Slim Adaptation Modules: A Simple yet Strong Baseline for Continual Foundation Models

Elif Ceren Gok Yildirim
TU Eindhoven
e.c.gok@tue.nl

Murat Onur Yildirim
TU Eindhoven
m.o.yildirim@tue.n

Joaquin Vanschoren
TU Eindhoven
j.vancshoren@tue.nl

Abstract

Continual adaptation is essential for maintaining the relevance of foundation models (FMs) as data and tasks evolve. While recent research in class-incremental learning (CIL) has primarily focused on adapting pre-trained large-scale transformers, it remains unclear how well these methods actually perform compared to lightweight convolutional networks. Without such comparisons, it is difficult to know whether recent advances truly surpass strong ConvNet-based baselines. To address this gap, we propose Slim Adaptation Modules (SAM) which keeps the majority of the pre-trained backbone frozen while enabling rapid and scalable adaptation through sparse task-specific layers. SAM achieves up to a ~5× reduction in trainable parameters and a ~6× reduction in total parameters, substantially lowering the cost of continual updates. Extensive experiments across diverse benchmarks demonstrate that this simple yet strong design not only mitigates catastrophic forgetting but also consistently outperforms state-of-the-art CIL methods, including those based on foundation models. This positions SAM as a simple and strong baseline for guiding future FM-based CIL research.

1 Introduction

Recent research in CIL has increasingly adopted FMs, with a particular focus on Vision Transformers (ViTs) [1], which are known to have robust generalization capabilities and serve as a powerful base for incremental learning [2–5]. However, sequential fine-tuning of these models often alters their original representations, leading to substantial catastrophic forgetting [6–9]. To alleviate this, parameter-efficient fine-tuning (PEFT) methods, such as prompt-based and adapter-based approaches, have been introduced [3, 4, 10, 11] for ViTs. These methods freeze the FMs and restrict updates to a subset of parameters by introducing task-specific modules, aiming to mitigate catastrophic forgetting.

However, existing approaches face several critical limitations. First, their reliance on large-scale ViT architectures (e.g., 86M parameters) makes deployment in real-world scenarios challenging due to substantial computational and memory demands. Second, the task-specific modules introduced by these methods remain parameter-inefficient; for example, state-of-the-art prompt- and adapter-based techniques require approximately ~3M and ~1M additional parameters per task, respectively. Third, and most importantly, it remains unclear how these FM-based methods compare against strong ConvNets, leaving open the question of whether they truly provide superior performance.

To address this, we propose a novel PEFT-like approach that leverages pretrained ResNets [12] to significantly reduce both the total and trainable parameters compared to existing techniques [3, 10, 11, 13]. Specifically, we freeze all layers except for the final feature extraction layer to leverage the transferable general features across tasks. Then, for each new task, we instantiate a dedicated final ResNet layer as a task-specific module, enabling efficient specialization in task-specific adaptation. To minimize both the total parameter count and the number of trainable parameters, we apply structured sparsity on these modules, referring to them as 'Slim Adaptation Modules', or shortly SAM.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

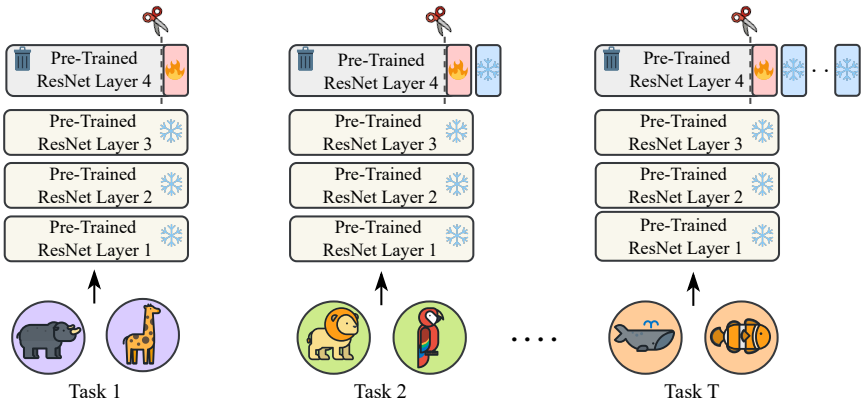


Figure 1: SAM freezes the first three layers of a pre-trained ResNet to preserve general knowledge while dynamically adding a task-specific last layer for each new task. To improve parameter efficiency, each last layer is structurally pruned to become 'slim' before training on its corresponding task. After training, the weights are frozen to prevent forgetting.

Our contributions are three-fold:

I. We introduce Slim Adaptation Modules (SAM), a PEFT-inspired method specifically for ConvNets, providing an alternative to existing approaches that primarily target FMs.

II. SAM achieves a 2–5$\times$ reduction in trainable parameters and a 2–6$\times$ reduction in total parameters compared to existing PEFT methods, enabling more efficient continual learning.

III. Across multiple benchmarks, SAM consistently outperforms adapter- and prompt-based methods, establishing itself as a simple yet strong baseline to FM-based CIL research.

2 Related Work

FM-based CIL has recently gained prominence in research by leveraging large-scale pre-trained models for efficient adaptation to incremental tasks [14–16]. Recent works focus on PEFT methods which preserve frozen FM weights while integrating lightweight modules like prompts or adapters. L2P [11] adapts a technique from natural language processing by introducing a learnable prompt pool, where instance-specific prompts are selected via a key-query matching mechanism to guide the response of pre-trained models. DualPrompt [10] extends L2P by incorporating G-Prompt and E-Prompt, designed to capture task-invariant and task-specific information, respectively. CODA-Prompt [3] employs contrastive loss to decorrelate prompt representations, mitigating interference, and integrates them using an attention-based weighting mechanism. APER [9] systematically explores various PEFT methods, including adapters, and demonstrates that prototypical classifiers serve as a strong baseline. EASE [13] enhances PTMs by attaching adapters to each layer which creates expandable subspaces and then aggregates feature representations from all adapter sets. Although these recent advancements have improved the performance, they are posing challenges in efficiency, and it remains unclear how these FM-based methods compare against pre-trained ConvNets.

3 Method

Architecture Overview. We build our method on the ResNets [12], and break the model down into three core components: a shared extractor Φ, task-specific modules γ, and a unified classifier $W^\top$. We keep the shared pre-trained extractor $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ frozen throughout all learning sessions and serves as a generic task-invariant feature representation, allowing faster adaptation. Specifically, it corresponds to the initial three residual layers of a ResNet backbone. For each task $\mathcal{D}_b$, a task-specific embedding module $\gamma_b : \mathbb{R}^d \to \mathbb{R}^d$ is appended to the shared frozen extractor Φ. The unified classifier $W^\top : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{Y}_b|}$ maps the final task-specific embeddings to class logits. Then, the overall model can be represented as given in Eq 1, where $\hat{y}_i$ produces the predictions with an activation function σ for a given input $\mathbf{x_i}$ from task b.

$$\hat{y}_i = \arg\max \sigma(W^\top \gamma_b(\Phi(\mathbf{x}_i))) \tag{1}$$

Slim Adaptation Modules. We adopt a structured pruning strategy early in training to enhance computational efficiency. Specifically, after the first epoch, we identify the most salient channels within each task-specific embedding module γ and prune the least important ones. Formally, for a given convolutional layer with weight tensor $W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times K \times K}$, the saliency of each output channel c is measured using its L_1-norm as $s_c = \sum |W_c^i|$ and those with the lowest L_1-norm values are pruned until the predefined sparsity level p is reached. This approach substantially reduces the number of learnable parameters, ensuring that training updates remain confined to a minimal yet highly informative subset of weights. As a result, it alleviates the computational burden while preserving sufficient expressive capacity for effective adaptation. Following pruning, our architectural notation and flow are slightly modified. The shared pre-trained extractor $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ remains unchanged, as does the unified classifier $W^\top : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{Y}_b|}$. However, the task-specific module γ_b is now replaced with a slim adaptation module $\mathcal{S}_b$. The resulting architecture is then expressed as Eq. 2.

$$\hat{y_i} = \arg\max \sigma(W^\top \mathcal{S}_b(\Phi(\mathbf{x}_i))). \tag{2}$$

Training Protocol. Instead of modifying the entire network, we update only the parameters of the slim task-specific module γ and the shared classifier $W^\top$, enabling efficient adaptation with minimal interference between tasks. The parameters of these components are optimized using the standard cross-entropy loss, as defined in Eq. 3.

$$\ell_{CE} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) = -\sum_{i=1}^{N} y_i \log \sigma(W^\top \mathcal{S}_b(\Phi(\mathbf{x}_i))) \tag{3}$$

Inference Protocol. The model must select the most appropriate module $\mathcal{S}$ for a given test batch $\mathbf{x}_{test}$ since it does not have access to task IDs. Therefore, we employ a confidence-based selection strategy. First, the $\mathbf{x}_{test}$ is processed by the frozen feature extractor Φ, returning initial feature representations $\Phi(\mathbf{x}_{test})$. These representations are then passed through each slim adaptation module $\mathcal{S}_b$ and the shared classifier $W^\top$, yielding task-conditioned class probability distributions as $p_b(x_{test}) = \sigma(W^\top \mathcal{S}_b(\Phi(\mathbf{x}_{test})))$. Then, we use it to capture the certainty of each module, where one with the highest confidence is selected for inference as formulated in Eq. 4.

$$\hat{b} = \arg\max_b \frac{1}{|\mathbf{x}_{test}|} \sum_{x_i \in \mathbf{x}_{test}} \max_{y \in \mathcal{Y}_b} p_b(y \mid x_i) \tag{4}$$

4 Experimental Setup

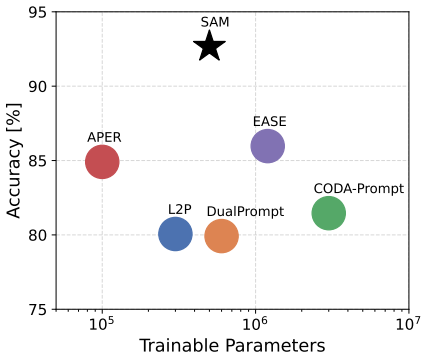
Datasets and Setting. We include two standard benchmarks CIFAR-100 and ImageNet-R as well as two fine-grained datasets CUB-200 and Cars-196. Specifically, CIFAR-100 [17] comprises 100 natural image classes with 500 training images per class, while ImageNet-R [18] contains 200 classes with 24,000 training images and 6,000 test images. In addition, CUB-200 [19] consists of 200 bird species, with approximately 60 images per class (equally divided between training and testing), and Cars-196 [20] includes 196 car models, with 8,144 training images and 8,040 test images. We adopt the 'B-m Inc-n' notation to describe the class split, where m denotes the number of classes in the initial stage and n indicates the number of classes introduced at each incremental stage. These diverse datasets allow us to robustly evaluate our method across standard class-incremental learning scenarios and fine-grained classification tasks.

Comparison Methods. We benchmark our approach against state-of-the-art PTM-based class-incremental learning methods built on the ViT architecture. Specifically, we compare with SimpleCIL [21], L2P [11], DualPrompt [10], CODA-Prompt [3], APER [21], and EASE [13]. Additionally, we include sequential finetuning as a baseline to assess the impact of continual learning strategies.

Evaluation Metrics. Following the benchmark protocol [13], we denote the model's accuracy after the $b-$th incremental stage as $\mathcal{A}_b$. In our evaluation, we report both the final accuracy $\mathcal{A}_B$ (i.e., the performance after the last stage) and the average accuracy across all stages, defined as $\bar{\mathcal{A}} = \frac{1}{B}\sum_{b=1}^{B} A_b$, where B is the total number of incremental stages.

Table 1: Average and final accuracy [%] with existing methods using **ViT-B/16-IN21K** and SAM using **ResNets**.

Method	CIFAR B0 Inc5		CUB B0 Inc10		IN-R B0 Inc20		Cars B0 Inc10	
	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$	$\bar{\mathcal{A}}$	$\mathcal{A}_B$
Finetune	60.65 ± 5.6	48.12 ± 3.3	55.78 ± 2.8	33.13 ± 3.3	59.09 ± 3.7	49.46 ± 3.3	41.90 ± 1.0	19.47 ± 2.7
SimpleCIL	86.48 ± 0.8	81.28 ± 0.1	91.58 ± 1.3	86.73 ± 0.1	61.31 ± 0.4	54.55 ± 0.1	54.95 ± 0.8	35.43 ± 0.0
L2P	84.90 ± 1.2	80.06 ± 1.4	73.22 ± 1.8	61.55 ± 1.7	75.92 ± 0.7	70.88 ± 0.7	42.06 ± 2.0	30.07 ± 0.8
DualPrompt	85.61 ± 1.3	79.92 ± 0.4	81.36 ± 1.8	70.51 ± 1.1	71.48 ± 0.5	66.09 ± 1.3	45.30 ± 1.1	30.15 ± 0.9
CODA-Prompt	87.64 ± 0.4	81.46 ± 0.3	77.65 ± 1.0	68.44 ± 1.0	76.25 ± 0.3	71.39 ± 0.3	36.22 ± 0.6	25.44 ± 0.3
APER-Adapter	89.57 ± 0.9	84.91 ± 0.2	91.62 ± 1.2	86.72 ± 0.2	74.81 ± 0.8	66.97 ± 0.8	47.91 ± 0.8	35.49 ± 0.0
EASE	90.79 ± 0.8	85.97 ± 0.6	**92.51** ± 1.3	86.49 ± 1.2	**80.35** ± 1.0	75.74 ± 0.8	49.32 ± 1.0	34.75 ± 0.3
SAM (RN18)	**91.40** ± 2.1	**88.51** ± 3.4	87.40 ± 1.3	83.69 ± 3.4	68.54 ± 0.4	65.76 ± 0.4	**79.09** ± 1.6	**64.82** ± 1.6
SAM (RN50)	**93.06** ± 1.5	**92.50** ± 2.1	85.40 ± 3.0	82.67 ± 3.0	73.22 ± 0.6	72.83 ± 0.5	**77.41** ± 1.5	**62.23** ± 8.2
SAM (RN101)	**94.16** ± 1.5	**93.05** ± 1.7	89.76 ± 1.1	**87.26** ± 1.7	77.75 ± 0.7	**77.03** ± 0.8	**80.16** ± 2.1	**77.30** ± 2.6
SAM (RN152)	**94.17** ± 1.4	**93.79** ± 1.7	89.91 ± 1.4	**88.35** ± 1.5	79.33 ± 1.0	**78.95** ± 0.5	**83.10** ± 0.9	**77.23** ± 3.5



Method	Trainable Parameters	Total Parameters	Accuracy [%]
L2P	6 M	92 M	80.06 ± 1.1
DualPrompt	12 M	98 M	79.92 ± 0.4
CODA-Prompt	60 M	146 M	81.46 ± 0.3
APER	300 K	86 M	84.91 ± 0.2
EASE	24 M	110 M	85.97 ± 0.6
SAM (RN18)	12 M	15 M	88.51 ± 3.4
SAM (RN50)	12 M	21 M	92.50 ± 2.1
SAM (RN101)	12 M	40 M	93.05 ± 1.7
SAM (RN152)	12 M	56 M	93.79 ± 1.7

Figure 2: Parameter size vs. accuracy: The left panel shows the relationship between parameter size and accuracy, while the right panel presents numerical results for different PEFT methods.

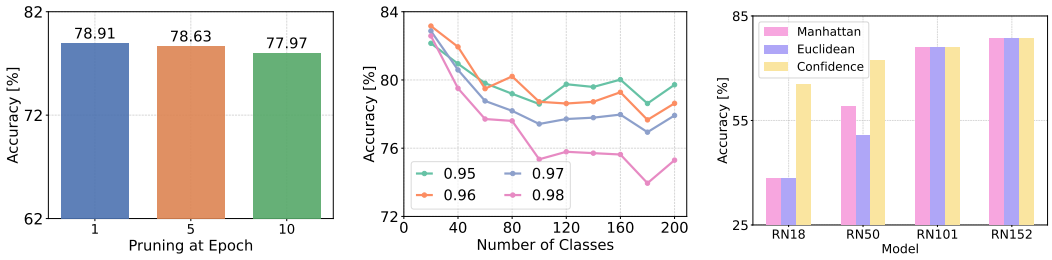
Implementation Details. All experiments were conducted on an NVIDIA A100 GPU using PyTorch [22] and the Pilot framework [23]. While existing methods utilize the pre-trained **ViT-B/16-IN1K** model which initially trained on ImageNet-21K and subsequently fine-tuned on ImageNet-1K, we employ pre-trained **ResNet18**, **ResNet50**, **ResNet101** and **ResNet152** models that are trained solely on ImageNet-1K. For our method, SAM, we train the models for 25 epochs using the Adam optimizer with a batch size of 48 and a learning rate of 0.001. We perform the pruning immediately after the first epoch, enforcing a pruning magnitude of 96%, and continue training with the slim module for the remaining epochs. We use the default parameters of existing approaches and repeat experiments five times with different random seeds. We report the mean and standard deviation of the performance metrics.

5 Results

In this section, we present a benchmark comparison of our method in terms of both performance and parameter efficiency. We then provide an ablation study followed by an in-depth analysis and discussion of our findings. Please see our Appendix for more results and discussion.

5.1 State-of-art Comparison

Incremental Performance. Table 1 presents the average and final accuracy of various continual learning methods across four benchmarks, demonstrating the effectiveness of SAM. On simpler datasets like CIFAR100, even a small ResNet18 backbone achieves competitive performance. For more challenging datasets such as CUB and ImageNet-R, larger ResNets further improve results while still remaining smaller than ViT-B/16-IN21K used by other methods. These results establish SAM as a simple yet strong baseline for future FM-based continual learning, consistently outperforming existing approaches and providing a parameter-efficient, robust strategy for scalable continual adaptation. Please see our Appendix for more experimental results.



(a) Pruning schedule (b) Pruning magnitude (c) Inference time module selection

Figure 3: Ablations of different components for the SAM method. (a) Effect of pruning timing on the performance. (b) Impact of different sparsity levels on performance. (c) Comparison of task-specific adaptation module selection strategies during inference.

Parameter Size Comparison. Figure 2 compares FM-based CIL methods on the CIFAR B0 Inc5 benchmark in terms of accuracy and parameter efficiency. SAM uses $5\times$ and $2\times$ fewer trainable parameters than the state-of-the-art prompt-based CODA-Prompt and adapter-based EASE, respectively, while delivering superior performance. Considering total parameters, including the frozen pre-trained backbone and task-specific modules, SAM requires 2–$6\times$ fewer parameters than these baselines. These results show that ResNet models can perform just as well, suggesting that current FM-based CIL methods are not fully leveraging their potential.

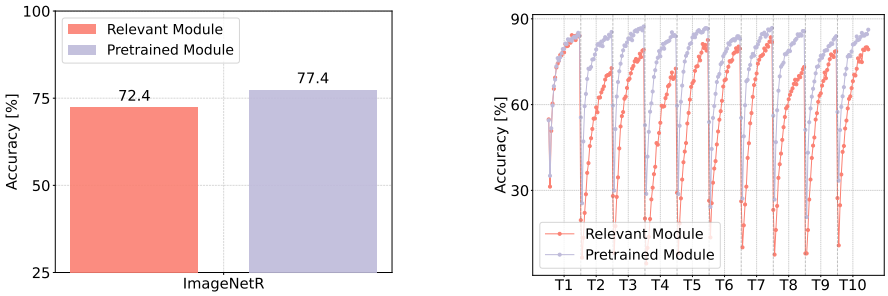
5.2 Ablation Study

To systematically evaluate the impact of core design choices for SAM, we conduct ablations across three critical dimensions: pruning schedule, pruning magnitude, and slim adaptation module selection strategies during inference. Furthermore, we also evaluate knowledge transfer between modules to see if 'warm-starting' is beneficial.

Pruning schedule. We investigate the impact of *'when to prune'* and evaluate three scenarios where pruning is performed after the 1^{st}, 5^{th}, and 10^{th} training epoch, respectively. As shown in Figure 3a, these experiments reveal the sensitivity of the model's performance to the timing of pruning, thereby informing the optimal schedule. Our observations indicate that applying pruning to the task-specific module in the early stages of training is more beneficial than applying it in later epochs, thereby motivating us to implement module pruning at epoch 1.

Pruning magnitude. The pruning magnitude applied to the task-specific module is another critical factor. In this experiment, we vary the pruning magnitude applied to the task-specific module γ_b across four settings: 0.95, 0.96, 0.97, and 0.98. Figure 3b illustrates how the incremental performance changes as a function of the pruning level, providing insights into the trade-off between parameter reduction and predictive accuracy. Our analysis shows that introducing an extreme pruning level harms the performance and therefore we opt for an optimal pruning magnitude of 0.96.

Module selection during inference. Finally, we explore three strategies; two distance-based and one confidence-based for selecting the appropriate slim adaptation module $\mathcal{S}_b$ at inference. In particular, the distance-based approaches, which compute the distance between the centroid of the test batch derived from the frozen backbone Φ and the stored centroids of previously encountered tasks, perform effectively on larger backbones ResNet101 and ResNet152. However, they fail to generalize well on smaller backbones ResNet18 and ResNet50. Consequently, we propose a refined, confidence-based approach that determines the correct slim adaptation module $\mathcal{S}_b$ by leveraging the maximum softmax probability over the test batch. As shown in Figure 3c, this confidence-based method consistently outperforms the distance-based strategies across all model sizes, ensuring more robust module selection at inference. Together, these findings offer empirical guidelines and highlight the efficiency of our design choices, and they provide a clear understanding of how each component contributes to the overall performance of our continual learning approach SAM.



(a) Final incremental accuracy on test set

(b) Per task training accuracy

Figure 4: Analysis of different initialization strategies for slim adaptation modules where (a) shows the final incremental accuracy on the test set after all tasks are completed, (b) demonstrates per-task training accuracy, showing how different initialization strategies affect the model's ability to adapt.

Knowledge Transfer between the modules. Beyond our main experiments, we also conduct an additional investigation into the knowledge transfer dynamics within our SAM approach, as this aspect warrants further discussion. Specifically, we examine two different strategies for initializing newly added slim adaptation modules: (i) initializing from the most similar previously learned task (Relevant Module), and (ii) using pre-trained weights (Pretrained Module). As illustrated in Figure 4a, the "Pretrained Module" approach achieves a final accuracy of 77.4% on ImageNet-R, outperforming the 72.4% obtained by the "Relevant Module." To understand this behavior we investigate the learning curves across all tasks in Figure 4b that shows the "Pretrained Module" strategy maintains a higher training accuracy. We hypothesize that pre-trained weights offer broader, more general feature representations, enabling more effective adaptation to new tasks. Reusing existing modules from other related tasks limits the model's ability to adapt to new task effectively, as their weights and feature spaces are already specialized for previous tasks. This makes it harder to align their representations with the current task compared to leveraging pre-trained general features, which offer a more flexible and transferable foundation. These findings emphasize the importance of selecting an appropriately general initialization strategy to promote stronger incremental learning performance.

5.3 Discussion and Further Analysis

In our experiments, we evaluate various ResNet models and observe consistently strong final incremental accuracy, indicating a robust balance between retaining previously learned knowledge and adapting to new tasks. This is particularly noteworthy given that existing methods rely on much larger Vision Transformer backbones to reach high performance. Despite this strength in final accuracy, SAM occasionally lags slightly in average accuracy compared to FM-based approaches. We attribute this difference to representational capacity where larger architectures tend to learn richer features. Specifically, SAM utilizes a ResNet backbone with fewer pre-trained parameters (3M-48M), whereas existing methods leverage larger pre-trained models (86M) that offer richer representational capacity. Indeed, when we scale up our model from pre-trained ResNet18 (3M) to pre-trained ResNet152 (48M), we observe improvements in both final and average accuracy. Overall, these findings illustrate that while large capacity backbones are more effective for adaptation, current FM-based approaches are falling short from that perspective since smaller ResNets can still offer competitive accuracy while being more efficient.

6 Conclusion

In this work, we introduced Slim Adaptation Modules (SAM), a PEFT-inspired approach designed for pre-trained model-based class-incremental learning in replay-free settings. Unlike recent methods that rely on large-scale foundational models, SAM leverages pre-trained ConvNets, particularly ResNets, showing that smaller architectures can match or or even surpass the performance of FM-based CIL methods while substantially reducing resource demands. Overall, SAM serves as a simple yet strong baseline for future FM-based continual learning research, highlighting that existing approaches may not be fully exploiting the powerful generalization capabilities of foundation models. Future work can explore other continual learning scenarios and adaptive pruning mechanisms that dynamically adjust based on task complexity and dataset characteristics.

Broader Impact

This paper presents work whose goal is to advance the field of machine learning, especially on the subject of FM-based CIL. Besides the advancements in the field, it reduces both total and trainable parameter count, thereby diminishing memory, computation, and scalability concerns.

Acknowledgements

This work is supported by; TAILOR, a project funded by the EU Horizon programme under GA No. 952215; SYNERGIES, a project funded by the EU Horizon programme under GA No. 101146542; Dutch national e-infrastructure with the support of SURF Cooperative using grant no. EINF-4568, and Turkish MoNE scholarship.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[2] Weiguo Pian, Shentong Mo, Yunhui Guo, and Yapeng Tian. Audio-visual class-incremental learning. In *ICCV*, 2023.

[3] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *CVPR*, 2023.

[4] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In *NeurIPS*, 2022.

[5] Andrés Villa, Juan León Alcázar, Motasem Alfarra, Kumail Alhamoud, Julio Hurtado, Fabian Caba Heilbron, Alvaro Soto, and Bernard Ghanem. Pivot: Prompting for video continual learning. In *CVPR*, 2023.

[6] Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *NeurIPS*, 2024.

[7] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *ICVV*, 2023.

[8] Aristeidis Panos, Yuriko Kobe, Daniel Olmeda Reino, Rahaf Aljundi, and Richard E Turner. First session adaptation: A strong replay-free baseline for class-incremental learning. In *ICVV*, 2023.

[9] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *IJCV*, 2024.

[10] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, 2022.

[11] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, 2022.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[13] Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *CVPR*, 2024.

[14] Mark D McDonnell, Dong Gong, Ehsan Abbasnejad, and Anton van den Hengel. Premonition: Using generative models to preempt future data changes in continual learning. *arXiv preprint arXiv:2403.07356*, 2024.

[15] Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. 2024.

[16] Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. *arXiv preprint arXiv:2401.16386*, 2024.

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 2021.

[19] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[20] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV*, 2013.

[21] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 2024.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[23] Hai-Long Sun, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Pilot: A pre-trained model-based continual learning toolbox. *arXiv preprint arXiv:2309.07117*, 2023.

[24] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 2017.

A Appendix

A.1 Adaptive Approach for Initializing SAM

In our original implementation, a new slim adaptation module $\mathcal{S}_b$ was added for each incoming task. In this section, we try a more adaptive and efficient strategy that reuses existing slim adaptation modules when a new task exhibits high similarity to previously encountered ones. This design yields a significantly more compact model by initializing fewer adaptation modules.

To quantify task similarity, we compute a task centroid c_b by averaging the feature representations extracted from the shared frozen extractor Φ (i.e., the first three residual layers of a pre-trained ResNet) overall training samples in $\mathcal{D}_b$, assuming that the mean feature representation effectively captures the overall data distribution. We then measure the similarity between c_b and the centroids of all previously encountered tasks $\{c_1, \ldots, c_{b-1}\}$, using the Manhattan distance as in Eq 5.

$$d(\mathcal{D}_b, \mathcal{D}_i) \simeq \|c_b - c_i\|_1, \quad c_b = \frac{1}{n_b} \sum_{i=1}^{n_b} \Phi(x_i) \tag{5}$$

We then compute the average distance across all previously encountered tasks as in Eq. 6 and then we scale it by a hyperparameter β which is a pre-determined hyperparameter to obtain a similarity threshold τ given in Eq. 7.

$$\bar{d} = \frac{1}{b} \sum_{b=1}^{b} d(\mathcal{D}_b, \mathcal{D}_i). \tag{6}$$

$$\tau = \beta \cdot \bar{d}. \tag{7}$$

If the minimum distance $d_{\min} = \min_{1 \leq i < b} \|c_b - c_i\|_1$ between the c_b and any previous centroid c_i is less than τ, then the new task is deemed sufficiently similar to a previously encountered task. In such cases, rather than allocating a new slim adaptation module, we reuse the module associated with the most similar task. To mitigate the risk of catastrophic forgetting when reusing a module, we incorporate a knowledge distillation loss during the training, similar to LwF [24].

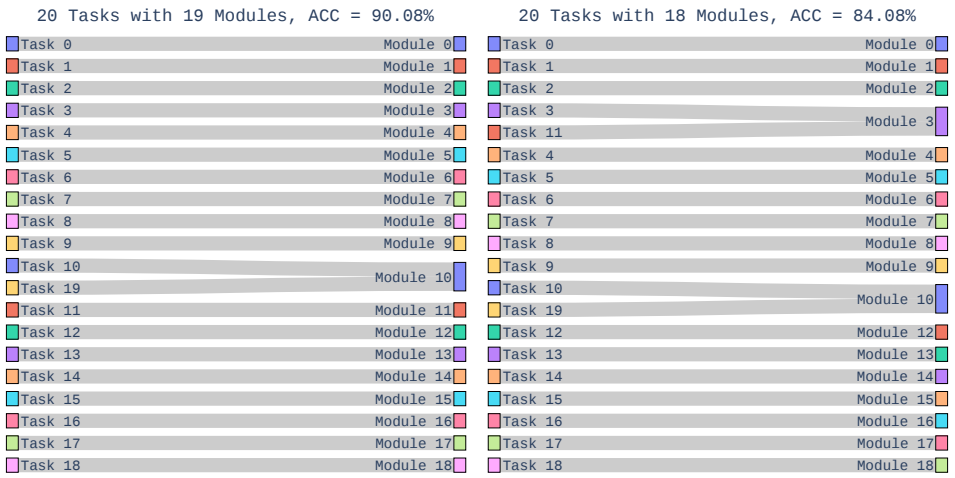
The results presented in Figure C highlight a fundamental tradeoff between continual learning performance and parameter efficiency. By selectively reusing existing slim adaptation modules, our approach reduces the total number of modules but this comes at the cost of diminished incremental accuracy.

For instance, when the threshold hyperparameter β is set to 0.70, Module 10 is shared across Task 10 and Task 19 and yields 90.08% accuracy. On the other hand, when β is set to 0.73 it leads to more reusing across tasks where Module 3 is used for Task 3 and Task 11 and Module 10 for Task 10 and Task 19 with 76.16% accuracy. Although it results lower in accuracy, we argue that developing such an adaptive mechanism is crucial for scalable continual learning. Our method SAM provides a flexible framework for task reuse, making it straightforward to integrate into various applications.

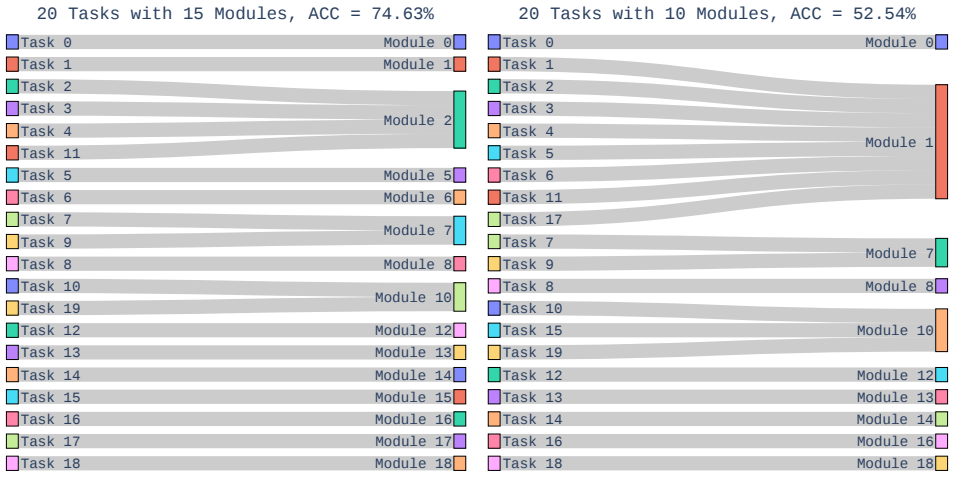
A.2 More Results on Inference

In this section, we provide additional analysis for the inference. Specifically, we analyze how many times the correct slim adaptation module is selected during the test time. To investigate this, we conduct an experiment on CIFAR B0 Inc5 setup, examining the consistency of slim adaptation module selection during inference.

In Figure Da, the matrix illustrates how frequently the correct slim adaptation module is chosen for a given task. Each row corresponds to a ground truth module identifier, and each column indicates the module selected by the model. The strong diagonal alignment reveals that the model predominantly selects the correct module for a given task except for a few batches. In other words, the model exhibits a high degree of task recognition and employs the corresponding slim adaptation module with minimal confusion across tasks.



(a) β=0.70 (b) β=0.73



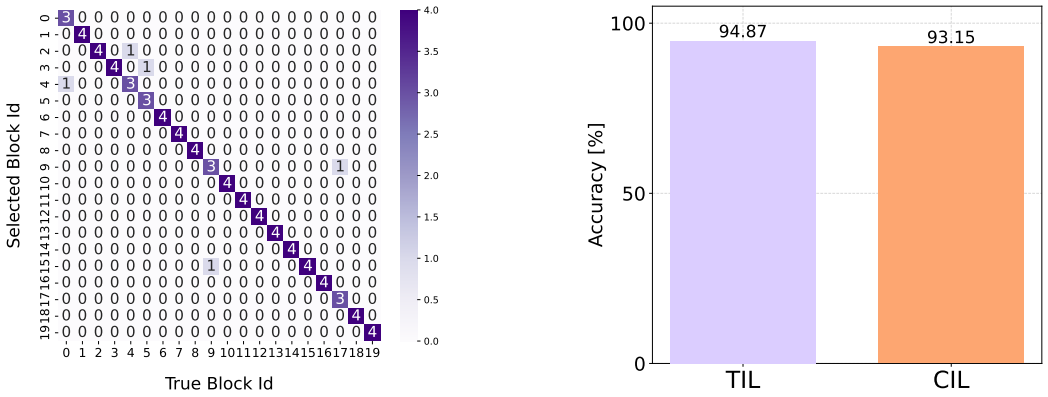
(c) β=0.75 (d) β=0.77

Figure C: Analysis of the effect of threshold values on the adaptive initialization of slim adaptation modules. As the threshold increases from (a) 0.70 to (d) 0.77, the model reuses existing modules more frequently, thereby reducing the total number of newly instantiated modules.

To further investigate the impact of module selection on performance, we conduct experiments in a task-incremental learning (TIL) setting, where the correct slim adaptation module is explicitly assigned for each test batch. As shown in Figure Db, the accuracy difference between CIL and TIL scenarios is minimal and negligible. This finding reinforces SAM's effectiveness in CIL settings, demonstrating its ability to achieve correct module-task alignment and maintain overall system performance.

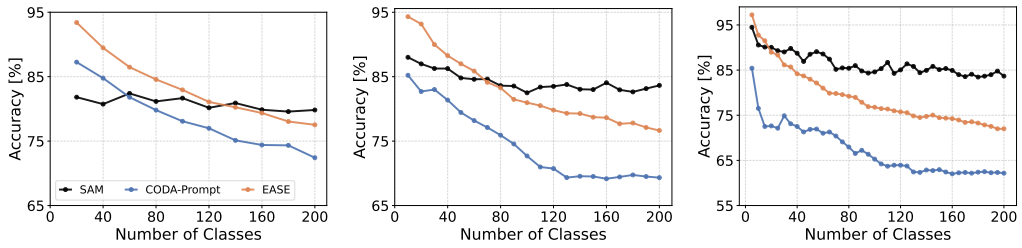
A.3 More Comparison on ImageNet-R

We conduct an additional analysis to assess the effect of the SAM approach over longer learning sessions. To that end, we run experiments on ImageNet-R and compare SAM with two strong baselines; CODA-Prompt (prompt-based) and EASE (adapter-based). We observe that, across all three setups, SAM demonstrates more stable performance as the number of classes increases. In particular, while EASE often starts with slightly higher accuracy in the beginning, its performance declines more sharply than SAM. CODA-Prompt shows the steepest drop overall. These results highlight the robustness of SAM in maintaining competitive accuracy in longer scenarios.



(a) Heatmap for module selection during inference (b) Incremental accuracy for TIL and CIL

Figure D: Inference analysis for SAM where (a) SAM mostly selects the correct adaptation module, making it well-suited for CIL scenarios and (b) SAM's performance when true the task identifiers are provided.



(a) 10 Tasks (b) 20 Tasks (c) 40 Tasks

Figure E: Results on IN-R across (a) 10 tasks, (b) 20 tasks, and (c) 40 tasks. SAM maintains a more stable performance than both EASE and CODA-Prompt, highlighting its robustness under long scenarios.

A.4 Details of Compared Methods

- **Finetune:** This baseline directly fine-tunes the pre-trained ViT backbone on each new task using the standard cross-entropy loss. While simple, it typically suffers from severe catastrophic forgetting as the entire model is updated without any constraints.
- **L2P [11]:** L2P introduces pre-trained ViT into continual learning by freezing the backbone weights and using visual prompt tuning to capture features of new tasks. It constructs instance-specific prompts through a prompt pool organized via key-value mapping.
- **DualPrompt [10]:** As an extension of L2P, DualPrompt refines the prompt mechanism by employing two types of prompts: general and expert prompts, while maintaining the instance-specific prompt construction process.
- **CODA-Prompt [3]:** To overcome limitations in instance-specific prompt selection, CODA-Prompt replaces the selection process with an attention-based prompt recombination strategy, effectively eliminating the need for explicit prompt selection.
- **SimpleCIL [21]:** This method utilizes a vanilla pre-trained ViT model as initialization and builds a prototype-based classifier for each class, employing a cosine classifier for final prediction.
- **ADAM [9]:** Extending SimpleCIL, ADAM aggregates the pre-trained and adapted models by treating the first incremental stage as the sole adaptation phase. This design unifies generalizability and task-specific adaptation within a single framework.
- **EASE [13]:** This method trains a distinct, lightweight adapter for each new task, thereby constructing task-specific subspaces. These subspaces enable joint decision-making while preserving prior knowledge, and a semantic-guided prototype complement strategy is employed to update old class features without requiring access to previous instances.