FROM ISOLATION TO INTEGRATION: BUILDING AN ADAPTIVE EXPERT FOREST FOR PRE-TRAINED MODEL-BASED CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Class-Incremental Learning (CIL) requires models to learn new classes without forgetting old ones. A common method is to freeze a pre-trained model and train a new, lightweight adapter for each task. While this prevents forgetting, it treats the learned knowledge as a simple, unstructured collection and fails to use the relationships between tasks. To this end, we propose the Semantic-guided Adaptive Expert Forest (SAEF), a new method that organizes adapters into a structured hierarchy for better knowledge sharing. SAEF first groups tasks into conceptual clusters based on their semantic relationships. Then, within each cluster, it builds a balanced expert tree by creating new adapters from merging the adapters of similar tasks. At inference time, SAEF finds and activates a set of relevant experts from the forest for any given input. The final prediction is made by combining the outputs of these activated experts, weighted by how confident each expert is. Experiments on several benchmark datasets show that SAEF achieves SOTA performance.

1 INTRODUCTION

Class-Incremental Learning (CIL) enables models to continually acquire knowledge of new classes while preserving performance on previously learned ones McCloskey & Cohen (1989); Kirkpatrick et al. (2017). While mitigating catastrophic forgetting is a primary objective, the growing application of CIL in resource-sensitive contexts, such as on-device and robotic systems, has also underscored the importance of computational efficiency Soltoggio et al. (2024); Liu et al. (2025). CIL methodologies can be broadly categorized into two paradigms: training from scratch and leveraging Pre-trained Models (PTMs). The former category, which includes regularization, architectural, and rehearsal-based methods Li & Hoiem (2017); Serra et al. (2018); Rebuffi et al. (2017a), often encounters practical limitations concerning scalability, data privacy, and overall performance Wang et al. (2022b). Consequently, a significant portion of recent research has shifted towards PTM-based CIL Zhou et al. (2024a). This method utilizes the powerful representations learned during large-scale pre-training as a robust foundation for incremental tasks, frequently leading to state-of-the-art (SOTA) results. A prominent strategy within this paradigm involves freezing the PTM backbone and applying Parameter-Efficient Fine-Tuning (PEFT) techniques, such as training lightweight adapters Sun et al. (2025) or prompts Wang et al. (2022c); Smith et al. (2023). This method offers a compelling balance of high performance and low computational overhead, making it a highly influential and widely adopted method in the field.

A common strategy within this PTM-based paradigm is to train a separate, lightweight adapter for each incremental task Zhou et al. (2024b). During inference, systems may either retrieve the most suitable task adapter Smith et al. (2023) or ensemble predictions from the entire collection Zhou et al. (2024b). While separating task-specific parameters helps prevent catastrophic forgetting, this method treats the learned knowledge as a simple, unstructured collection, as shown in Fig. 1(a). This design fails to use the important semantic relationships between tasks (for example, between a 'dog' task and a 'cat' task) for better knowledge transfer. Some methods attempt to address this by adding a global module to learn shared knowledge from all tasks Huai et al. (2025). However, we argue that sharing knowledge globally across all tasks is a suboptimal method. True knowledge transfer should be more selective and structured. It should be context-specific and hierarchical, sharing information only between related tasks, as illustrated in Fig. 1(b).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To this end, we propose the Semantic-guided Adaptive Expert Forest (SAEF). SAEF is a method that automatically organizes the set of task-specific adapters into a structured expert forest for better knowledge sharing. This process consists of three main stages. First, SAEF groups tasks into conceptual clusters based on their semantic relationships. Second, within each cluster, it constructs a balanced expert tree by recursively creating new adapters from merging the adapters of the most similar tasks. This process builds a hierarchy of experts at different levels of abstraction. Finally, the third stage involves adaptive inference, where SAEF employs a parallel search strategy across this hierarchy to dynamically activate a subset of relevant experts for any given sample. The final prediction is produced by an entropy-guided fusion of these experts' outputs. In summary, our main contributions are as follows:

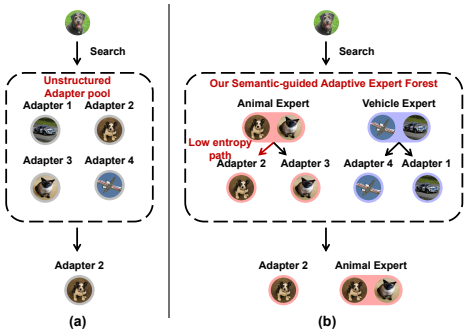


Figure 1: Adapter-based CIL: **(a)** an unstructured adapter pool; **(b)** SAEF's semantic expert forest with adaptive search.

- We propose SAEF, a novel method that automatically constructs a semantic expert hierarchy from previously isolated task adapters to enable structured knowledge reuse.
- We design a dynamic inference strategy that adaptively navigates this hierarchy, using a multi-path search and an entropy-guided fusion mechanism for robust predictions.
- We conduct extensive experiments on major CIL benchmarks, where SAEF consistently outperforms SOTA methods with accuracy gains of up to 2.96%.

2 Related Work

CIL methods can be broadly divided into two main categories: training models from scratch and using Pre-trained Models (PTMs).

2.1 Training Models from Scratch

The traditional method for CIL is to train a model from scratch. These methods are typically grouped into three categories. **Regularization-based** methods Yu et al. (2020); Serra et al. (2018); Li & Hoiem (2017); Yang et al. (2024); Chaudhry et al. (2018); Aljundi et al. (2018) add a loss term to penalize changes to important parameters from past tasks. **Architecture-based** methods Ebrahimi et al. (2020); Ke et al. (2020); Loo et al. (2021); Wang et al. (2020); Zhao et al. (2022) add new parameters for each new task to keep knowledge separate, for example, by growing the network or using different parts of it for different tasks. **Rehearsal-based** methods Aljundi et al. (2019); Buzzega et al. (2020); Cha et al. (2021); Chaudhry et al. (2021; 2019) store a small number of old samples and replay them during training on new tasks. These methods have limitations in scalability and performance, which has led to a shift towards PTM-based methods Buzzega et al. (2020); Zhu et al. (2021); Wang et al. (2022a); Shokri & Shmatikov (2015); Smith et al. (2021); Wang et al. (2022b); Li et al. (2024).

2.2 Pre-Trained Model-based CIL

To avoid the high cost of training from scratch, modern CIL often relies on a frozen pre-trained model (PTM) with Parameter-Efficient Fine-Tuning (PEFT). This involves training only a small set of new parameters for each task. This makes CIL practical for many resource-constrained applications. Common PEFT methods include prompt tuning, which learns to select or combine prompts Wang et al. (2022c;b); Smith et al. (2023), and adapter-based tuning, which adds a new adapter for each task Sun et al. (2025); Zhou et al. (2024b). Other methods also follow this paradigm, such as fine-tuning only the final layer Zhang et al. (2023) or using random projections McDonnell et al. (2023).

Recent work explores knowledge sharing across task-specific modules. MoE methods route inputs among experts Fan et al. (2022); Hazimeh et al. (2021); Ho et al. (2022); Jacobs et al. (1991), but

often overlook task-semantic hierarchy. MOS shares knowledge across tasks Sun et al. (2025) without explicitly modeling this hierarchy. SD-LoRA learns magnitudes for frozen LoRA directions Wu et al. (2025); SEMA expands adapters with a soft router Wang et al. (2025); and HAM groups pruned adapters for α-merging Coleman et al. (2025). In contrast, SAEF organizes experts into semantics-guided concept trees, merges related experts into usable internal nodes, and selects relevant paths for inference.

3 METHODOLOGY

3.1 PRELIMINARIES

Class-Incremental Learning (CIL). A model learns from a sequence of T tasks $\{\mathcal{D}_t\}_{t=1}^{T}$, where each task t provides training data $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^{N_t}$ with disjoint class sets $\mathcal{C}_t$ ($\mathcal{C}_t \cap \mathcal{C}_{t'} = \emptyset$ for $t \neq t'$). We follow the exemplar-free setting Wang et al. (2022c;b); Zhou et al. (2024a): no past samples are stored. The model comprises a feature extractor ϕ and a classifier weight matrix W, with trainable parameters θ_t for task t. Training at task t uses only $\mathcal{D}_t$ and never accesses test data; the training objectives are specified below. After learning task t, the model is evaluated on all seen classes $\mathcal{Y}_t = \bigcup_{i=1}^{t} \mathcal{C}_i$.

PTM-based CIL with Task-Specific Adapters. We build on the PTM-based CIL framework Zhou et al. (2024a); Sun et al. (2025), where a frozen ViT Dosovitskiy et al. (2020) backbone is augmented with a lightweight adapter $\mathcal{A}_t = \{W_{\text{down}}^{(l,t)}, W_{\text{up}}^{(l,t)}\}_{l=1}^{L}$ per task. Each adapter is a bottleneck module ($W_{\text{down}} \in \mathbb{R}^{d\times r}$, $W_{\text{up}} \in \mathbb{R}^{r\times d}$, $r \ll d$) inserted in parallel to the MLP of each Transformer block:

$$\mathbf{x}_o = \text{MLP}(\mathbf{x}_{in}) + \text{ReLU}(\mathbf{x}_{in} W_{\text{down}}) W_{\text{up}}, \tag{1}$$

where $\mathbf{x}_{in} \in \mathbb{R}^d$ is the input feature and $\mathbf{x}_o \in \mathbb{R}^d$ is the output. Training combines a cross-entropy loss $\mathcal{L}_{\text{cls}}$ over all seen classes $\mathcal{Y}_t$:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N_t} \sum_{(\mathbf{x},y)\in\mathcal{D}_t} \log \frac{\exp(\mathbf{w}_y^\top \phi(\mathbf{x}; \mathcal{A}_t))}{\sum_{c\in\mathcal{Y}_t} \exp(\mathbf{w}_c^\top \phi(\mathbf{x}; \mathcal{A}_t))}, \tag{2}$$

where $\phi(\mathbf{x}; \mathcal{A}_t)$ is the feature vector and $\mathbf{w}_c$ is the classifier weight for class c, with an orthogonality regularizer that discourages interference between adapters:

$$\mathcal{L}_{\text{orth}} = \sum_{l=1}^{L} \sum_{i=1}^{t-1} \left\| W_{\text{up}}^{(l,t)} (W_{\text{up}}^{(l,i)})^\top \right\|_F, \quad \mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{orth}}, \tag{3}$$

where $\|\cdot\|_F$ is the Frobenius norm and λ balances the two objectives.

3.2 OUR METHOD: SEMANTIC-GUIDED ADAPTIVE EXPERT FOREST

Our method begins by following the standard PTM-based CIL paradigm, where a separate, task-specific adapter is trained for each incremental task using the objective in Eq. (3). Rather than treating the resulting set of adapters as a flat collection during inference, we propose the Semantic-guided Adaptive Expert Forest (SAEF), which organizes these adapters into a structured, hierarchical knowledge base. This process consists of three main stages, as illustrated in Figure 2: *Conceptual Clustering*, *Hierarchical Construction*, and *Adaptive Inference*.

Conceptual Clustering. The first stage of SAEF partitions the T learned tasks into semantically coherent groups. We define two complementary prototypes for each task t: a semantic prototype $\mathbf{p}_t^{\text{s}}$ and a visual prototype $\mathbf{p}_t^{\text{f}}$. Semantic prototypes are derived from a frozen CLIP Radford et al. (2021) text encoder g_t (see Appendix N for an ablation of visual-only clustering), using the prompt template t_c: "a photo of a [CLASS]":

$$\mathbf{p}_t^{\text{s}} = \frac{1}{|\mathcal{C}_t|} \sum_{c\in\mathcal{C}_t} g_t(t_c). \tag{4}$$

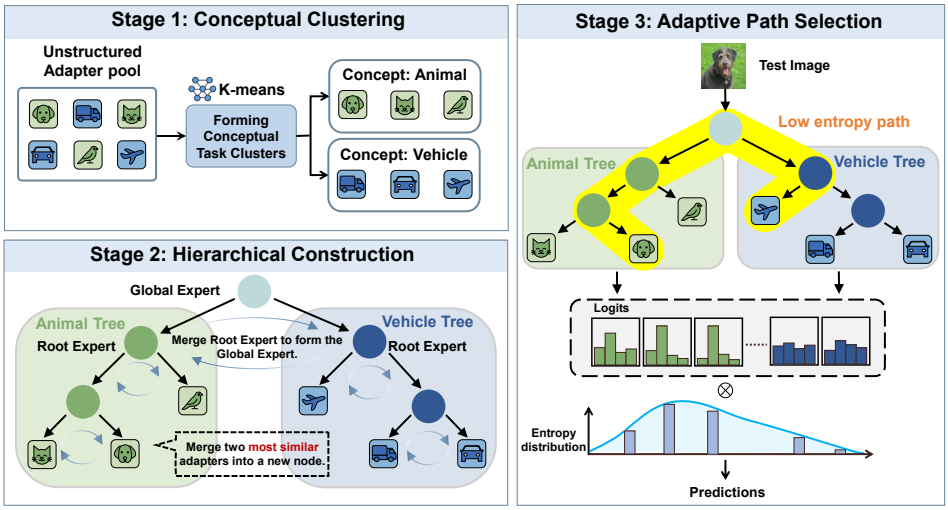


Figure 2: Overview of the SAEF. Our method organizes individually trained task adapters into a structured knowledge hierarchy. **(1) Stage 1: Conceptual Clustering.** Based on semantic prototypes, tasks are grouped into high-level conceptual clusters (e.g., 'Animal', 'Vehicle'). **(2) Stage 2: Hierarchical Construction.** Within each cluster, a balanced expert tree is built via a bottom-up process of recursively fusing the most visually similar nodes. The roots of these trees are then merged to form a single Global Expert. **(3) Stage 3: Adaptive Inference.** For a given input, SAEF performs a parallel, multi-path search to dynamically activate a set of relevant experts. The final prediction is derived from an entropy-guided fusion of their outputs.

Visual prototypes are extracted using the PTM's feature extractor ϕ with the first task's adapter $\mathcal{A}_1$ as a domain-adapted backbone (see Appendix O for an ablation of alternative feature sources):

$$\mathbf{p}_t^{\mathrm{f}} = \frac{1}{|\mathcal{D}_t|} \sum_{\mathbf{x}_i \in \mathcal{D}_t} \phi(\mathbf{x}_i; \mathcal{A}_1). \tag{5}$$

We apply K-Means to the semantic prototypes $\{\mathbf{p}_t^{\mathrm{s}}\}_{t=1}^{T}$ to partition tasks into K clusters $\{G_1, \ldots, G_K\}$, with K automatically selected by maximizing the average silhouette score Shahapure & Nicholas (2020). Each cluster G_k forms one tree in the subsequent *Hierarchical Construction* stage. This selects $K = 2$ on all four benchmarks; an ImageNet-R sensitivity analysis is provided in Appendix R.

Hierarchical Construction. The initial clustering provides a coarse-grained organization of tasks. To enable more effective, fine-grained knowledge transfer, this second stage constructs a balanced binary tree of experts for each conceptual cluster G_k, and then merges their roots into a single global expert. This transforms the initially unstructured collection of adapters into a unified, multi-level hierarchy. We reconstruct the expert forest dynamically after training each task and before inference, utilizing solely seen tasks without accessing future labels. Thus, all reported metrics reflect a strict online CIL setting.

For each cluster G_k, we start with a set of leaf experts, where each expert corresponds to a single task $t \in G_k$ and holds its adapter parameters $\theta_t \in \mathbb{R}^D$ and visual prototype $\mathbf{p}_t^{\mathrm{f}}$. The tree is built using a bottom-up, iterative process. To determine the merge order, each leaf is assigned a subtree height of zero, and each newly created parent is assigned a height one greater than the maximum height of its children. At each step, we merge the most visually similar pair among nodes with the smallest available subtree height. If only one node remains at that height, we merge it with the most similar node at the next smallest available height. This height-based rule keeps the resulting tree depth at $O(\log |G_k|)$; its precise construction is given in Appendix D.1. Pairwise similarity is computed as:

$$\mathrm{sim}(i, j) = \frac{{\mathbf{p}_i^{\mathrm{f}}}^{\top} \mathbf{p}_j^{\mathrm{f}}}{\|\mathbf{p}_i^{\mathrm{f}}\|_2 \|\mathbf{p}_j^{\mathrm{f}}\|_2}, \tag{6}$$

where $\|\cdot\|_2$ denotes the L2-norm. Once the most similar pair is identified, a new parent expert is created. The parent's parameter vector, $\theta_p \in \mathbb{R}^D$, is formed by merging the parameters of its two children, θ_i and θ_j, using a vectorized operation designed to preserve the dominant signal from both:

$$\theta_p \leftarrow \text{sign}(\theta_i + \theta_j) \odot \max(|\theta_i|, |\theta_j|), \tag{7}$$

where $\text{sign}(\cdot)$, $|\cdot|$, and $\max(\cdot,\cdot)$ are element-wise sign, absolute value, and maximum operations, and $\odot$ denotes the Hadamard product. This operation is theoretically motivated in Section 3.3, and its robustness to alternative fusion strategies is analyzed in Appendix M. The visual prototype for the new parent is computed as a weighted average of its children's prototypes, with weights proportional to the number of original leaf tasks each child represents. The two child experts are then replaced by their new parent in the set. This process continues until only a single root expert, R_k, remains for the cluster. Full construction details are in Appendix C.1.

Finally, to create a single entry point for inference, we merge the K tree roots into one global root expert R_G. The parameters of this global root, $\theta_G \in \mathbb{R}^D$, are formed using a vectorized operation that captures both the consensus and the maximum signal strength from all trees:

$$\theta_G \leftarrow \text{sign}\left(\sum_{k=1}^{K} \theta_{R_k}\right) \odot \max_{k\in\{1,\ldots,K\}} |\theta_{R_k}|, \tag{8}$$

where $\text{sign}(\cdot)$ is the element-wise sign function, $\max_k$ finds the element-wise maximum across the K root vectors, and $\odot$ denotes the Hadamard product. This completes the *Hierarchical Construction* of the full SAEF hierarchy.

Adaptive Inference. SAEF uses a dynamic inference strategy to classify any given input x. The strategy finds an optimal path in each of the K trees and then fuses the predictions from all experts along these paths.

For any expert in the hierarchy, represented by its adapter $\mathcal{A}_n$, we first compute its class-wise logits for a given input x. Specifically, the logit for each class $c \in \mathcal{Y}_T$ is computed as $s_n(c) = \mathbf{w}_c^\top \phi(x; \mathcal{A}_n)$, where $\mathbf{w}_c$ is the classifier weight for class c. These logits are then transformed into a predictive distribution $\mathbf{z}_n$ via the softmax function:

$$z_n(c) = \frac{\exp(s_n(c))}{\sum_{j\in\mathcal{Y}_T} \exp(s_n(j))}. \tag{9}$$

The expert's predictive confidence is inversely measured by the entropy of this distribution:

$$H(z_n) = -\sum_{c\in\mathcal{Y}_T} z_n(c) \log z_n(c). \tag{10}$$

The inference process begins with a parallel search, performed independently for each of the K trees. Starting from each tree's root, the search path follows the child expert that shows lower predictive entropy for the given input. This recursive selection defines an optimal inference path $\mathcal{P}_k$ from root to leaf for each tree. Further details on this entropy-guided selection process are provided in Appendix C.2.

The final prediction is derived from an activated set of experts $\mathcal{E}$ which includes the global root R_G and all experts from the K identified paths:

$$\mathcal{E} = \{R_G\} \cup \bigcup_{k=1}^{K} \mathcal{P}_k. \tag{11}$$

The fusion weight w_n for each activated expert is based on its confidence (negative entropy), normalized across the set with a temperature τ:

$$w_n = \frac{\exp(-H(z_n)/\tau)}{\sum_{j\in\mathcal{E}} \exp(-H(z_j)/\tau)}. \tag{12}$$

The final predictive distribution z_f is the weighted average of the predictions from all activated experts:

$$z_f = \sum_{n\in\mathcal{E}} w_n \mathbf{z}_n. \tag{13}$$

This entropy-guided fusion enables robust predictions by dynamically weighting experts across all hierarchy levels.

3.3 Theoretical Analysis

In this section, we provide theoretical intuition and analytical support for our parameter fusion strategy and the prototype-guided hierarchical construction. While the analysis involves idealized assumptions, it offers principled motivation for the design choices in SAEF. Detailed proofs are in Appendix E.

Property 1 (*Adapter Subspace as a Proxy*)**.** *The subspace spanned by the optimized adapter weights* $\mathbf{w}$ *lies within the space spanned by the input data* X*, suggesting* $\mathbf{w}$ *as a useful proxy for the original input subspace.*

Our primary motivation for parameter fusion is to prevent the response of the original data X on the fused principal feature directions from decaying. However, in the Exemplar-Free CIL setting, X of past tasks is inaccessible. Inspired by Property 1 and following Yang et al. (2025), we suggest using adapter weights as approximate proxies for task input subspaces; all adapter matrices are treated as flattened vectors $\mathbf{w}_i \in \mathbb{R}^D$, and we define the Energy Retention ratio as follows.

Definition 1 (*Energy Retention*)**.** Let $\mathbf{w}_i \in \mathbb{R}^D$ denote the vectorized (flattened) adapter parameters capturing the principal feature directions of task i, and $\mathbf{w}_j \in \mathbb{R}^D$ the corresponding parameters for task j. The Energy Retention ratio ρ for task i after merging into the fused adapter $\mathbf{w}_{i+j}$ is defined as:

$$\rho = \frac{\mathbf{w}_{i+j}^\top \mathbf{w}_i}{\|\mathbf{w}_i\|_2^2}. \tag{14}$$

When $\rho \geq 1$, the fused adapter preserves at least as much of task i's feature energy as the original adapter, which is favorable for knowledge retention and may support positive forward transfer.

Property 2 (*Sign-Aware Energy Retention*)**.** Based on Definition 1, the energy retention ratio ρ of the sign-aware fused expert admits the following decomposition:

$$\rho = 1 + \frac{\mathcal{G}_{\text{syn}} - \mathcal{L}_{\text{conf}}}{\|\mathbf{w}_i\|_2^2}, \tag{15}$$

where $\mathcal{G}_{\text{syn}} = \sum_{d \in \mathcal{A}^+} |\mathbf{w}_{i,d}|(|\mathbf{w}_{j,d}| - |\mathbf{w}_{i,d}|)$ represents the synergistic gain from the co-activated dimension set $\mathcal{A}^+$ with identical signs and $|\mathbf{w}_{j,d}| > |\mathbf{w}_{i,d}|$. Conversely, $\mathcal{L}_{\text{conf}} = \sum_{d \in \mathcal{C}^-} |\mathbf{w}_{i,d}|(|\mathbf{w}_{j,d}| + |\mathbf{w}_{i,d}|)$ is the conflict loss from the dimension set $\mathcal{C}^-$ with opposing signs and $|\mathbf{w}_{j,d}| > |\mathbf{w}_{i,d}|$. By Property 1, high prototype similarity $\text{sim}(i, j)$ implies aligned adapter subspaces, which reduces the conflict set $\mathcal{C}^-$ and suppresses $\mathcal{L}_{\text{conf}}$, making $\rho \geq 1$ more likely. Further discussion is provided in Appendix E.

Property 3 (*Controlled Structural Deviation*)**.** Let $\mathbf{w}_{i+j}$ denote the fused expert of two flattened adapter vectors $\mathbf{w}_i, \mathbf{w}_j \in \mathbb{R}^D$. Under the assumption that $|\mathbf{w}_{i,d}| \neq |\mathbf{w}_{j,d}|$ for almost every dimension d (a generic condition we assume holds in practice),

$$\|\mathbf{w}_{i+j} - \mathbf{w}_i\|_2^2 + \|\mathbf{w}_{i+j} - \mathbf{w}_j\|_2^2 = \|\mathbf{w}_i - \mathbf{w}_j\|_2^2. \tag{16}$$

Together with Property 1, this identity motivates our greedy construction: if prototype directions approximate adapter subspace proxies, maximizing prototype similarity approximately minimizes the pairwise distance between merged experts, thereby limiting the structural deviation introduced at each merge. A detailed derivation is provided in Appendix E.4.

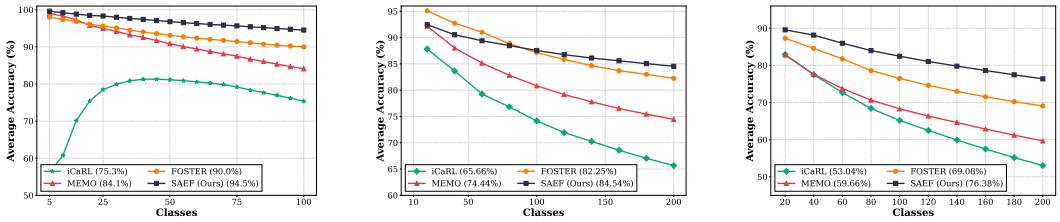
4 Experiments

4.1 Implementation Details

Datasets and Task Protocols. We evaluate on four benchmarks following Zhou et al. (2024a): CIFAR-100 Krizhevsky et al. (2009) (100 classes, 20 tasks of 5), ImageNet-R Hendrycks et al. (2021a), ImageNet-A Hendrycks et al. (2021b), and ObjectNet Barbu et al. (2019) (each 200 classes, 10 tasks of 20). The latter two exhibit significant domain shifts, providing a challenging out-of-distribution evaluation. Class orders are shuffled with a fixed seed of 1993 Rebuffi et al. (2017b).

Table 1: Average ($\bar{\mathcal{A}}$) and final-task ($\mathcal{A}_T$) Top-1 accuracy (%) on four exemplar-free CIL benchmarks using ViT-B/16-IN21K. Best results are in **bold**.

Method	CIFAR-100		ImageNet-R		ImageNet-A		ObjectNet	
	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
Finetune	38.90	20.17	32.32	22.78	24.28	14.51	19.14	8.73
L2P Wang et al. (2022c)	85.94	79.93	75.46	69.77	49.39	41.71	63.78	52.19
DualPrompt Wang et al. (2022b)	87.87	81.15	73.10	67.18	53.71	41.67	59.27	49.33
CODA-Prompt Smith et al. (2023)	89.11	81.96	77.97	72.27	53.54	42.73	66.07	53.29
SLCA Zhang et al. (2023)	92.49	88.55	81.17	77.00	68.66	58.74	72.55	61.30
SSIAT Tan et al. (2024)	93.52	90.07	83.20	78.85	70.83	62.23	73.65	62.45
SimpleCIL Zhou et al. (2024a)	87.57	81.26	61.26	54.55	59.77	48.91	65.45	53.59
APER + Adapter Zhou et al. (2024a)	90.65	85.15	75.82	67.95	60.47	49.37	67.18	55.24
EASE Zhou et al. (2024b)	91.51	85.80	81.74	76.17	65.34	55.04	70.84	57.86
MOS Sun et al. (2025)	93.30	89.25	82.96	77.93	67.08	56.22	74.69	63.62
SAEF (Ours)	**94.53**	**90.60**	**84.54**	**80.02**	**73.39**	**64.91**	**76.38**	**66.58**



(a) CIFAR-100 (b) ImageNet-R (c) ObjectNet

Figure 3: Comparison against rehearsal-based methods. Average accuracy ($\bar{\mathcal{A}}$) is plotted as a function of the number of learned tasks. Rehearsal-based methods use 20 exemplars per class. Despite being an exemplar-free method, SAEF consistently outperforms strong rehearsal-based competitors across multiple benchmarks, and the performance gap often widens over time, demonstrating superior knowledge retention and scalability.

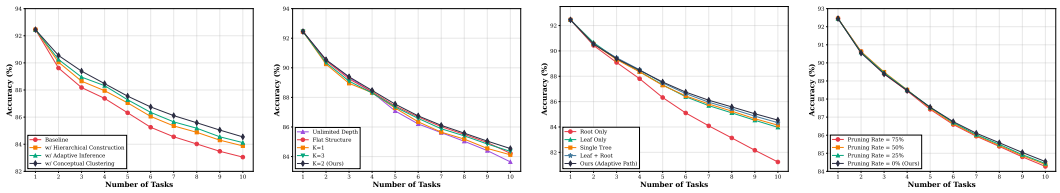
Training Details. All models use a ViT-B/16 backbone pre-trained on ImageNet-21K, implemented in PyTorch Paszke et al. (2019) via the PILOT codebase Sun et al. (2023). Each task is trained for 20 epochs (batch size 48, SGD with cosine annealing, lr=0.01). The adapter bottleneck dimension is fixed at $r = 16$ for all methods. SAEF's cluster count K is auto-selected via silhouette score; fusion temperature $\tau = 1.0$. The SAEF hierarchy is reconstructed after each task is trained, using only the adapters and prototypes accumulated up to that point. Results with a ViT-B/16 backbone pre-trained on ImageNet-1K are provided in Appendix I.

Comparison Methods. We benchmark SAEF against prominent PTM-based CIL methods: L2P Wang et al. (2022c), DualPrompt Wang et al. (2022b), CODA-Prompt Smith et al. (2023), SimpleCIL Zhou et al. (2024a), APER Zhou et al. (2024a), SLCA Zhang et al. (2023), SSIAT Tan et al. (2024), EASE Zhou et al. (2024b), and MOS Sun et al. (2025). We also include rehearsal-based methods FOSTER Wang et al. (2022a), MEMO Zhou et al. (2023), and iCaRL Rebuffi et al. (2017b), and a sequential Finetune baseline.

Evaluation Metrics. We report Average Accuracy ($\bar{\mathcal{A}}$) and Final Accuracy ($\mathcal{A}_T$) following the standard protocol Rebuffi et al. (2017b); formal definitions are in Appendix F.

4.2 Main Results

We first evaluate SAEF against SOTA exemplar-free methods, with detailed results in Table 1. Our method achieves the highest scores in both average accuracy ($\bar{\mathcal{A}}$) and final task accuracy ($\mathcal{A}_T$) across all four benchmarks. SAEF surpasses not only strong prompt-based methods like CODA-Prompt but also SOTA adapter-based competitors. For instance, on the challenging ObjectNet benchmark, SAEF achieves a final task accuracy of 66.58%, outperforming the strongest competitor MOS by a significant



(a) Ablation of SAEF Components (b) Analysis of Forest Structure (c) Path Strategy Comparison (d) Entropy-based Node Pruning

Figure 4: Comprehensive ablation and structural analysis of SAEF on ImageNet-R. Performance is measured by average accuracy ($\bar{A}$). **(a)** Ablation study demonstrating the progressive contribution of each core component. **(b)** Structural analysis validating our balanced forest structure ($K = 2$) over less structured alternatives. **(c)** Comparison of different path integration strategies, highlighting the necessity of combining intermediate and cross-domain semantic nodes. **(d)** Impact of pruning high-entropy nodes from the output space representation, confirming that parent nodes contribute non-redundant predictive information that is not recoverable from leaf experts alone.

2.96%. This advantage stems from SAEF's ability to organize experts into a semantic hierarchy, which facilitates targeted positive transfer among conceptually related tasks while preventing negative interference from irrelevant ones. In contrast, flat methods like MOS and EASE treat all prior knowledge equally, leading to suboptimal predictions from unstructured ensembles.

We further compare SAEF with strong rehearsal-based methods to test its scalability, even though SAEF operates in the more difficult exemplar-free setting. We follow Zhou et al. (2024b) to set the exemplar number to 20 per class for these methods. As shown in Figure 3, SAEF still outperforms these powerful competitors. On the long 10-task ObjectNet sequence, SAEF's final average accuracy reaches 76.38%, surpassing the strong FOSTER baseline by a substantial 7.3%. Critically, the performance gap between SAEF and rehearsal-based methods widens as more tasks are learned. This suggests that structured knowledge retrieval scales better than instance replay for long-sequence CIL.

4.3 Ablation Study

To dissect the contribution of each key component in SAEF, we conduct a progressive ablation study on ImageNet-R, with results visualized in Figure 4(a). Our analysis begins with a 'Baseline' that trains task-specific adapters and selects the final prediction via the maximum logit from a full ensemble. Building upon this, we incrementally propose our components. First, organizing the flat adapter collection into a single hierarchical tree ('w/ Hierarchical Construction') yields an immediate performance uplift. This demonstrates the inherent value of structured knowledge representation, as it allows for the creation of generalized experts at internal nodes, even without semantic pre-sorting. Next, incorporating our entropy-driven adaptive search and fusion strategy ('w/ Adaptive Inference') delivers a more substantial gain. This highlights the advantage of dynamically activating only the most relevant experts for a given sample, effectively pruning noisy or irrelevant knowledge paths. Finally, the full SAEF model is formed by introducing the initial stage of semantic clustering ('w/ Conceptual Clustering'), which achieves the highest performance. This final improvement confirms that pre-grouping tasks into conceptual domains prevents semantically disparate experts from being merged, reducing noise in the local hierarchies. Each component thus plays a distinct and synergistic role in the full SAEF pipeline.

4.4 Discussion

Impact of the Expert Forest's Structure. We analyze the impact of key structural design choices on SAEF's performance, with results on ImageNet-R presented in Figure 4(b). Our analysis first examines the number of conceptual clusters, K. The results show that our default configuration ($K = 2$), automatically determined via the silhouette coefficient, achieves the best performance. Deviating from this, both a single monolithic tree ($K = 1$) and a completely 'Flat Structure' ($K = T$, where each task is its own cluster) impair performance. This suggests that knowledge can be organized

Table 2: Analysis of the trade-off between inference efficiency and average accuracy by varying the entropy threshold τ_e on CIFAR-100. The 'Flat Structure' baseline represents a non-hierarchical method where all task adapters are queried. The setting of $\tau_e = 1.0$ (highlighted in **bold**) strikes an excellent balance.

Configuration	$\bar{A}$ (%) ↑	Avg. Depth ↓	Theo. Speedup ↑	Time (ms) ↓
Flat Structure (Baseline)	94.44	–	1.00×	2775.43
SAEF ($\tau_e = 0.0$)	94.53	4.04	2.20×	1346.51
SAEF ($\tau_e = 1.0$)	**94.24**	**1.19**	**5.92×**	**500.39**
SAEF ($\tau_e = 2.0$)	93.96	1.00	6.67×	443.00

in ways that are either too coarse or too fragmented, empirically validating our automated strategy for identifying the ideal number of conceptual domains.

We also evaluate 'Unlimited Depth', which greedily merges the two most similar nodes regardless of hierarchical level. This yields the poorest results, confirming that cross-level merges introduce semantic noise; our bottom-up construction avoids this by prioritizing same-level merges. From an efficiency standpoint, both the 'Flat Structure' and 'Unlimited Depth' strategies necessitate a search over all N experts, resulting in $O(N)$ inference complexity. Our balanced forest structure, however, enables a much faster logarithmic search within each tree ($O(\log N)$). This advantage in both accuracy and efficiency underscores the importance of SAEF's carefully designed hierarchical organization.

Trade-off between Inference Efficiency and Accuracy. To study the inference efficiency–accuracy trade-off, we introduce an optional entropy-based early-exit setting: search along a path stops when a node's predictive entropy falls below τ_e. Raising τ_e increases early exits and reduces search depth. As shown in Table 2, at $\tau_e = 1.0$, this setting reduces average search depth to 1.19 and yields a nearly $6\times$ theoretical speedup over the 'Flat Structure' baseline, which queries all N task adapters, with only a 0.29% drop in accuracy. Measured inference-time reductions closely match the theoretical estimate (see Appendix G), suggesting that upper-level experts often suffice for confident predictions. Further efficiency and resource analyses, including training throughput and storage overhead, are reported in Appendix J.

Role and Semantic Meaning of Parent Nodes. We verify that intermediate (parent) nodes contribute meaningful predictions rather than serving merely as routing heuristics. First, we compare five path strategies on ImageNet-R (Figure 4(c)): *Root Only*, *Leaf Only*, *Single Tree* (best single tree only), *Leaf + Root*, and our *Full Adaptive Path*. The advantage of *Leaf + Root* over *Leaf Only* confirms the Global Root's semantic value, while *Full Adaptive Path* outperforming *Single Tree* demonstrates that cross-domain knowledge integration is beneficial. Second, we prune high-entropy nodes from the activated path at varying ratios $q \in \{0\%, 25\%, 50\%, 75\%\}$, affecting only the output aggregation (not routing). As shown in Figure 4(d), accuracy monotonically decreases with pruning ratio, confirming that parent nodes encode geometrically distinct, task-relevant features rather than ambiguous noise. Full experimental details and analysis are provided in Appendix L. Additional analyses quantify expert-selection accuracy and assess the importance of selecting the exact adapter path, showing that SAEF maintains strong classification through shared internal-node evidence even when leaf selection is imperfect; see Appendix Q for details.

5 Conclusion

Effectively reusing knowledge in adapter-based CIL remains challenging. Although adapters mitigate catastrophic forgetting, treating them as a flat collection leads to indiscriminate knowledge sharing. We propose SAEF, which organizes adapters into a semantic hierarchy by automatically constructing a forest of balanced expert trees that group and progressively merge related tasks. During inference, SAEF activates relevant expert paths for targeted transfer while limiting interference from unrelated tasks. Experiments on major CIL benchmarks show that SAEF outperforms current SOTA methods, validating hierarchical knowledge organization for CIL. Limitations and broader impacts are discussed in Appendices A and B.

AI USE STATEMENT

We used large language models to assist with manuscript drafting, language editing, formatting, and literature review.

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision*, pp. 139–154, 2018.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *NeurIPS*, 32, 2019.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International conference on computer vision*, pp. 9516–9525, 2021.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision*, pp. 532–547, 2018.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *International Conference on Learning Representations*, 2019.

Arslan Chaudhry, Albert Gordo, Puneet Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6993–7001, 2021.

Eric Nuertey Coleman, Luigi Quarantiello, Samrat Mukherjee, Julio Hurtado, and Vincenzo Lomonaco. Ham: Hierarchical adapter merging for scalable continual learning. *arXiv preprint arXiv:2509.13211*, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim K Goh, Ho-Kin Tang, Daojing He, et al. Parameter competition balancing for model merging. *Advances in Neural Information Processing Systems*, 37:84746–84776, 2024.

Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. In *Proceedings of the European conference on computer vision*, pp. 386–402. Springer, 2020.

Zhiwen Fan, Rishov Sarkar, Ziyu Jiang, Tianlong Chen, Kai Zou, Yu Cheng, Cong Hao, Zhangyang Wang, et al. M^3vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. *Advances in Neural Information Processing Systems*, 35:28441–28457, 2022.

Yasir Ghunaim, Adel Bibi, Kumail Alhamoud, Motasem Alfarra, Hasan Abed Al Kader Hammoud, Ameya Prabhu, Philip HS Torr, and Bernard Ghanem. Real-time evaluation in online continual learning: A new hope. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11888–11897, 2023.

Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34:29335–29347, 2021.

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 *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8340–8349, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15262–15271, 2021b.

Nhat Ho, Chiao-Yu Yang, and Michael I Jordan. Convergence rates for gaussian mixtures of experts. *Journal of Machine Learning Research*, 23(323):1–81, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Tianyu Huai, Jie Zhou, Xingjiao Wu, Qin Chen, Qingchun Bai, Ze Zhou, and Liang He. Cl-moe: Enhancing multimodal large language model with dual momentum mixture-of-experts for continual visual question answering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19608–19617, 2025.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Zixuan Ke, Bing Liu, and Xingchang Huang. Continual learning of a mixed sequence of similar and dissimilar tasks. *Advances in neural information processing systems*, 33:18493–18504, 2020.

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

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

Qiwei Li, Yuxin Peng, and Jiahuan Zhou. Fcs: Feature calibration and separation for non-exemplar class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 28495–28504, 2024.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 40(12):2935–2947, 2017.

Hangda Liu, Boyu Diao, Wenxin Chen, and Yongjun Xu. A resource-aware workload scheduling method for unbalanced gemms on gpus. *The Computer Journal*, 68(3):273–282, 2025.

Noel Loo, Siddharth Swaroop, and Richard E Turner. Generalized variational continual learning. *International Conference on Learning Representations*, 2021.

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. In *European Conference on Computer Vision*, pp. 379–395. Springer, 2024.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton Van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36:12022–12053, 2023.

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*, pp. 8026–8037, 2019.

Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet K Dokania, Philip HS Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally budgeted continual learning: What does matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3698–3707, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *NeurIPS*, 30, 2017a.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pp. 2001–2010, 2017b.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pp. 4548–4557. PMLR, 2018.

Ketan Rajshekhar Shahapure and Charles Nicholas. Cluster quality analysis using silhouette score. In *2020 IEEE 7th international conference on data science and advanced analytics (DSAA)*, pp. 747–748. IEEE, 2020.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321, 2015.

James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9374–9384, 2021.

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*, pp. 11909–11919, 2023.

Andrea Soltoggio, Eseoghene Ben-Iwhiwhu, Vladimir Braverman, Eric Eaton, Benjamin Epstein, Yunhao Ge, Lucy Halperin, Jonathan How, Laurent Itti, Michael A Jacobs, et al. A collective ai via lifelong learning and sharing at the edge. *Nature Machine Intelligence*, 6(3):251–264, 2024.

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.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20699–20707, 2025.

Yuwen Tan, Qinhao Zhou, Xiang Xiang, Ke Wang, Yuchuan Wu, and Yongbin Li. Semantically-shifted incremental adapter-tuning is a continual vitransformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23252–23262, 2024.

Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *European conference on computer vision*, pp. 398–414. Springer, 2022a.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10087–10098. IEEE, 2025.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023.

Zifeng Wang, Tong Jian, Kaushik Chowdhury, Yanzhi Wang, Jennifer Dy, and Stratis Ioannidis. Learn-prune-share for lifelong learning. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 641–650. IEEE, 2020.

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. *ECCV*, 2022b.

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*, pp. 139–149, 2022c.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning. In *International Conference on Learning Representations*, volume 2025, pp. 58337–58353, 2025.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Chuanguang Yang, Zhulin An, Libo Huang, Junyu Bi, Xinqiang Yu, Han Yang, Boyu Diao, and Yongjun Xu. Clip-kd: An empirical study of clip model distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15952–15962, 2024.

Enneng Yang, Anke Tang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, and Jie Zhang. Continual model merging without data: Dual projections for balancing stability and plasticity. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6982–6991, 2020.

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 *ICCV*, 2023.

Tingting Zhao, Zifeng Wang, Aria Masoomi, and Jennifer Dy. Deep bayesian unsupervised lifelong learning. *Neural Networks*, 149:95–106, 2022.

Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *ICLR*, 2023.

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*, 2024a.

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*, pp. 23554–23564, 2024b.

Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, 34:14306–14318, 2021.

A LIMITATIONS

Use of semantic priors. SAEF uses CLIP Radford et al. (2021) text embeddings to group tasks. Ambiguous or domain-specific class names may yield less reliable semantic prototypes and affect the resulting hierarchy.

Hierarchical construction. The bottom-up greedy strategy makes hierarchy construction practical, but does not guarantee a globally optimal tree. Alternative construction strategies may produce different expert groupings.

Adapter storage. SAEF retains task-specific adapters and adds internal-node adapters, so storage grows approximately linearly with the number of tasks. This overhead may matter for long task sequences or memory-constrained deployments.

Backbone architectures. Our experiments use a ViT-B/16 backbone. Whether the observed gains extend to other architectures remains to be evaluated.

B BROADER IMPACTS

SAEF contributes to more efficient and accurate class-incremental learning, allowing models to continuously incorporate new knowledge without extensive retraining. This capability is particularly useful for resource-constrained and dynamically changing environments, including on-device learning, robotics, and edge AI applications. In addition, the adaptive inference strategy reduces unnecessary computation and improves deployment efficiency, which may facilitate broader practical adoption of continual learning systems. Meanwhile, as with general recognition and adaptation techniques, SAEF should be applied with care in sensitive domains involving personal data or high-stakes decisions. Practical deployment should follow appropriate ethical guidelines, privacy protections, and relevant regulatory requirements.

C DETAILED IMPLEMENTATION OF SAEF OPERATIONS

In this appendix, we provide the detailed implementation specifics for the *Hierarchical Construction* and *Adaptive Inference* stages, which are described at a higher level of abstraction in the main text. We also detail the prototype-based classifier and its alignment mechanism, which are foundational to our method.

C.1 HIERARCHICAL CONSTRUCTION DETAILS

The tree construction process operates on a working set of nodes $\mathcal{N}_k$ for each cluster G_k. Each node stores an expert adapter, a visual prototype, its number of descendant leaf tasks, and a subtree height h used to order merges. Parent nodes also store pointers to their children. This construction-time height is distinct from a node's depth measured from the root of the completed tree.

Node Representation. A leaf corresponding to task t is initialized as $node_t = (\mathcal{A}_t, \mathbf{p}_t^{\mathrm{f}}, m_t = 1, h_t = 0)$. A parent formed from $node_i$ and $node_j$ is represented as $node_p = (\mathcal{A}_p, \mathbf{p}_p^{\mathrm{f}}, m_p, h_p, node_i, node_j)$, where

$$m_p = m_i + m_j, \qquad h_p = 1 + \max\{h_i, h_j\}.$$

Node Selection. At each iteration, we first consider the available nodes with the smallest subtree height:

$$h_{\min} = \min_{n \in \mathcal{N}_k} h(n), \qquad \mathcal{L}_k = \{n \in \mathcal{N}_k : h(n) = h_{\min}\}. \tag{17}$$

If $|\mathcal{L}_k| \geq 2$, we merge the pair in $\mathcal{L}_k$ with the highest visual-prototype similarity. Otherwise, letting $\mathcal{L}_k = \{n_i\}$, we pair n_i with its most similar node at the next smallest available subtree height:

$$(n_i^*, n_j^*) = \begin{cases} \arg\max_{\{n_i, n_j\} \subseteq \mathcal{L}_k} \mathrm{sim}(n_i, n_j), & |\mathcal{L}_k| \geq 2, \\ \left(n_i, \arg\max_{n_j \in \mathcal{L}_k'} \mathrm{sim}(n_i, n_j)\right), & \mathcal{L}_k = \{n_i\}, \end{cases} \tag{18}$$

where $h' = \min_{n \in \mathcal{N}_k \setminus \{n_i\}} h(n)$ and $\mathcal{L}'_k = \{n \in \mathcal{N}_k : h(n) = h'\}$ in the second case. Thus, merges between nodes of equal subtree height take priority; cross-height merges occur only when a single node remains at the smallest available height.

Prototype Merging. When a parent is created from $node_i^*$ and $node_j^*$, its visual prototype is the leaf-count-weighted average of its children's prototypes. Let $C(node)$ be the set of leaf tasks descending from $node$. Then

$$\mathbf{p}_p^{\mathrm{f}} = \frac{|C(node_i^*)|\mathbf{p}_{i^*}^{\mathrm{f}} + |C(node_j^*)|\mathbf{p}_{j^*}^{\mathrm{f}}}{|C(node_i^*)| + |C(node_j^*)|}. \tag{19}$$

The selected children are removed from $\mathcal{N}_k$ and replaced with their parent. The process continues until one node remains.

C.2 Adaptive Inference Details

Entropy-Guided Path Selection. During inference, SAEF performs a search to find the optimal path in each of the K trees. The search begins at the root of a tree and recursively descends to one of its children. At any non-leaf node (indexed by p) with children (indexed by L and R for left and right), the search proceeds to the child node that exhibits lower predictive entropy for the given input sample x. The formal rule for selecting the next node in the path is:

$$node_{\text{next}} = \underset{n \in \{node_L, node_R\}}{\arg\min} H(\mathbf{z}_n), \tag{20}$$

where $H(\mathbf{z}_n)$ is the Shannon entropy of the prediction from the expert at node n, as defined in Eq. (10). This process is repeated until a leaf node is reached, defining a complete path from root to leaf.

C.3 Prototype-based Classifier and Alignment

Our method relies on a non-parametric classifier constructed from class prototypes. Here, we detail how these prototypes are created and aligned across different task-specific feature spaces.

Classifier Construction for New Classes. For any new class c introduced in the current task t with training samples $\mathcal{D}_c$, its prototype $\mathbf{p}_c$ is computed as the mean feature vector of its samples, extracted using the current task's adapter $\mathcal{A}_t$:

$$\mathbf{p}_c = \frac{1}{|\mathcal{D}_c|} \sum_{x \in \mathcal{D}_c} \phi(x; \mathcal{A}_t). \tag{21}$$

The corresponding classifier weight $\mathbf{w}_c$ is simply its L2-normalized prototype: $\mathbf{w}_c = \mathbf{p}_c / \|\mathbf{p}_c\|_2$.

Classifier Alignment for Old Classes. The non-parametric classifier requires class prototypes to be comparable in a shared feature space. To avoid storing past exemplars, we follow the distributional replay strategy used in prior work such as MOS Sun et al. (2025). For each past class c', we store Gaussian feature statistics $\mathcal{N}(\mu_{c'}, \Sigma_{c'})$ computed when the class is first learned. When learning task t, we sample pseudo-features from these stored distributions and use them to update the prototypes of old classes. This yields an aligned global classifier matrix $W_t = [\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{Y}_t|}]$, where all class weights are comparable for logit computation.

D SAEF Algorithm Pseudocode

We present SAEF as two separate algorithms. Algorithm 1 updates the hierarchy after each completed task using only tasks observed so far. Algorithm 2 performs adaptive inference for each test sample with the hierarchy available at the current task.

Algorithm 1 SAEF: Hierarchy Update After Each Task

Inputs: Completed task t, its adapter $\mathcal{A}_t$ with parameters θ_t, current data $\mathcal{D}_t$ and classes $\mathcal{C}_t$; frozen text encoder g_{text}, frozen feature extractor ϕ, cluster search range K_{range}
Persistent state: Cached prototypes $\mathcal{P}_s, \mathcal{P}_f$ (empty before task 1), and parameters $\{\theta_s\}_{s=1}^{t-1}$
Output: Current hierarchy $\text{Hierarchy}_t = (\mathcal{R}_t, R_{G,t})$

procedure UPDATEHIERARCHY($t, \mathcal{A}_t, \mathcal{D}_t, \mathcal{C}_t, g_{\text{text}}, \phi, K_{\text{range}}$)
▷ Stage 1: update prototypes using only the current task
  $\mathbf{p}_t^{\text{s}} \leftarrow \frac{1}{|\mathcal{C}_t|} \sum_{c \in \mathcal{C}_t} g_{\text{text}}(\text{"a photo of a [c]"})$
  $\mathbf{p}_t^{\text{f}} \leftarrow \frac{1}{|\mathcal{D}_t|} \sum_{\mathbf{x} \in \mathcal{D}_t} \phi(\mathbf{x}; \mathcal{A}_1)$
  Cache $(t, \mathbf{p}_t^{\text{s}})$ in $\mathcal{P}_s$ and $(t, \mathbf{p}_t^{\text{f}})$ in $\mathcal{P}_f$
  $\mathcal{K}_t \leftarrow K_{\text{range}} \cap \{2, \ldots, t-1\}$
  if $\mathcal{K}_t = \emptyset$ **then**
    $K_t \leftarrow 1$ ▷ Silhouette score is undefined otherwise
  else
    $K_t \leftarrow \text{FINDOPTIMALK}(\mathcal{P}_s, \mathcal{K}_t)$ ▷ Maximize silhouette score
  $\{G_1, \ldots, G_{K_t}\} \leftarrow \text{KMEANS}(\mathcal{P}_s, K_t)$ ▷ Partition tasks $1, \ldots, t$
▷ Stage 2: rebuild trees from cached information
  $\mathcal{R}_t \leftarrow \emptyset$
  for $k = 1, \ldots, K_t$ **do**
    $R_k \leftarrow \text{BUILDTREE}(G_k, \{\theta_s, \mathbf{p}_s^{\text{f}}\}_{s \in G_k})$
    $\mathcal{R}_t \leftarrow \mathcal{R}_t \cup \{R_k\}$
  if $K_t = 1$ **then**
    $R_{G,t} \leftarrow R_1$
  else
    $\theta_G \leftarrow \text{sign}\left(\sum_{k=1}^{K_t} \theta_{R_k}\right) \odot \max_{k=1,\ldots,K_t} |\theta_{R_k}|$ ▷ Eq. (8)
    $R_{G,t} \leftarrow \text{NEWEXPERT}(\theta_G)$
  return $(\mathcal{R}_t, R_{G,t})$

procedure BUILDTREE($G, \{\theta_s, \mathbf{p}_s^{\text{f}}\}_{s \in G}$)
  $\mathcal{N} \leftarrow \{\text{LEAF}(\theta_s, \mathbf{p}_s^{\text{f}}, m = 1, h = 0) : s \in G\}$ ▷ m: leaf count; h: subtree height
  while $|\mathcal{N}| > 1$ **do**
    $h_{\min} \leftarrow \min_{n \in \mathcal{N}} h_n$; $\mathcal{L} \leftarrow \{n \in \mathcal{N} : h_n = h_{\min}\}$
    if $|\mathcal{L}| \geq 2$ **then**
      $(n_i, n_j) \leftarrow \arg\max_{\{a,b\} \subseteq \mathcal{L}} \text{sim}(a, b)$ ▷ Same-height pair first
    else
      $n_i \leftarrow$ the sole node in $\mathcal{L}$
      $h' \leftarrow \min_{n \in \mathcal{N} \setminus \{n_i\}} h_n$
      $n_j \leftarrow \arg\max_{n \in \mathcal{N}:\, h_n = h'} \text{sim}(n_i, n)$ ▷ Next-smallest-height fallback
    $\theta_p \leftarrow \text{sign}(\theta_i + \theta_j) \odot \max(|\theta_i|, |\theta_j|)$ ▷ Eq. (7)
    $m_p \leftarrow m_i + m_j$; $\mathbf{p}_p^{\text{f}} \leftarrow (m_i \mathbf{p}_i^{\text{f}} + m_j \mathbf{p}_j^{\text{f}})/m_p$
    $h_p \leftarrow 1 + \max(h_i, h_j)$
    $n_p \leftarrow \text{NEWNODE}(\theta_p, \mathbf{p}_p^{\text{f}}, m_p, h_p, \text{children} = (n_i, n_j))$
    $\mathcal{N} \leftarrow (\mathcal{N} \setminus \{n_i, n_j\}) \cup \{n_p\}$
  return the sole node in $\mathcal{N}$

D.1 ALGORITHMIC DETAILS

Hierarchy Update After Each Task. After learning task t, Algorithm 1 computes its semantic prototype from $\mathcal{C}_t$ and its visual prototype from $\mathcal{D}_t$, then caches both prototypes. It reclusters the semantic prototypes of tasks $1, \ldots, t$, rebuilds each cluster's tree from cached prototypes and adapter parameters, and forms the current global expert. No past training data or future task labels are accessed. When no valid silhouette-score candidate exists, the tasks form a single cluster.

Algorithm 2 SAEF: Online Adaptive Inference

Inputs:
Test sample x
The pre-built 'Hierarchy'$= (\mathcal{R}, R_G)$ from Algorithm 1
Frozen feature extractor ϕ, Classifier W_T
Fusion temperature τ, Entropy threshold τ_e
Output: The final predicted class label y_{pred}

procedure INFERENCE($x, (\mathcal{R}, R_G), \phi, W_T, \tau, \tau_e$)
▷ — **Stage 3: Adaptive Inference** —
  Initialize activated expert set: $\mathcal{E} \leftarrow \{R_G\}$ ▷ Start with the Global Expert
  for each tree root $R_k \in \mathcal{R}$ **do**
    $\mathcal{P}_k \leftarrow$ FindPath$(R_k, x, \phi, W_T, \tau_e)$ ▷ Find low-entropy path in each tree
    $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{P}_k$ ▷ Collect all experts on the paths, Eq. (11)
  Initialize results list: $\mathcal{S} \leftarrow []$
  for each expert $n \in \mathcal{E}$ **do**
    $s_n \leftarrow W_T^\top \phi(x; \mathcal{A}_n)$ ▷ Compute logits for all classes $\mathcal{Y}_T$
    $\mathbf{z}_n \leftarrow$ Softmax(s_n) ▷ Get prediction distribution, Eq. (9)
    $H_n \leftarrow -\sum_c z_n(c) \log z_n(c)$ ▷ Calculate confidence via entropy, Eq. (10)
    Add $(\mathbf{z}_n, H_n)$ to $\mathcal{S}$
  $Z_{sum} \leftarrow \sum_{(\mathbf{z}_j, H_j) \in \mathcal{S}} \exp(-H_j/\tau)$ ▷ Compute normalization term for weights
  $\mathbf{z}_f \leftarrow \mathbf{0}$
  for each $(\mathbf{z}_n, H_n) \in \mathcal{S}$ **do**
    $w_n \leftarrow \exp(-H_n/\tau)/Z_{sum}$ ▷ Compute fusion weight, Eq. (12)
    $\mathbf{z}_f \leftarrow \mathbf{z}_f + w_n \cdot \mathbf{z}_n$ ▷ Aggregate predictions, Eq. (13)
  return $\arg\max \mathbf{z}_f$

procedure FINDPATH($node, x, \phi, W_T, \tau_e$) ▷ Recursive helper function for path selection
  $s_{node} \leftarrow W_T^\top \phi(x; \mathcal{A}_{node})$
  $H_{node} \leftarrow$ Entropy(Softmax(s_{node}))
  if $node$ is a leaf **then** ▷ Early exit if confident or at a leaf
    return $\{node\}$
  else
    $(n_L, n_R) \leftarrow$ children of $node$
    $s_L \leftarrow W_T^\top \phi(x; \mathcal{A}_L)$; $H_L \leftarrow$ Entropy(Softmax(s_L))
    $s_R \leftarrow W_T^\top \phi(x; \mathcal{A}_R)$; $H_R \leftarrow$ Entropy(Softmax(s_R))
    if $H_L < H_R$ **then**
      return $\{node\} \cup$ FindPath$(n_L, x, \phi, W_T, \tau_e)$
    else
      return $\{node\} \cup$ FindPath$(n_R, x, \phi, W_T, \tau_e)$

Height-Aware Tree Construction. The construction-time subtree height h of a leaf is zero, and a parent's height is one plus the maximum height of its children. BUILDTREE processes nodes at the smallest available height first, merging the most visually similar pair at that height. If one node is left without a same-height partner, it is merged with the most similar node at the next smallest available height. Each parent's visual prototype is averaged using the numbers of original leaf tasks represented by its children. This merge order prioritizes similarly sized subtrees and keeps the resulting tree height $O(\log |G_k|)$.

Online Adaptive Inference. Algorithm 2 details the online prediction mechanism for a given input sample x. The `Inference` procedure first identifies the set of all relevant experts, $\mathcal{E}$ (Lines 4-8). This involves starting with the global expert and then traversing each of the K^* trees to find the optimal path using the recursive helper function `FindPath` (Line 6). The `FindPath` procedure (Lines 23-35) implements the entropy-guided traversal, including the early-exit mechanism controlled by the threshold τ_e (Line 26). Once the set of activated experts is finalized, the algorithm computes the predictive distribution $\mathbf{z}_n$ and corresponding confidence (entropy H_n) for each expert (Lines

10-14). Finally, it calculates the normalized, confidence-based fusion weights w_n and computes the final prediction as a weighted average of all activated experts' outputs (Lines 16-20), directly implementing the fusion logic from Eq. (12) and Eq. (13).

E Detailed Theoretical Proofs

In this section, we provide the complete mathematical proofs and expanded discussions for the properties stated in Section 3.3 of the main paper.

E.1 Proof of Property 1

Property 1 (*Adapter Subspace as a Proxy*)**.** *The subspace spanned by the optimized adapter weights* $\mathbf{w}$ *lies within the space spanned by the input data* X*, suggesting* $\mathbf{w}$ *as a useful proxy for the original input subspace.*

Proof. Consider a linear layer representing the adapter's projection, denoted as $Z = \tilde{X}\mathbf{w}$, where $\tilde{X}$ is the projected input representation, and $\mathbf{w}$ is the weight matrix.

By applying the chain rule, the gradient of the loss function $\mathcal{L}$ with respect to the weight matrix $\mathbf{w}$ is:

$$\nabla_{\mathbf{w}}\mathcal{L} = \tilde{X}^{\top}\nabla_{Z}\mathcal{L}. \tag{22}$$

This directly indicates that the gradient space resides within the input subspace spanned by the data $\tilde{X}$. Prior work Hu et al. (2022); Wang et al. (2023) has shown that the low-rank subspace of adapter weights can serve as a proxy for the gradient subspace. Motivated by this observation and following Yang et al. (2025), we suggest treating the optimized adapter weights $\mathbf{w}$ as an approximate proxy for the principal directions of the input subspace. ■

E.2 Proof of Property 2: Sign-Aware Energy Retention

Property 2 (*Sign-Aware Energy Retention*)**.** *Based on Definition 1, the energy retention ratio* ρ *of the sign-aware fused expert admits the following decomposition:* $\rho = 1 + \frac{\mathcal{G}_{\text{syn}} - \mathcal{L}_{\text{conf}}}{\|\mathbf{w}_i\|_2^2}$.

Proof. We analyze the energy retention dimension-wise. Let $u = \mathbf{w}_{i,d}$ and $v = \mathbf{w}_{j,d}$. Our fusion operation is $\mathbf{w}_{i+j,d} = \text{sign}(u+v)\max(|u|,|v|)$.

The inner product is $\mathbf{w}_{i+j}^{\top}\mathbf{w}_i = \sum_d \mathbf{w}_{i+j,d}\mathbf{w}_{i,d}$. We rewrite the term for each dimension as $\mathbf{w}_{i,d}^2 + \Delta_{i,d}$, where $\Delta_{i,d}$ is the response increment. We partition all parameter dimensions into specific sets based on signs and magnitudes:

- **Set $\mathcal{A}^+$ (Synergistic Enhancement):** where $\text{sign}(u) = \text{sign}(v)$ and $|v| > |u|$. The fused weight is $\mathbf{w}_{i+j,d} = \text{sign}(u)|v|$. The increment is positive: $\Delta_{i,d} = |u|(|v| - |u|) > 0$. Summing these yields the Synergistic Gain $\mathcal{G}_{\text{syn}}$.
- **Set $\mathcal{C}^-$ (Conflict with Opponent Dominant):** where $\text{sign}(u) \neq \text{sign}(v)$ and $|v| > |u|$. The fused weight takes the sign of the opposing task j, yielding $\mathbf{w}_{i+j,d} = -\text{sign}(u)|v|$. The product becomes $-|u||v|$, and the increment is strictly negative: $\Delta_{i,d} = -|u|(|v| + |u|) < 0$. The absolute sum of these yields the Conflict Loss $\mathcal{L}_{\text{conf}}$.
- **Other Sets:** For dimensions where $|u| \geq |v|$ (regardless of sign), the fusion strictly retains u. Thus, $\mathbf{w}_{i+j,d} = u$, resulting in $\Delta_{i,d} = 0$.

Summing the increments over all dimensions, the inner product becomes $\mathbf{w}_{i+j}^{\top}\mathbf{w}_i = \|\mathbf{w}_i\|_2^2 + \mathcal{G}_{\text{syn}} - \mathcal{L}_{\text{conf}}$. Dividing both sides by $\|\mathbf{w}_i\|_2^2$ directly yields the formula for ρ. ■

Discussion. Highly similar visual prototypes indicate a strong overlap in their input subspaces. By explicitly maximizing prototype similarity $\text{sim}(i,j)$ during the hierarchical construction, our algorithm tends to prevent the merging of conflicting gradient directions. This reduces the conflict set $\mathcal{C}^-$, suppressing $\mathcal{L}_{\text{conf}}$ and making high energy retention ($\rho \geq 1$) more likely. Consequently, the prototype-guided fusion encourages subspace feature preservation and potentially exploits synergistic gains for positive knowledge transfer.

E.3 Proof of Property 3: Controlled Structural Deviation

Property 3 (*Controlled Structural Deviation*)**.** *Let $\mathbf{w}_{i+j}$ be the fused expert from $\mathbf{w}_i$ and $\mathbf{w}_j$, where $\mathbf{w}_i, \mathbf{w}_j \in \mathbb{R}^D$ are flattened adapter parameter vectors. Under the assumption that $|\mathbf{w}_{i,d}| \neq |\mathbf{w}_{j,d}|$ for almost every dimension d, the following identity holds:*

$$\|\mathbf{w}_{i+j} - \mathbf{w}_i\|_2^2 + \|\mathbf{w}_{i+j} - \mathbf{w}_j\|_2^2 = \|\mathbf{w}_i - \mathbf{w}_j\|_2^2. \tag{23}$$

Proof. Let us evaluate the deviation dimension by dimension. Let $u = \mathbf{w}_{i,d}$ and $v = \mathbf{w}_{j,d}$. Our fusion operation yields $w = \mathbf{w}_{i+j,d} = \text{sign}(u+v)\max(|u|,|v|)$.

Under the assumption that $|u| \neq |v|$ almost everywhere, we analyze the resulting value of w:

- If u and v have the same sign, w takes the value of the parameter with the larger absolute magnitude. Thus, $w \in \{u, v\}$.
- If u and v have opposite signs, $u + v$ takes the sign of the parameter with the larger absolute magnitude. Multiplying this sign by $\max(|u|,|v|)$ perfectly reconstructs the original parameter with the larger magnitude. Thus, $w \in \{u, v\}$ holds almost everywhere.

Since w equals either u or v for almost every dimension d, the sum of squared deviations for a single dimension is $(w-u)^2 + (w-v)^2$. If $w = u$: $0^2 + (u-v)^2 = (u-v)^2$. If $w = v$: $(v-u)^2 + 0^2 = (u-v)^2$.

In all cases, the local deviation sum is exactly $(u-v)^2$. Summing over all dimensions d yields:

$$\begin{aligned}\|\mathbf{w}_{i+j} - \mathbf{w}_i\|_2^2 + \|\mathbf{w}_{i+j} - \mathbf{w}_j\|_2^2 &= \sum_d \left((\mathbf{w}_{i+j,d} - \mathbf{w}_{i,d})^2 + (\mathbf{w}_{i+j,d} - \mathbf{w}_{j,d})^2\right) \\ &= \sum_d (\mathbf{w}_{i,d} - \mathbf{w}_{j,d})^2 \\ &= \|\mathbf{w}_i - \mathbf{w}_j\|_2^2.\end{aligned} \tag{24}$$

This completes the proof. ■

Discussion. This identity provides a rigorous justification for our greedy bottom-up construction algorithm. By actively selecting task pairs with maximum prototype similarity (which corresponds to the minimum squared Euclidean distance between their approximated subspaces), our algorithm minimizes the structural deviation at every merging step, supporting the stability and robustness of the resulting hierarchical tree.

E.4 Greedy Merging Justification

This section provides an idealized justification for using prototype similarity as the greedy merge criterion. We proceed under two successive approximations. First, following Property 1 and the suggestion in Yang et al. (2025), we treat the adapter weights $\mathbf{w}_i$ as an approximate proxy for the input subspace of task i. Second, we further assume that the visual prototype $\mathbf{p}_i^{\text{f}}$—the mean feature vector of task i's training samples—approximately captures the dominant direction of this input subspace. Under these two approximations, we posit that the cosine similarity between the parameter vectors approximately equals the cosine similarity between their corresponding visual prototypes:

$$\frac{\mathbf{w}_i^\top \mathbf{w}_j}{\|\mathbf{w}_i\|_2 \|\mathbf{w}_j\|_2} \approx \text{sim}(\mathbf{p}_i^{\text{f}}, \mathbf{p}_j^{\text{f}}). \tag{25}$$

We emphasize that this is an idealized simplification, and the actual relationship may be more complex in practice.

Under the further assumption that adapter norms are comparable ($\|\mathbf{w}_i\|_2 \approx \|\mathbf{w}_j\|_2 \approx \gamma$, a reasonable assumption for adapters trained under the same protocol), we have:

$$\|\mathbf{w}_i - \mathbf{w}_j\|_2^2 = \|\mathbf{w}_i\|_2^2 + \|\mathbf{w}_j\|_2^2 - 2\|\mathbf{w}_i\|_2\|\mathbf{w}_j\|_2 \frac{\mathbf{w}_i^\top \mathbf{w}_j}{\|\mathbf{w}_i\|_2\|\mathbf{w}_j\|_2} \approx 2\gamma^2 \left(1 - \text{sim}(\mathbf{p}_i^{\text{f}}, \mathbf{p}_j^{\text{f}})\right). \tag{26}$$

Since $2\gamma^2 > 0$, maximizing $\mathrm{sim}(\mathbf{p}_i^{\mathrm{f}}, \mathbf{p}_j^{\mathrm{f}})$ approximately minimizes $\|\mathbf{w}_i - \mathbf{w}_j\|_2^2$ under the above approximations. By Property 3, this in turn suggests a reduction in the structural deviation $\|\mathbf{w}_{i+j} - \mathbf{w}_i\|_2^2 + \|\mathbf{w}_{i+j} - \mathbf{w}_j\|_2^2$ introduced by the merge.

This does not imply global optimality of the entire hierarchy, but provides an idealized motivation for why the local greedy rule may favor low-conflict merges. ■

F Evaluation Metrics

Let $A_{i,j}$ denote the accuracy on task j after the model has trained up to task i. We report two primary metrics. The **Average Accuracy** ($\bar{\mathcal{A}}$) quantifies performance throughout the entire learning sequence:

$$\bar{\mathcal{A}} = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{t}\sum_{j=1}^{t} A_{t,j}\right). \tag{27}$$

The **Final Accuracy** ($\mathcal{A}_T$) measures average performance across all tasks once the full training sequence has concluded:

$$\mathcal{A}_T = \frac{1}{T}\sum_{j=1}^{T} A_{T,j}. \tag{28}$$

Higher values for both metrics indicate a more effective CIL method.

G Theoretical Speedup Calculation

In our experiments, we report the theoretical speedup of SAEF's inference process relative to a baseline 'Flat Ensemble' method. The baseline requires querying all N available task-specific adapters. Therefore, its computational complexity is proportional to N.

The inference process of SAEF involves two main steps:

1. A mandatory forward pass through the global root adapter, $\mathcal{A}_R$.
2. A parallel search within each of the K conceptual trees. In each tree, the entropy-guided routing evaluates the predictive entropy of both child nodes at every internal split. Consequently, traversing to a depth d in a tree requires querying $2d - 1$ adapters (the root adapter of that tree plus two children for each subsequent level).

Let $\bar{d}_i$ be the average search depth in the i-th tree. The average total number of adapter queries for SAEF is then

$$\text{Queries}_{\text{SAEF}} = 1 + \sum_{i=1}^{K}(2\bar{d}_i - 1). \tag{29}$$

In the specific configuration considered in our main experiments (Table 2), $K = 2$ and the search is heavily concentrated in one primary tree, while the second tree typically terminates very early, yielding $\bar{d}_2 \approx 1$. Under these conditions, the total query count simplifies to $1+(2\bar{d}-1)+(2\times 1-1) = 1+2\bar{d}$, where $\bar{d}$ denotes the average depth of the primary tree.

The theoretical speedup (S) is defined as the ratio of the number of queries required by the baseline to the number of queries required by SAEF. It is formulated as:

$$S = \frac{\text{Queries}_{\text{Baseline}}}{\text{Queries}_{\text{SAEF}}} = \frac{N}{1 + 2\bar{d}}, \tag{30}$$

where N is the total number of task-specific adapters (leaf nodes) and $\bar{d}$ is the average search depth in the primary tree, which is controlled by the entropy threshold τ_e. This formula provides a principled way to estimate the efficiency gains from SAEF's hierarchical structure and its adaptive search mechanism.

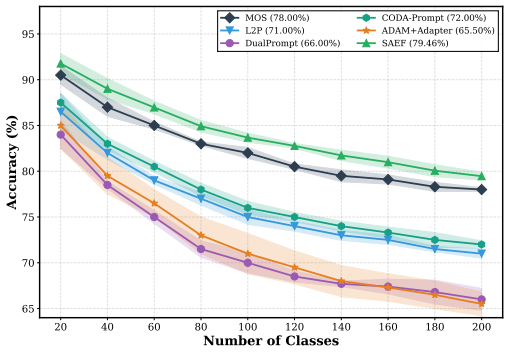


Figure A1: Performance across five random seeds on ImageNet-R (10-task setting). Points show the mean accuracy after each task under different class orders.

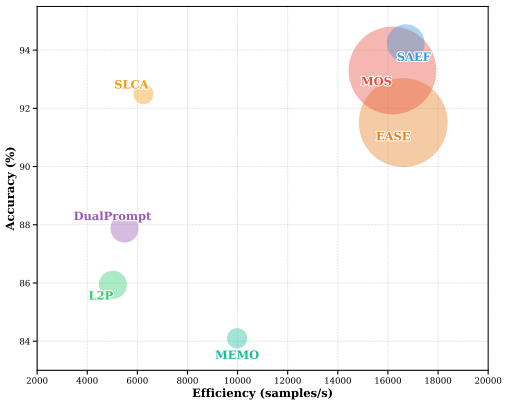
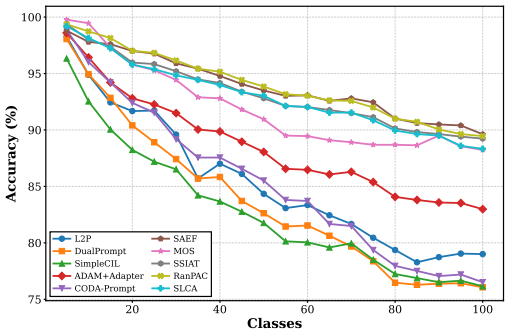
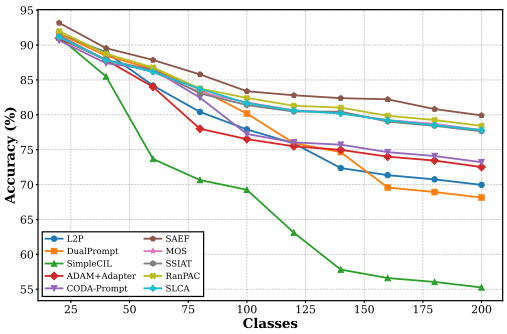


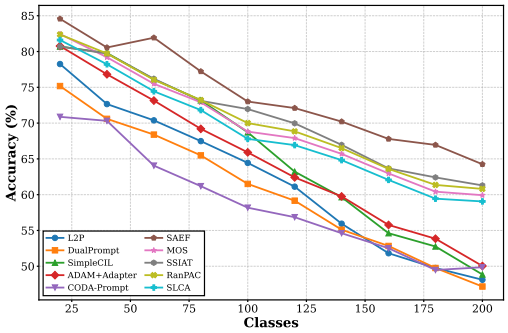
Figure A2: Accuracy–efficiency trade-off on CIFAR-100. Average accuracy is plotted against training throughput (samples/s); bubble size indicates inference time (ms/sample).



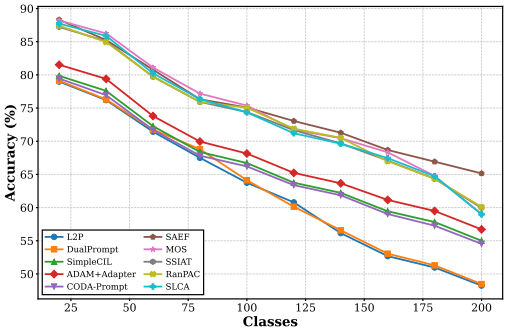
(a) CIFAR-100



(b) ImageNet-R



(c) ImageNet-A



(d) ObjectNet

Figure A3: Incremental accuracy trends of SAEF compared to SOTA CIL methods on four standard benchmarks: CIFAR-100, ImageNet-R, ImageNet-A, and ObjectNet. Across all scenarios, SAEF demonstrates consistently superior performance. This highlights its robust ability to mitigate catastrophic forgetting and maintain a clear advantage as the number of learned classes increases.

H Performance Stability Across Multiple Runs

Our main experiments follow the standard CIL protocol Rebuffi et al. (2017b), using a fixed random seed of 1993 for class ordering to ensure comparable evaluation with prior work. To assess robustness to variations in class order and task composition, we repeat the experiments on CIFAR-100,

Table A1: Mean $\pm$ standard deviation of average accuracy ($\bar{\mathcal{A}}$) and final accuracy ($\mathcal{A}_T$), in %, across five random seeds.

Dataset	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
CIFAR-100	94.61 ± 0.12	90.76 ± 0.13
ImageNet-R	84.42 ± 0.25	79.49 ± 0.42
ImageNet-A	73.48 ± 0.72	65.18 ± 0.66
ObjectNet	76.25 ± 0.37	66.42 ± 0.27

ImageNet-R, ImageNet-A, and ObjectNet using five seeds: $\{1993, 1994, 1995, 1996, 1997\}$. Each seed produces a different class ordering and corresponding task partition.

Table A1 summarizes the results across all four benchmarks. SAEF exhibits limited variation across runs, with standard deviations below 0.8% for both metrics on every dataset. These results indicate that its performance remains stable under the tested changes in class order and task composition.

Figure A1 further compares the mean performance of SAEF and competing methods across the five ImageNet-R runs. SAEF achieves higher mean performance, supporting the effectiveness of its semantic hierarchy across different incremental task sequences.

I Detailed Incremental Performance Analysis

To provide a more detailed view of SAEF's performance progression, we present the incremental average accuracy trends against leading CIL methods in Figure A3. These plots, based on a ViT-B/16 backbone pre-trained on ImageNet-1K, track the performance as the number of learned classes increases across our four primary benchmarks.

Across all evaluated settings, the plots clearly show that SAEF not only achieves the highest final accuracy but also maintains a consistent and often widening performance margin throughout the entire learning sequence. For instance, on the challenging ImageNet-R benchmark (Figure A3b), the performance of many competing methods degrades more sharply over time, whereas SAEF's trajectory remains stable and superior. This robust performance underscores the benefit of SAEF's hierarchical knowledge organization, which allows for more effective and targeted knowledge reuse compared to methods that treat adapters as an unstructured collection.

J Efficiency and Resource Analysis

We evaluate efficiency using the PILOT codebase Sun et al. (2023) and measure wall-clock time under the same hardware and data protocol for all compared methods. Figure A2 summarizes accuracy, training throughput, and inference latency on CIFAR-100. Prompt-based methods such as L2P and DualPrompt have relatively low inference cost but lower accuracy, whereas adapter ensembles such as EASE and MOS incur higher inference latency.

Table A2 shows that SAEF has lower measured training and inference times than MOS and EASE on all four benchmarks, with up to a $1.9\times$ inference speedup over EASE in the standard setting. The optional early-exit experiment at $\tau_e = 1.0$ is a separate operating point: it yields an inference speedup of over $5.5\times$ over EASE in the CIFAR-100 measurement shown in Fig. A2; see Appendix G for the corresponding analysis.

Forest construction accounts for less than 1% of total training time. Its main additional resource cost is adapter parameters, not retained training data. For T tasks partitioned into $K \geq 2$ conceptual clusters, SAEF stores T leaf adapters and $T - K + 1$ internal adapters, including the global root, for a total of $2T - K + 1$ adapters. When $K = 1$, the single tree root also serves as the global root, giving $T - 1$ internal adapters and $2T - 1$ adapters in total. All benchmarks reported here use $K = 2$, for which the total adapter count is $2T - 1$. Table A3 distinguishes the parameters in all stored adapters from the additional parameters introduced by internal nodes. The FP32 storage figures in Table A2 are conditional estimates; actual storage depends on parameter dtype and serialization. Reporting both parameter counts and wall-clock costs reflects the importance of computational budgets in continual learning Prabhu et al. (2023); Ghunaim et al. (2023).

Table A2: Adapter parameter counts and wall-clock training and inference time. FP32 storage is an estimate assuming four bytes per parameter, not a measurement of stored files or runtime memory. SAEF inference uses the standard setting without optional entropy-based early exit. Parameter counts are approximate.

Dataset	Metric	MOS	EASE	**SAEF (Ours)**
CIFAR-100	Adapter params (M)	9.2	9.2	17.9
	FP32 storage (MB)	36.8	36.8	71.6
	Train (s)	8612.4	8750.3	**8583.7**
	Infer (s)	6837.5	6890.1	**3600.4**
ImageNet-R	Adapter params (M)	4.6	4.6	8.7
	FP32 storage (MB)	18.4	18.4	34.8
	Train (s)	3156.7	3240.5	**3124.8**
	Infer (s)	1145.1	1172.3	**782.1**
ImageNet-A	Adapter params (M)	4.6	4.6	8.7
	FP32 storage (MB)	18.4	18.4	34.8
	Train (s)	1794.5	1845.2	**1673.2**
	Infer (s)	408.9	425.6	**343.2**
ObjectNet	Adapter params (M)	4.6	4.6	8.7
	FP32 storage (MB)	18.4	18.4	34.8
	Train (s)	11561.1	11785.4	**11415.1**
	Infer (s)	2342.1	2380.5	**1663.7**

Table A3: SAEF adapter counts and approximate parameter counts across benchmarks ($K = 2$). Each adapter has approximately 0.46M parameters. "Additional" counts internal-node parameters beyond the T leaf adapters; neither parameter column includes the frozen model.

Dataset	T	**Leaf**	**Internal**	**Total params (M)**	**Additional params (M)**
CIFAR-100	20	20	19	17.9	8.7
ImageNet-R	10	10	9	8.7	4.1
ImageNet-A	10	10	9	8.7	4.1
ObjectNet	10	10	9	8.7	4.1

Table A4: Robustness analysis of the fusion temperature τ on ImageNet-R. The optimal results are marked in bold.

Temperature (τ)	1.0	0.5	0.2	0.1
$\bar{\mathcal{A}}$ **(%)**	**84.54**	84.49	84.32	84.31
$\mathcal{A}_T$ **(%)**	80.02	**80.08**	79.78	79.87

K Parameter Robustness.

SAEF's fusion mechanism involves a single key hyperparameter: the temperature τ in the entropy-weighted softmax (Eq. (12)), which controls the sharpness of the weighting distribution. To evaluate its robustness, we conducted an analysis on the challenging ImageNet-R benchmark by varying τ across a range of values: $\{1.0, 0.5, 0.2, 0.1\}$. The results, presented in Table A4, demonstrate that SAEF's performance is remarkably stable.

L Detailed Analysis of Parent Node Semantics

This section expands the parent-node analysis in Section 4.4 by examining path strategies, contributions at different depths, and the effect of pruning intermediate experts.

Path Strategy and Depth Ablations. We first compare five inference strategies on ImageNet-R (Figure 4(c)): *Root Only* uses the Global Root R_G; *Leaf Only* uses the leaf on the lowest-entropy path; *Single Tree* restricts inference to the tree with the lowest root entropy; *Leaf + Root* combines the selected leaf with R_G; and *Full Adaptive Path* (SAEF) fuses experts on all K selected paths together

Table A5: Path-strategy and depth ablations on ImageNet-R. The *Single Tree* comparison is shown in Figure 4(c).

Configuration	$\bar{A}$ (%)	A_T (%)
Leaf Only	83.97	78.98
Root Only	81.24	72.92
Leaf + Root	84.34	79.72
Shallow 50% of path	83.24	75.36
Deep 50% of path	84.37	79.68
SAEF (Full Adaptive Path)	84.54	**80.02**

Table A6: Mean predictive entropy by expert depth on ImageNet-R. A dash indicates that the depth is not present at that task count.

T	Depth 0	Depth 1	Depth 2	Depth 3	Depth 4
2	0.5604	0.6084	–	–	–
4	0.6963	0.7041	0.6557	–	–
6	0.8130	0.8008	0.6997	0.6157	–
8	0.8641	0.8523	0.7458	0.6433	0.5309
10	0.9104	0.8984	0.7980	0.6670	0.6281

with R_G. To examine the contribution of hierarchy depth more directly, we additionally retain only the shallowest or deepest 50% of path experts for fusion while leaving path selection unchanged. Table A5 reports these results alongside the corresponding path-strategy results.

The deepest half of the path outperforms the shallowest half in both metrics, indicating that deeper experts provide stronger task-specific predictions. Nevertheless, the shallowest half improves final accuracy over *Root Only* (75.36% versus 72.92%), suggesting that its predictions add useful information beyond the root. Full-path fusion achieves the highest average and final accuracy in Table A5 (84.54% and 80.02%), outperforming both *Deep 50%* (84.37% and 79.68%) and *Leaf Only* (83.97% and 78.98%). These results indicate that combining experts across hierarchy depths is more effective than relying on either the selected leaf or a depth-restricted subset alone. The *Single Tree* comparison in Figure 4(c) further examines the contribution of paths from multiple conceptual clusters.

Predictive Entropy Across Depths. Table A6 reports the mean predictive entropy of experts at each available depth on ImageNet-R. At six, eight, and ten tasks, entropy decreases with depth; at ten tasks, it falls from 0.9104 at depth 0 to 0.6281 at depth 4. This pattern is consistent with deeper experts making more confident predictions, in line with the stronger performance of *Deep 50%* in Table A5. The trend is not strictly monotonic at two or four tasks, however, and lower entropy alone does not establish higher classification accuracy. The depth ablation provides the complementary performance evidence.

Entropy-Based Pruning Experiment. To test whether intermediate parent nodes contribute to the final prediction beyond guiding path selection, we prune high-entropy nodes from output aggregation at ratios $q \in \{0\%, 25\%, 50\%, 75\%\}$, without changing the routing decisions. If these nodes served only for routing and added unhelpful predictions to the fusion, excluding them should maintain or improve accuracy. Instead, Figure 4(d) shows the highest accuracy when all intermediate nodes are retained ($q = 0\%$), with accuracy declining as the pruning ratio increases. Together with the depth ablation, this indicates that parent-node predictions contribute useful information to full-path fusion, even though the deepest experts provide the strongest depth-restricted results.

M Robustness to Parameter Fusion Strategies

A natural question is whether SAEF's performance stems from the hierarchical structure itself or from the specific parameter fusion operation in Eq. (7). While our sign-aware max-magnitude fusion is algorithmically identical to MagMax Marczak et al. (2024), we replace it with four alternative merging methods—Task Arithmetic Ilharco et al., DARE Yu et al. (2024), PCB-Merging Du et al. (2024), and TIES-Merging Yadav et al. (2023)—keeping the forest structure and adaptive inference unchanged. As shown in Table A7, all variants achieve competitive performance within a narrow

band ($\leq$0.39% on $\bar{\mathcal{A}}$), underscoring that SAEF's gains are primarily attributable to its hierarchical organization rather than the specific fusion operator.

M.1 Results and Analysis

Table A7: Comparison of different parameter fusion methods integrated into our framework. $\bar{\mathcal{A}}$ denotes average accuracy and $\mathcal{A}_T$ denotes last-phase accuracy. All variants perform within a narrow band, confirming the structural robustness of SAEF.

Method	**Task Arithmetic**	**DARE**			**PCB-Merging**			**TIES-Merging**			**Ours (SAEF Fusion)**
Pruning Rate (q)	0%	25%	50%	75%	25%	50%	75%	25%	50%	75%	0%
$\bar{\mathcal{A}}$ **(%)**	84.51	84.32	84.51	84.33	84.17	84.21	84.17	84.36	**84.56**	84.40	84.54
$\mathcal{A}_T$ **(%)**	79.98	79.78	79.88	79.70	79.78	79.92	79.55	79.72	**80.03**	79.73	80.02

The only variant that marginally surpasses our default fusion is TIES-Merging ($q = 50\%$), which applies L_1-norm trimming before the sign-guided merge. This suggests that pre-filtering low-magnitude noise can slightly refine the subspace approximation. However, the gain is marginal ($+0.02\%$ on $\bar{\mathcal{A}}$), and our default fusion requires no pruning hyperparameter, making it the preferred choice for simplicity and reproducibility.

M.2 Compared Methods

The core challenge addressed by these methods is *parameter interference*: when fine-tuning creates task-specific parameter variations, merging them can cause conflicts (e.g., opposing gradient directions or redundant noise masking crucial features). Below, we detail how each method merges two task adapters $\mathbf{w}_i$ and $\mathbf{w}_j$ within our architecture.

Task Arithmetic Ilharco et al. treats adapter parameters as task vectors and merges them via element-wise averaging: $\mathbf{w}_{i+j} = \frac{1}{2}(\mathbf{w}_i + \mathbf{w}_j)$.

DARE Yu et al. (2024) exploits extreme parameter redundancy by randomly dropping a fraction q of weights and rescaling the rest. Formally, a binary mask $M \sim \text{Bernoulli}(1-q)$ is applied to each adapter, and the remaining weights are rescaled by $\frac{1}{1-q}$ to preserve expected energy. The two sparsified adapters are then averaged.

PCB-Merging Du et al. (2024) prunes based on functional importance rather than randomness. For each adapter, the bottom q fraction of weights by absolute magnitude are dropped. The remaining salient parameters of $\mathbf{w}_i$ and $\mathbf{w}_j$ are combined via element-wise averaging.

TIES-Merging Yadav et al. (2023) addresses interference through trimming and sign resolution. In our adaptation, we apply the "Trim" phase of TIES-Merging before our sign-based max-magnitude fusion: for both $\mathbf{w}_i$ and $\mathbf{w}_j$, we prune the bottom q fraction of weights by L_1 norm, then apply our fusion operation (Eq. (7)) on the remaining parameters.

N Ablation: Semantic vs. Visual Prototype Sources for Clustering

SAEF uses CLIP text embeddings as semantic prototypes for task clustering, which requires class names as input. A natural question is whether this additional side information is necessary, or whether visual features alone can achieve comparable clustering quality. To answer this, we compare two prototype sources for the clustering stage across all four benchmarks:

- **CLIP text (Ours)**: Semantic prototypes from the frozen CLIP text encoder using the prompt "a photo of a [CLASS]".
- **ViT visual**: Visual prototypes from the frozen PTM backbone ϕ only, without any task adapter.

As shown in Table A8, CLIP text semantics provide a consistent advantage across most benchmarks, validating the effectiveness of semantic clustering. Notably, even without CLIP, the ViT visual

Table A8: Comparison of prototype sources for task clustering. CLIP text semantics provide a consistent advantage, while ViT visual features alone remain competitive, demonstrating that SAEF does not critically depend on CLIP.

Source	CIFAR-100		ImageNet-R		ImageNet-A		ObjectNet	
	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
ViT visual (no CLIP)	94.44	90.47	84.25	79.25	72.00	60.90	76.38	66.58
CLIP text (Ours)	**94.53**	**90.60**	**84.54**	**80.02**	**73.39**	**64.91**	**76.38**	**66.58**

variant remains highly competitive, e.g., 76.38% vs. 76.38% on ObjectNet, confirming that SAEF's hierarchical structure is the primary driver of performance gains rather than the CLIP signal alone.

O Ablation: Visual Prototype Feature Source

SAEF uses the first task's adapter $\mathcal{A}_1$ as a domain-adapted backbone to extract visual prototypes for all tasks. This design choice is motivated by the need for a consistent feature space across tasks: $\mathcal{A}_1$ is the only adapter available when computing prototypes for all tasks in a single pass, and it serves as a lightweight bridge from the pre-trained feature space to the downstream task distribution Zhou et al. (2024a); Sun et al. (2025).

A potential concern is that this introduces a dependency on the first task's domain. To assess the sensitivity of SAEF to this choice, we compare four prototype feature sources on ImageNet-R using Average Accuracy as the evaluation metric:

- **Backbone**: Features from the frozen PTM backbone ϕ only, without any task adapter.
- **Curr. Adapter**: Features extracted using each task's own adapter $\mathcal{A}_t$.
- **Avg. Adapter**: Features extracted using the average of all task adapters.
- **Ours ($\mathcal{A}_1$)**: Features extracted using the first task's adapter $\mathcal{A}_1$.

Table A9: Ablation of visual prototype feature sources on ImageNet-R. We report Average Accuracy. The consistent results confirm that SAEF is robust to this design choice.

Average Accuracy (%)	84.42	84.51	84.54	**84.54**

As shown in Table A9, all four feature sources yield nearly identical Average Accuracy values, with differences within 0.12%. This confirms that SAEF is robust to the choice of prototype extractor. We use $\mathcal{A}_1$ as the default because it is always available at clustering time and requires no additional computation.

P Comparison with Related Adapter-Combination Methods

Beyond the adapter-preserving methods compared in the main text, we evaluate SAEF against two adapter-combination methods, SD-LoRA Wu et al. (2025) and SEMA Wang et al. (2025), under the exemplar-free protocol. Table A10 reports average and final accuracy on ImageNet-R and ImageNet-A.

SAEF achieves the highest average and final accuracy among the compared methods on both benchmarks. On ImageNet-A, it exceeds SD-LoRA by 8.44% in average accuracy and 8.95% in final accuracy. SAEF does store more adapters because its internal nodes are also inference experts; under the exemplar-free protocol, this additional storage consists of model parameters rather than replay samples. Computational budgets are also important in continual deployment Prabhu et al. (2023); Ghunaim et al. (2023). We therefore report adapter storage alongside measured training and inference costs in Appendix J.

Table A10: Average accuracy ($\bar{\mathcal{A}}$) and final accuracy ($\mathcal{A}_T$), in %, under the exemplar-free protocol.

Method	ImageNet-R $\bar{\mathcal{A}}$	ImageNet-R $\mathcal{A}_T$	ImageNet-A $\bar{\mathcal{A}}$	ImageNet-A $\mathcal{A}_T$
SD-LoRA	82.04	77.34	64.95	55.96
SEMA	81.39	74.82	62.50	51.35
SAEF (Ours)	**84.54**	**80.02**	**73.39**	**64.91**

Table A11: Expert-selection accuracy (%) on ImageNet-R as the number of learned tasks increases. Internal-node accuracy measures whether the evaluated node's subtree contains the ground-truth task.

Number of tasks	2	4	6	8	10
Learned gating	88.42	74.66	66.39	58.97	53.27
Flat lowest-entropy leaf	76.84	72.74	74.16	72.22	69.32
SAEF leaf accuracy	81.84	75.22	66.41	60.78	54.52
SAEF internal-node accuracy	–	85.54	78.44	79.13	76.02

Q Expert Selection and Path Interpretation

We investigate how expert selection affects classification on ImageNet-R and how the selected paths reflect task relationships. We compare SAEF with two alternatives: a learned MoE gate over all leaf experts and a flat rule that selects the globally lowest-entropy leaf. We report both task-selection accuracy and classification accuracy, followed by an analysis of traversal depth using an optional entropy-based early-exit setting.

Expert-selection accuracy. Leaf-selection accuracy measures whether the selected leaf corresponds to the task containing the ground-truth class. For SAEF, we also report internal-node accuracy, which measures whether a traversed internal node's subtree contains the ground-truth task. These metrics assess task matching at different levels: a leaf identifies an individual task, whereas an internal node identifies a broader group of tasks.

Table A11 shows that learned gating becomes less accurate as the expert set grows, declining from 88.42% at two tasks to 53.27% at ten tasks. The flat entropy rule achieves higher leaf-selection accuracy at ten tasks than either learned gating or SAEF. Within SAEF, internal-node accuracy remains substantially higher than leaf accuracy, reaching 76.02% versus 54.52% at ten tasks. This indicates that identifying a task group is easier than identifying the exact task, although the internal-node criterion is also less restrictive because each subtree covers multiple tasks.

Why accurate classification does not require accurate leaf selection. Table A12 compares expert-selection strategies in terms of both routing and classification performance. At ten tasks, the flat lowest-entropy rule achieves higher leaf-selection accuracy than SAEF (69.32% versus 54.52%), yet SAEF obtains higher average accuracy (84.54% versus 83.47%) and final accuracy (80.02% versus 78.67%). This result shows that selecting the exact task-specific adapter is not sufficient to explain the quality of the final prediction.

The three strategies behave differently as the number of tasks increases. Learned gating must maintain a single flat routing function over an expanding set of task-specific experts. When new experts are introduced, the routing boundaries for previously learned tasks become harder to preserve, causing interference and routing forgetting. Its leaf-selection accuracy therefore decreases from 88.42% with two tasks to 53.27% with ten tasks. The flat lowest-entropy rule avoids learning this global router and achieves 69.32% leaf-selection accuracy at ten tasks, but it selects only one leaf and does not provide internal experts for sharing knowledge among related tasks.

SAEF addresses this limitation by organizing task-specific adapters into a semantic hierarchy. Related adapters are merged into internal nodes, and the activated internal and leaf experts jointly contribute to the final prediction through the confidence-based fusion in Eqs. (12) and (13). Consequently, a leaf-selection error does not necessarily remove the evidence associated with the ground-truth task. For example, the selected leaf may belong to a neighboring task, while an earlier internal node on the same path contains the adapter associated with the ground-truth task. The internal expert can then provide useful class evidence even though the final leaf is not the exact task-specific adapter.

Table A12: Average accuracy ($\bar{\mathcal{A}}$) and final accuracy ($\mathcal{A}_T$), in %, for different expert-selection configurations on ImageNet-R.

Configuration	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
Learned gating	83.12	78.08
Flat lowest-entropy leaf	83.47	78.67
SAEF (Ours)	**84.54**	**80.02**

Table A13: Effect of the optional entropy-based early exit on ImageNet-R. Higher thresholds shorten traversal and increase internal-node task coverage, while reducing average classification accuracy.

τ_e	Internal-node accuracy (%)	Average depth	$\bar{\mathcal{A}}$ (%)
0	76.02	3.16	**84.54**
0.5	90.73	1.77	83.24
1.0	99.65	1.16	82.13

To quantify this effect, the internal-node accuracy reported in Table A11 is computed as follows: a selected internal node is counted as correct whenever its descendant subtree contains the adapter associated with the ground-truth task. This criterion evaluates whether the traversal reaches the correct semantic region; it does not require the selected node itself to be the exact task adapter. At ten tasks, SAEF obtains 76.02% internal-node accuracy, substantially higher than its 54.52% leaf-selection accuracy. The gap indicates that the hierarchy can identify a relevant group of adapters even when the exact leaf remains ambiguous.

Let R denote correct leaf selection and C denote correct classification. The final classification probability can be decomposed as

$$\Pr(C) = \Pr(R)\Pr(C \mid R) + \Pr(\neg R)\Pr(C \mid \neg R). \tag{31}$$

For a flat selector, an incorrect leaf provides limited opportunity to recover from a routing error. In SAEF, however, $\Pr(C \mid \neg R)$ can remain substantial because internal experts along the selected path preserve and combine knowledge from related adapters. The final prediction therefore depends on the evidence collected across the hierarchy rather than on an exact leaf match alone. This explains why SAEF reaches 80.02% final accuracy, exceeding learned gating by 1.94% and the flat lowest-entropy rule by 1.35%, despite having lower exact leaf-selection accuracy than the flat rule.

The comparison between the routing and classification metrics should also be interpreted at the same incremental stage. At ten tasks, SAEF achieves 54.52% leaf-selection accuracy, 76.02% internal-node accuracy, and 80.02% final classification accuracy. The 84.54% value is its average classification accuracy across all incremental stages. These results show that SAEF benefits from retrieving knowledge at multiple hierarchy levels: the internal node identifies a semantically relevant adapter region, while deeper experts provide task-specific refinement. Thus, accurate classification does not require the router to identify the exact leaf in every case; it requires the activated hierarchy to provide sufficiently relevant and complementary evidence for the final prediction.

Traversal depth and path interpretation. To examine the contribution of different hierarchy levels, we introduce an optional entropy threshold τ_e for this analysis. Traversal stops when a node's predictive entropy falls below τ_e; increasing the threshold therefore favors earlier exits. The $\tau_e = 0$ setting corresponds to traversal without this optional early exit.

As shown in Table A13, increasing τ_e from 0 to 1.0 reduces average traversal depth from 3.16 to 1.16 and raises internal-node accuracy from 76.02% to 99.65%, but lowers average classification accuracy from 84.54% to 82.13%. Shallower nodes cover larger task groups, making ground-truth task membership easier to satisfy. Their higher membership accuracy therefore does not imply more discriminative predictions. The accompanying classification decrease suggests that deeper experts provide useful specialized evidence that is lost when traversal stops early.

These results support interpreting a selected path as a progression from broader task groups to more specialized experts. CLIP text semantics guide the grouping, and each internal node represents the related tasks among its descendants. The hierarchy thus provides an interpretable organization of the evidence used for prediction: internal experts contribute shared knowledge, while deeper

Table A14: Sensitivity to the number of conceptual clusters on ImageNet-R. $K = 2$ is selected automatically by the silhouette criterion.

K	$\bar{A}$ (%)	A_T (%)
1	84.12	79.60
2 (selected)	**84.54**	**80.02**
3	84.35	79.70
4	83.94	79.28
5	84.14	79.57

Table A15: Effect of global-root initialization on ImageNet-R.

Method	$\bar{A}$ (%)	A_T (%)
SAEF (Ours)	84.54	80.02
SAEF + global-root initialization	**84.61**	**80.28**

experts refine it. This organization does not guarantee correct routing at every level, but it allows classification to benefit from a relevant internal expert even when the exact leaf is missed. The associated inference-cost trade-off is further examined in Appendix J.

R Sensitivity to the Number of Conceptual Clusters

The number of conceptual clusters, K, determines how task semantic prototypes are partitioned before constructing the expert forest. SAEF selects K automatically by maximizing the average silhouette score over the semantic prototypes, rather than tuning it against classification accuracy. This procedure selects $K = 2$ on CIFAR-100, ImageNet-R, ImageNet-A, and ObjectNet. To assess sensitivity to this choice, we override the selected value and evaluate $K \in \{1, 2, 3, 4, 5\}$ on ImageNet-R. The $K = 1$ setting is included as a single-cluster ablation; it is not a candidate for silhouette-based selection, since the silhouette score requires at least two clusters.

As shown in Table A14, average accuracy ranges from 83.94% to 84.54% across the five settings. All settings exceed the strongest baseline, SSIAT (83.20% average accuracy), indicating that SAEF's performance on ImageNet-R is not dependent on a single choice of K. The automatically selected $K = 2$ also achieves the best average and final accuracy among the evaluated settings. These results support the silhouette criterion for this benchmark without assuming that it is optimal for every dataset or task sequence.

S Initializing New Adapters with Prior Knowledge

SAEF normally trains each task adapter independently before incorporating it into the expert forest. This keeps previously learned adapters unchanged and transfers knowledge through hierarchical construction and inference. To examine whether prior knowledge can also help during adapter training, we evaluate a variant that initializes the new adapter with the global root R_G constructed from all previously learned tasks. For task $t > 1$, this initialization is applied before training on the current task data $\mathcal{D}_t$; previous adapters remain frozen, and no past-task samples are used. The subsequent hierarchy construction and inference procedure are unchanged.

As shown in Table A15, global-root initialization improves average accuracy from 84.54% to 84.61% and final accuracy from 80.02% to 80.28%. This suggests that the consolidated knowledge in the expert forest can provide a useful starting point for new task adapters, making training-time reuse a promising extension of SAEF.

Table A16: Performance on ImageNet-A with longer task sequences.

	50 tasks		100 tasks	
Method	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$
MOS	66.42	53.33	66.40	53.27
SAEF (Ours)	**68.41**	**56.74**	**68.13**	**54.95**

Table A17: Average forgetting (%) after the final task. Lower is better.

Dataset	MOS	SAEF (Ours)
CIFAR-100	9.52	**4.60**
ImageNet-R	11.67	**6.16**
ImageNet-A	32.62	**11.41**
ObjectNet	15.26	**8.07**

T Performance with Longer Task Sequences

As the number of tasks grows, the expert forest contains more nodes, raising the question of whether its increasing complexity affects classification. We evaluate SAEF on ImageNet-A under longer sequences of 50 and 100 tasks, using MOS as a comparison.

As shown in Table A16, SAEF outperforms MOS in both average and final accuracy at both sequence lengths. From 50 to 100 tasks, SAEF's average accuracy changes from 68.41% to 68.13%, while its final accuracy changes from 56.74% to 54.95%. Thus, the longer sequence causes a modest performance decline, but does not eliminate SAEF's advantage. Structurally, balancing limits the depth of a path within a conceptual tree to $O(\log |G_k|)$ for a cluster containing $|G_k|$ tasks. This keeps the number of experts on an individual path manageable as the forest grows, consistent with the observed performance on longer sequences.

U Forgetting Analysis

To assess retention of previously learned tasks, we report average forgetting after the final task. Using $A_{i,j}$ as defined in Appendix F, we compute

$$\mathcal{F}_T = \frac{1}{T-1} \sum_{j=1}^{T-1} \left(\max_{i \in \{j,\ldots,T-1\}} A_{i,j} - A_{T,j} \right). \tag{32}$$

This measures the average change from each previous task's best accuracy before the final stage to its accuracy after learning all tasks. Lower values indicate less forgetting.

As shown in Table A17, SAEF exhibits lower forgetting than MOS on all four benchmarks. The reduction is largest on ImageNet-A, from 32.62% to 11.41%. These results complement the accuracy comparisons by showing that SAEF also better retains performance on previously learned tasks.