NOT MORE PARAMETERS, BETTER ORGANIZATION: STRUCTURED COMPRESSION MEMORY NETWORK FOR CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Class-incremental learning (CIL) aims to learn new classes sequentially while preserving discriminability over previously seen categories. While fine-tuning pre-trained models (PTMs) yields stronger generalization than training from scratch, existing approaches suffer from an inherent trade-off between parameter overhead and feature regulation. Expandable architectures incur steady parameter inflation as tasks grow, whereas fixed-parameter alternatives struggle to maintain reliable recognition of historical classes. To address this dilemma, we propose the Structured Compression Memory Network (SCMNet), a rehearsal-free, fixed-parameter framework built upon PTMs. Instead of growing parameters with tasks, SCMNet spends a fixed budget on organizing the feature and prototype spaces. Specifically, dynamic neighborhood compression constructs label-constrained batch graphs, suppressing intra-class variance while preserving inter-class separability. The adaptive prototype constraint uses stored prototypes as anchors to stabilize historical decision boundaries. These two modules form a mutually reinforcing loop to mitigate performance degradation over long task sequences. Evaluations across six benchmarks and three pre-trained backbones demonstrate that SCMNet surpasses state-of-the-art methods with only a fixed budget of $\sim$1.7M trainable parameters, validating the benefits of our structured feature organization strategy. Our code is included in the supplementary materials and will be released publicly upon acceptance.

1 INTRODUCTION

Class-incremental learning (CIL) requires a model to learn disjoint class sets in sequence and to classify all seen categories uniformly, without task identifiers at test time, with catastrophic forgetting as its central obstacle (Dohare et al., 2024). Since deployments such as autonomous driving and medical diagnosis impose strict storage constraints, replaying old training data is impractical (Liu et al., 2025; Zhou et al., 2025b; Luo et al., 2026), which makes rehearsal-free frameworks that reconcile novel-class learning with old-knowledge preservation a practical necessity.

Related research on class-incremental learning can be divided into two main stages, as illustrated on the left of Figure 1. The earlier stage trains the backbone from random initialization, either expanding the classification head to separate the representations of successive tasks or sharing a single fixed head among them. Without a generalizable visual representation, such models struggle to attain even basic discriminative ability, let alone sustain it as tasks accumulate. Recent methods instead build on PTMs, whose representations are learned from web-scale data and transfer strongly across domains; they therefore offer a much better starting point and considerably greater potential, and adapting to new classes requires only parameter-efficient fine-tuning. Nevertheless, keeping the memory of old classes stable while balancing it against the new ones remains a central challenge.

Within the PTMs-based regime, the same opposition recurs at the level of tunable modules. Expansion-based methods such as L2P (Wang et al., 2022c) and SEMA (Wang et al., 2025a) insert a new adapter for each incoming task, which shields the memory of old classes from the learning of new ones, at the price of a parameter budget and an inference path that both lengthen with the task sequence. Fixed-budget methods such as SimpleCIL (Zhou et al., 2025a) keep parameters constant,

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.

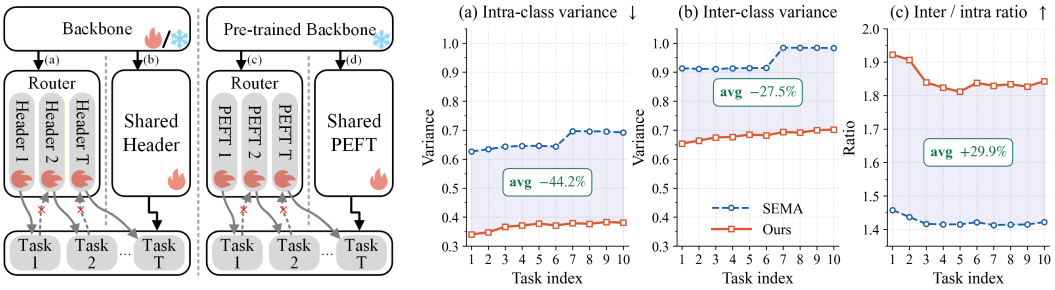

Figure 1: Left: four class-incremental learning pipelines. Right: intra/inter-class variance and their ratio over 10-task IN-R.

but each task then updates the same parameters, and repeated updates make the balance between old and new classes progressively harder to hold. Neither family governs the persistent geometry of the feature space: how tightly samples gather around their prototype, how far apart the prototypes lie, and how both quantities evolve as tasks accumulate are all left to the adaptation itself, even though they are exactly what an explicit regulation of the prototype space could control.

The right panel of Figure 1 shows why this geometry cannot be ignored and more analysis in Appendix A.2. Under the 10-task protocol of IN-R, SEMA already strikes a careful balance between the two families by expanding its parameters sublinearly. Both quantities of SEMA remain at a high level, and because the parameter budget is enlarged across tasks, the feature space is re-established rather than progressively refined, so that both curves shift abruptly. Our method instead relies on a prototype library that contracts each class towards its own prototype while preserving the layout of the prototypes accumulated so far. Both variances drop substantially and their curves stay nearly flat, so the space is organized thoroughly and the transition across tasks is smoothed. The resulting widening of the inter/intra-class variance ratio leaves considerably more room for continual learning.

We therefore treat stored class prototypes not as a passive by-product of training but as an actively organized memory buffer, and propose SCMNet, a rehearsal-free framework with a fixed parameter budget built on a frozen PTM backbone. Dynamic Neighborhood Compression (DNC) applies label-restricted graph convolution to contract the samples of a class around their prototype while preserving inter-class boundaries. Adaptive Prototype Constraint (APC) supervises the classifier with crowding-adaptive prototype targets, counteracting the drift that accumulating classes exert on boundaries established earlier. The two reinforce each other: sharper prototypes give APC more reliable supervision, and the resulting stability lets the next round of compression proceed on a better-behaved space. Both modules operate within a fixed budget, and the number of trainable parameters stays at about 1.7M regardless of how many tasks have been learned, so that adaptation to the incoming classes proceeds smoothly without any expansion of the parameters. Our main contributions are summarized as follows:

- We formulate CIL as a sequence space organization problem and propose SCMNet, a fixed-parameter framework built upon frozen pre-trained backbones that balances novel feature adaptation and old knowledge preservation.
- DNC module compresses intra-class redundancies for compact novel-class representations via graph propagation, while APC module stabilizes historical boundaries to retain old-class discriminative information. Integrated as a synergistic closed-loop paradigm for CIL.
- Evaluations across six benchmarks and three backbones demonstrate that SCMNet achieves consistent accuracy gains against SOTA methods, yielding a +5.68pt average gain on IN-R over the capacity-expanding baseline while holding parameters fixed.

2 Related Work

2.1 Class-Incremental Learning

Class-incremental learning (CIL) requires sequentially learning disjoint class sets while preserving discriminability on all previously seen categories, with catastrophic forgetting (Birashk & Khan,

2026) as its central obstacle. Early methods approached this challenge through three strategies. Replay-based methods (Gao et al., 2026; Iqbal et al., 2026) retain a subset of old-class samples for joint training, achieving strong retention at the cost of growing storage and privacy concerns. Regularization-based methods either penalize updates to important parameters or distill old model predictions (An et al., 2024; Ye et al., 2025; Du et al., 2026), but these indirect constraints gradually lose efficacy as task sequences lengthen. Architecture expansion (Wang et al., 2022a; Dong et al., 2026) allocates dedicated parameter subspaces per task, avoiding cross-task interference at the expense of unbounded model growth and restricted knowledge reuse. All three paradigms assume training from random initialization, limiting their representational capacity relative to large-scale pretrained models. The advent of Vision Transformers (Dosovitskiy et al., 2021) and CLIP (Radford et al., 2021) has shifted the field toward a new paradigm built on frozen pretrained backbones, which provide stronger generalization and substantially higher performance ceilings (Zhou et al., 2025b).

2.2 Pre-trained Model Based Class-Incremental Learning

The adoption of pre-trained Vision Transformers (Dosovitskiy et al., 2021) and CLIP (Radford et al., 2021) has shifted class-incremental learning toward parameter-efficient fine-tuning on frozen backbones, which offer stronger generalization than training from scratch. Sufficient representations for CIL are shown to be attainable using only frozen pretrained features (Zhou et al., 2025a;b), motivating the widespread practice of coupling a frozen backbone with lightweight learnable modules. At the simpler end of existing solutions, prototype-based approaches including SimpleCIL and ADAM (Zhou et al., 2025a) demand minimal parameter updates, yet they lack active mechanisms to regularize feature spaces for incremental discrimination. Higher performance is pursued via two mainstream routes: expanding model capacity per task, or adapting within a fixed parameter budget.

Capacity-expansion methods avoid cross-task interference by allocating new learnable modules on demand. SEMA (Wang et al., 2025a) pairs each adapter with a representation descriptor as a distribution-shift detector, dynamically inserting adapters with an expandable mixture router; CODA-Prompt (Smith et al., 2023) extends the prompt pool through decomposed attention. ACMap (Fukuda et al., 2025) and TUNA (Wang et al., 2025b) explore adapter fusion to mitigate expansion overhead. These methods trade unbounded parameter growth for easier conflict avoidance, yet frozen legacy modules inhibit cross-task knowledge reuse. Fixed-parameter methods instead constrain adaptation under a constant budget: L2P (Wang et al., 2022c) and DualPrompt (Wang et al., 2022b) select from a fixed prompt pool via key-query matching; InfLoRA (Liang & Li, 2024) uses low-rank adaptation with rehearsal buffers; SA-LoRA (Bing et al., 2026) shares a single down-projection across tasks; SR^2-LoRA (Wan et al., 2026) constrains inter-layer relation drift through singular value alignment. While diverse, these methods regulate model dynamics rather than directly organizing the feature space. Orthogonal to both routes, KCEA (Tran et al., 2026) follows a consolidation paradigm, merging task-wise LoRA modules into a unified backbone via task arithmetic and realigning the classifier with data-free evolutionary optimization, without exemplars.

Departing from these methods that regulate model dynamics or consolidate parameters, we instead directly organize the feature space, modeling class-incremental learning as two complementary objectives: compressing intra-class redundancy within each task and preserving old-class discriminability across tasks. SCMNet instantiates both objectives as explicit, mutually reinforcing feature-space operators atop a frozen backbone with a constant parameter set.

3 Method

3.1 Problem Formulation

Class-incremental learning (CIL) involves a sequence of tasks $\{(\mathcal{X}_t, \mathcal{Y}_t)\}_{t=0}^{T-1}$, where each task t introduces a mutually disjoint set of novel categories $\mathcal{Y}_t$ without access to historical training data. Upon completing task t, the model must uniformly classify all encountered classes $\bigcup_{k=0}^{t} \mathcal{Y}_k$ without task identifiers. The core challenge is the stability-plasticity tension: adapting to new classes risks overwriting discriminability on old ones (catastrophic forgetting), while freezing prior knowledge impedes new-class learning.

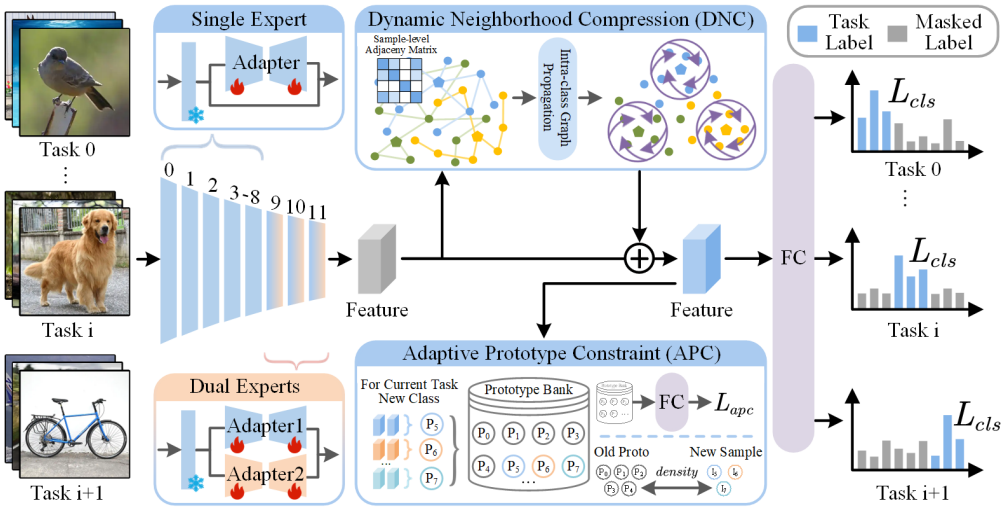

Figure 2: Overall architecture of SCMNet. A frozen backbone with fixed adapters extracts features, which are refined by DNC through label-constrained graph propagation to suppress intra-class variance. APC locks historical decision boundaries by anchoring the classifier on stored prototypes.

We adopt the prevailing PTM-based CIL paradigm: a frozen pre-trained ViT backbone with parameter-efficient fine-tuning (PEFT) through adapters, and a unified classification head $\mathbf{W} \in \mathbb{R}^{C_{\text{total}} \times d}$ pre-allocated for all C_{total} classes. Unlike expansion-based methods that grow capacity per task (Wang et al., 2025a; Wan et al., 2026), SCMNet keeps total trainable parameters constant and directly organizes the feature space through two structural modules: Dynamic Neighborhood Compression (DNC) for intra-class compaction and Adaptive Prototype Constraint (APC) for old-class boundary preservation. The overall framework is shown in Figure 2.

3.2 Fixed Adapter Configuration

SCMNet inserts lightweight bottleneck adapters (Houlsby et al., 2019) in parallel at every Transformer layer of a frozen ViT-B/16 backbone. Following SEMA's (Wang et al., 2025a) insight that moderately enriched capacity at selected layers benefits long-sequence adaptation, we equip layers 8–11 with two expert adapters while the remaining layers carry a single adapter. Differently, two adapters at each dual-expert layer are fused by simple element-wise addition with no learnable router; this design inherits only the lightweight parallelism of SEMA's expert structure while discarding its representation descriptors, expansion triggers, and mixture routing components that together drive unbounded parameter growth ($\sim$223K per expansion).

3.3 Dynamic Neighborhood Compression

Frozen backbone features carry class-relevant structure alongside class-irrelevant nuisances (pose, illumination, background) that inflate intra-class variance. Under incremental learning, this uncompressed variance wastes capacity in an increasingly crowded feature space and introduces estimation noise into class prototypes, directly degrading the precision of downstream constraints. DNC addresses this via label-constrained graph propagation, instantiating the compression objective of minimizing intra-class redundancy. The construction and transfer of the graph are shown in Figure 3.

Graph Construction. For a batch $\{x_i, y_i\}_{i=1}^{N}$ with backbone features $\mathbf{X} \in \mathbb{R}^{N \times d}$, DNC first computes pairwise cosine similarity between all samples:

$$\mathbf{S}_{ij} = \frac{\mathbf{X}_i^\top \mathbf{X}_j}{\|\mathbf{X}_i\| \cdot \|\mathbf{X}_j\|}, \quad i \neq j; \quad \mathbf{S}_{ii} = 1. \tag{1}$$

A label-consistency mask then zeros out any remaining edges between samples of different classes:

$$\mathbf{M}_{ij}^{\text{label}} = \begin{cases} 1, & y_i = y_j \\ 0, & y_i \neq y_j \end{cases}, \quad \tilde{\mathbf{S}} = \mathbf{S} \odot \mathbf{M}^{\text{label}}. \tag{2}$$

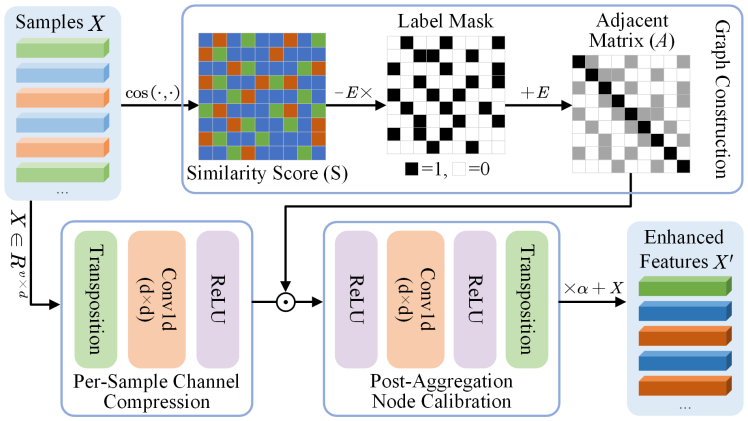

Figure 3: Workflow of DNC. A weighted, label-constrained adjacency matrix is constructed from each training batch. Features undergo per-sample channel compression, graph aggregation, node calibration, and residual fusion with weight α for class-guided information aggregation.

Finally, the resulting adjacency matrix $\tilde{\mathbf{S}}$, which encodes weighted intra-class-only edges, is subject to row-wise normalization followed by the addition of self-loops to yield the adjacency matrix $\mathbf{A} = \frac{\tilde{\mathbf{S}}}{\sum_j \tilde{\mathbf{S}}_{ij}} + \mathbf{I}$.

Two-Stage Graph Convolution. Operating on transposed features $\mathbf{X} \in \mathbb{R}^{d\times N}$, a single GCN layer decomposes compression into two stages. *Channel compression:* a shared 1×1 Conv1d $\mathbf{W}_s$ is applied per-node, learning a class-agnostic profile that suppresses nuisance-dominated dimensions while retaining object-structure dimensions: $\hat{\mathbf{X}} = \mathbf{X} + \mathbf{W}_s(\mathbf{X})$. *Node calibration:* features undergo graph message passing, where same-class neighbors pull each node toward the class consensus, dominated by its own signal through the strong self-loop. ReLU activation and a second transformation $\mathbf{W}_d$ extract corrective patterns from the post-aggregation context, formulated as:

$$\mathbf{X}' = \mathrm{ReLU}\Big(\mathbf{W}_d\Big(\mathrm{ReLU}\Big(\hat{\mathbf{X}} \cdot \mathbf{A}\Big)\Big)\Big). \tag{3}$$

The output is fused through a residual connection $\tilde{\mathbf{X}} = \mathbf{X} + \alpha \cdot \mathbf{X}'$. Unlike standard PEFT adapters that adopt small scaling factors $\alpha \ll 1$ for subtle feature adjustment, we adopt a considerably larger α, since DNC is meant to actively reshape the feature distribution rather than apply a minor correction. A value of $\alpha = 4$ already attains the best performance on every benchmark, and increasing α further strengthens the graph constraint and brings additional gains as the number of classes per task grows. The full sweep is reported in Table 8 of the appendix.

3.4 Prototype Memory Bank

After each task training, SCMNet constructs prototypes from DNC-enhanced features. For class c with N_c training samples: $\boldsymbol{\mu}_c = \frac{1}{N_c}\sum_{i:y_i=c} \tilde{\mathbf{f}}_i$, where $\tilde{\mathbf{f}}_i$ are extracted in eval mode with DNC. Prototypes remain unnormalized for scale consistency with training features. The memory bank $\mathcal{M} = \{(\boldsymbol{\mu}_c, c)\}$ accumulates across tasks and serves as the bridge between DNC and APC: DNC's compression yields lower-variance prototypes, which APC then uses as high-fidelity anchors for classifier regularization.

3.5 Adaptive Prototype Constraint

A primary forgetting mechanism in CIL is classifier drift: as gradient updates from new-class data revise the classification head $\mathbf{W} = [\mathbf{w}_0, \ldots, \mathbf{w}_{K-1}]$, old-class weight vectors deviate from their optimal configurations relative to historical feature distributions. APC counters this by treating stored prototypes as virtual training samples, instantiating the retention objective of preserving historical discriminability. During training of task t ($t > 0$), APC feeds all old-class prototypes $\{\boldsymbol{\mu}_j\}_{j<K}$

through the current classification head and applies cross-entropy with their ground-truth labels:

$$\mathcal{L}_{\text{apc}} = -\frac{1}{|\mathcal{M}_{\text{old}}|} \sum_{(\boldsymbol{\mu}_j, c_j) \in \mathcal{M}_{\text{old}}} \log \frac{\exp(\mathbf{w}_{c_j}^\top \boldsymbol{\mu}_j)}{\sum_{k=0}^{K-1} \exp(\mathbf{w}_k^\top \boldsymbol{\mu}_j)}. \tag{4}$$

This single term enforces dual constraints: high probability on the correct class (pull) and low probability on all other old classes (push), jointly locking the historical decision boundaries.

The strength of the consolidation constraint follows the state of the prototype space. We quantify how crowded that space is by the average nearest-neighbour cosine similarity among the stored prototypes:

$$\gamma = \frac{1}{|\mathcal{M}|} \sum_{c \in \mathcal{M}} \max_{c' \neq c} \frac{\boldsymbol{\mu}_c^\top \boldsymbol{\mu}_{c'}}{\|\boldsymbol{\mu}_c\| \cdot \|\boldsymbol{\mu}_{c'}\|}. \tag{5}$$

From the first incremental task onwards, $\mathcal{L}_{\text{apc}}$ carries the weight $\lambda_{\text{apc}} = 1.0/(\gamma + \epsilon)$ with $\epsilon = 0.1$. Early in the sequence, when few classes have been observed and the feature space is still ample, γ takes a small value and the weight a large one, so that the classes are consolidated into regions lying as far apart from one another as the space allows and the resulting layout reserves room for the classes yet to arrive. As more classes are placed into the same space, γ increases and the anchoring applied to the prototypes accumulated so far weakens progressively rather than remaining constant, so that incoming classes are accommodated without contending for the limited space available.

3.6 Training Procedure and Inference

The classification head $\mathbf{W} \in \mathbb{R}^{C_{\text{total}} \times 768}$ is pre-allocated at initialization; only rows corresponding to seen classes receive gradients.

Task 0. With an empty memory bank, training uses standard cross-entropy over DNC-enhanced features:

$$\mathcal{L}^{(0)} = \mathcal{L}_{cls} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\mathbf{w}_{y_i}^\top \tilde{\mathbf{f}}_i)}{\sum_{k=0}^{C_0-1} \exp(\mathbf{w}_k^\top \tilde{\mathbf{f}}_i)}. \tag{6}$$

After training, prototypes $\{\boldsymbol{\mu}_c\}_{c=0}^{C_0-1}$ are constructed in Secion 3.4.

Task t ($t > 0$). Old-class logits are masked to a large negative value in $\mathcal{L}_{\text{cls}}$. The total loss combines classification with APC:

$$\mathcal{L}^{(t)} = \mathcal{L}_{\text{cls}} + \lambda_{\text{apc}} \cdot \mathcal{L}_{\text{apc}}. \tag{7}$$

After training, new-class prototypes are appended to $\mathcal{M}$. The two modules form a closed loop: DNC suppresses intra-class variance, yielding cleaner prototypes for APC; APC stabilizes decision boundaries, providing a more regularized feature space for subsequent DNC rounds.

Inference. At test stage, the memory bank is disabled and graph construction is omitted; for single-sample inputs, the graph structure degenerates to an identity graph. The final prediction is obtained by taking the argmax over the dimensions corresponding to all seen classes. No rehearsal of past samples is required, and both the parameter count and inference overhead remain constant.

4 Experiments

4.1 Datasets and Experimental Setup

We evaluate on four standard class-incremental datasets: CIFAR-100 (Krizhevsky, 2009) (100 classes), ImageNet-R (Hendrycks et al., 2021a) (IN-R, 200 classes of artistic renditions), ImageNet-A (Hendrycks et al., 2021b) (IN-A, 200 natural adversarial samples), and VTAB (Zhai et al., 2020) (50 classes across heterogeneous domains). All datasets are evaluated under 5-, 10-, and 20-task incremental splits with shuffled class orders.

Implementation details. All experiments use an ImageNet-1K pretrained ViT-B/16 as the frozen backbone. Bottleneck adapters are inserted at every Transformer layer, with layers 8–11 carrying dual adapters and the remaining layers single adapters, totaling $\sim$1.7M trainable parameters. We optimize with SGD (initial learning rate 5×10^{-3}, cosine annealing to 10^{-4}, batch size 64, 20

Table 1: Comparison with state-of-the-art methods across six benchmarks. $\bar{A}$ and A_N denote average and last-task accuracy. All methods use ViT-B/16-IN1K backbone. Best results in **bold**, second best underlined.

Method	10-Task CIFAR		5-Task IN-R		10-Task IN-R		20-Task IN-R		20-Task IN-A		5-Task VTAB	
	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N
FT Adapter (Chen et al., 2022)	47.88	30.90	53.91	41.23	45.31	30.93	38.51	24.22	29.78	17.64	59.98	43.50
L2P (Wang et al., 2022c)	84.77	77.87	77.40	73.59	66.97	62.72	70.67	62.90	47.16	38.48	81.19	80.83
DualPrompt (Wang et al., 2022b)	86.60	80.43	76.39	72.29	72.83	66.75	62.33	61.97	59.54	50.23	82.89	79.79
CODA-P (Smith et al., 2023)	91.55	86.11	81.63	76.98	81.11	75.25	75.00	70.02	47.29	35.02	79.88	81.58
SimpleCIL (Zhou et al., 2025a)	82.31	76.21	65.83	61.31	67.09	61.35	67.59	61.35	60.05	49.24	85.29	83.61
ADAM (Zhou et al., 2025a)	90.55	85.62	79.91	74.25	79.11	73.15	75.84	69.10	60.15	49.24	85.29	83.61
InfLoRA (Liang & Li, 2024)	90.51	85.05	78.58	72.58	81.39	75.32	78.87	72.60	59.71	46.21	88.90	87.63
SEMA (Wang et al., 2025a)	91.37	86.98	84.75	79.78	83.56	78.00	81.75	74.53	64.53	53.32	91.26	89.64
SR^2-LoRA (Wan et al., 2026)	94.60	91.98	86.01	81.72	85.93	81.14	84.32	79.54	69.41	61.01	–	–
KCEA (Tran et al., 2026)	94.67	91.19	-	-	86.01	80.70	-	-	-	-	94.69	89.36
SCMNet	**94.80**	**92.26**	**91.14**	**87.48**	**89.91**	**85.68**	**86.07**	**80.92**	**72.17**	**61.49**	**95.06**	**95.49**

Table 2: Component ablation on IN-R. "✓" denotes the component is active.

Component		5-Task		10-Task		20-Task	
DNC	APC	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N
–	–	84.78	80.42	84.47	81.07	83.04	79.66
✓	–	89.41	85.47	88.81	84.32	85.75	81.55
–	✓	84.90	80.65	84.48	79.67	83.55	79.90
✓	✓	**91.14**	**87.48**	**89.91**	**85.68**	**86.07**	**80.92**

epochs per task). We report average accuracy $\bar{A}$ (mean of all-class accuracies after each task) and last-task accuracy A_N (accuracy over all seen classes after the final task), following standard CIL protocol (Wang et al., 2025a; Wan et al., 2026). More detailes can be found in Appendix A.1.

4.2 Comparison with State-of-the-Art Methods

Table 1 compares SCMNet with recent PTM-based CIL methods under the IN1K backbone. SCMNet attains the best results on all of the six benchmarks.

ImageNet-R. IN-R presents the largest domain gap from ImageNet-1K pretraining. Prompt-based methods (L2P, DualPrompt) degrade sharply with longer sequences (DualPrompt drops from 76.39% to 62.33% $\bar{A}$), as they modulate attention without compacting intra-class features. Prototype-based approaches (SimpleCIL, ADAM) stay stable but plateau at low accuracy. Expansion-based SEMA drops from 84.75% to 81.75% as frozen legacy adapters block cross-task sharing, while SR^2-LoRA's model-level regularization cannot rival DNC's direct feature-space compaction.

CIFAR and VTAB. On benchmarks with smaller domain gaps, the competitive landscape is tighter. On CIFAR, CODA-P and InfLoRA exhibit near-saturated performance, while KCEA and SR^2-LoRA operate near the empirical upper bound; even under such favorable conditions for existing competitors, SCMNet still attains the leading average accuracy. On VTAB with limited incremental learning stages, SCMNet achieves the best average accuracy across all tasks, and its accuracy on the final task even exceeds this overall average. As pre-trained representations naturally align well with downstream data distributions on these datasets, explicit feature-space compression yields comparatively moderate improvements, yet SCMNet remains competitive with consolidation-based methods while keeping a constant parameter budget.

ImageNet-A. IN-A contains natural adversarial examples that pretrained models systematically misclassify, collapsing prompt-based methods (L2P and CODA-P) and limiting the second-best method to $\bar{A} = 69.41\%$. SCMNet achieves $\bar{A} = 72.17\%$, exceeding SEMA by 7.64 and SR^2-LoRA by 2.76 points. Unlike SEMA's reconstruction-error descriptors, which cannot separate adversarial perturbations from natural intra-class variation, DNC's label-constrained graph propagation pulls clean and adversarial samples toward shared class centroids, mitigating classifier fragility.

Table 3: Adapter configuration ablation on IN-R. The adopted configuration is shaded.

Adapters	Location	5-Task		10-Task		20-Task	
		$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N
Single	all layers	91.05	87.27	89.84	85.60	85.85	80.55
Double	11 – 11	90.96	86.95	87.78	85.35	85.81	80.98
Double	10 – 11	90.56	86.23	89.63	84.70	85.60	80.98
Double	9 – 11	90.39	86.38	89.75	85.22	86.12	80.52
Double	8 – 11	91.14	**87.48**	89.91	85.68	86.07	80.92
Double	7 – 11	91.10	87.03	**90.09**	85.37	86.21	**81.13**
Double	6 – 11	**91.42**	87.35	89.91	**85.75**	**86.44**	81.05

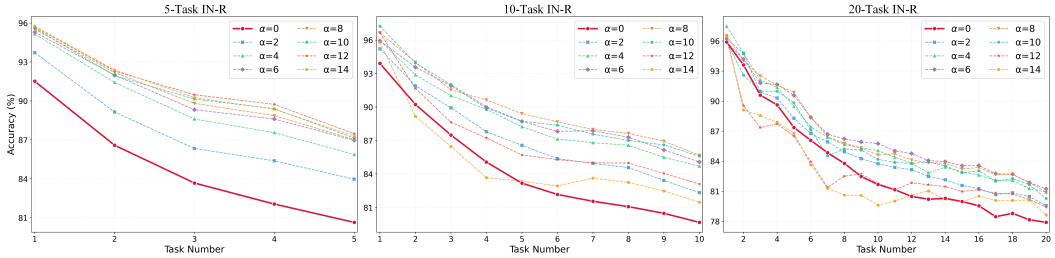

Figure 4: Accuracy versus different DNC residual weights α on IN-R under 5/10/20-task settings.

4.3 Ablation Studies

Component contribution. Table 2 isolates DNC and APC. The adapter-only baseline (DNC off, APC off) already achieves $\bar{A} = 84.78\%$ on 5-task IN-R, confirming that a frozen backbone with lightweight adapters provides a strong foundation. Enabling DNC alone yields a substantial gain ($+4.63$ points $\bar{A}$ on 5-task), confirming that intra-class compression directly benefits discrimination even without explicit old-class constraints. This gain persists across all task lengths, indicating that DNC's effect is structural rather than task-count-dependent. Enabling APC alone produces only a marginal improvement ($+0.12$ points), which is expected: without DNC-compressed features, the prototypes fed to APC carry inflated estimation variance, severely limiting the constraint's precision. When both modules are active, performance jumps to $\bar{A} = 91.14\%$, a gain of $+6.36$ points over the baseline. The improvement exceeds the combined performance of standalone modules, verifying super-additivity and the mutual interplay of DNC and APC.

Adapter configuration. Table 3 compares adapter placements. Across all task lengths, the gap between single and the best dual-adapter configuration never exceeds 0.59 points in $\bar{A}$, confirming that adapter capacity is not the primary driver. Narrowly scoped dual adapters (layers 11–11 or 10–11) even underperform the single-adapter baseline, as a representation-depth mismatch disrupts cross-layer consistency; widening the range (9–11 through 6–11) resolves this and yields monotonic gains. We adopt layers 8–11, balancing capacity and overhead, confirming that SCMNet's advantage stems from DNC and APC rather than the dual-expert setup.

DNC residual weight α. The residual weight α governs the strength of DNC's remodulation of the backbone features. As shown in Figure 4, accuracy rises sharply with α and peaks in the range $\alpha \in [4, 12]$, while $\alpha = 0$ reduces the model to the APC-only ablation and confirms that DNC is necessary. At $\alpha = 4$, our model already outperforms every method compared in Table 7. When a single task has to learn more classes, raising α further and thereby strengthening the organization of the prototype library still brings additional gains, as in the 5-task split of IN-R, which assigns 40 classes per task against 10 in the 20-task split. The full sweep is deferred to Appendix A.3.

Extended comparison with SEMA and SR^2-LoRA. Table 4 compares three datasets across three task splits, where SCMNet leads on 8 of 9 settings in $\bar{A}$. The sole exception is CIFAR at 20 tasks, where SR^2-LoRA overtakes SCMNet because CIFAR's small domain gap leaves little intra-class variance for DNC to compress, while SVD-based alignment directly benefits from inter-layer drift

Table 4: Extended task-sequence comparison with SEMA and SR^2-LoRA.

Dataset	Method	5-Task		10-Task		20-Task	
		$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N
CIFAR	SEMA	92.39	88.42	91.63	86.87	87.97	81.32
	SR^2-LoRA	94.91	92.26	94.60	91.98	**94.08**	**90.45**
	SCMNet	**95.66**	**93.45**	**94.80**	**92.26**	92.73	88.38
IN-R	SEMA	84.75	79.78	83.56	78.00	81.75	74.53
	SR^2-LoRA	86.01	81.72	85.93	81.14	84.32	79.54
	SCMNet	**91.14**	**87.48**	**89.91**	**85.68**	**86.07**	**80.92**
IN-A	SEMA	70.17	61.16	70.70	60.17	64.53	53.32
	SR^2-LoRA	70.67	63.27	70.03	**63.47**	69.41	61.01
	SCMNet	**74.77**	**64.65**	**72.79**	61.42	**72.17**	**61.49**

Table 5: Backbone generalization on IN-R. "SCM" denotes DNC and APC enabled.

Backbone	SCM	5-Task		10-Task		20-Task	
		$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N
ViT-B/16-IN1K	–	84.78	80.42	84.47	81.07	83.04	79.66
	✓	91.14	87.48	89.91	85.68	86.07	80.92
ViT-B/16-CLIP	–	86.04	80.83	85.30	78.87	83.66	76.42
	✓	91.25	85.63	87.78	80.70	85.12	77.05
ViT-B/16-DINOv3	–	91.50	87.28	90.92	85.57	89.81	83.70
	✓	**94.81**	**91.10**	**94.32**	**89.78**	**92.05**	**85.57**

accumulated. On IN-R and IN-A, SCMNet's feature-space organization provides decisive advantages.

Backbone generalization. Table 5 examines whether SCMNet amplifies stronger pretrained representations or is specific to the IN1K backbone. Across ViT-B/16 backbones pretrained on IN1K, CLIP, and DINOv3, enabling DNC and APC yields consistent and substantial gains; under DINOv3, SCMNet boosts the 5-task IN-R $\bar{A}$ from 91.50% to 94.81%. This confirms SCMNet as a backbone-agnostic structural enhancer atop any frozen visual encoder.

Parameter efficiency. Table 6 compares two budgets. SEMA assigns one adapter per layer ($\sim$304K) and adds auxiliary autoencoders and expandable routers ($\sim$600K at Task 0), each expansion contributing a further $\sim$223K. SCMNet uses $8 \times 1 + 4 \times 2$ adapters ($\sim$405K) and the DNC enhancer ($\sim$1180K), so about 1.7M parameters are trainable from the first task onwards and stay that way. Notably, our lead does not come from the larger adapter stack, since with the same single adapter per layer as SEMA the model already outperforms it (Table 3). The two budgets diverge as the sequence grows, since three expansions bring SEMA level with the 1.7M we spend once and for all and further ones take it past that figure, while ours stays fixed however many tasks arrive.

Table 6: Parameter budget comparison.

Method	ViT (frozen)	Adapters (trainable)	Auxiliary (trainable)	FC (trainable)	Extra per Expansion
SEMA	$\sim$86M	$\sim$304K	$\sim$600K	$\sim$153K	$+$223K
SCMNet	$\sim$86M	$\sim$405K	$\sim$1180K	$\sim$153K	0

5 Conclusion

This work presents SCMNet, a rehearsal-free framework for class-incremental learning that decouples intra-class feature compression and old-class knowledge retention into two mutually reinforcing modules under a strictly fixed parameter budget. Extensive evaluations across six benchmarks and three pre-trained backbones verify that structured feature-space regulation is a more effective CIL solution than unbounded parameter expansion or standalone model-level regularization.

REFERENCES

Xu An, Yongming Wu, and Yang Zheng. A regularized class incremental learning method based on self-supervision with distillation constraints. *Journal of Computer-Aided Design and Computer Graphics*, 36(5):775–785, 2024. doi: 10.3724/SP.J.1089.2024.19848.

Xiaohuan Bing, Yuanhao Xi, and Ramin Yahyapour. SA-LoRA: Shared-A decoupled low-rank adaptation for class-incremental learning. *Journal of King Saud University - Computer and Information Sciences*, 38:511, 2026. doi: 10.1007/s44443-026-00925-x.

Amin Birashk and Latifur Khan. Federated continual learning for task-incremental and class-incremental problems: A survey. *Expert Systems with Applications*, 297:129278, 2026. doi: 10.1016/j.eswa.2025.129278.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. AdaptFormer: Adapting vision transformers for scalable visual recognition. In *Adv. Neural Inform. Process. Syst.*, pp. 16664–16678, 2022. doi: 10.52202/068431-1212.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024. doi: 10.1038/s41586-024-07711-7.

Mingda Dong, Zhizhong Zhang, Xin Tan, Jiling Qiu, and Yuan Xie. Dynamic expansion orthogonal network for class-incremental learning. *Knowledge-Based Systems*, 341:115868, 2026. doi: 10.1016/j.knosys.2026.115868.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. Learn. Represent.*, 2021. doi: 10.48550/arXiv.2010.11929.

Kaile Du, Junzhou Xie, Fan Lyu, Yifan Zhou, Zihan Ye, Wei Li, Yuyang Li, and Guangcan Liu. Negative-weighted knowledge distillation regularized graph convolutional network for multi-label class-incremental learning. *Pattern Recognition*, 179:113657, 2026. doi: 10.1016/j.patcog.2026.113657.

Takuma Fukuda, Hiroshi Kera, and Kazuhiko Kawamoto. Adapter merging with centroid prototype mapping for scalable class-incremental learning. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 4884–4893, 2025. doi: 10.1109/CVPR52734.2025.00460.

Xueze Gao, Quan Feng, Jianhua Zhang, Sen Yang, Jian Zhao, and Haiyan Gao. Incremental learning with dynamic memory replay for crop-disease classification across diverse environments. *Knowledge-Based Systems*, 340:115703, 2026. doi: 10.1016/j.knosys.2026.115703.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Int. Conf. Comput. Vis.*, pp. 8320–8329, 2021a. doi: 10.1109/ICCV48922.2021.00823.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 15257–15266, 2021b. doi: 10.1109/CVPR46437.2021.01501.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Int. Conf. Mach. Learn.*, pp. 2790–2799, 2019.

Saeed Iqbal, Muhammad Attique Khan, Ghulam Mustafa, Sarra Ayouni, Abeer Aljohani, and Amir Hussain. Fedcapd: Federated class-incremental learning via capsule distillation and diffusion replay. *Neurocomputing*, 683(C), 2026. doi: 10.1016/j.neucom.2026.133365.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Yan-Shuo Liang and Wu-Jun Li. InfLoRA: Interference-free low-rank adaptation for continual learning. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 23638–23647, 2024. doi: 10.1109/CVPR52733.2024.02231.

Zeyi Liu, Xiao He, Biao Huang, and Donghua Zhou. Incremental learning-enabled fault diagnosis of dynamic systems: A comprehensive review. *IEEE Transactions on Cybernetics*, 55(12):5633–5649, 2025. doi: 10.1109/TCYB.2025.3586643.

Gen Luo, Jiamu Sun, Lei Jin, Yiyi Zhou, Qiang Xu, Rongrong Fu, Xiaoshuai Sun, and Rongrong Ji. Domain incremental learning for object detection. *Pattern Recognition*, 170:111882, 2026. doi: 10.1016/j.patcog.2025.111882.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Int. Conf. Mach. Learn.*, pp. 8748–8763, 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 *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 11909–11919, 2023. doi: 10.1109/CVPR52729.2023.01146.

Tung Tran, Marko Zolo Gozano Untalan, Zikang Wan, and Danilo Vasconcellos Vargas. Knowledge consolidation with evolutionary alignment for class-incremental learning. *Neurocomputing*, 696: 134175, 2026. doi: 10.1016/j.neucom.2026.134175.

Fengqiang Wan, Yipeng Lin, Kan Lv, and Yang Yang. SR^2-LoRA: Self-rectifying inter-layer relations in low-rank adaptation for class-incremental learning. *arXiv*, 2026. doi: 10.48550/arXiv.2605.07420.

Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. FOSTER: Feature boosting and compression for class-incremental learning. In *Eur. Conf. Comput. Vis.*, pp. 398–414, 2022a. doi: 10.1007/978-3-031-19806-9_23.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. In *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 10087–10098, 2025a. doi: 10.1109/CVPR52734.2025.00943.

Yan Wang, Da-Wei Zhou, and Han-Jia Ye. Integrating task-specific and universal adapters for pre-trained model-based class-incremental learning. In *Int. Conf. Comput. Vis.*, pp. 806–816, 2025b.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *Eur. Conf. Comput. Vis.*, pp. 631–648, 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 *IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pp. 139–149, 2022c. doi: 10.1109/CVPR52688.2022.00024.

Mang Ye, Wenke Huang, Zekun Shi, Zhiwei Ye, and Bo Du. BiaMix: Contrastive learning and memory similarity distillation in class-incremental learning. *CAAI Transactions on Intelligence Technology*, 10(6):1745–1758, 2025. doi: 10.1049/cit2.70064.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. The visual task adaptation benchmark. *ICLR*, 2020.

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. *Int. J. Comput. Vis.*, 133(3):1012–1032, 2025a. doi: 10.1007/s11263-024-02218-0.

Da-Wei Zhou, Yuanhan Zhang, Yan Wang, Jingyi Ning, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Learning without forgetting for vision-language models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 47(6):4489–4504, 2025b. doi: 10.1109/TPAMI.2025.3540889.

A Appendix

A.1 Reproducibility Statement

Code and configuration release. To foster reproducibility, we release the complete source code together with all configuration files as an anonymized code package (a compressed archive) attached as supplementary material. The package contains the full training and evaluation pipeline of SCMNet as well as the reproduced baseline implementations. Each configuration file under `exps/` corresponds one-to-one to a row in the main results table (Table 1), so every reported number can be reproduced by running a single command with the matching config.

Reproducibility design. We make our results deterministic and easily reproducible by design. All runs fix the random seed and disable non-deterministic CUDA kernels; the optimizer, learning-rate schedule, batch size, and epoch counts are stated in Section4.1 and pinned in the released configs. All backbones are publicly released pre-trained models (ViT-B/16 on ImageNet-1K, CLIP, and DINOv3), and all six benchmarks (CIFAR-100, ImageNet-R, ImageNet-A, and VTAB) are public datasets processed under the standard class-incremental protocol described in Section 4.1.

Fairness design. Our comparison is anchored on SEMA, the capacity-expanding baseline analysed throughout this paper, and is conducted under a single protocol: every method in Table 1 is evaluated with the same frozen ViT-B/16-IN1K backbone, the same task splits with a fixed random class ordering, and the same two metrics ($\bar{A}$ and A_N) defined in Section 4.1. The numbers reported for the compared methods are quoted from the SEMA (Wang et al., 2025a), where they were obtained under exactly this protocol; the only exceptions are SR^2-LoRA (Wan et al., 2026) and KCEA (Tran et al., 2026), whose results are quoted from their own papers. Cells left as "$-$" indicate settings that the corresponding source paper does not report. All numbers for SCMNet itself are produced by running the released configuration files.

A.2 Variance Trends over Growing Task Sequences

The right panel of Figure 1 reports the variance analysis on the 10-task protocol of ImageNet-R, and this appendix repeats the same measurement on the 5- and 20-task protocols so that the claim can be examined on short and long sequences alike. The protocol is identical everywhere: once a task finishes training, the current model is frozen and features are extracted over the test set of all classes seen so far. The intra-class variance averages, over classes, the squared distances between each sample and the mean of its own class, and the inter-class variance measures how far the class means spread around the global mean. Their ratio is what ultimately matters, because it is high exactly when samples gather tightly around their prototypes while the prototypes stay separated. Both methods are measured under this protocol, SCMNet with the adopted configurations.

Figure 5 shows that the two methods differ less in the level at which they start than in the way they evolve. The variances of SEMA run high and do not evolve smoothly, since its curves hold one level for several tasks and then shift to another. Consequently, the space is repeatedly re-established rather than progressively refined, so the geometry reached at the end of one stretch of the sequence is not the geometry that the following tasks continue from. SCMNet stays well below SEMA on both variances and keeps its own curves close to flat from the first task to the last. Its reduction of the intra-class term is consistently about one and a half times its reduction of the inter-class term, so samples are pulled together far more than classes are drawn towards each other, and the inter-to-intra ratio therefore stays clearly above that of SEMA on every protocol.

The two ratios also respond differently to the length of the sequence, and this is where the value of a continuously maintained space becomes visible. On the 5- and 10-task protocols both drift downward after the first tasks, the one of SCMNet staying at a much higher level throughout. At 20 tasks they part, with the ratio of SEMA continuing the decline observed at shorter lengths while that of SCMNet climbs slowly and finishes above where it started. Compression and prototype anchoring therefore do not behave as a one-off correction that wears off as tasks accumulate, but

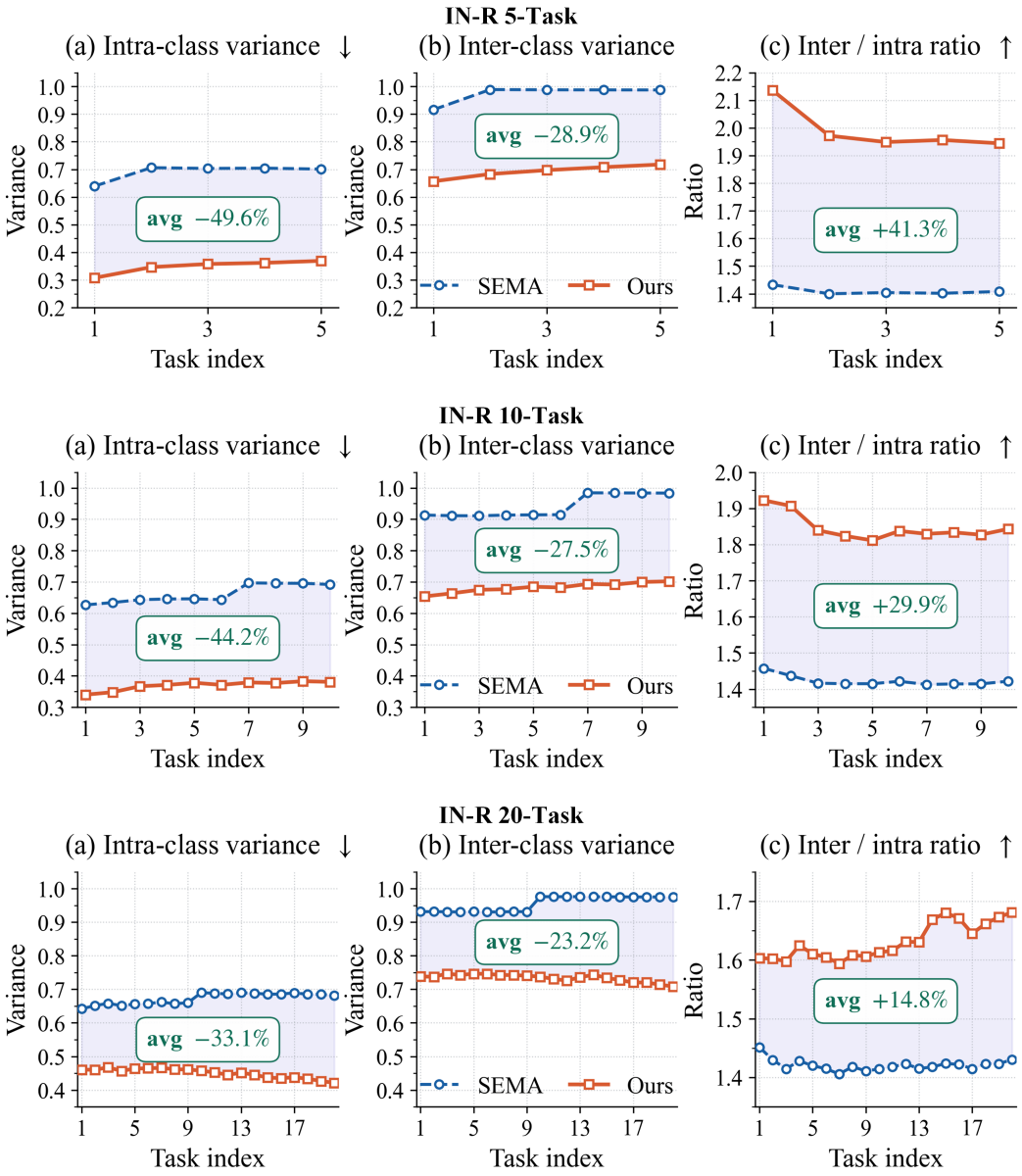

Figure 5: Intra-class variance (left), inter-class variance (middle), and their ratio (right) measured after each task on the 5-, 10- and 20-task protocols of ImageNet-R. Shaded bands mark the average gap between SEMA and SCMNet.

as an organization that keeps working along the sequence, since every task inherits a space that earlier tasks have already tightened and leaves it slightly more separated than it found it. This is the restrained organization the method rests on, the same fixed budget maintaining the space rather than enlarging it, so that incoming classes are absorbed by refining the geometry instead of re-establishment.

A.3 DNC Residual Weight Sweep

The residual weight α controls how strongly DNC remodulates the backbone features, and Table 8 sweeps it from 0, which removes the module entirely, to 14 on all six benchmarks. Removing DNC always costs accuracy, by 0.6 points on 20-task ImageNet-A and by up to 6.2 points on 5-task

ImageNet-R, so the module contributes the gain rather than decorating it. Accuracy rises quickly once α leaves zero and reaches its maximum somewhere in $\alpha \in [4, 12]$ on every benchmark. Values in this range are far larger than the scaling factors carried by ordinary PEFT adapters, which is consistent with what DNC does, since it reorganizes the feature space rather than applying a mild correction to it. How far the weight can be pushed, however, is not decided by the module alone, because it also depends on how much relational structure each incremental step offers to the graph, and it is here that the benchmarks begin to differ.

The decisive quantity is the number of classes a single task has to absorb, which is 40 on 5-task ImageNet-R, 20 on 10-task ImageNet-R, and about 10 on the remaining four settings. When a task brings in many classes at once, one graph is built over a much larger set of prototypes, so more relations are available for the message passing to exploit and more structure is there for a strong constraint to organize, which is why strengthening the constraint keeps paying off. On 5-task ImageNet-R the average accuracy is still increasing at $\alpha = 12$ and stays within 0.2 points of its maximum at $\alpha = 14$, and on 10-task ImageNet-R the optimum moves to $\alpha = 8$ with the three neighbouring values $\alpha \in \{6, 8, 10\}$ lying within half a point of one another. Figure 6 further shows that the gain is not deferred to the end of the sequence, since the curve obtained at $\alpha = 12$ on 5-task ImageNet-R lies above the one obtained at $\alpha = 4$ at every step rather than only at the last one.

The picture reverses when a task contributes only about ten classes. Within one step there are then far fewer prototypes to relate, so the graph transmits less information and imposes a weaker constraint, and the task itself is easier, which leaves a strong constraint little to correct and much to disturb. Accuracy still peaks early, between $\alpha = 4$ and $\alpha = 6$, but it falls away steeply afterwards, with the last-task accuracy dropping from 92.26 to 85.25 on 10-task CIFAR and from 61.49 to 54.44 on 20-task ImageNet-A between $\alpha = 4$ and $\alpha = 14$, against a change of 1.42 points in the opposite direction on 5-task ImageNet-R. Large α amplifies the fluctuation instead of damping it in this regime, and Figure 6 locates the effect precisely, since the curves of the settings with few classes per task start together at the first task and then fan out from the second one onwards, whereas those of the settings with many classes per task stay close to each other throughout.

We therefore tie the adopted weight to the size of a task and use $\alpha = 12$, 8 and 4 for the 40, 20 and roughly 10 classes per task settings, as marked by the shaded cells of Table 8. A single value $\alpha = 4$ is kept for the whole ten-class group because it is the best or within 0.5 points of the best value on each of those four benchmarks, while the same choice would leave 1.4 and 1.1 points unused on 5-task and 10-task ImageNet-R. Even with $\alpha = 4$ fixed for all six benchmarks, the model already outperforms every compared method in Table 7.

Table 7: Comparison with state-of-the-art methods across six benchmarks. $\bar{A}$ and A_N denote average and last-task accuracy. All methods use ViT-B/16-IN1K backbone. Best results in **bold**, second best underlined.

Method	10-Task CIFAR		5-Task IN-R		10-Task IN-R		20-Task IN-R		20-Task IN-A		5-Task VTAB	
	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N
FT Adapter (Chen et al., 2022)	47.88	30.90	53.91	41.23	45.31	30.93	38.51	24.22	29.78	17.64	59.98	43.50
L2P (Wang et al., 2022c)	84.77	77.87	77.40	73.59	66.97	62.72	70.67	62.90	47.16	38.48	81.19	80.83
DualPrompt (Wang et al., 2022b)	86.60	80.43	76.39	72.29	72.83	66.75	62.33	61.97	59.54	50.23	82.89	79.79
CODA-P (Smith et al., 2023)	91.55	86.11	81.63	76.98	81.11	75.25	75.00	70.02	47.29	35.02	79.88	81.58
SimpleCIL (Zhou et al., 2025a)	82.31	76.21	65.83	61.31	67.09	61.35	67.59	61.35	60.05	49.24	85.29	83.61
ADAM (Zhou et al., 2025a)	90.55	85.62	79.91	74.25	79.11	73.15	75.84	69.10	60.15	49.24	85.29	83.61
InfLoRA (Liang & Li, 2024)	90.51	85.05	78.58	72.58	81.39	75.32	78.87	72.60	59.71	46.21	88.90	87.63
SEMA (Wang et al., 2025a)	91.37	86.98	84.75	79.78	83.56	78.00	81.75	74.53	64.53	53.32	91.26	89.64
SR^2-LoRA (Wan et al., 2026)	94.60	91.98	86.01	81.72	85.93	81.14	84.32	79.54	69.41	61.01	–	–
KCEA (Tran et al., 2026)	94.67	91.19	-	-	86.01	80.70	-	-	-	-	94.69	89.36
SCMNet ($\alpha = 4$)	**94.80**	**92.26**	89.72	85.88	88.85	84.70	**86.07**	**80.92**	**72.17**	**61.49**	**95.06**	**95.49**
SCMNet (adopted α)	**94.80**	**92.26**	**91.14**	**87.48**	**89.91**	**85.68**	**86.07**	**80.92**	**72.17**	**61.49**	**95.06**	**95.49**

A.4 Disclosure of Generative AI Use

In this work, generative AI tools were used for two purposes only, namely to assist with the translation of manuscript drafts and to edit them for grammar, clarity and consistency of terminology. Both activities act on text that the authors had already written, and every suggested formulation was read, checked against the corresponding experiment or derivation, and either accepted or rewritten by the authors before it entered the manuscript.

Table 8: DNC residual weight α sweep across all benchmarks. $\alpha = 0$ disables DNC. The adopted configuration is shaded, best results in **bold**, second best underlined.

α	5-Task IN-R		10-Task IN-R		20-Task IN-R		10-Task CIFAR		20-Task IN-A		5-Task VTAB	
	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N	$\bar{A}$	A_N
0	84.90	80.65	84.48	79.67	83.55	77.90	92.21	88.12	71.57	61.49	90.92	90.16
2	87.71	83.98	87.20	82.33	85.08	79.58	93.81	90.91	72.04	61.49	93.88	94.29
4	89.72	85.88	88.85	84.70	86.07	80.92	**94.80**	92.26	**72.17**	61.49	**95.06**	**95.49**
6	90.44	86.97	89.42	85.08	**86.53**	**81.25**	94.75	**92.52**	71.69	**61.55**	93.29	93.28
8	90.68	87.03	**89.91**	**85.68**	86.37	80.90	93.35	91.37	70.98	60.76	92.55	93.03
10	90.85	87.15	89.71	85.60	85.68	80.28	91.34	89.64	69.53	59.12	87.79	91.54
12	**91.14**	**87.48**	87.23	83.10	83.46	79.43	87.45	87.69	68.41	58.39	86.20	90.71
14	91.01	87.30	85.22	81.48	82.79	78.65	83.82	85.25	63.93	54.44	87.19	89.44

Generative AI tools were not used for any of the remaining tasks that require disclosure. The research idea, the formulation of SCMNet and of its two modules, the mathematical statements made about them, the design of the architectures, the choice of the training protocol and of the hyperparameters, and the implementation of the method were all produced by the authors, as were the collection and processing of the datasets, the execution of the experiments, and the reading and interpretation of the results. The reported numbers come from the runs described in Appendix A.1, the figures and tables were drawn by the authors from those runs, and the literature discussed in this paper was located and read through a manual survey. All AI-assisted text was reviewed by the authors, who checked that no technical claim, number, citation or figure was introduced, altered or removed by the assistance. We take responsibility for the final content of this work, including the wording produced with the aid of generative AI.

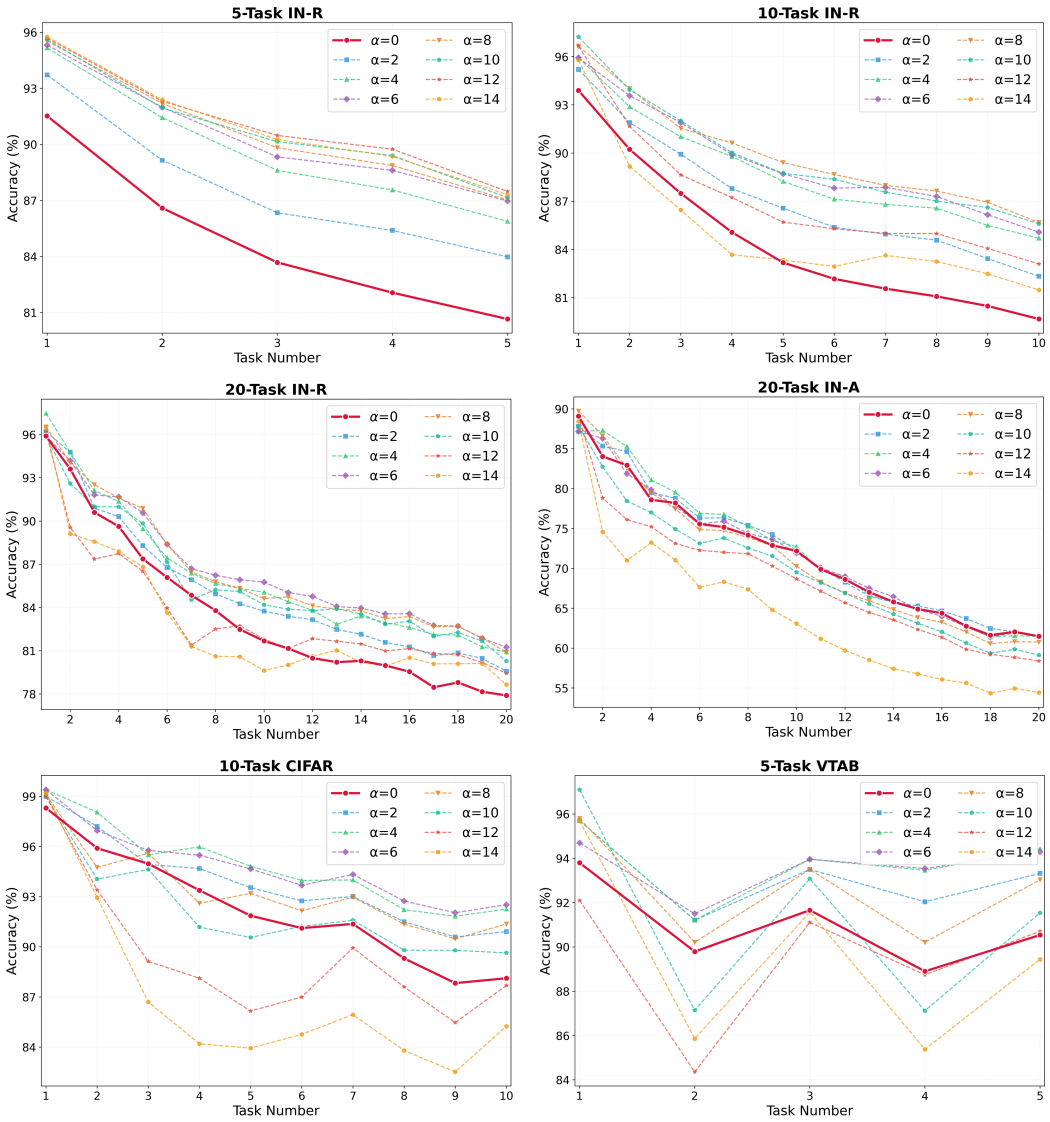

Figure 6: Accuracy versus different DNC residual weights α on six benchmarks.