Representation-Steered Incremental Adapter-Tuning for Class-Incremental Learning with Pre-Trained Models

Supplementary Material

A. Pseudo Code

We summarize the training pipeline of **RSIAT** in Algorithm 1. At the *base task*, we optimize the shared adapter and cosine classifier with jointly minimize the cosine loss and the warm-up representation-steering loss (Lines 6–13) and then record class prototypes (Line 14). For each *incremental task*, we carry over the same adapter and instantiate a RAE projector (Line 17); we then jointly minimize the cosine loss, the alignment loss, and the orthogonal loss to align evolving spaces while protecting old prototypes (Lines 18–26). After training each task, we update old-class via semantic-shift estimation, add prototypes of the new classes, and retrain the unified classifier using synthesized features (Lines 27–29). Unlike adapter-per-task approaches, RSIAT maintains a shared adapter throughout, avoiding parameter growth and module retrieval during inference.

Algorithm 1 RSIAT: Base & Incremental Training

1: **Inputs:** Frozen PTM extractor ϕ, shared adapter A, cosine classifier W; incremental datasets $\{\mathcal{D}_t\}_{t=1}^{T}$ with class sets $\{\mathcal{Y}_t\}_{t=1}^{T}$; residual projector P^t.
2: **Outputs:** Incrementally trained model with frozen PTM extractor ϕ, tuned adapter A^T and classifier W^T.
3: # *Base task*
4: Initialize $\mathcal{M} \leftarrow \emptyset$;
5: Initialize $A^1 \leftarrow A$; expand W^1 for $\mathcal{Y}_1$;
6: **for** epoch $= 1$ to E_1 **do**
7: **for** each batch $\mathcal{B}_1 \subset \mathcal{D}_1$ **do**
8: Compute cosine loss $\mathcal{L}_{\cos}^{(1)}$ using Eq. 3;
9: Compute RS loss $\mathcal{L}_{\mathrm{RS}}$ using Eq. 4;
10: Compute base task training loss $\mathcal{L}^{(1)}$ using Eq. 6;
11: Update (A^1, W^1) by minimizing $\mathcal{L}^{(1)}$;
12: **end for**
13: **end for**
14: Estimate class prototypes $\mathcal{M}_1$ on $\mathcal{D}_1$; update $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_1$;
15: # *Incremental tasks*
16: **for** $t = 2$ to T **do**
17: Initialize $A^t \leftarrow A^{t-1}$; expand W^t for $\mathcal{Y}_t$; freeze copy $\tilde{A} \leftarrow A^{t-1}$; initialize projector P^t;
18: **for** epoch $= 1$ to E_t **do**
19: **for** each mini-batch $\mathcal{B}_t \subset \mathcal{D}^t$ **do**
20: Compute cosine loss $\mathcal{L}_{\cos}^{(t)}$ using Eq. 3;
21: Compute alignment loss using Eq. 8;
22: Compute orthogonal loss $\mathcal{L}_{\mathrm{orth}}^{(t)}$ using Eq. 9;
23: Compute incremental task training loss $\mathcal{L}^{(t)}$ using Eq. 10;
24: Update (A^t, W^t, P^t) by minimizing $\mathcal{L}^{(t)}$;
25: **end for**
26: **end for**
27: Estimate semantic shift for old class prototypes in $\mathcal{M}$ and update $\mathcal{M}$;
28: Estimate new class prototypes $\mathcal{M}_t$ on $\mathcal{D}_t$ and update $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_t$;
29: Synthesize features from $\mathcal{M}$ and retrain W^t with cross-entropy;
30: **end for**
31: **return** the incrementally trained mode;

B. Dataset Configurations

We conducted our experiments on the following six datasets to verify the effectiveness of our method from different perspectives.

CIFAR100 [19]. CIFAR100 contains 60,000 color images at 32×32 resolution, evenly split into 100 classes (600 images per class, 500 train and 100 test). Despite its small image size, the benchmark stresses long-tail generalization and continual updates under low-resolution constraints.

ImageNet-R [12]. ImageNet-R contains 30,000 images across 200 ImageNet classes, collected from non-photographic "rendition" domains such as paintings, cartoons, embroidery, and graphics. It serves as an OOD robustness benchmark: models trained on standard photographs must recognize the same semantics under strong style shift.

ImageNet-A [13]. ImageNet-A is a curated set of natural adversarial examples: ∼7,500 real-world images drawn from 200 ImageNet classes that systematically fool standard classifiers despite being unaltered photographs. It evaluates robustness to naturally occurring, close-to-decision-boundary inputs.

OmniBenchmark [63]. Following prior PTM-based CIL protocols [66], we construct the dataset by sampling 300 classes from OmniBenchmark to form a cross-realm class-incremental sequence. OmniBenchmark is an "omni-vision" benchmark that aggregates 21 realm-wise datasets (over one million images) spanning heterogeneous visual domains and semantics, thereby stressing cross-domain robustness in rehearsal-free settings.

CUB200 [39]. CUB200 is a fine-grained bird classification

dataset with 200 species and 11,788 images, supporting research on fine-grained recognition.

VTAB [59]. Following prior PTM-based CIL protocols [66], we select five VTAB datasets with 10 classes each, yielding 50 classes in total to model cross-domain class-incremental learning. VTAB itself is a diverse benchmark designed to evaluate representations by their ability to adapt to unseen tasks with few examples across Natural, Specialized, and Structured families.

C. Baseline Introduction

We compare our approach against two categories of baselines: PTM-based CIL methods and traditional CIL methods, each encompassing several public and recently proposed approaches. The details of all baselines are provided below.

PTM-based CIL methods leverage the strong generalization capability of large pre-trained models and adapt them using parameter-efficient tuning strategies. **L2P** freezes a pre-trained ViT and learns a key–value prompt pool [44]. Each input retrieves a few prompts for training/inference, enabling instance-specific adaptation across tasks. **DualPrompt** introduces complementary G-Prompt and E-Prompt to encode task-invariant and task-specific instructions, attached to selected transformer layers [43]. Uses a query–matching mechanism to pick the E-Prompt at test time. **CODA-Prompt** decomposes prompts into learnable components and composes instance-specific prompts via attention-weighted aggregation, replacing discrete prompt pools [35]. Keys/queries are trained end-to-end with a frozen ViT, enabling flexible, rehearsal-free continual adaptation. **RanPAC** injects a frozen random-projection layer with nonlinearity between pre-trained features and a prototype-based head. Updates Gram matrix and class prototypes incrementally, using ridge-inverted scores for decorrelation. Trains PETL only on the first session; freezes backbone; rehearsal-free and forgetting-robust [28]. **SLCA** proposes a "Slow Learner" that selectively reduces the representation layer's learning rate to mitigate progressive overfitting when updating pre-trained models [62]. Then performs post-hoc Classifier Alignment by modeling class-wise Gaussian statistics and normalizing logits, recalibrating the head without storing data. **APER+Adapter** adapts the pre-trained model once using adapter-based parameter-efficient tuning to acquire task-specific features, then merges adapted and frozen embeddings [66]. Classifier weights are class prototypes with cosine similarity. Subsequent tasks freeze encoders and update prototypes. **MOS** proposes "model surgery" for PTM-based CIL: trains task-specific adapters and progressively merges them to curb parameter drift; at inference, a training-free self-refined adapter retrieval mechanism corrects mistaken module selection [37]. **EASE** introduces an expandable subspace ensemble for PTM-based CIL: a lightweight adapter is learned per task to form task-specific subspaces while the backbone is frozen [67]. Old-class prototypes are synthesized in new subspaces via semantic mapping, and subspace logits are reweighted at inference for unified prediction. **SSIAT** incrementally tunes a single shared adapter on a frozen ViT, avoiding parameter growth and rehearsal [38]. It estimates semantic shift of older classes via prototype updates, then retrains a unified classifier to align decision boundaries across sessions.

In contrast, traditional CIL baselines rely on a small exemplar memory to replay past knowledge and mitigate forgetting. To ensure a fair comparison, we evaluate all such methods under the same pre-trained backbone as used in our framework. iCaRL performs class-incremental learning by keeping a small exemplar set per class and classifying with a nearest-mean-of-exemplars rule [32]. It updates the feature representation using knowledge distillation while selecting exemplars via herding, enabling joint feature/classifier learning under a fixed memory budget. **DER** freezes previously learned representation and appends a new feature extractor per step to form a "dynamically expandable representation" [47]. An auxiliary loss promotes novel, diverse features, while differentiable channel masks prune redundancy; the classifier is then retrained on a class-balanced subset to mitigate bias. **FOSTER** [40] addresses class-incremental learning via a two-phase "feature boosting and compression" framework: it first expands capacity to enrich representations for novel classes with exemplar replay and distillation, then compresses knowledge back into the base network, preserving past knowledge while enabling efficient adaptation. **MEMO** decomposes the backbone into shared generalized blocks and task-specific specialized blocks [65]. At each increment, it freezes prior specialized blocks and adds a new one, aggregating features for inference. By sharing shallow layers and trading saved capacity for exemplars, MEMO improves memory efficiency with a simple auxiliary classification objective. **TagFex** expands a task-specific model per task while continually learning a separate task-agnostic SSL model [64]. A merge-attention module fuses task-agnostic and task-specific features; knowledge is transferred to the task-specific classifier via KL.

D. Training Configurations

In this section, we summarize the hyperparameters used for training on each dataset. As shown in Tab. D1, we fix the adapter bottleneck dimension to $d_a = 64$ for all datasets, while the remaining hyperparameters are tuned in different datasets. Here, E_{base} and E_{inc} denote the number of training epochs for the base task and each incremental task, respec-

Table D1. Training configurations using *ViT-B/16-IN21K* as the backbone. Dataset-specific hyperparameters are reported, where 'HP' denotes the hyperparameter.

HP	CIFAR	IN-R	IN-A	OmniBench	CUB	VTAB
E_{base}	10	30	20	30	20	30
E_{inc}	30	30	15	10	30	20
lr	0.018	0.007	0.007	0.004	0.038	0.036
B	64	48	48	96	128	48
d_a	64	64	64	64	64	64
d_p	256	384	128	768	768	256
s	40.0	50.0	30.0	50.0	20.0	20.0
m	0.1	0.2	0.3	0.1	0.1	0.3
α	0.5	1.5	1.0	2.0	1.0	2.0
λ_{RS}	0.2	0.5	1.0	0.2	0.2	2.0
E_w	7	4	3	10	9	10
β	1.0	3.0	3.0	3.0	1.5	0.2
γ	0.4	1.5	6.0	2.0	0.8	1.0

tively; lr is the initial learning rate; and B is the batch size. Among the method-level hyperparameters, d_p denotes the residual projector code dimension; s and m are the scale and margin factors used in the cosine loss of Eq. 3, with m acting as a penalty margin to enforce tighter decision boundaries; α is the balancing coefficient in the RS loss of Eq. 4; λ_{RS} and E_w are the target weight and warm-up duration in Eq. 5; and β and γ control the alignment and orthogonality terms in Eq. 10, respectively.

E. More Experiments

We conduct four additional experiments to further examine RSIAT. First, we evaluate multiple CIL incremental settings (large base classes and long task sequences) on five CIL benchmarks to test the robustness of RSIAT under diverse protocols. Second, we report results averaged over multiple random seeds to confirm the stability of our approach. Third, we perform t-SNE visualizations with Recall@1 measurements to investigate how different methods shape and preserve feature geometry over 10 tasks. Fourth, we carry out a more detailed ablation study, reporting per-task Recall@1 when removing components from RSIAT to isolate the contribution of each architectural design.

E.1. Multiple CIL incremental settings

We complement Sec. 5.2 by reporting full numerical results on five datasets under the two multiple CIL incremental settings. Tab. E1 ("large base classes") summarizes the case where each stream starts from a large base task (*e.g.*, CIFAR $B50I10$, IN-R/IN-A $B100I20$, OmniBench $B150I30$, CUB $B100I10$). Across all benchmarks, **RSIAT** achieves the best average accuracy $\bar{A}$, and attains the best final accuracy A_B on four out of five datasets; on CIFAR, it still yields the highest $\bar{A}$ while its A_B is only marginally below MOS (within 0.04% absolute). The margins are especially clear on robustness-oriented datasets (IN-R/IN-A) and the diverse-domain OmniBench, suggesting that shaping a well-structured base representation makes later adaptation easier even when the initial label space is already large.

Tab. E2 ("long sequence tasks") reports results for the complementary setting with zero base and long streams (CIFAR $B0I5$, IN-R/IN-A $B0I10$, OmniBench $B0I15$, CUB $B0I10$). All methods exhibit performance degradation as classes accumulate, but **RSIAT** maintains the highest $\bar{A}$ and A_B on every dataset, particularly on IN-R/IN-A, where long-term prototype drift is most severe. These results confirm that our representation-steering and residual alignment mechanism not only works under the standard protocol, but also scales favorably to scenarios with large base classes and long incremental sequences.

E.2. Stability across Multiple Seeds

To address concerns regarding the stability of CIL methods against variations in task ordering, we repeated our experiments across 3 random seeds. Tab. E3 and Tab. E4 summarize the detailed results on five benchmarks, where we compare RSIAT with several strong recent baselines, including RanPAC, MOS, EASE, and SSIAT. Even on datasets like CIFAR-100 and CUB-200, where performance gaps among top methods are typically narrowest, RSIAT consistently achieves higher average accuracy $\bar{A}$ and final accuracy A_B. Furthermore, our method maintains comparable or even lower standard deviations across all evaluated datasets. These comprehensive multi-seed results confirm the strong robustness and stability of our approach.

E.3. T-SNE for method comparison

To complement the visualizations study in Sec. 5.4, we further plot t-SNE embeddings for all ten tasks on ImageNet-R ($B0I20$) in Fig. E1 (Tasks 1–5) and Fig. E2 (Tasks 6–10). In each figure, columns correspond to MOS, RanPAC, SSIAT, and RSIAT (ours), while rows correspond to different tasks. In addition to the visual patterns, we quantitatively assess cluster quality using the nearest-neighbor metric $\mathrm{Recall@1}$. For each feature vector in the original representation space (before t-SNE), we find its nearest neighbor under cosine similarity and check whether the two samples share the same label; $\mathrm{Recall@1}$ is then computed as the percentage of features whose nearest neighbor belongs to the same class. In other words, higher $\mathrm{Recall@1}$ indicates tighter within-class aggregation and clearer between-class separation in the learned feature space. Across the entire sequence, MOS quickly develops highly entangled and overlapping clusters, and its $\mathrm{Recall@1}$ drops steadily as classes accumulate. RanPAC maintains relatively compact

Table E1. Large base classes results on five CIL datasets with **ViT-B/16-IN21K** as the backbone. We report average accuracy $\bar{A}$ over tasks and final accuracy A_B. All methods are implemented without using exemplars. Best performance is shown in **bold**, and second-best performance is shown in underline.

Method	CIFAR $B50I10$		IN-R $B100I20$		IN-A $B100I20$		OmniBench $B150I30$		CUB $B100I10$	
	$\bar{A}$	A_B	$\bar{A}$	A_B	$\bar{A}$	A_B	$\bar{A}$	A_B	$\bar{A}$	A_B
L2P [44]	86.87	80.43	73.27	66.70	50.96	46.28	69.07	58.78	73.02	58.99
DualPrompt [43]	86.11	81.02	69.92	64.88	56.14	50.03	70.18	61.95	78.98	70.40
CODA-Prompt [35]	90.09	85.22	78.00	74.18	61.76	56.16	74.56	66.87	81.62	73.83
RanPAC [28]	94.13	92.49	82.28	80.15	68.88	64.78	83.25	80.63	91.57	90.88
SLCA [62]	93.29	91.64	82.65	80.55	61.49	56.68	81.18	74.90	89.42	87.15
APER+Adapter [66]	91.90	89.66	74.13	70.78	61.64	56.48	77.74	75.52	89.72	87.62
MOS [37]	94.15	**93.00**	81.16	79.43	67.26	63.40	83.44	80.80	91.39	90.54
EASE [67]	90.46	87.47	77.16	74.22	63.67	57.14	78.33	76.19	89.02	86.17
SSIAT [38]	93.92	92.35	83.38	80.73	68.95	62.34	82.15	78.81	92.15	90.29
RSIAT (Ours)	**94.28**	92.96	**86.19**	**83.97**	**71.42**	**65.96**	**84.05**	**81.75**	**92.52**	**91.48**

Table E2. Long sequence tasks results on five CIL datasets with **ViT-B/16-IN21K** as the backbone. We report average accuracy $\bar{A}$ over tasks and final accuracy A_B. All methods are implemented without using exemplars. Best performance is shown in **bold**, and second-best performance is shown in underline.

Method	CIFAR $B0I5$		IN-R $B0I10$		IN-A $B0I10$		OmniBench $B0I15$		CUB $B0I10$	
	$\bar{A}$	A_B	$\bar{A}$	A_B	$\bar{A}$	A_B	$\bar{A}$	A_B	$\bar{A}$	A_B
L2P [44]	84.18	77.73	75.91	70.18	50.00	37.26	70.75	60.74	67.05	56.25
DualPrompt [43]	84.87	77.41	72.55	66.02	52.06	40.62	72.42	61.22	77.47	66.54
CODA-Prompt [35]	88.06	80.70	78.63	72.05	47.77	36.41	74.74	64.71	84.00	73.37
RanPAC [28]	93.90	90.04	82.69	77.27	63.73	53.46	86.08	78.80	93.13	89.40
SLCA [62]	93.43	88.93	82.33	76.67	62.37	50.56	80.00	69.37	89.51	82.19
APER+Adapter [66]	90.55	84.97	74.15	66.00	60.66	48.98	80.31	73.57	92.21	86.73
MOS [37]	93.46	89.34	81.12	75.50	64.15	53.32	85.89	79.88	93.49	90.12
EASE [67]	91.56	85.83	81.17	74.87	60.26	47.53	80.83	73.87	92.23	86.81
SSIAT [38]	94.27	90.67	82.95	78.47	71.26	61.09	83.70	75.86	93.38	89.53
RSIAT (Ours)	**94.66**	**91.24**	**85.57**	**80.65**	**72.87**	**62.80**	**86.35**	**80.33**	**93.99**	**90.59**

Table E3. Multiple Seeds results on CIFAR, ImageNet-R (IN-R), and ImageNet-A (IN-A) datasets. We report average accuracy $\bar{A}$ over tasks and final accuracy A_B. Best performance is shown in **bold**, and second-best performance is shown in underline.

Method	CIFAR $B0I10$		IN-R $B0I20$		IN-A $B0I20$	
	$\bar{A}$	A_B	$\bar{A}$	A_B	$\bar{A}$	A_B
RanPAC [28]	94.14±0.58	91.33±0.12	82.84±0.26	77.97±0.18	67.01±2.15	58.92±0.54
MOS [37]	93.96±0.82	91.29±0.20	82.03±0.38	77.55±0.28	67.54±1.66	59.71±0.25
EASE [67]	91.85±0.60	88.20±0.49	81.21±0.35	75.76±0.36	63.83±0.67	54.31±0.62
SSIAT [38]	94.16±0.66	91.50±0.14	83.81±0.35	79.65±0.07	70.81±1.30	61.95±0.28
RSIAT	**94.47**±0.67	**92.06**±0.11	**86.63**±0.21	**82.69**±0.20	**72.85**±1.45	**65.33**±1.05

clusters at early tasks but gradually exhibits increasing fragmentation and overlap between classes. SSIAT preserves tighter groupings than the former two methods, yet still exhibits noticeable drift of old-class manifolds in later tasks. In contrast, RSIAT consistently produces the most compact and well-separated clusters from Task 1 to Task 10, with Recall@1 decreasing more slowly and remaining highest at each stage. These full-sequence visualizations corroborate

Table E4. Multiple Seeds results on OmniBenchmark and CUB datasets. We report average accuracy $\bar{A}$ over tasks and final accuracy A_B. Best performance is shown in **bold**, and second-best performance is shown in underline.

Method	OmniBench $B0I30$		CUB $B0I10$	
	$\bar{A}$	A_B	$\bar{A}$	A_B
RanPAC [28]	85.64±0.45	79.43±0.29	92.93±0.49	89.39±0.12
MOS [37]	85.59±0.69	80.06±0.04	93.11±0.53	89.83±0.11
EASE [67]	80.55±1.27	74.42±0.34	91.35±1.25	86.62±0.08
SSIAT [38]	84.02±0.98	77.71±0.07	92.83±0.55	88.87±0.47
RSIAT	**85.92**±0.68	**80.98**±0.30	**93.42**±0.61	**90.11**±0.54

Table E5. Task-wise Recall@1 on ImageNet-R $B0I20$ for different settings removing components from RSIAT.

Method	Task										Avg ↑
	1	2	3	4	5	6	7	8	9	10	
Baseline	89.1	84.2	81.1	78.4	74.7	74.6	74.5	73.2	71.9	72.3	77.40
+ IRS	89.1	86.4	83.6	81.6	78.2	76.7	76.1	75.7	75.1	74.3	79.68
+ BRS	94.5	85.1	82.5	79.3	75.9	74.7	74.4	74.0	72.6	72.6	78.56
+ IRS (wo_res)	94.5	82.7	81.2	79.0	75.5	74.8	74.1	73.8	72.1	72.3	78.00
+ IRS + $\mathcal{L}_{\text{RS}}$	91.7	86.3	84.0	81.8	78.9	77.5	76.9	76.2	75.7	75.0	80.40
RSIAT	94.5	87.4	84.3	82.7	79.1	78.2	77.3	77.0	76.0	75.3	81.18

our claim that representation steering plus residual alignment effectively reduces prototype drift and preserves a stable, discriminative geometry throughout CIL streams.

E.4. More ablation study

To further extend the ablation study in Sec. 5.3, Tab. E5 report task-wise Recall@1 on ImageNet-R ($B0I20$) for different settings removing components from RSIAT. Here, Recall@1 measures the percentage of features whose nearest neighbour (by cosine similarity in the original feature space) belongs to the same class, thus reflecting how well class manifolds remain compact and separated at each stage. Across all ten tasks, the full model RSIAT attains the highest average Recall@1 (81.18%), indicating the most coherent feature geometry over time. The Baseline variant, which only trains the cosine head and shared adapter, shows consistently lower values (77.40%), confirming that representation steering and projector are both necessary to preserve neighbourhood structure. Removing Base Representation Steering (+ IRS) mainly harms early tasks, whereas discarding Incremental Representation Steering (+ BRS) causes faster degradation on later tasks. Eliminating the projector residual path (+ IRS (wo_res)) or the warm-up schedule (+ IRS + $\mathcal{L}_{RS}$) also reduces Recall@1 across stages. Overall, any component removal yields worse per-task neighbourhood purity, reinforcing that all parts of RSIAT contribute to stable, well-structured representations.

F. Discussion

F.1. Relation to EWC

While regularization-based methods like Elastic Weight Consolidation (EWC)[18] aim to preserve historical knowledge, RSIAT operates on a fundamentally different level. Specifically, EWC regularizes network weights to directly prevent parameter drift. In contrast, our proposed orthogonal representation steering operates strictly at the feature level. It regularizes the representation space through geometric constraints between newly extracted features and old stored prototypes. This design maintains spatial separation and structural alignment without imposing rigid constraints on the shared adapter's parameter updates, thus preserving higher plasticity for learning new tasks.

F.2. Clarification on Feature Collapse

A potential concern in incremental learning is whether aligning new features to an evolving space might induce feature collapse for previously learned classes. In RSIAT, the residual autoencoder (projector) is intentionally re-initialized to zero before learning each new task, ensuring it specifically models the incremental drift of the current stage starting from an exact identity mapping. More importantly, this projector serves exclusively as a temporary training aid for space alignment and is entirely discarded during the inference phase. Because the projector does not participate in inference, this mechanism fundamentally prevents feature

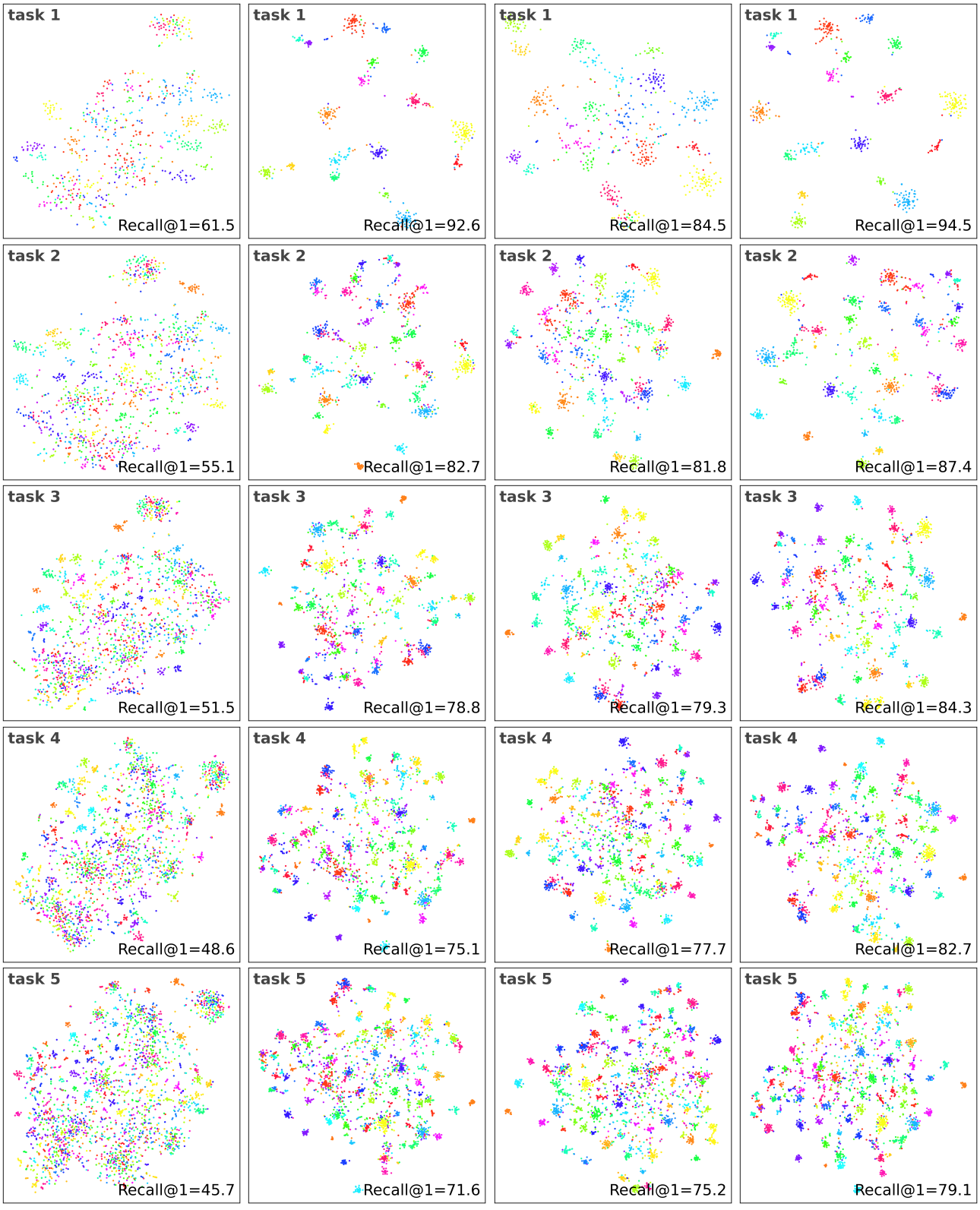

Figure E1. t-SNE embeddings for Tasks 1–5 across methods. Columns correspond to MOS, RanPAC, SSIAT, and RSIAT (ours); rows correspond to different tasks. Numbers in each panel denote Recall@1 measured in the original feature space before t-SNE. RSIAT consistently forms compact clusters and preserves separation as tasks accumulate.

Figure E2. t-SNE embeddings for Tasks 6–10 across methods. Columns correspond to MOS, RanPAC, SSIAT, and RSIAT (ours); rows correspond to different tasks. Numbers in each panel denote Recall@1 measured in the original feature space before t-SNE. RSIAT consistently forms compact clusters and preserves separation as tasks accumulate.

collapse for past classes.

F.3. Why the Baseline is Highly Competitive

As observed in our experiments, our baseline (built upon SSIAT[38]) already demonstrates strong competitiveness and frequently outperforms many existing state-of-the-art methods. This superior performance stems from two primary factors. First, incrementally tuning a single shared adapter avoids the instance-level routing errors commonly seen in multi-prompt or multi-adapter methods. Second, the baseline incorporates a semantic shift compensation mechanism that estimates and updates the distribution of old classes without requiring explicit data replay. This provides a remarkably robust foundation for the unified classifier, which our RSIAT framework further enhances through explicit geometric representation steering.

F.4. Limitations

A key requirement of an ideal CIL system is preventing parameter bloat during inference. RSIAT strictly maintains a constant inference-time model size and computational parameter count, as inference relies solely on the frozen backbone, the shared adapter, and the classifier. However, our method does rely on storing class-wise covariance matrices as frozen statistics to facilitate incremental learning without raw data replay. We acknowledge that this introduces an additional training storage footprint (*e.g.*, approximately 60MB for 100 classes). It is crucial to emphasize that these buffers are strictly used offline for prototype updating and are completely detached from the inference forward pass. In future work, we plan to explore more storage-efficient prototype preservation strategies or lighter generative mechanisms to further compress this training-stage memory overhead while maintaining high performance.