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

Elif Ceren Gok Yildirim[1], Murat Onur Yildirim[1], Joaquin Vanschoren[1]
[1]AMOR/e Lab, Eindhoven University of Technology
e.c.gok@tue.nl, m.o.yildirim@tue.nl, j.vanschoren@tue.nl

The continual learning literature has rapidly shifted from traditional class-incremental learning (CIL) techniques to foundation model (FM)-based CIL methods without a clear understanding of how these newer approaches compare to strong, lightweight convolutional baselines. This abrupt transition has created a substantial methodological gap, making it difficult to assess whether recent FM-based CIL progress reflects genuine advances or merely the absence of rigorous baselines. To address this gap, we introduce *Pruned Adaptation Modules* (PAM), a simple yet effective method that freezes the vast majority of the pre-trained ResNet while enabling scalable continual adaptation through sparse task-specific layers. PAM yields up to a ~5× reduction in trainable parameters and a ~6× reduction in total parameters, significantly reducing the cost of continual updates. Across diverse benchmarks, PAM consistently mitigates catastrophic forgetting and outperforms state-of-the-art FM-based CIL approaches. Our findings position PAM as a strong and transparent baseline that helps bridge the gap between traditional and FM-based CIL, guiding future research for a more accurate assessment of true progress in continual adaptation.

1. Introduction

Class-incremental learning (CIL) aims to enable models to acquire new knowledge over time without forgetting previously learned tasks. Recent work in this area has increasingly adopted foundation models (FMs), particularly Vision Transformers (ViTs) [1], due to their strong generalization properties and their demonstrated effectiveness as bases for incremental learning [2, 3].

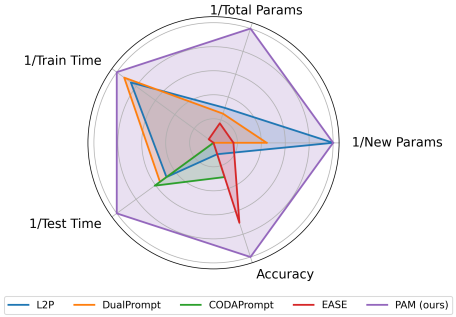


Figure 1: PAM is a simple yet powerful bridge that challenges the progress in FM–based CIL. It achieves better accuracy with ResNets, which significantly reduces runtime and parameters.

However, sequential fine-tuning of these models disrupts their pretrained representations, leading to pronounced catastrophic forgetting [4–7]. To mitigate this, parameter-efficient fine-tuning (PEFT) strategies, such as prompt-based [8–11] and adapter-based methods [12–14], freeze the FM and introduce small task-specific modules. These techniques restrict updates to a limited set of parameters, thereby enabling stabilized adaptation.

Despite their success, three key limitations remain. First, FM-based CIL methods typically rely on large-scale ViT backbones (e.g., 86M parameters), making training, storage, and deployment expensive in practical settings. Second, although designed to be lightweight, their task-specific modules are still relatively parameter-heavy; state-of-the-art prompt and adapter approaches require roughly ~3M and ~1M parameters per task, respectively. Third and most critically, the field has transitioned too quickly from traditional ConvNet-based CIL to FM-based CIL. As a result, it remains unclear whether these newer approaches truly outperform strong convolutional baselines. This rapid paradigm shift has created a methodological gap, obscuring the community's ability to measure actual progress in continual learning with FMs.

Third Conference on Parsimony and Learning (CPAL 2026).

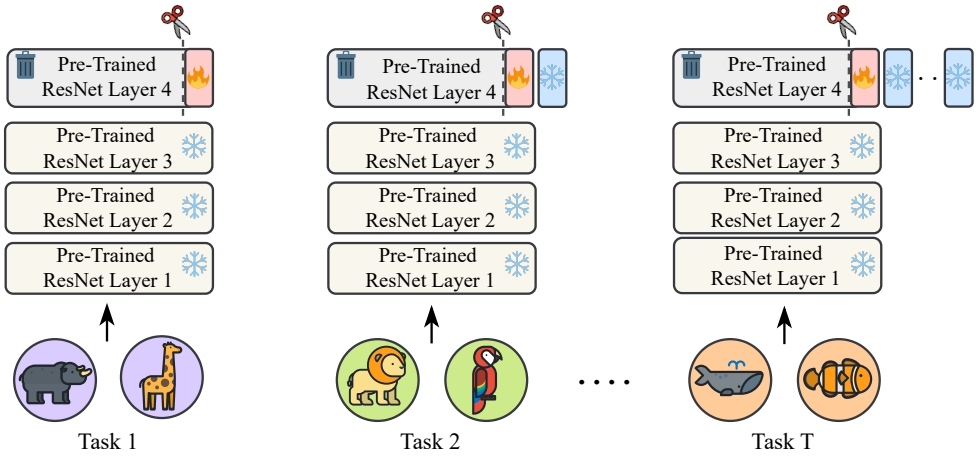


Figure 2: PAM freezes the first three layers of a pre-trained ResNet to preserve general knowledge while dynamically adding a task-specific last layer for each new task. To improve parameter efficiency, each last layer is structurally pruned to become 'slim' before training on its corresponding task. After training, the weights are frozen to prevent forgetting.

To address this, we propose a novel PEFT-like approach that leverages pretrained ResNets [15] to significantly reduce both the total and trainable parameters compared to existing FM-based techniques. Specifically, we freeze all layers except for the last one to leverage the transferable general features across tasks. Then, for each new task, we instantiate a dedicated last 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 'Pruned Adaptation Modules', or shortly PAM.

By incorporating these scalable design choices, we demonstrate a strong and principled bridging baseline that anchors future FM-based CIL research to a clear point of comparison, enabling the community to more reliably measure and advance true progress in continual learning.

Our contributions are three-fold:

I. We introduce 'Pruned Adaptation Modules' (PAM), a PEFT-inspired design for ConvNets, particularly ResNets, offering an alternative to existing FM-based approaches.

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

III. Across multiple benchmarks, PAM consistently outperforms adapter- and prompt-based methods, establishing itself as a simple yet strong baseline for evaluating and guiding future FM-based CIL research.

2. Related Work

Traditional CIL. Traditional approaches seek to learn new classes sequentially from scratch while simultaneously mitigating catastrophic forgetting of previously acquired knowledge [16–18]. Existing approaches include rehearsal-based methods, which retain or synthesize exemplars from past tasks to balance data distributions during updates [18–22]; knowledge distillation-based techniques, leveraging teacher-student frameworks to align logits or features across incremental stages [23–28]; and regularization-based strategies, which constrain parameter updates to preserve critical weights [29–32]. Architectural adaptations address CIL through model rectification [33–37] to correct decision biases or dynamic expansion [38–45] by incrementally adding task-specific networks.

FM-Based CIL. FM-based approaches have emerged as a powerful direction in CIL, enabling efficient adaptation to new classes by building upon the rich representations encoded in extensively pretrained models [46–49]. Recent works focus on PEFT methods which preserve the pretrained 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 pretrained models. DualPrompt [10] extends L2P by incorporating G-Prompt and E-Prompt, designed to capture task-invariant and task-specific information, respectively. CODA-Prompt [8] employs contrastive loss to decorrelate prompt representations, mitigating interference, and integrates them using an attention-based weighting mechanism. APER [7] 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.

While these methods have advanced the state of the art, they are almost exclusively designed for and evaluated on large-scale pre-trained ViT backbones. However, the field's direct leap to these massive structures has left a significant gap in the literature regarding a simple but efficient baseline that we can truly assess their performance.

3. Method

In this section, we first define the problem of CIL, outlining the challenge of sequentially learning from a stream of non-overlapping tasks. Next, we introduce our architectural design and module pruning. Finally, we detail our training and inference protocol, demonstrating how the model learns while dynamically selecting the appropriate module during evaluation without task identifiers. For a detailed description of PAM's algorithm, please refer to the Appendix A.2.

3.1. Problem Formulation

CIL addresses the challenge of learning from a sequence of tasks $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_B$, where each task $\mathcal{D}_b = \{(x_i, y_i)\}_{i=1}^{n_b}$ introduces a set of non-overlapping classes. Here, $x_i \in \mathbb{R}^D$ represents a training instance, $y_i \in Y_b$ denotes its corresponding label, and $Y_b \cap Y_{b'} = \emptyset$ for $b \neq b'$. During training on task b the model has access only to $\mathcal{D}_b$, and the objective is two-fold: (i) acquiring new knowledge by learning to classify instances from the current task $\mathcal{D}_b$, and (ii) to preserve old knowledge by retaining performance on all previously seen tasks $\mathcal{D}_1, \ldots, \mathcal{D}_{b-1}$. After training on task $\mathcal{D}_b$, the model is evaluated on the cumulative label space $\mathcal{Y}_b = Y_1 \cup \cdots \cup Y_b$. Specifically, the aim is to obtain a model $f(\mathbf{x}) : X \to \mathcal{Y}_b$ that is able to classify all test dataset **without task indices** in the **exemplar-free setting** [7–13].

3.2. Architecture Overview

We build our method on the ResNets [15], 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 pretrained 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.

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

3.3. Pruned Adaptation Modules

We adopt a structured pruning strategy early in training to improve parameter efficiency. Specifically, after the first epoch, we evaluate the saliency of channels within each task-specific embedding module γ and remove the least informative ones. Consider a convolutional layer with weight tensor $W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times K \times K}$. For each output channel c, we compute its importance using the L_1-norm of its associated kernel weights. Formally, let $W_c \in \mathbb{R}^{C_{\text{in}} \times K \times K}$ denote the kernel corresponding to channel c, and let W_c^i denote its individual weight values. The saliency score s_c is defined as in Eq. 2. Channels are then ranked according to their saliency s_c, and the lowest-scoring ones are pruned until the desired sparsity level is reached. This structured pruning step substantially reduces the number of learnable parameters, ensuring that subsequent updates operate on a compact yet highly informative subset of weights. As a result, the method reduces parameter size while preserving sufficient expressive capacity for effective task adaptation.

$$s_c = \sum |W_c^i|. \tag{2}$$

Following pruning, our architectural notation and flow are slightly modified. The shared pretrained 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 pruned adaptation module $\mathcal{S}_b$. The resulting architecture per task is then expressed as in Eq. 3.

$$\hat{y_i} = \arg\max \sigma(W^\top \mathcal{S}_b(\Phi(\mathbf{x}_i))). \tag{3}$$

3.4. Training and Inference Protocol

For each learning session with dataset $\mathcal{D}_b$, we maintain the general feature extractor Φ in a frozen state to retain transferable and generalizable representations across tasks. This design choice ensures that the core knowledge learned from the pretraining remains intact while adapting to new tasks. Instead of modifying the entire network, we update only the parameters of the pruned 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. 4.

$$\ell_{CE} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) = -\sum_{i=1}^{N} y_i \log \sigma(W^\top \mathcal{S}_b(\Phi(\mathbf{x}_i))) \tag{4}$$

During inference, the model does not have access to task identities, necessitating a mechanism to select the most appropriate pruned task-specific module $\mathcal{S}$ for a given unlabeled test batch $\mathbf{x}_{test}$. To address this challenge, we employ a confidence-based selection strategy. First, the test batch $\mathbf{x}_{test}$ is processed by the frozen general feature extractor Φ, returning initial feature representations $\Phi(\mathbf{x}_{test})$. These representations are then passed through each pruned adaptation module $\mathcal{S}_b$ and the shared classifier $W^\top$, yielding task-conditioned class probability distributions as given in Eq 5:

$$p_b(x_{test}) = \sigma(W^\top \mathcal{S}_b(\Phi(\mathbf{x}_{test}))) \tag{5}$$

Then, the confidence score is defined as the average maximum softmax probability across the batch, effectively capturing the certainty of each module in its predictions. Finally, the module with the highest confidence score is selected for inference, as formulated in Eq. 6. This selection mechanism allows the model to dynamically adapt to different tasks without requiring explicit task identifiers, leveraging internal confidence measures to infer the most appropriate task-specific module.

$$\hat{b} = \arg\max_b \frac{1}{|\mathbf{x}_{test}|} \sum_{x_i \in \mathbf{x}_{test}} \max_{y \in \mathcal{Y}_b} p_b(y \mid x_i) \tag{6}$$

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 [50] comprises 100 natural image classes with 500 training images per class, while ImageNet-R [51] contains 200 classes with 24,000 training images and 6,000 test images. In addition, CUB-200 [52] consists of 200 bird species, with approximately 60 images per class (equally divided between training and testing), and Cars-196 [53] 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 [12], L2P [11], DualPrompt [10], CODA-Prompt [8], APER [12], 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.

Implementation Details. All experiments were conducted on an NVIDIA A100 GPU using PyTorch [54] and the PILOT framework [55]. 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, PAM, 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 pruned module for the remaining epochs. To ensure fairness and robustness, we use the default parameters of existing approaches and repeat each experiment five times with different random seeds, where each seed also alters the task order. We report the mean and standard deviation of the performance metrics.

5. Results

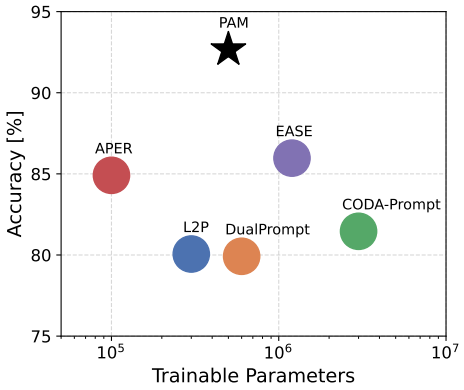
5.1. State-of-art Comparison

Incremental Performance. Table 1 presents the average and final accuracy of various continual learning methods across four benchmarks, demonstrating the effectiveness of PAM. We observe that on simpler datasets (e.g., CIFAR100), even a small ResNet18 backbone achieves competitive performance. For more challenging datasets such as CUB and ImageNet-R, larger ResNets yields improved results while still remaining smaller than ViT-B/16-IN21K used by other methods. These results establish PAM 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.

Parameter Size Comparison. Figure 3 compares FM-based CIL methods on the CIFAR B0 Inc5 benchmark in terms of accuracy and parameter efficiency. PAM 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, PAM 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.

Table 1: Average and final accuracy [%] with methods using **ViT-B/16-IN21K** and PAM 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 $\pm$ 5.6	48.12 $\pm$ 3.3	55.78 $\pm$ 2.8	33.13 $\pm$ 3.3	59.09 $\pm$ 3.7	49.46 $\pm$ 3.3	41.90 $\pm$ 1.0	19.47 $\pm$ 2.7
SimpleCIL	86.48 $\pm$ 0.8	81.28 $\pm$ 0.1	91.58 $\pm$ 1.3	86.73 $\pm$ 0.1	61.31 $\pm$ 0.4	54.55 $\pm$ 0.1	54.95 $\pm$ 0.8	35.43 $\pm$ 0.0
L2P	84.90 $\pm$ 1.2	80.06 $\pm$ 1.4	73.22 $\pm$ 1.8	61.55 $\pm$ 1.7	75.92 $\pm$ 0.7	70.88 $\pm$ 0.7	42.06 $\pm$ 2.0	30.07 $\pm$ 0.8
DualPrompt	85.61 $\pm$ 1.3	79.92 $\pm$ 0.4	81.36 $\pm$ 1.8	70.51 $\pm$ 1.1	71.48 $\pm$ 0.5	66.09 $\pm$ 1.3	45.30 $\pm$ 1.1	30.15 $\pm$ 0.9
CODA-Prompt	87.64 $\pm$ 0.4	81.46 $\pm$ 0.3	77.65 $\pm$ 1.0	68.44 $\pm$ 1.0	76.25 $\pm$ 0.3	71.39 $\pm$ 0.3	36.22 $\pm$ 0.6	25.44 $\pm$ 0.3
APER-Adapter	89.57 $\pm$ 0.9	84.91 $\pm$ 0.2	91.62 $\pm$ 1.2	86.72 $\pm$ 0.2	74.81 $\pm$ 0.8	66.97 $\pm$ 0.8	47.91 $\pm$ 0.8	35.49 $\pm$ 0.0
EASE	90.79 $\pm$ 0.8	85.97 $\pm$ 0.6	**92.51** $\pm$ 1.3	86.49 $\pm$ 1.2	**80.35** $\pm$ 1.0	75.74 $\pm$ 0.8	49.32 $\pm$ 1.0	34.75 $\pm$ 0.3
PAM (RN18)	**91.40** $\pm$ 2.1	**88.51** $\pm$ 3.4	87.40 $\pm$ 1.3	83.69 $\pm$ 3.4	68.54 $\pm$ 0.4	65.76 $\pm$ 0.4	**79.09** $\pm$ 1.6	**64.82** $\pm$ 1.6
PAM (RN50)	**93.06** $\pm$ 1.5	**92.50** $\pm$ 2.1	85.40 $\pm$ 3.0	82.67 $\pm$ 3.0	73.22 $\pm$ 0.6	72.83 $\pm$ 0.5	**77.41** $\pm$ 1.5	**62.23** $\pm$ 8.2
PAM (RN101)	**94.16** $\pm$ 1.5	**93.05** $\pm$ 1.7	89.76 $\pm$ 1.1	**87.26** $\pm$ 1.7	77.75 $\pm$ 0.7	**77.03** $\pm$ 0.8	**80.16** $\pm$ 2.1	**77.30** $\pm$ 2.6
PAM (RN152)	**94.17** $\pm$ 1.4	**93.79** $\pm$ 1.7	89.91 $\pm$ 1.4	**88.35** $\pm$ 1.5	79.33 $\pm$ 1.0	**78.95** $\pm$ 0.5	**83.10** $\pm$ 0.9	**77.23** $\pm$ 3.5



Method	Trainable Params Per Task	Total Params After All Tasks	Final Accuracy [%]
L2P	300 K	92 M	80.06 $\pm$ 1.1
DualPrompt	600 K	98 M	79.92 $\pm$ 0.4
CODA-Prompt	3 M	146 M	81.46 $\pm$ 0.3
APER	100 K	86 M	84.91 $\pm$ 0.2
EASE	1.2 M	110 M	85.97 $\pm$ 0.6
PAM (RN18)	600 K	15 M	88.51 $\pm$ 3.4
PAM (RN50)	600 K	21 M	92.50 $\pm$ 2.1
PAM (RN101)	600 K	40 M	93.05 $\pm$ 1.7
PAM (RN152)	600 K	56 M	93.79 $\pm$ 1.7

Figure 3: Parameter size vs. accuracy: The left panel shows that PAM challenges existing and future FM-based methods; and the right panel presents the parameter count for different methods after completing all sessions.

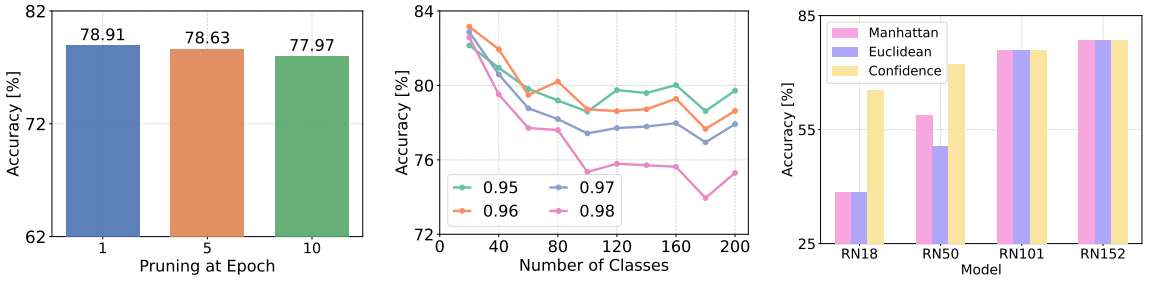
5.2. Ablation Study

To systematically evaluate the impact of core design choices for PAM, we conduct ablations across three critical dimensions: pruning schedule, pruning magnitude, and pruned 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 4a, 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.

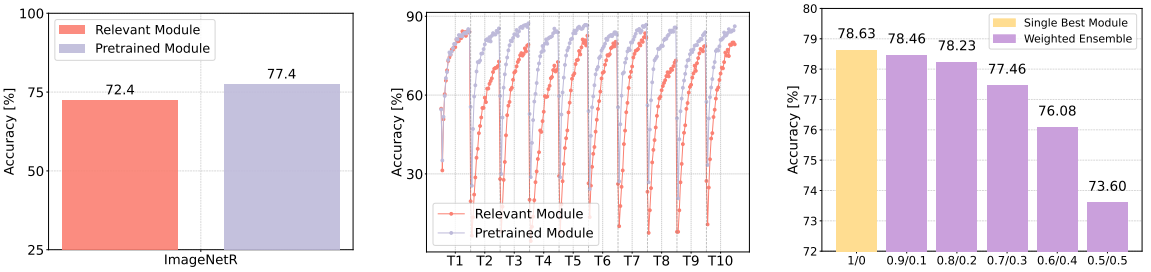
Pruning magnitude. The pruning magnitude applied to the task-specific module is another critical factor. In this experiment, we vary the pruning magnitude applied to the task-specific module γ_b across four settings: $0.95, 0.96, 0.97$, and 0.98. Figure 4b illustrates how the incremental performance 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 pruned 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



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

Figure 4: Ablations of different components for the PAM 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.

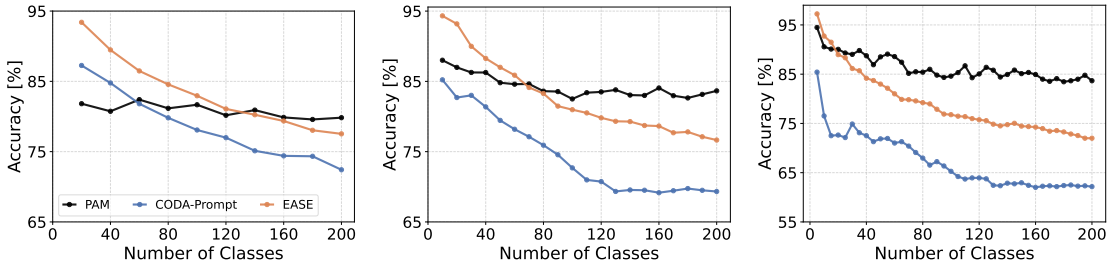


(a) Final incremental accuracy (b) Per-task training accuracy (c) Ensemble performance

Figure 5: Analysis of PAM across different evaluation settings. (a) Final incremental accuracy after all tasks, (b) per-task training accuracy illustrating the effect of initialization strategies for pruned adaptation modules, and (c) performance of weighted ensemble strategies compared to the single best module. Ratios show the relative contribution of the most confident module and the remaining modules (e.g., 0.9/0.1).

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 pruned adaptation module $\mathcal{S}_b$ by leveraging the maximum softmax probability over the test batch. As shown in Figure 4c, 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 PAM.

Knowledge transfer between the modules. Beyond our main ablations, we also conduct an additional investigation into the knowledge transfer dynamics within our PAM approach, as this aspect warrants further discussion. Specifically, we examine two different strategies for initializing newly added pruned 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 5a, 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 5b 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.



(a) IN-R B0 Inc20 (b) IN-R B0 Inc10 (c) IN-R B0 Inc5

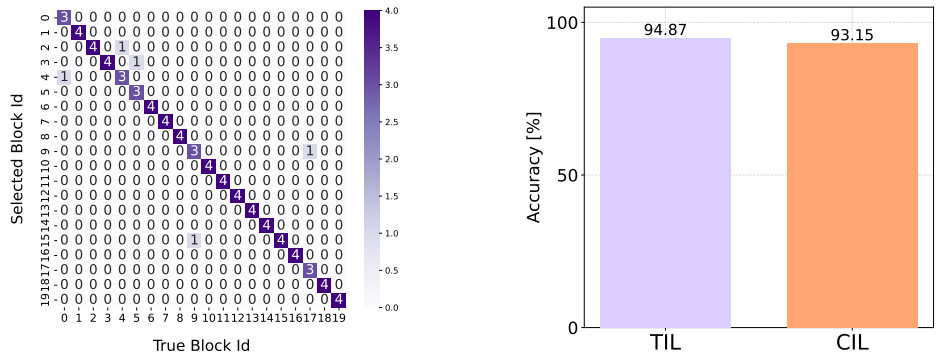
Figure 6: PAM achieves noticeably more stable accuracy throughout long-horizon experiments with challenging ImageNet-R (IN-R) dataset, reflecting its robustness against performance degradation as tasks accumulate.

5.3. Discussion and Further Analysis

Representation capacity. In our experiments, we evaluate various ResNet models and observe consistently strong final incremental accuracy. This is particularly noteworthy given that existing methods rely on much larger ViT backbone to reach high performance. Despite this strength in final accuracy, PAM 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, PAM 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 to pre-trained ResNet152, we observe improvements in both final and average accuracy. Overall, these findings illustrate that while large capacity backbones are more effective for adaptation but current FM-based approaches are falling short from that perspective since smaller ResNets can still offer competitive accuracy while being more efficient.

Longer learning sessions. Analyzing performance under longer learning sessions is critical in the continual learning paradigm, where models must remain robust as the number of tasks grows. To evaluate this, we conduct an extended study on the challenging ImageNet-R benchmark using varying numbers of tasks. We compare PAM against two strong state-of-the-art baselines representing different PEFT families: the prompt-based CODA-Prompt and the adapter-based EASE. Across all experimental setups, PAM consistently exhibits more stable behavior as the number of classes increases. Although EASE begins with slightly higher initial accuracy, its performance deteriorates more rapidly over time, indicating weaker resilience in extended learning scenarios. CODA-Prompt suffers the steepest decline overall, highlighting the difficulty of maintaining prompt-based solutions under long sequences of tasks. In contrast, PAM maintains competitive accuracy throughout, demonstrating its suitability for realistic continual learning scenarios with slower performance decay and better long-term retention.

Proximity to the upper bound. To contextualize our results, we compare our performance in the challenging class-incremental learning setting against the task-incremental learning upper bound. Since task-incremental learning provides an idealized scenario with explicit task identities, this comparison highlights how close our approach gets to the best-possible performance. Using the CIFAR B0 Inc5 setup, we measure how often the model chooses the appropriate module for each task. As depicted in Figure 7a, the confusion matrix shows a strong diagonal pattern, indicating that the model consistently selects the correct module with only occasional mismatches. This reflects a high level of implicit task recognition and minimal confusion across tasks. To understand how much this implicit selection affects overall performance, we also evaluate the system under a task-incremental learning setting, which serves as an upper bound because the correct module is explicitly provided for every test batch. As shown in Figure 7b, the gap between class-incremental learning and task-incremental learning accuracy is extremely small, demonstrating that PAM performs nearly as well as the ideal task-incremental learning scenario.



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

Figure 7: Correct module alignment of the PAM: (a) The confusion matrix shows that PAM reliably selects the correct adaptation module. (b) Comparing CIL and TIL results illustrates that providing true task identifiers yields a marginal improvement, indicating that PAM's performance is already close to the TIL upper bound.

Ensemble strategy. Finally, we study different inference strategies, examining how to best leverage task-specific modules during prediction. In our default inference protocol, we rely on the prediction of the single most confident task-specific module, which consistently provides the strongest performance in our experiments. To reduce the reliance on a single module's decision, we also explored an ensemble-based alternative. In this approach, we implemented a weighted ensemble strategy, assigning dominant weight to the most confident module while allowing the remaining modules to contribute proportionally through fixed ratios. This pilot study serves as an initial step toward assessing whether ensembles of task-specific experts can improve generalization. As reported in Figure 5c, our findings show that naive ensembling does not outperform the single-module baseline, while using a principled weighting on ensembles shows greater promise.

6. Conclusion

In this work, we introduce Pruned Adaptation Modules (PAM), a simple yet effective approach that bridges traditional CIL methods with emerging FM-based continual learning. Unlike recent strategies that depend heavily on large foundation models, PAM demonstrates that compact, pre-trained backbones such as ResNets can achieve competitive, or even superior, performance in CIL while dramatically reducing computational and storage costs. By freezing early layers and applying structured pruning to a lightweight, task-specific module, PAM achieves a 2 - $5\times$ reduction in trainable parameters and a 2 - $6\times$ smaller overall parameter footprint compared to state-of-the-art FM-based CIL methods. Overall, PAM 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 may explore extending PAM to transformer-based backbones, other continual learning scenarios, incorporating dynamic pruning strategies, or integrating PAM into larger FMs to combine efficiency with broad generalization capabilities.

Broader Impact

Proposed approach reduces memory and computation requirements, enabling more accessible and energy-efficient continual learning research. As with any adaptive machine learning system, care should be taken to ensure responsible and fair deployment in real-world applications.

Acknowledgements

This work is supported by the EU Horizon programme through SYNERGIES, a project under GA No. 101146542; and ELLIOT, a project under GA No. 101214398; and Dutch e-infrastructure with the support of SURF Cooperative using GA no. EINF-10242; 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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[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] 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.

[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] Hai-Long Sun, Da-Wei Zhou, Hanbin Zhao, Le Gan, De-Chuan Zhan, and Han-Jia Ye. Mos: Model surgery for pre-trained model-based class-incremental learning. In *AAAI*, 2024.

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

[16] Gido M. Van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*.

[17] Ceren Gok Yildirim, Murat Onur Yildirim, Mert Kilickaya, and Joaquin Vanschoren. Adacl: Adaptive continual learning. In *Proceedings of the 1st ContinualAI Unconference*. PMLR, 2023.

[18] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[19] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2018.

[20] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, 2020.

[21] Hanbin Zhao, Hui Wang, Yongjian Fu, Fei Wu, and Xi Li. Memory-efficient class-incremental learning for image classification. *TNNLS*, 2021.

[22] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *NeurIPS*, 2019.

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

[24] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[25] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

[26] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *CVPR*, 2019.

[27] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, 2020.

[28] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *WACV*, 2020.

[29] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

[30] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *CVPR*, 2019.

[31] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. In *PNAS*, 2016.

[32] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICLR*, 2017.

[33] Quang Pham, Chenghao Liu, and Steven Hoi. Continual normalization: Rethinking batch normalization for online continual learning. In *ICLR*, 2022.

[34] Yujun Shi, Kuangqi Zhou, Jian Liang, Zihang Jiang, Jiashi Feng, Philip HS Torr, Song Bai, and Vincent YF Tan. Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. In *CVPR*, 2022.

[35] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019.

[36] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *CVPR*, 2020.

[37] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *CVPR*, 2020.

[38] Xiuwei Chen and Xiaobin Chang. Dynamic residual classifier for class incremental learning. In *ICCV*, 2023.

[39] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *CVPR*, 2022.

[40] Zhiyuan Hu, Yunsheng Li, Jiancheng Lyu, Dashan Gao, and Nuno Vasconcelos. Dense network expansion for class incremental learning. In *CVPR*, 2023.

[41] Bingchen Huang, Zhineng Chen, Peng Zhou, Jiayin Chen, and Zuxuan Wu. Resolving task confusion in dynamic expansion architectures for class incremental learning. In *AAAI*, 2023.

[42] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *CVPR*, 2021.

[43] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *ECCV*, 2022.

[44] Murat Onur Yildirim, Elif Ceren Gok, Ghada Sokar, Decebal Constantin Mocanu, and Joaquin Vanschoren. Continual learning with dynamic sparse training: Exploring algorithms for effective model updates. In *CPAL*, 2024.

[45] Murat Onur Yildirim, Elif Ceren Gok Yildirim, Decebal Constantin Mocanu, and Joaquin Vanschoren. Self-regulated neurogenesis for online data-incremental learning. *arXiv:2403.14684*, 2025.

[46] 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:2403.07356*, 2024.

[47] 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.

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

[49] Murat Onur Yildirim, Elif Ceren Gok Yildirim, and Joaquin Vanschoren. Sculpting [CLS] features for foundation model-based class-incremental learning. In *AI That Keeps Up: NeurIPS Workshop on Continual and Compatible Foundation Model Updates*, 2025.

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

[51] 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.

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

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

[54] 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.

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

A. Appendix

In this appendix, we present a supplementary strategy to our approach PAM for adaptively reusing pruned adaptation modules based on task similarity rather than defining entirely new ones for each novel task and demonstrate how this adaptive strategy influences overall model performance. We also provide a comprehensive explanation of the existing methods for both prompt- and adapter-based which are used in the evaluations and presented in the main paper, together with our pseudocode.

A.1. Adaptive Approach for Initializing PAM

In our original implementation, a new pruned 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 pruned 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 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 7.

$$d(\mathcal{D}_b, \mathcal{D}_i) \simeq \|c_b - c_i\|_1, \quad c_b = \frac{1}{n_b} \sum_{i=1}^{n_b} \Phi(x_i) \tag{7}$$

We then compute the average distance across all previously encountered tasks as in Eq. 8 and we scale it by a hyperparameter β which is a pre-determined hyperparameter to obtain a similarity threshold τ given in Eq. 9.

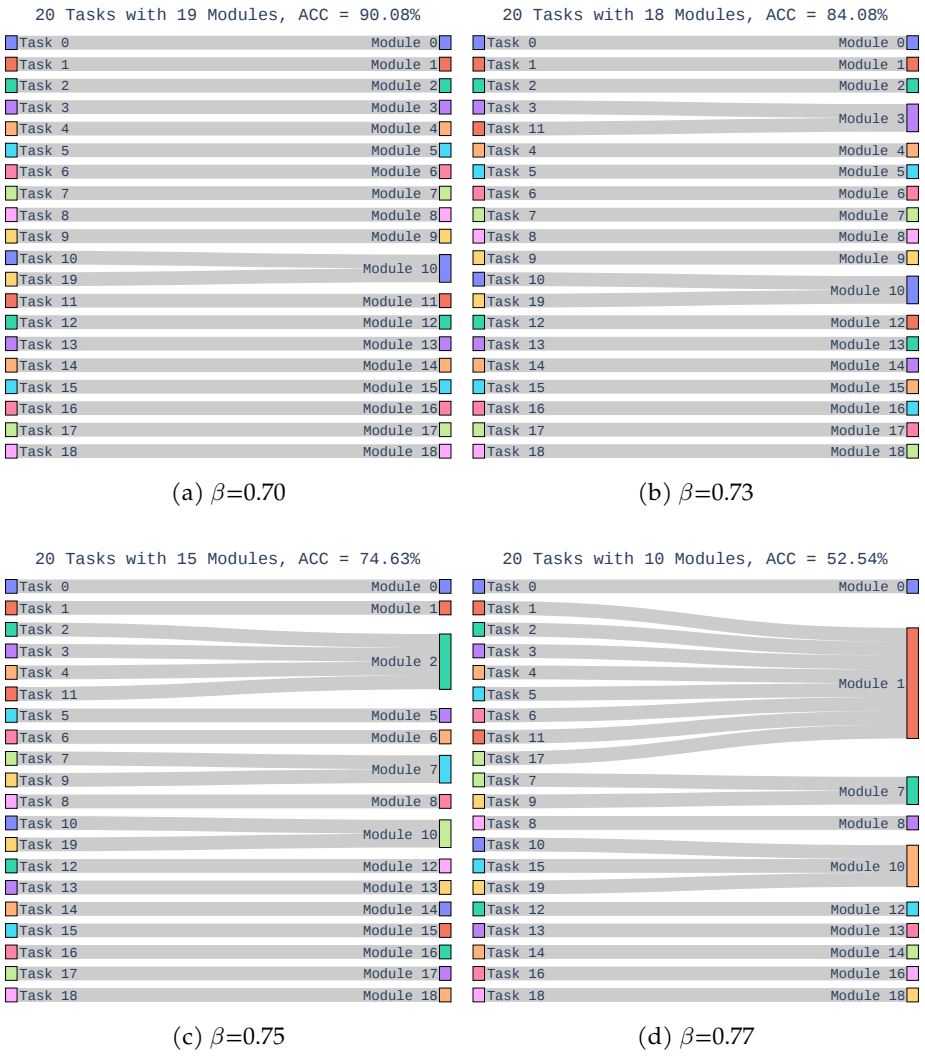
$$\bar{d} = \frac{1}{b} \sum_{b=1}^{b} d(\mathcal{D}_b, \mathcal{D}_i). \tag{8}$$

$$\tau = \beta \cdot \bar{d}. \tag{9}$$

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 pruned 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 [23].

The results presented in Figure A highlight a fundamental tradeoff between continual learning performance and parameter efficiency. By selectively reusing existing pruned 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 PAM provides a flexible framework for task reuse, making it straightforward to integrate into various applications.



(a) β=0.70 (b) β=0.73

(c) β=0.75 (d) β=0.77

Figure A: Analysis of the effect of threshold values on the adaptive initialization of pruned 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.

A.2. Details of Compared Methods and PAM

Below, we briefly describe the baseline methods we used to evaluate our approach. Then, we share the algorithm of PAM.

- **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.

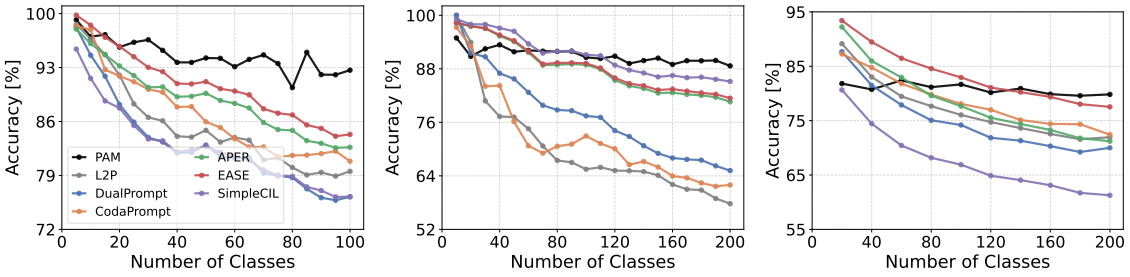
- **DualPrompt [10]:** As an extension of L2P, DualPrompt refines the prompt mechanism by employing two types of prompts: general and expert prompts, while maintaining the instance-specific prompt construction process.

- **CODA-Prompt [8]:** 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 [12]:** 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.
- **APER [7]:** Extending SimpleCIL, APER 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.

Algorithm 1 PAM: Pruned Adaptation Modules

Training Phase:
for each task $D_b \in \{D_1, D_2, \ldots, D_B\}$ **do**
  for e in range epoch **do**
    if $e = 1$ **then**
      Train γ_b using the loss function in Eq. 4
      Rank saliency of the filters in γ_b using Eq. 2
      Obtain $\mathcal{S}_b$ with the saliency-based pruning
    else
      Train $\mathcal{S}_b$ with remaining filters using Eq. 4
    end if
  end for
end for

Inference Phase:
for each test batch $\mathbf{x}_{test}$ **do**
  Get class probabilities for each $\mathcal{S}_b$ using Eq. 5
  Select the most confident $\mathcal{S}_{\hat{b}}$ using Eq. 6
  Get predictions for $\mathbf{x}_{test}$ using Eq. 3 with $\mathcal{S}_{\hat{b}}$
end for



(a) CIFAR B0 Inc5 (b) CUB B0 Inc10 (c) ImagenetR B0 Inc20

Figure B: Incremental accuracy flow of each method over sequential tasks where comparison methods utilize ViT-B/16 and PAM employs ResNet152. PAM maintains a more stable SOTA performance across tasks.