre-trained on ImageNet-1K [56], which is a commonly used pre-trained weight, as the backbone for all baselines and our method. Discussions on other pre-trained weights will be offered in the supplementary materials. We train the adapters with 5 epochs, followed by the training of representation descriptors with 20 epochs, both with a batch size of 32. SGD is used as the optimizer with the initial learning rate set to 0.005 and 0.01 for adapters and representation descriptors, respectively, decaying with cosine annealing. For all datasets, module expansion is enabled only in the last 3 transformer blocks by default.

4.2 Experimental Results

We evaluate the performance of our method by comparing with several ViT-based fine-tuning works in continual learning. We use the average accuracy of

Table 1: Comparision with ViT-based fine-tuning methods in class-incremental learning. $\mathcal{A}_N$ denotes the average accuracy of all seen tasks after training on the last task and $\bar{\mathcal{A}}$ denotes the average performance during training on all tasks.

Method	CIFAR-100		ImageNet-R		ImageNet-A		VTAB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$
Finetune Adapter	47.88	30.9	38.51	24.22	29.78	17.64	59.98	43.5
L2P	84.77	77.87	70.67	62.90	47.16	38.48	81.19	80.83
DualPrompt	86.60	80.43	62.33	61.97	59.54	50.23	82.89	79.79
CODA-P	**91.55**	86.11	75.00	70.02	47.29	35.02	79.88	81.58
SimpleCIL	82.31	76.21	67.59	61.35	60.05	49.24	85.29	83.61
ADAM	90.55	85.62	75.84	69.10	60.15	49.24	85.29	83.61
SEMA	91.37	**86.98**	**81.75**	**74.53**	**64.53**	**53.32**	**91.26**	**89.64**

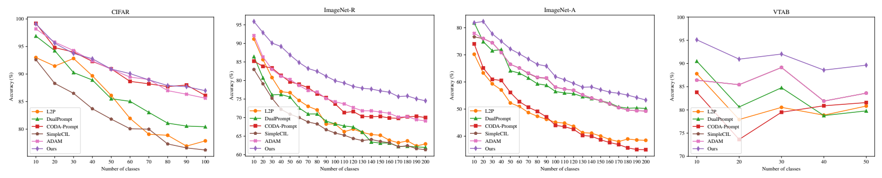


Fig. 3: Incremental performance of different methods on class-incremental learning benchmarks. All models adopt ViT-B/16-IN1K as the backbone.

all tasks [8] and average incremental accuracy [52] as our evaluation metrics, denoted as $\mathcal{A}_N$ and $\bar{\mathcal{A}}$ respectively. As shown in Tab. 1, our method significantly outperforms the recent state-of-the-art methods in terms of average accuracy, which is the most important metric in continual learning, on all datasets. We also achieve better average incremental accuracy in most cases. We further report the comparison of incremental accuracy in Fig. 3. Although approaches like CODA-P exhibit strong performance on CIFAR-100 and VTAB, their effectiveness might diminish on datasets containing adversarial samples similar to those found in ImageNet, due to its reliance on embeddings produced by ViT. Also, having only one adapter trained on the first task limits ADAM's ability to adapt to future tasks with intra/inter-task distribution shifts, as shown by ImageNet-R and VTAB. With dynamic expansion strategy, SEMA consistently outperforms other parameter-efficient fine-tuning methods in CL throughout the incremental training stages, especially on datasets which could not be well-handled by ViT (ImageNet-A) or with large inter-task distribution shifts (VTAB), demonstrating its capacity in accommodating to downstream tasks with various data distributions.

4.3 Ablation Studies and Analysis

Ablation studies on module expansion and adapter weighting. Tab. 2 thoroughly compares SEMA against non-expandable adaptation and its other

Table 2: Performance comparison of SEMA with non-expandable adaptation and other variants. No Expansion denotes the case where only one adapter is used in both training and inference and no adapter expansion is allowed. Average Weighting and Random Weighting replace the routing strategy with average or random weights. Random Selection and Top-1 Selection selects only one adapter in both training and inference, whilst Top-1 Sel. Inf. executes normal training as SEMA but inference with only the top-1 choice among the adapters.

Method	ImageNet-A		VTAB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$
SEMA	**64.53**	**53.32**	**91.26**	**89.64**
No Expansion	61.20	49.9	86.21	83.66
Average Weighting	56.88	44.31	90.84	89.14
Random Weighting	62.95	49.77	88.87	85.17
Random Selection	61.70	50.36	90.82	88.51
Top-1 Selection	62.00	50.56	90.83	88.61
Top-1 Sel. Inf.	61.96	50.36	90.95	88.84

variants. No Expansion is an equivalent version to ADAM, which uses only one adapter in each transformer layer, with slight difference in implementation details. SEMA significantly outperforms No Expansion, with the self-expansion strategy which allows the model to dynamically add new adapters to accommodate novel tasks with distribution shifts, alleviating the limitation of inadequate adaptation capacity provided by a single adapter.

Average Weighting, Random Weighting, Random Selection, Top-1 Selection are four variants of SEMA, where we remove the expandable weighting router during both training and inference time while keeping the dynamic expansion mechanism with representation descriptors for capturing distributional shifts in incoming tasks. Instead of weighting the contribution of adapters via the routing function, Average Weighting and Random Weighting assign average or randomly generated weight to each adapter respectively. Both methods benefit from additional parameters introduced by adapter expansion and outperforms No Expansion on VTAB. However, they fail to perform well on ImageNet-A with non-ideal adapter weights, showing that merely increasing the model parameters does not necessarily lead to performance gain. Random Selection and Top-1 Selection purely selects one of the adapter for training and inference, with random choice or the top-1 choice produced by the router, which do not mix the outputs of multiple adapters on the same layer. Top-1 Sel. Inf. executes the same routing strategy during training but only selects the top-1 adapter determined by the router. While these selection methods are able to obtain better performance than Random Weighting and No Expansion, SEMA outperforms them by training and inference with mixture of adapters weighted through the expandable router that allows SEMA to gain benefits from outputs of different adapters.

Analysis on dynamic expansion process. We visualize the dynamic expansion process of the last transformer layer on ImageNet-A dataset with the minimum value of z-scores computed with all existing representation descriptors. As shown in Fig. 4, the expansion signal is triggered for Task 2 with distribution shift estimated by z-score over the threshold, and a new adapter is added for training. With the training of the newly added representation descriptor, the z-score gradually decreases and falls below the expansion threshold. As the z-score stays below the threshold from Task 3 to Task 8, no adapter is added during this period until the expansion signal is triggered by Task 9. This showcases that SEMA refrains itself from performing adapter expansion for each incoming task and only adds adapters while significant distribution shifts have been detected by the representation descriptor. Hence, SEMA is an efficient adaptation methods that only requires expansion while necessary.

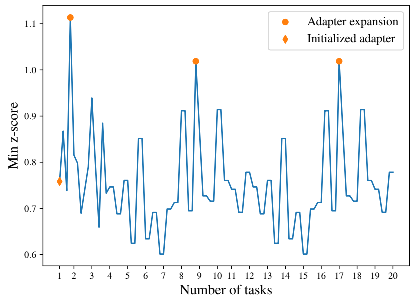
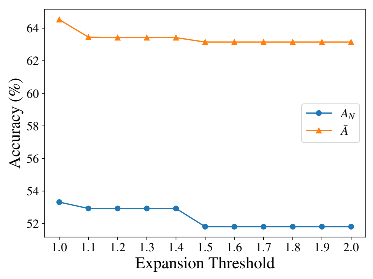


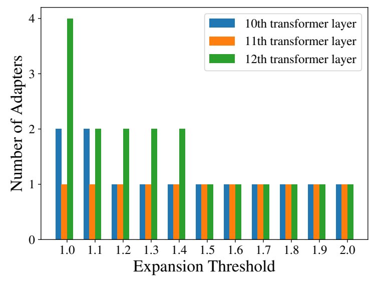
Fig. 4: Example of dynamic expansion process of ImageNet-A. The expansion threshold is 1 and expansion is performed for the 2nd, 9th and 17th task at the last layer.

Expansion threshold. In Fig. 5, we explore the effect of expansion threshold on the classification accuracy and number of adapters added to the network with ImageNet-A and VTAB. As shown in Fig. 5a and Fig. 5b, lower expansion thresholds enable more frequent adapter expansion which results in larger model capacity and potentially better performance. However, when the expansion threshold is over 1.5, it is unlikely for data in ImageNet-A to be considered as an outlier to previous task distribution hence expansion is not performed throughout the incremental training process. High expansion threshold minimizes the chance for expansion, which might lead to insufficient adaptation in some cases. Similarly, in Fig. 5d, with low expansion threshold such as 1 and 2, expansion is executed for every task at the last tranformer block layer, which also brings a little boost in performance (Fig. 5c). The experiments show that SEMA is robust in the choice of expansion threshold. It can perform well in a large range of settings for the hyperparameter. A proper expansion threshold in a wide range can lead to a balance between the performance gain and the parameter size.

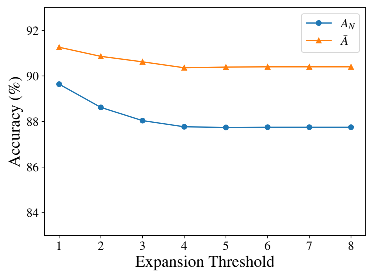
Analysis of multi-layer expansion. In Fig. 6, we explore the effects on accuracy by implementing expansion across varying numbers of the final transformer blocks, ranging from last 2 layers (#11-#12) to last 4 layers (#9-#12). It is intuitive that allowing dynamic expansion in deeper layers enables greater adaptation to different downstream tasks, resulting in higher performance. However, as shown in Fig. 6b and Fig. 6d, permitting expansion in early transformer layers also increases the overall number of added adapters, without significant boost in



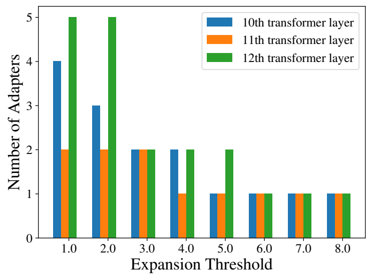
(a) Accuracy



(b) Num. of added adapters



(c) Accuracy



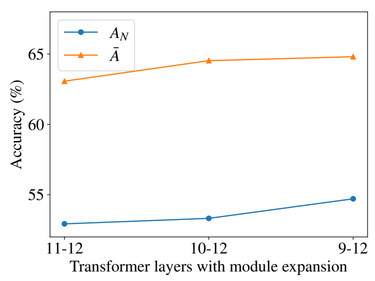
(d) Num. of added adapters

Fig. 5: Analysis of the impact of expansion threshold with (a)(b) ImageNet-A and (c)(d) VTAB. (a) and (c) show that SEMA can produce good accuracy stably with slight variation w.r.t. varying expansion threshold. (b) and (d) reports how the number of added adapters (on the specific Transformer layers #10, #11, #12) changes with the varying threshold value, corresponding to (a) and (c), respectively. It shows that the proposed method is insensitive to the threshold. While adding more adapters may lead to higher accuracy, a proper threshold can achieve a balance between the performance and model size.

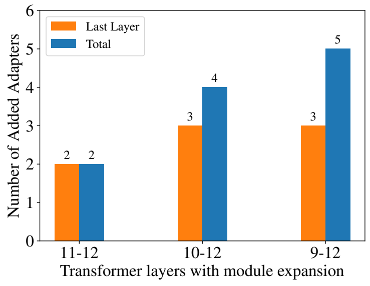
performance as earlier layers tend to behave in a similar manner despite distribution shifts. Also, enforcing addition of too many adapters may cause difficulty in training, especially in early transformer layers.

Ablation studies on adapter variants. To further validate the efficacy of our dynamic expansion mechanism, we extend our experiment to assess other variants of adapters. By default, we use AdaptFormer, a commonly used adaptation formed by two linear layers with one activation in between. We extend our evaluation to replace it with Convpass [32], a convolutional adapter designed specifically for visual tasks. As shown in Tab. 3, our proposed approach is robust to the selection of adapter methods, showing the broad applicability and effectiveness of our dynamic expansion strategy across different adapter methods.

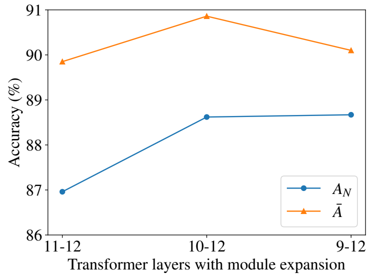
Hidden dimension in AE. We test different values for hidden dimensions in the AE as representation descriptors. The AE-based representation descriptors enable the capture of the characteristics of the data for decision-making on whether to add a new adapter during continual training. According to Fig. 7, the proposed method can perform well with a wide range of settings on AE's hidden dimension. We use 128 as the default setting in our implementation.



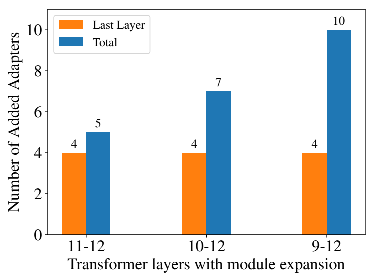
(a) Accuracy



(b) Num. of added adapters



(c) Accuracy

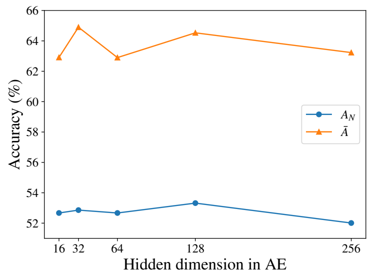


(d) Num. of added adapters

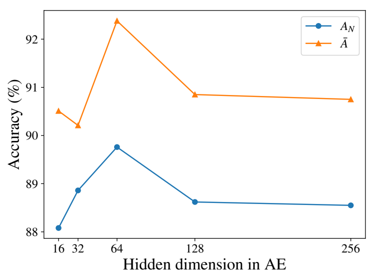
Fig. 6: Analysis of the effect of multi-layer expansion, with (a)(b) ImageNet-A and (c)(d) VTAB. By enabling automatic self-expansion on multiple Transformer layers, SEMA can achieve better performance than restricting that on a single layer.

Table 3: Assessment of different types of adapters. AdaptFormer is our default adapter whilst Convpass is a convolutional variant designed specifically for ViT. Both adapters achieve consistently good performance on ImageNet-A and VTAB with SEMA.

Method	ImageNet-A		VTAB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$
AdaptFormer	64.53	53.32	91.26	89.64
Convpass	63.48	51.74	90.68	88.62



(a) ImageNet-A



(b) VTAB

Fig. 7: Ablation on representation descriptor on (a) ImageNet-A and (b) VTAB.

5 Conclusion

In this paper, we propose a novel self-expandable modularized adaptation approach for continual learning. SEMA learns to reuse and add modules in an automated manner without memory replay. We incorporate an efficient expansion strategy with detection for feature distribution shifts in different layers of transformer-based models, successfully mitigating the forgetting problem of jointly using the fixed set of parameters. Experiment results demonstrate the outstanding performance of SEMA to datasets with different levels of distribution shifts. One limitation is that we perform expansion at most once per layer for each task for parameter efficiency concerns, which may constraint the flexibility of model expansion.

References

1. Agarap, A.F.: Deep learning using rectified linear units (relu). arxiv 2018. arXiv preprint arXiv:1803.08375 (1803) 6
2. Ahn, H., Cha, S., Lee, D., Moon, T.: Uncertainty-based continual learning with adaptive regularization. Advances in neural information processing systems **32** (2019) 4
3. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Proceedings of the European conference on computer vision (ECCV). pp. 139–154 (2018) 4
4. Aljundi, R., Chakravarty, P., Tuytelaars, T.: Expert gate: Lifelong learning with a network of experts. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3366–3375 (2017) 4
5. Andreas, J., Rohrbach, M., Darrell, T., Klein, D.: Neural module networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 39–48 (2016) 5
6. Ardywibowo, R., Huo, Z., Wang, Z., Mortazavi, B.J., Huang, S., Qian, X.: Varigrow: Variational architecture growing for task-agnostic continual learning based on bayesian novelty. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., Sabato, S. (eds.) International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA. Proceedings of Machine Learning Research, vol. 162, pp. 865–877. PMLR (2022), `https://proceedings.mlr.press/v162/ardywibowo22a.html` 2
7. Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S.: Dark experience for general continual learning: a strong, simple baseline. Advances in neural information processing systems **33**, 15920–15930 (2020) 2, 3
8. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with A-GEM. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), `https://openreview.net/forum?id=Hkf2_sC5FX` 2, 10
9. Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: On tiny episodic memories in continual learning. arXiv preprint arXiv:1902.10486 (2019) 3, 5
10. Chen, S., Ge, C., Tong, Z., Wang, J., Song, Y., Wang, J., Luo, P.: Adaptformer: Adapting vision transformers for scalable visual recognition. Advances in Neural Information Processing Systems **35**, 16664–16678 (2022) 4, 6

11. Chen, Z., Ding, M., Shen, Y., Zhan, W., Tomizuka, M., Learned-Miller, E., Gan, C.: An efficient general-purpose modular vision model via multi-task heterogeneous training. arXiv preprint arXiv:2306.17165 (2023) 2
12. Chenshen, W., Herranz, L., Xialei, L., et al.: Memory replay gans: Learning to generate images from new categories without forgetting [c]. In: The 32nd International Conference on Neural Information Processing Systems, Montréal, Canada. pp. 5966–5976 (2018) 3
13. Chronopoulou, A., Peters, M.E., Fraser, A., Dodge, J.: Adaptersoup: Weight averaging to improve generalization of pretrained language models. In: Vlachos, A., Augenstein, I. (eds.) Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia, May 2-6, 2023. pp. 2009–2018. Association for Computational Linguistics (2023). https://doi.org/10.18653/V1/2023.FINDINGS-EACL.153, https://doi.org/10.18653/v1/2023.findings-eacl.153 4
14. Cui, Y., Yu, Z., Cai, R., Wang, X., Kot, A.C., Liu, L.: Generalized few-shot continual learning with contrastive mixture of adapters. arXiv preprint arXiv:2302.05936 (2023) 2, 5
15. De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE transactions on pattern analysis and machine intelligence **44**(7), 3366–3385 (2021) 2, 3, 5
16. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021), https://openreview.net/forum?id=YicbFdNTTy 1, 9
17. Dou, S., Zhou, E., Liu, Y., Gao, S., Zhao, J., Shen, W., Zhou, Y., Xi, Z., Wang, X., Fan, X., et al.: Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. arXiv preprint arXiv:2312.09979 (2023) 4, 7
18. Ermis, B., Zappella, G., Wistuba, M., Rawal, A., Archambeau, C.: Memory efficient continual learning with transformers. Advances in Neural Information Processing Systems **35**, 10629–10642 (2022) 5
19. Gao, C., Chen, K., Rao, J., Sun, B., Liu, R., Peng, D., Zhang, Y., Guo, X., Yang, J., Subrahmanian, V.: Higher layers need more lora experts (2024) 7
20. Gao, Q., Zhao, C., Sun, Y., Xi, T., Zhang, G., Ghanem, B., Zhang, J.: A unified continual learning framework with general parameter-efficient tuning. arXiv preprint arXiv:2303.10070 (2023) 5
21. Gou, Y., Liu, Z., Chen, K., Hong, L., Xu, H., Li, A., Yeung, D.Y., Kwok, J.T., Zhang, Y.: Mixture of cluster-conditional lora experts for vision-language instruction tuning. arXiv preprint arXiv:2312.12379 (2023) 4
22. He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., Neubig, G.: Towards a unified view of parameter-efficient transfer learning. In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net (2022), https://openreview.net/forum?id=0RDcd5Axok 4
23. Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al.: The many faces of robustness: A critical analysis of out-of-distribution generalization. In: ICCV. pp. 8340–8349 (2021) 9
24. Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., Song, D.: Natural adversarial examples. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15262–15271 (2021) 9

25. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. science **313**(5786), 504–507 (2006) 3, 6
26. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Lifelong learning via progressive distillation and retrospection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 437–452 (2018) 4
27. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: International Conference on Machine Learning. pp. 2790–2799. PMLR (2019) 4
28. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021) 4
29. Hung, C.Y., Tu, C.H., Wu, C.E., Chen, C.H., Chan, Y.M., Chen, C.S.: Compacting, picking and growing for unforgetting continual learning. Advances in Neural Information Processing Systems **32** (2019) 4
30. Jha, S., Gong, D., Zhao, H., Yao, L.: NPCL: Neural processes for uncertainty-aware continual learning. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), https://openreview.net/forum?id=huh0XmSdBK 2
31. Jia, M., Tang, L., Chen, B.C., Cardie, C., Belongie, S., Hariharan, B., Lim, S.N.: Visual prompt tuning. In: European Conference on Computer Vision. pp. 709–727. Springer (2022) 4, 5
32. Jie, S., Deng, Z.H.: Convolutional bypasses are better vision transformer adapters. arXiv preprint arXiv:2207.07039 (2022) 4, 13
33. Ke, Z., Liu, B., Ma, N., Xu, H., Shu, L.: Achieving forgetting prevention and knowledge transfer in continual learning. Advances in Neural Information Processing Systems **34**, 22443–22456 (2021) 4
34. Kemker, R., Kanan, C.: Fearnet: Brain-inspired model for incremental learning. arXiv preprint arXiv:1711.10563 (2017) 3
35. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017) 2, 4
36. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto (2009) 9
37. Lee, K., Lee, K., Shin, J., Lee, H.: Overcoming catastrophic forgetting with unlabeled data in the wild. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 312–321 (2019) 4
38. Lee, Y., Chen, A.S., Tajwar, F., Kumar, A., Yao, H., Liang, P., Finn, C.: Surgical fine-tuning improves adaptation to distribution shifts. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net (2023), https://openreview.net/pdf?id=APuPRxjHvZ 7
39. Li, X.L., Liang, P.: Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190 (2021) 4
40. Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In: International Conference on Machine Learning. pp. 3925–3934. PMLR (2019) 4
41. Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on

Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 3925–3934. PMLR (2019), http://proceedings.mlr.press/v97/li19m.html 4
42. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017) 2, 4
43. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. Advances in neural information processing systems **30** (2017) 3
44. Mallya, A., Davis, D., Lazebnik, S.: Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In: Proceedings of the European conference on computer vision (ECCV). pp. 67–82 (2018) 4
45. McDonnell, M., Gong, D., Parvaneh, A., Abbasnejad, E., van den Hengel, A.: RanPAC: Random projections and pre-trained models for continual learning. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), https://openreview.net/forum?id=aec58UfBzA 2
46. Mi, L., Wang, H., Tian, Y., He, H., Shavit, N.N.: Training-free uncertainty estimation for dense regression: Sensitivity as a surrogate. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 10042–10050 (2022) 5
47. Nguyen, C.V., Li, Y., Bui, T.D., Turner, R.E.: Variational continual learning. arXiv preprint arXiv:1710.10628 (2017) 4
48. Ostapenko, O., Rodriguez, P., Caccia, M., Charlin, L.: Continual learning via local module composition. Advances in Neural Information Processing Systems **34**, 30298–30312 (2021) 2, 4, 5
49. Pelosin, F.: Simpler is better: off-the-shelf continual learning through pretrained backbones. arXiv preprint arXiv:2205.01586 (2022) 5
50. Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., Gurevych, I.: Adapterfusion: Non-destructive task composition for transfer learning. In: Merlo, P., Tiedemann, J., Tsarfaty, R. (eds.) Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021. pp. 487–503. Association for Computational Linguistics (2021). https://doi.org/10.18653/V1/2021.EACL-MAIN.39, https://doi.org/10.18653/v1/2021.eacl-main.39 4
51. Qin, C., Chen, C., Joty, S.: Lifelong sequence generation with dynamic module expansion and adaptation. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 6701–6714. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-main.414, https://aclanthology.org/2023.emnlp-main.414 2
52. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017) 2, 3, 10
53. Riemer, M., Klinger, T., Bouneffouf, D., Franceschini, M.: Scalable recollections for continual lifelong learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 1352–1359 (2019) 3
54. Rostami, M., Kolouri, S., Pilly, P.K.: Complementary learning for overcoming catastrophic forgetting using experience replay. arXiv preprint arXiv:1903.04566 (2019) 3
55. Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., Gurevych, I.: Adapterdrop: On the efficiency of adapters in transformers. In: Moens, M., Huang, X., Specia, L., Yih, S.W. (eds.) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event

/ Punta Cana, Dominican Republic, 7-11 November, 2021. pp. 7930–7946. Association for Computational Linguistics (2021). https://doi.org/10.18653/V1/2021.EMNLP-MAIN.626, https://doi.org/10.18653/v1/2021.emnlp-main.626 4
56. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**, 211–252 (2015) 9
57. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: International conference on machine learning. pp. 4548–4557. PMLR (2018) 4
58. Shen, Y., Zhang, Z., Cao, T., Tan, S., Chen, Z., Gan, C.: Moduleformer: Learning modular large language models from uncurated data. arXiv preprint arXiv:2306.04640 (2023) 2
59. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. Advances in neural information processing systems **30** (2017) 3
60. Smith, J.S., Karlinsky, L., Gutta, V., Cascante-Bonilla, P., Kim, D., Arbelle, A., Panda, R., Feris, R., Kira, Z.: Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11909–11919 (2023) 2, 5, 9, 24
61. Veniat, T., Denoyer, L., Ranzato, M.: Efficient continual learning with modular networks and task-driven priors. arXiv preprint arXiv:2012.12631 (2020) 2, 4
62. Wang, L., Zhang, X., Su, H., Zhu, J.: A comprehensive survey of continual learning: Theory, method and application. arXiv preprint arXiv:2302.00487 (2023) 2, 3
63. Wang, Y., Huang, Z., Hong, X.: S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. Advances in Neural Information Processing Systems **35**, 5682–5695 (2022) 5
64. Wang, Y., Agarwal, S., Mukherjee, S., Liu, X., Gao, J., Awadallah, A.H., Gao, J.: AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (eds.) Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. pp. 5744–5760. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Dec 2022). https://doi.org/10.18653/v1/2022.emnlp-main.388, https://aclanthology.org/2022.emnlp-main.388 4
65. Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.Y., Ren, X., Su, G., Perot, V., Dy, J., et al.: Dualprompt: Complementary prompting for rehearsal-free continual learning. In: European Conference on Computer Vision. pp. 631–648. Springer (2022) 2, 5, 9, 24
66. Wang, Z., Zhang, Z., Lee, C.Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., Pfister, T.: Learning to prompt for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 139–149 (2022) 2, 5, 9
67. Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., Farhadi, A.: Supermasks in superposition. Advances in Neural Information Processing Systems **33**, 15173–15184 (2020) 4
68. Wu, X., Huang, S., Wei, F.: MoLE: Mixture of loRA experts. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=uWvKBCYh4S 4
69. Yan, Q., Gong, D., Liu, Y., van den Hengel, A., Shi, J.Q.: Learning bayesian sparse networks with full experience replay for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 109–118 (2022) 2

70. Yan, S., Xie, J., He, X.: Der: Dynamically expandable representation for class incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3014–3023 (2021) 2, 4
71. Ye, F., Bors, A.G.: Task-free continual learning via online discrepancy distance learning. Advances in Neural Information Processing Systems **35**, 23675–23688 (2022) 4
72. Ye, F., Bors, A.G.: Self-evolved dynamic expansion model for task-free continual learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 22102–22112 (2023) 4
73. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 (2017) 4
74. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International conference on machine learning. pp. 3987–3995. PMLR (2017) 2, 4
75. Zeno, C., Golan, I., Hoffer, E., Soudry, D.: Task agnostic continual learning using online variational bayes. arXiv preprint arXiv:1803.10123 (2018) 4
76. Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A.S., Neumann, M., Dosovitskiy, A., et al.: A large-scale study of representation learning with the visual task adaptation benchmark. arXiv preprint arXiv:1910.04867 (2019) 9, 27
77. Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., Kuo, C.C.J.: Class-incremental learning via deep model consolidation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1131–1140 (2020) 4
78. Zhou, D.W., Sun, H.L., Ye, H.J., Zhan, D.C.: Expandable subspace ensemble for pre-trained model-based class-incremental learning. In: CVPR (2024) 5
79. Zhou, D.W., Ye, H.J., Zhan, D.C., Liu, Z.: Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need (Mar 2023), https://arxiv.org/abs/2303.07338v1 2, 5, 6, 9
80. Zhou, D.W., Zhang, Y., Ning, J., Ye, H.J., Zhan, D.C., Liu, Z.: Learning without forgetting for vision-language models (2023) 2

A More Details about SEMA

A.1 More Details of SEMA Training

At the start of the training, each transformer block at different layers is equipped with one adapter module containing one adapter and one representation descriptor, as well as an expandable weighting router. After the first task, for the incoming new tasks, SEMA monitors the representations of each batch of samples at each layer with the AE-based representation descriptor. New adapters are added if a significant enough representation/distribution shift is detected at each layer. Adding the adapters expands the model's representation ability for handling the new patterns. For simplicity, we allow the expansion at the task level, which means at most one time of expansion can happen for each task. As shown in Fig. 1 and Fig. 8, the detection and expansion operation starts from the transformer blocks closest to the input. Once a distribution shift that could not be handled by all adapters in a certain block is observed, the detection phase is paused, and an expansion signal is triggered in this block. A new adapter module will be added to the block where the expansion signal is triggered, with an expansion of the weighting router, and activated for training. After sufficient training, the detection phase will be restarted for later blocks. If no distribution shift is reported for a task in any transformer blocks, no adapter module will be added, and no training of adapters is required for this task. We provide a detailed example of the training and expansion process in Fig. 8.

A.2 More Discussions on the z-score-based Expansion Signal

In our self-expansion strategy, we use the statistical z-score of reconstruction error to determine whether the expansion signal should be triggered. z-score serves as a good outlier detector. A high absolute value of z-score reflects that the feature is greatly different from others in the feature space. The z-score is computed with

$$z = \frac{e_{\text{recon}} - \mu_{\text{recon}}}{\sigma_{\text{recon}}},$$

where e_{recon} is the mean reconstruction error of current batch and μ_{recon} and σ_{recon} are mean and standard deviation of past reconstruction errors respectively. We only evaluate z-score of large batches instead of evaluating on a per-sample basis since the latter approach easily leads to unstable training and expansion, due to the fact that there is a high chance encountering outlier samples with high z-score in a task. We prefer to add limited number of adapters according to the distribution shifts of tasks, instead of samples, for parameter efficiency concerns. Furthermore, μ_{recon} and σ_{recon} are calculated with reconstruction error of the last 500 samples during training as reconstruction error data collected in earlier training stages may not properly reflect the distribution of reconstruction error on the representation descriptor after complete training.

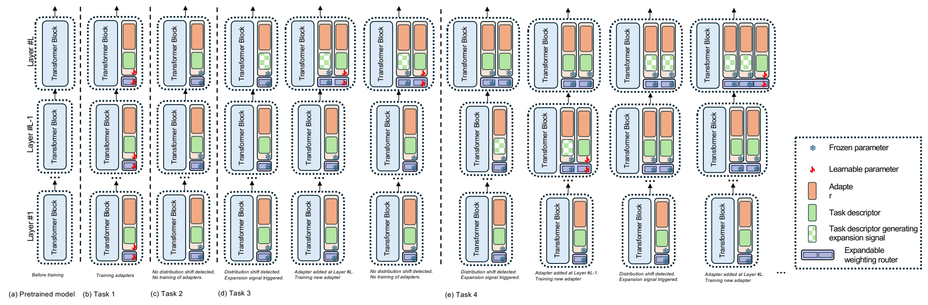

Layer #L
Layer #L-1
Layer #1
Transformer Block
Before training
Training adapters
No distribution shift detected. No training of adapters.
Distribution shift detected. Expansion signal triggered.
Adapter added at Layer #L. Training new adapter
No distribution shift detected. No training of adapters.
Distribution shift detected. Expansion signal triggered.
Adapter added at Layer #L-1. Training new adapter
Distribution shift detected. Expansion signal triggered.
Adapter added at Layer #L. Training new adapter
(a) Pretrained model
(b) Task 1
(c) Task 2
(d) Task 3
(e) Task 4
Frozen parameter
Learnable parameter
Adapte
r
Task descriptor
Task descriptor generating expansion signal
Expandable weighting router

Fig. 8: Detailed illustration of the training process. (a) The pretrained model with L transformer layers is provided for adaptation. (b) At the start of training, each transformer layer is equipped with one expandable weighting router and one adapter module, including one functional adapter and its paired representation descriptor. All modules are trainable at this stage. (c) All modules and routers are frozen after the training on Task 1. When Task 2 arrives, a detection of distribution shift is performed with all frozen representation descriptors in each transformer layer for all batches in Task 2. Since no distribution shift is observed, module addition is not performed and all modules are frozen. (d) As Task 3 arrives, the detection for distribution shift is executed again and distribution shift is observed in the L-th layer. Expansion signal is triggered and an adapter module is added in the L-th layer with expanded router. Training for the newly added adapter and router is performed. Since the addition is performed at the last transformer layer, no further detection for distribution shift is required. (e) When Task 4 arrives, expansion signal is triggered in the $L-1$-th layer during the detection phase. After sufficient training, the newly added module is frozen and detection for distribution shift in later layers is executed. When both representation descriptors in the L-th layer consider the incoming feature as an outlier, expansion signal will be triggered. A new module is added for training in the L-th layer while all other modules are frozen.

We have provided a detailed illustration of the dynamic expansion process with z-score in Fig. 4, which displays the movement of the minimum z-score across all adapters on the last transformer layer. As expansion is only executed if the incoming data distribution is not similar with any of the seen distributions in the past tasks, we compute the z-scores of all adapters and extract the minimum value. If the minimum z-score is still higher than the expansion threshold, then the incoming data is considered to be different from all seen distributions and expansion is deemed to be necessary. Once the expansion signal is triggered, the detection of distribution shift is immediately stopped, and the new adapter and representation descriptor are added for training. After training the representation descriptor, the reconstruction error of AE decreases and the z-score computed with the newly trained representation descriptor will be below the threshold. Hence, expansion signal will only be triggered again by future tasks with significant distribution shifts whose minimum z-score among all representation descriptors exceeds the expansion threshold.

B More Details about Implementation and Evaluation

B.1 Details of Datasets

CIFAR-100 contains 100 classes with 500 training samples and 100 testing samples per class.
ImageNet-R contains renditions of 200 ImageNet classes, which is a challenging CL benchmark introduced by with great intra-class diversity.
ImageNet-A contains real-world images filtered from ImageNet in an adversarial manner which are hard to be classified by models pre-trained with ImageNet.
VTAB consists of 50 classes from 5 domains with 10 classes from each domain.

To construct class-incremental setting, for results reported in Tab. 1, all datasets are splitted in a manner where each task consists of 10 distinct classes.

B.2 Implementations of Compared Methods

For SimpleCIL and ADAM, we use the official implementation at `https://github.com/zhoudw-zdw/RevisitingCIL`. For other prompting methods, namely L2P, DualPrompt and CODA-P, we adopt the open-source implementation from PILOT toolbox, available at `https://github.com/sun-hailong/LAMDA-PILOT`. In our experiments, we adhere to the hyperparameter configurations as specified in the original publications for each of the compared methods.

B.3 Details on Evaluation Metrics

Denote the accuracy of the i-th task after training on the N-th task as $\mathcal{A}_{i,N}$. The average accuracy $\mathcal{A}_N$ represents the average accuracy of all seen tasks after training on the N-th task:

$$\mathcal{A}_N = \frac{1}{N}\sum_{i=1}^{N}\mathcal{A}_{i,N},$$

which is often considered as the most important evaluation metric in continual learning.

The average incremental accuracy $\bar{\mathcal{A}}$ is the average accuracy along incremental stages, defined as:

$$\bar{\mathcal{A}} = \frac{1}{N}\sum_{t=1}^{N}\mathcal{A}_t.$$

Forgetting $\mathcal{F}_N$ measures the extent of catastrophic forgetting along incremental training stages, defined as:

$$\mathcal{F}_N = \frac{1}{N-1}\sum_{i=1}^{N-1} f_i^N,$$

where f_i^N represents the forgetting on the i-th task after training on the N-th task, defined as:

$$f_i^N = \max_{j\in\{1,..,N-1\}} \mathcal{A}_{i,j} - \mathcal{A}_{i,N}.$$

C More Experiment Results and Ablation Studies

C.1 Influence of Pre-trained Weights

In the main paper, we experiment SEMA and other methods with ViT-B/16-IN1K in Tab. 1. To study the influence of pre-trained weights, we further experiment SEMA with another commonly used pre-trained ViT weight, i.e., ViT-B/16-IN21K. We evaluate the performance using average accuracy $\mathcal{A}_N$ and average incremental accuracy $\bar{\mathcal{A}}$. As shown in Tab. 4, SEMA consistently outperforms prompting and adaptation methods in class-incremental learning. This indicates that our model is robust in performance regardless of different choices of pre-trained weights.

Table 4: Experiments on four class-incremental learning benchmarks with ViT-B/16-IN21K weight. $\mathcal{A}_N$ denotes the average accuracy of all seen tasks after training on the last task and $\bar{\mathcal{A}}$ denotes the average performance during training on all tasks.

Method	CIFAR-100		ImageNet-R		ImageNet-A		VTAB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$
L2P	89.51	85.02	74.49	65.82	46.67	39.30	79.17	63.56
DualPrompt	90.39	85.64	73.67	68.88	58.45	48.78	88.11	77.58
CODA-P	91.01	86.20	70.36	65.32	50.73	37.06	85.13	85.85
SimpleCIL	87.13	81.26	61.92	54.33	60.50	49.44	85.99	84.38
ADAM	92.18	87.47	75.08	67.30	60.53	49.57	85.95	84.35
SEMA	**92.23**	**87.84**	**77.84**	**69.60**	**62.50**	**51.35**	**91.99**	**90.86**

C.2 Further Analyses on the Effectiveness of Self-Expansion

The proposed method SEMA enables the model to add parameters and expand its capacity on demand. It allows the model to handle samples that could not be handled before by adding a small number of parameters. In continual learning, this process helps to alleviate forgetting by avoiding the interference from new patterns while encouraging knowledge reuse, comparing to some methods [60, 65] that add parameters sequentially by task. Unlike these methods, adding parameters *linearly* w.r.t. number of seen tasks/classes, SEMA adds parameters *sublinearly* at a very restricted rate. To further analyze the effectiveness of this self-expansion process, we conducted comparisons with other related methods and the "expansion-by-task" variant of SEMA, in this section.

Unlike previous prompt-tuning methods such as DualPrompt and CODA-P which incrementally expands prompts by task, SEMA efficiently performs expansion on a sub-linear basis only where distribution shifts occur. To evaluate the effectiveness of our expansion strategy, we provide a comparison with an "expansion-by-task‘ variant of SEMA which incrementally adding one adapter on

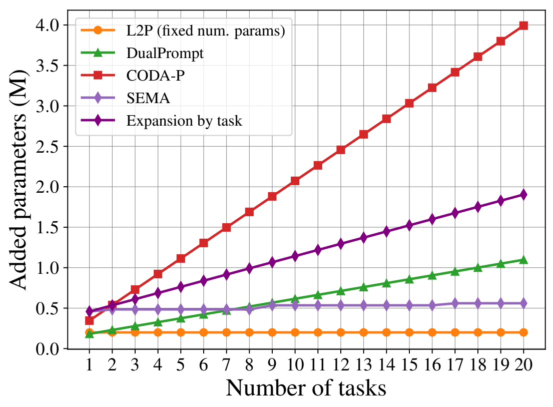


Fig. 9: Analysis on added parameters (in Millions) during model deployment on ImageNet-A. We compare with methods using fixed number of prompts like L2P, and methods like DualPrompt and CODA-P that incrementally expand like SEMA but with prompts and on a linear basis according to tasks. Expansion by task adds adapters for every incoming task, whilst SEMA executes expansion on demand, which increments parameters on a sub-linear basis. Specifically, SEMA added more parameters (with expansions at more layers) at Task 9 than other steps with expansion.

the layers that allow expansion for each incoming task in Tab. 5. Adding adapters on a per-task basis results in comparable performance as SEMA, however, the number of parameters added is significantly higher, especially while encountering longer task sequence as illustrated with ImageNet-R and ImageNet-A (20 tasks). Note that adding too many unnecessary adapters may cause task interference and result in increased difficulty in the training of expandable weighting routers, which leads to a slightly worse performance as shown in Tab. 5. Thus, we demonstrate that our expansion strategy is efficient in both controlling the size of added parameters regardless of the length of task sequence, encouraging knowledge reuse and reducing potential task interference in adapter weighting.

We additionally compare the size of added parameters with prompt-tuning continual learning methods in Tab. 6. While L2P uses a fixed size of prompt pool with small amount of added parameters, the fixed size of trainable parameters limit its capability to adapt to more distribution shifts in continual learning and comes with higher chance of forgetting. Compared to other methods which incrementally add prompts for each task, such as CODA-P and DualPrompt, SEMA involves much less added parameters during deployment (representation descriptors are only used for expansion in training and are discarded in deployment) while achieving better performance, demonstrating the efficiency of our dynamic expansion strategy which only adds parameters while necessary. We further illustrates the advantage in parameter controlling with sublinearly adding parameters on demand in Fig. 9.

Table 5: Comparison of added parameters used in model deployment and average accuracy with different expansion strategies. Expansion by Task adds one adapter for every new task. SEMA only expands if a distribution shift is detected by the representation descriptor.

Dataset	Expansion by Task		SEMA	
	Params (M)	$\mathcal{A}_N$	Params (M)	$\mathcal{A}_N$
CIFAR-100	1.066	86.86	0.645	86.98
ImageNet-R	1.904	74.08	0.617	74.53
ImageNet-A	1.904	52.80	0.560	53.32
VTAB	0.647	89.09	0.554	89.64

Table 6: Number of added parameters used in model deployment compared with other prompt-tuning continual learning methods, measured in Millions. L2P uses a fixed size of prompts. DualPrompt and CODA-P adds prompts sequentially by task. SEMA adds the smallest number of parameters, comparing to methods which allows expansion in training, with its dynamic expansion strategy.

Type	Method	CIFAR-100		ImageNet-R		ImageNet-A		VTAB	
		Params (M)	$\mathcal{A}_N$	Params (M)	$\mathcal{A}_N$	Params (M)	$\mathcal{A}_N$	Params (M)	$\mathcal{A}_N$
Fixed Param Size	L2P	0.123	77.87	0.200	62.90	0.200	38.48	0.085	80.83
Expandable Param Size	DualPrompt	1.022	80.43	1.098	61.97	1.098	50.23	0.983	79.79
	CODA-P	3.917	86.11	3.994	70.02	3.994	35.02	3.878	81.58
	SEMA	**0.645**	**86.98**	**0.617**	**74.53**	**0.560**	**53.32**	**0.554**	**89.64**

C.3 Further Discussions on the Weighting Router

Routing relying on representation descriptor. In SEMA, we use the representation descriptor to detect novel patterns that only trigger the expansion signal in a discrete way, while the weighting router is used to mix the adapters. Although the weighting router can naturally encourage more reuse and combination of knowledge from the learned adapters, the AE-based representation descriptors can also provide the identity information of each sample w.r.t. the adapters. We thus study the potential of only using the AE-based representation descriptors and the corresponding reconstruction error for routing. Since the representation descriptors are not directly involved in the training of adapters with cross-entropy loss, there exists a gap between the objective of training representation descriptors and obtaining the ideal mixture pattern for classification. As shown in Tab. 7, while routing with the reconstruction error of representation descriptors achieves sound performance on most datasets due to the good representation ability of AEs and their reconstruction errors, it is still outperformed by SEMA which learns in a more optimal manner with expandable weighting routers.

Analyzing adapter usage with a controlled special case. We use a special case under control to show how the learned adapters and weighting router can be used to handle different tasks. Considering that multiple facts can influence the adapter usage of different samples in a complex way, we conducted the controlled

Table 7: Comparison between routing with expandable weighting router and representation descriptor (RD).

Method	CIFAR-100		ImageNet-R		ImageNet-A		VTAB	
	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$
SEMA	**91.37**	**86.98**	**81.75**	**74.53**	**64.53**	**53.32**	**91.26**	**89.64**
Routing with RD	90.91	83.61	81.02	74.13	61.80	50.36	90.83	88.53

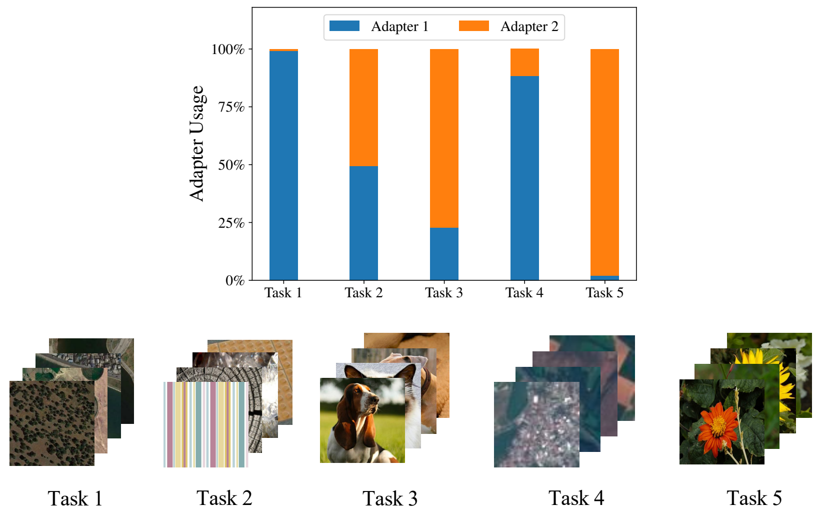


Fig. 10: Adapter usage with a controlled special case on VTAB. We limit the expansion of adapters and only add two adapters for clear and simplified visualization. We report the usage proportion of each adapter with the highest weighting per task, in the last transformer layer. Below, we provide visual illustrations of sample images from each VTAB task.

experiments by only allowing adding two adapters for the Transformer block at the last layer. VTAB dataset [76] is used considering its visibly diverse data distributions. In this way, it can be clearly visualized how the different adapters may be added and used by different samples in Fig. 10. With two adapters, Adapter 1 and Adapter 2 are trained with Task 1 and Task 2 respectively. Due to the similarity in data distribution between Task 1 and Task 4, Task 4 largely employs Adapter 1 during inference. Tasks 3 and 5, which comprise natural images, exhibit a preference for Adapter 2, instead of Adapter 1 which mainly contains satelite knowledge. For Task 2 that consists of texture image data, both adapters can be used to handle this kind of low-level data. Hence, we show that our expandable weighting router can effectively choose optimal adapters according to different data distributions.

Table 8: Per-image inference time of each method measured in milliseconds.

Method	Inference Time (ms)			
	CIFAR-100	ImageNet-R	ImageNet-A	VTAB
L2P	9.44	9.53	9.86	9.46
DualPrompt	9.44	9.51	9.84	9.44
CODA-P	9.45	9.47	9.85	9.43
ADAM	9.95	10.03	10.36	9.45
SEMA	**4.48**	**7.39**	**9.01**	**7.38**

C.4 Results on Inference Time

We evaluate the inference efficiency and report the average inference time of each image measured in milliseconds in Tab. 8. We show that SEMA is an efficient adaptation approach in continual learning as it significantly outperforms other methods on all datasets.

The inference latenty of the listed prompting continual learning methods is caused by the extra procedure of processing the image with a frozen pre-trained model for the query function. Similarly, ADAM requires extra feature extraction with a frozen pre-trained model for the concatenation of pre-trained features and adapted features. SEMA relieves the dependency on frozen pre-trained model as we focus on the intermediate feature distribution of each transformer block. The inference time of SEMA is related to the number of adapter modules added into the ViT, which can be regulated by adjusting the expansion threshold.

Table 9: Average accuracy(%) at each incremental stage on 10-task ImageNet-R.

Method	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10
L2P	86.36	66.99	65.54	66.81	64.25	64.66	63.64	64.88	63.86	62.72
DualPrompt	84.03	78.35	75.87	74.02	71.76	70.45	69.44	69.23	68.35	66.75
CODA-P	90.71	88.27	84.12	82.31	79.87	78.74	77.67	77.69	76.41	75.25
SimpleCIL	79.10	72.22	70.01	68.29	65.83	64.36	64.10	63.22	62.42	61.35
ADAM	91.87	84.94	82.36	80.02	77.76	76.46	75.61	74.97	73.99	73.15
SEMA	**93.61**	**90.08**	**86.97**	**84.71**	**82.58**	**81.26**	**80.23**	**79.57**	**78.68**	**78.00**

C.5 Additional Results on 10-Task Setting

Apart from Tab. 1 which reports ImageNet-R and ImageNet-A with 20-task setting, we conduct further experiments on 10-task setting where each task contains 20 classes. We report the average accuracy $\mathcal{A}_N$ at each incremental stage

Table 10: Average accuracy(%) at each incremental stage on 10-task ImageNet-A.

Method	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10
L2P	70.29	59.44	55.46	53.72	49.35	50.77	49.06	48.48	45.81	45.56
DualPrompt	77.71	71.94	66.39	62.89	57.91	57.74	56.20	53.53	51.47	51.42
CODA-P	70.86	70.00	62.82	61.46	57.31	56.51	53.52	51.52	49.53	49.11
SimpleCIL	76.00	70.83	65.13	61.60	58.03	56.92	54.06	51.84	49.68	49.24
ADAM	76.57	70.83	65.13	61.75	58.26	57.03	54.15	52.00	49.75	49.37
SEMA	**82.29**	**76.94**	**73.11**	**68.77**	**65.76**	**65.23**	**62.89**	**61.30**	**59.50**	**58.46**

in Tab. 9 and Tab. 10. SEMA outperforms all other methods in all incremental stages, which demonstrates that our method is competitive regardless of the length of tasks in continual learning.