PLEAT: PROGRESSIVE LORA EXPERT AGGREGATION AND TRANSPORT FOR CONTINUAL MODEL MERGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning aims to acquire knowledge from new tasks while retaining previously learned capabilities. One way to pursue this goal is through model merging, which combines the weights of independently trained experts without additional training. A central challenge in continual model merging is to integrate new experts without retaining an expanding expert pool. In resource-constrained settings, merging experts trained with low-rank adaptation (LoRA) must also preserve their compact representation. Existing merging strategies face three limitations in this setting: (1) joint merging requires retaining and revisiting an expanding collection of historical experts; (2) methods designed for fully fine-tuned models typically operate on dense updates, giving up LoRA's compact representation when applied directly to LoRA experts; and (3) fixed-rank LoRA merging preserves compactness but can become a capacity bottleneck as knowledge accumulates over long task sequences. We propose **PLEAT**, **P**rogressive **L**oRA **E**xpert **A**ggregation and **T**ransport, a training-free approach that exploits the low-rank structure of LoRA experts to maintain a factorized merged update with progressively adjustable capacity. PLEAT incrementally updates a shared input basis under a prescribed rank schedule, aligns each incoming expert to the updated representation, and transports historical coefficients before aggregation. Experiments across multiple backbone models show that PLEAT outperforms a range of existing merging methods, while retaining the parameter-efficient representation of LoRA and enabling continual merging without revisiting historical expert weights.

1 INTRODUCTION

Large pretrained models exhibit strong general capabilities, but often fall short on downstream tasks that require specialized knowledge. Fine-tuning on task-specific data offers an effective way to improve their performance, producing specialized experts from a shared base model. However, storing and deploying a separate expert for each task becomes increasingly costly as the number of tasks grows. Model merging offers an alternative by combining the parameters or parameter updates of these experts into a single model that aims to retain their respective capabilities (Wortsman et al., 2022; Ilharco et al., 2023). Knowledge from separate fine-tuning runs can then be consolidated in one deployed model, without maintaining a separate inference model for every task.

Existing methods explore different ways to combine expert updates. Task Arithmetic (Ilharco et al., 2023) adds task vectors to the base model, while TIES (Yadav et al., 2023) and DARE (Yu et al., 2024) modify these updates before aggregation to reduce interference. However, new experts continue to emerge as models are adapted to additional tasks. Reapplying joint merging to an expanding expert pool requires retaining and revisiting historical expert weights, increasing storage and computation costs. Continual merging addresses this problem by incorporating experts sequentially and maintaining a state that summarizes earlier updates (Tang et al., 2025; Yang et al., 2025). Extending this capability to parameter-efficient experts is particularly valuable when many task-specific models must be maintained.

Low-rank adaptation (LoRA) represents each expert update with two small matrices, reducing the parameters needed to store task-specific adaptations (Hu et al., 2022). Preserving this compact form is therefore an important objective when merging LoRA experts continually. Compactness matters throughout the sequence: retaining all past adapters would recreate the growing expert pool that

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.

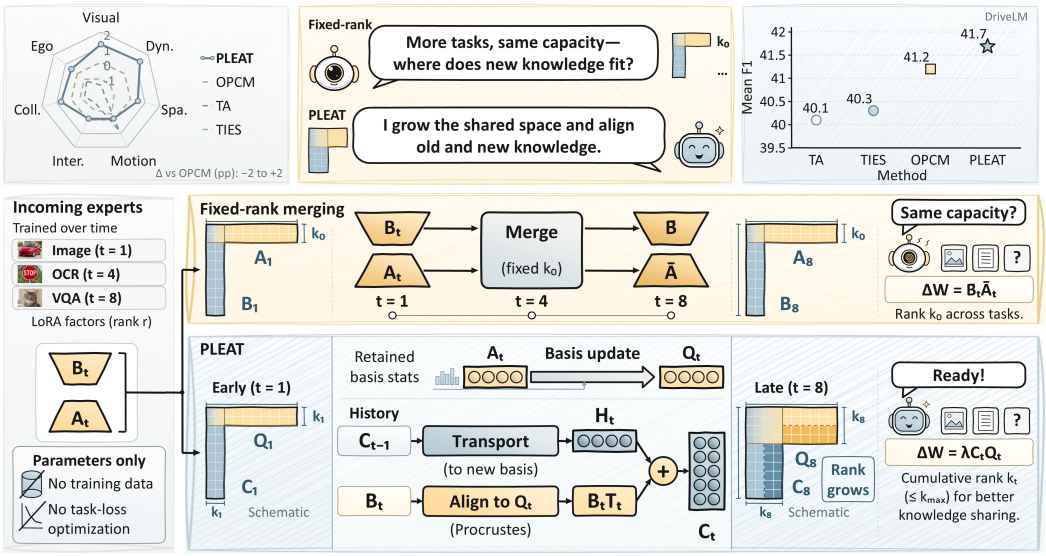


Figure 1. **Motivation and effect overview.** A ***fixed representation*** and a ***growing shared basis*** receive the same arriving experts. PLEAT aligns incoming factors and transports history with Procrustes maps in the complete current basis. DriveLM radar: per-family Token-F1 differences from OPCM, with a common radial range of -2 to $+2$ percentage points. The right panel compares mean Token-F1 for the same methods on a 39.5–42.0 axis.

continual merging aims to avoid. Yet methods designed around full parameter updates do not directly exploit this factorized structure. Expanding LoRA factors into dense updates for merging sacrifices their storage advantage. Methods such as KnOTS and LoRA-LEGO explore how to coordinate LoRA expert representations (Stoica et al., 2025; Zhao et al., 2025), but continual accumulation also raises a capacity question. Keeping the merged update at a fixed rank preserves compactness, yet leaves a limited subspace to accommodate increasingly diverse task updates. Even when each expert has a small rank, different tasks can contribute directions outside the span retained from earlier updates. Continual LoRA merging therefore requires both a compact representation and room for its capacity to grow, without revisiting historical experts.

An evolving representation also changes the coordinates of accumulated knowledge. Aligning only the incoming expert leaves stored coefficients attached to the previous basis, even when their dimensions remain compatible. Because these coefficients encode combinations of basis directions, historical coordinates must evolve with the shared basis to account for its changing directions.

We introduce PLEAT, Progressive LoRA Expert Aggregation and Transport, to couple basis updates with historical-state updates. PLEAT maintains a shared input basis through incremental SVD, aligns both coefficient branches with Procrustes maps, and adjusts capacity under a prescribed rank schedule. It transports historical coefficients before aggregation, retaining LoRA factors and incorporating each expert once without revisiting historical weights.

An evolving basis requires updating historical coordinates. *Rank growth controls the available capacity. Historical transport controls how accumulated coefficients use the updated basis. A basis can rotate even at fixed rank, so matching capacity alone does not ensure consistent historical coordinates.*

Our contributions are as follows:

❶ **Historical Coordinate Consistency.** We identify the mismatch between an evolving shared basis and historical coefficients that remain in its previous coordinates.

❷ **Progressive Expert Aggregation and Transport.** We develop PLEAT, combining progressive basis updates, incoming expert alignment, and historical coefficient transport in a factorized merging state.

❸ **Evaluation across Backbones and Task Sequences.** We evaluate PLEAT across multiple backbone models and task sequences, examining merging performance, storage costs, and the effects of rank growth and historical transport.

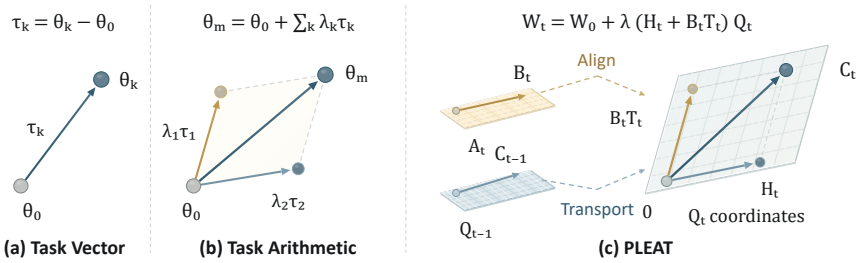


Figure 2. Weight-space addition and shared-coordinate aggregation. PLEAT transports history $H_t = C_{t-1}R_t$ and adds incoming coefficients B_tT_t before export; Section 3 defines both maps.

2 Related Work

Model fusion spans parameters, representations, and behaviors (Yang et al., 2026a; Cai et al., 2026). Studies extend merging to multimodal systems (Zhou et al., 2024; Zeng et al., 2025; Huang et al., 2026b) and quantify how model size and expert count affect performance (Yadav et al., 2025; Wang et al., 2026); MergeKit supports practical parameter operations (Goddard et al., 2024).

2.1 Model Merging

Model merging combines expert updates. Model soups average checkpoints; Task Arithmetic adds task deltas (Wortsman et al., 2022; Ilharco et al., 2023). TIES and DARE control signs and sparsity (Yadav et al., 2023; Yu et al., 2024); localization methods select coordinates (He et al., 2025; Wang et al., 2024; Chopra et al., 2025). Fisher weighting and RegMean use importance or response statistics (Matena & Raffel, 2022; Jin et al., 2023), while PCB balances parameter competition (Du et al., 2024). EMR uses task-specific masks and rescalers; FREE-Merging combines frequency filtering with lightweight experts (Huang et al., 2024; Zheng & Wang, 2025).

Spectral methods align task subspaces (Gargiulo et al., 2025; Marczak et al., 2025; Panariello et al., 2025), and directional methods align parameters and features (Chen et al., 2025). WUDI and OptMerge optimize expert-derived objectives; AdaMerging learns coefficients from unlabeled data (Cheng et al., 2025; Wei et al., 2026; Yang et al., 2024). Expert Merging aligns hidden states and predictions (Zhang et al., 2026); FRISM learns subspace weights for reasoning transfer (Huang et al., 2026a). Activation-based methods and FeatCal calibrate merged representations (Nobari et al., 2025; Yao et al., 2025; Gu et al., 2026). Merge-aware training improves compatibility (Tang et al., 2024; Porrello et al., 2026; Jin et al., 2025; Lee et al., 2025; Yang et al., 2026b).

2.2 LoRA and Continual Model Merging

LoRA factorizes expert updates (Hu et al., 2022). Adapter merging coordinates these factors and their ranks. KnOTS, LoRA-LEGO, and orthogonal-subspace methods coordinate adapter representations (Stoica et al., 2025; Zhao et al., 2025; Zhang & Zhou, 2025). DO-Merging separates magnitude and direction (Zheng et al., 2025), TC-LoRA combines clustering and tensor factorization (Su et al., 2025a). DiDi-Merging allocates rank budgets (Du & Lin, 2026); Poly-PRAG learns LoRA routing for retrieval (Su et al., 2025b).

Continual merging retains knowledge across expert arrivals. OPCM and dual-projection merging maintain historical state (Tang et al., 2025; Yang et al., 2025), while K-Merge retains adapter slots (Shenaj et al., 2026). Merge Before Forget couples learning to an evolving LoRA (Qiao & Mahdavi, 2026). PLEAT grows a shared basis and transports accumulated coefficients before incorporating incoming experts.

3 Method

PLEAT accumulates output coefficients in a shared input basis. Each arrival updates the basis, aligns the incoming expert, and transports historical coefficients before addition. Rank selection determines

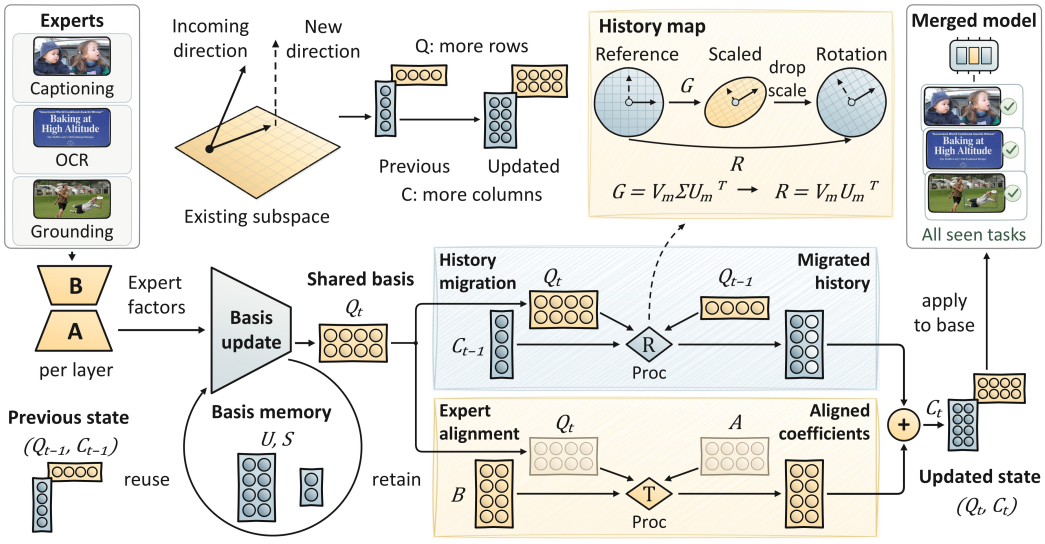


Figure 3. PLEAT updates a shared input basis. **Proc** denotes Procrustes alignment: R transports historical coefficients and T aligns the incoming expert to the complete updated basis, including new directions. The inset shows how the map retains orientation while removing singular-value scaling. Task examples depict intended coverage.

the exported basis and the decomposition state retained for later arrivals. Figure 7 in Appendix A summarizes the update sequence.

3.1 Preliminaries

A LoRA expert supplies factors $A_i \in \mathbb{R}^{r_i \times d_{\text{in}}}$ and $B_i \in \mathbb{R}^{d_{\text{out}} \times r_i}$ (Hu et al., 2022). Its effective update $\Delta W_i = s_i B_i A_i$ includes a loader multiplier s_i. PLEAT absorbs s_i into B_i on arrival and processes each factor pair once. Each adapted module maintains its own state, without training data, gradients, or repeated access to historical checkpoints.

Paired factors. For equal-rank experts with equal effective multipliers, independent averaging gives

$$\left(\frac{1}{t}\sum_i B_i\right)\left(\frac{1}{t}\sum_j A_j\right) = \frac{1}{t^2}\sum_i B_i A_i + \frac{1}{t^2}\sum_{i\neq j} B_i A_j. \tag{1}$$

Off-diagonal terms pair factors from different experts. Rescaling corrects the paired coefficient but cannot generally remove these cross terms. Independent factor merging therefore defines a different operator (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024).

Changing coordinates. An invertible rank-space matrix G_i preserves an expert update through

$$(B_i G_i)(G_i^{-1} A_i) = B_i A_i. \tag{2}$$

Independent factor averages generally depend on this choice. Shared-coordinate methods make alignment explicit (Stoica et al., 2025; Zhao et al., 2025); a continual merger must also transport accumulated coefficients whenever its basis changes. PLEAT uses Procrustes maps to align both branches. These maps align the supplied factor coordinates under orthogonality constraints. Canonicalization, permutation alignment, and cross-base rebasing remain distinct constructions (Han et al., 2026; Ainsworth et al., 2023; Rinaldi et al., 2025).

3.2 PLEAT

Shared state. At arrival t, a module stores Q_{t-1}, accumulated coefficients C_{t-1}, and retained singular values Σ_{t-1}. With $U_{t-1} = Q_{t-1}^\top$, an incremental SVD of $[U_{t-1}\Sigma_{t-1}, A_t^\top]$ yields candidate left singular vectors U_t and

$$Q_t = U_t[:, 1{:}k_t]^\top, \qquad Q_t Q_t^\top = I_{k_t}. \tag{3}$$

Both coefficient branches use the complete current basis Q_t (Figure 3); Algorithm 1 specifies the update order.

Incoming alignment. For $X \in \mathbb{R}^{m \times d_{\text{in}}}$ and a row-orthonormal $Q \in \mathbb{R}^{n \times d_{\text{in}}}$, let $XQ^\top = L\Sigma V^\top$ be a thin SVD with $p = \min(m, n)$ columns. We define

$$\text{Proc}(X, Q) = LV^\top, \qquad T_t = \text{Proc}(A_t, Q_t). \tag{4}$$

An identically zero A_t contributes no update and leaves the state unchanged. For nonzero inputs, all p singular-vector columns are retained, including an orthonormal completion for zero singular values. The map minimizes $\|X - TQ\|_F$ subject to $TT^\top = I_m$ when $m \le n$, or $T^\top T = I_n$ when $m > n$. PLEAT adds $B_t T_t$, with reconstruction residual bounded by

$$\|B_t A_t - B_t T_t Q_t\|_F \le \|B_t\|_2 \, \|A_t - T_t Q_t\|_F. \tag{5}$$

Incoming reconstruction is exact when A_t has orthonormal rows contained in the span of Q_t; Equation 4 defines the map for general factors. Appendix A gives the constrained optimum.

Historical transport. PLEAT maps stored coefficients into the complete updated basis before adding the arriving expert:

$$\begin{aligned} R_t &= \text{Proc}(Q_{t-1}, Q_t), \\ C_t &= C_{t-1} R_t + B_t T_t, \qquad C_1 = B_1 T_1. \end{aligned} \tag{6}$$

With nondecreasing rank, $R_t R_t^\top = I$. If the new span contains the old span, then $R_t Q_t = Q_{t-1}$ and historical transport preserves the represented update exactly. Otherwise its residual is $C_{t-1}(Q_{t-1} - R_t Q_t)$. The full-basis map can use new directions immediately, without appending zero columns after alignment. Neither branch reloads historical factors.

Transport objective. Procrustes preserves $\|C_{t-1} R_t\|_F = \|C_{t-1}\|_F$ under nondecreasing rank. Least squares uses $R_t^{\text{LS}} = Q_{t-1} Q_t^\top$, minimizing reconstruction error while potentially contracting historical coefficients. Both recover history under span containment. Procrustes thus maintains historical coefficient magnitude as the shared basis evolves. Appendix B derives the reconstruction trade-off and accumulated error.

Export scale. The deployed update is

$$\widehat{\Delta W}_{1:t} = \lambda C_t Q_t. \tag{7}$$

We use $\lambda = 1/t$ for sum-trained adapters and retain the loader multiplier inside each incoming coefficient. A validation-only scale is used when expert training follows a different normalization. Non-LoRA parameters use the uniform running mean specified in Appendix A.

3.3 Rank Selection and Retained State

Energy-based rank. For a nonzero updated retained spectrum with singular values $\sigma_{t,j}$, threshold τ selects

$$k_t^E = \min\left\{ k : \frac{\sum_{j=1}^{k} \sigma_{t,j}^2}{\sum_j \sigma_{t,j}^2} \ge \tau \right\}, \qquad k_t = \min\{k_{\max}, n_t, \max(k_{t-1}, k_t^E)\}. \tag{8}$$

Here $n_t = \min(d_{\text{in}}, k_{t-1} + r_t)$ bounds the candidate dimension. Starting from $k_0 = 0$, rank is nondecreasing within the cap. The energy ratio describes the updated retained input statistics, not all historical factors or task loss. A zero outside-span residual permits skipping residual QR, but basis rotation and transport still apply.

Fixed rank schedule. We choose the energy threshold and cap on a held-out validation split, then freeze them for every method, seed, order, and test milestone. The CLIP schedule requests ranks 64, 128, and 256 after 8, 14, and 20 experts; all other suites use the same energy rule in Equation 8. A scheduled rank remains limited by available directions; twenty rank-8 experts supply at most 160 directions per module.

Retained state and cost. Exported factors contain $k_t(d_{\text{in}} + d_{\text{out}})$ entries per module. PLEAT retains Q_t, C_t, and k_t singular values, requiring $O(k_t(d_{\text{in}} + d_{\text{out}}))$ memory. A rank-r_t arrival costs $O((k_{t-1} + r_t)^2 d_{\text{in}} + d_{\text{out}}(k_{t-1} + r_t)k_t)$ time for the block update, including historical transport and incoming alignment. The algorithm never stores prior experts and does not materialize a dense $d_{\text{out}} \times d_{\text{in}}$ update during LoRA merging.

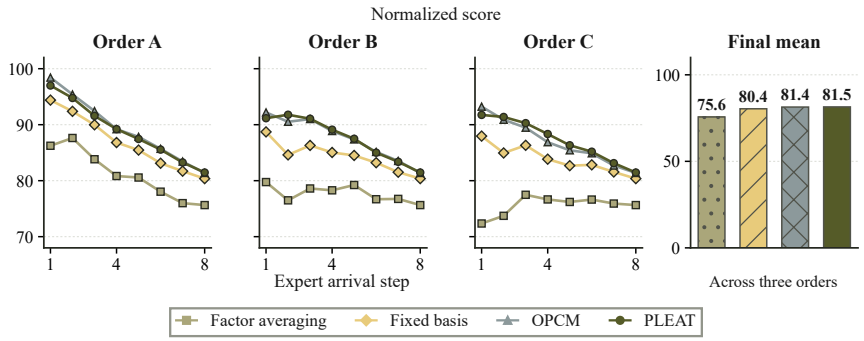


Figure 4. Arrival-order trajectories and final scores. Lines report the mean over tasks seen at each arrival. Bars show the final score averaged across the three orders. Each task score is divided by its single-expert reference and multiplied by 100, distinct from the base-to-expert normalization in the Norm. column of Table 2.

Suite	Task unit	Reporting contract
LLaVA-8	Eight named tasks	Native task metrics at prefix 8
DriveLM-7	Seven question families	Per-family Token-F1 and macro mean
CLIP-20	Twenty classification tasks	Strict seen accuracy at 8, 14, and 20 arrivals
Scene32	Thirty-two scene tasks	Seen and held-out Token-F1, with backward transfer

Table 1. **Dataset roles and reporting units.** The suites retain separate metrics and selection protocols.

4 Experiment

4.1 Experimental Setup

Tasks and metrics. We evaluate eight LLaVA tasks (Liu et al., 2023; 2024; Zeng et al., 2025), seven DriveLM question families (Sima et al., 2024; Caesar et al., 2020), and twenty CLIP datasets on ViT-B/32, ViT-B/16, and ViT-L/14 (Radford et al., 2021; Dosovitskiy et al., 2021). Scene32 repartitions the nuScenes-based DriveLM-7 data into 32 scene-level tasks to test longer sequences. LLaVA reports native metrics and a clipped base-to-expert normalized mean; DriveLM reports macro Token-F1 and CLIP mean seen-task accuracy. Appendix C defines normalization, scorers, and task membership.

Matched protocol. All methods receive identical experts, arrival orders, exported-rank caps, and non-LoRA parameters. We select one configuration per suite on a 10% validation split and freeze it before testing. Each tuned method receives at most 24 configurations. Results combine three training seeds and three orders. Baselines include averaging, task-vector methods, and continual or LoRA-specific mergers (Wortsman et al., 2022; Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024; Tang et al., 2025; Stoica et al., 2025; Yang et al., 2025; Shenaj et al., 2026). Dense updates are compressed to the common export rank; retained state is counted separately. Appendix C details the search and controls.

4.2 Main Results

Table 2 reports a task-normalized mean of 63.8 and an expert-relative score of 81.5 for PLEAT. DARE reaches a normalized mean of 62.7; task-wise leaders vary across the eight tasks. PLEAT exceeds its no-transport variant by 2.2 points on the reported normalized mean, providing a comparison of the two recurrences under the reported configuration.

PLEAT reaches 41.7 macro Token-F1 in Table 3, 1.9 points above uniform averaging and 0.5 above OPCM, the strongest matched baseline. Relative to OPCM, PLEAT improves visual, dynamic-object, spatial, collision, and ego questions, while scoring 0.1 points lower on motion and interaction questions. The same frozen scale and rank rule are used in every order.

Table 2. **LLaVA-8 results under the matched protocol.** Values are the reported task scores; Norm. maps each task's base to 0 and expert to 100 before clipping and averaging. Corresponding aggregates appear in Appendix D. Rel.% is 100 times the mean of the eight task-wise method/expert score ratios, without clipping.

	QA / classification (5 tasks)					Captioning		Ground.	8 tasks	
Method	**SQA**	**IN**	**IQA**	**OCR**	**VQA**	**Flickr**	**VizWiz**	**REC**	**Norm.**	**Rel.%**
Factor-wise and dense merging methods										
Uniform	63.4	62.4	38.6	64.5	67.8	50.3	45.5	67.9	56.6	78.1
Task Arith.	77.0	72.6	49.4	58.9	55.3	54.6	46.0	61.1	60.1	80.5
TIES	69.3	68.8	47.5	62.5	66.6	53.7	46.3	63.6	62.9	81.1
DARE	74.0	70.2	47.3	60.2	64.4	52.6	45.7	62.9	62.7	80.9
LoRA-aware merging methods										
Paired mean	58.5	58.9	32.7	65.2	68.0	48.5	44.8	69.7	51.4	75.6
KnOTS	73.3	69.3	45.6	59.9	65.0	52.5	45.6	62.9	61.5	80.3
Continual and stateful merging methods										
DOP	73.7	69.4	45.7	59.1	64.5	52.6	45.6	62.6	61.1	80.2
K-Merge	73.4	69.4	45.7	59.9	65.0	52.6	45.7	63.0	61.8	80.4
Online SVD	73.3	69.0	45.3	58.7	64.1	52.2	45.2	62.2	60.0	79.6
PLEAT w/o transport	73.4	69.3	45.6	59.9	65.0	52.6	45.7	62.9	61.6	80.4
PLEAT (Ours)	**74.7**	**70.3**	**46.6**	**60.1**	**65.4**	**53.5**	**46.5**	**63.5**	**63.8**	**81.5**

Table 3. **DriveLM results on seven question families.** Values are means over three training seeds and three arrival orders under equal validation budgets. Rel.% is 100 times the mean of the seven family-wise method/expert score ratios, without clipping.

	DriveLM (7 tasks), Token-F1							7 tasks	
Method	**Visual**	**Dyn.**	**Spatial**	**Motion**	**Inter.**	**Coll.**	**Ego**	**Mean**	**Rel.%**
Factor-wise and dense merging methods									
Uniform	11.8	35.1	54.5	38.1	31.5	81.4	26.6	39.8	69.6
TIES	10.8	33.4	53.9	41.1	34.0	81.5	27.8	40.3	70.1
Task Arith.	10.4	32.8	53.2	42.3	33.6	81.2	27.6	40.1	69.6
DARE	10.8	33.4	53.9	40.9	34.0	81.3	27.2	40.2	69.8
LoRA-aware merging methods									
Paired mean	11.1	33.7	54.2	41.0	34.1	81.4	27.6	40.4	70.4
KnOTS	11.7	34.4	54.7	41.2	34.5	81.8	28.1	40.9	71.5
Continual and stateful merging methods									
OPCM	11.9	34.6	55.0	41.5	34.8	82.1	28.4	41.2	72.1
DOP	11.6	34.5	54.5	41.1	34.4	81.5	27.2	40.7	71.1
K-Merge	11.9	34.9	54.8	41.4	34.7	81.8	27.4	41.0	71.7
Online SVD	10.9	33.8	53.9	40.5	33.8	80.9	26.8	40.1	69.7
PLEAT (Ours)	**13.1**	**35.9**	**55.5**	**41.4**	**34.7**	**82.8**	**28.9**	**41.7**	**73.7**

Table 4 compares fixed settings at 8, 14, and 20 tasks. PLEAT reaches the highest cross-backbone mean at twenty tasks. At twenty tasks it averages 59.9 across backbones, 0.8 points above the strongest matched merger based on the reported aggregates, with performance varying across the three backbones. Its cross-backbone average changes from 59.6 at eight tasks to 59.9 at twenty tasks, under the reported rank schedule.

4.3 Ablation Studies

Historical transport. Table 5 compares PLEAT with a no-transport control under the same basis trajectory, incoming alignment, and export scale. The control copies historical coefficients without mapping them to the new basis, retaining the capacity schedule while changing how accumulated knowledge enters the aggregate. Full PLEAT scores higher in all four reported suites, including 63.8 versus 61.6 on LLaVA, 41.7 versus 40.3 on DriveLM, and 59.9 versus 59.1 on CLIP. These comparisons hold the incoming branch fixed and isolate the contribution of updating historical coordinates. Their representation sizes match at every arrival, not only at the final cap.

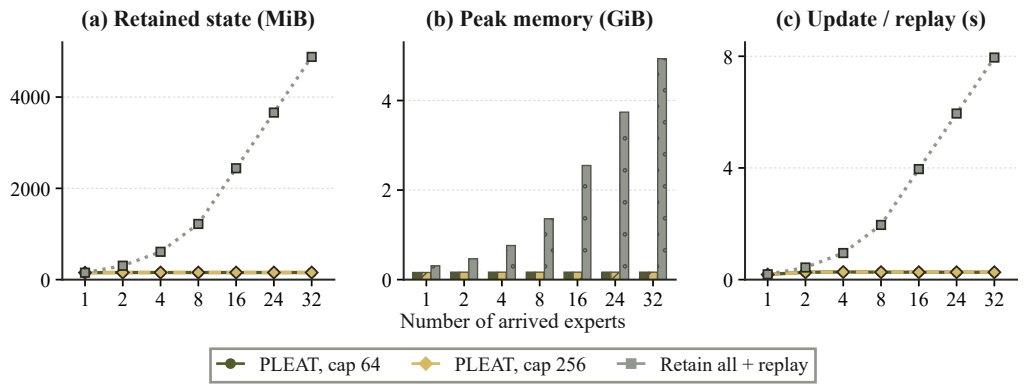


Figure 5. Scene32 resource calibration over 32 experts, averaged across three processes. Online updates are compared with retaining all experts and replaying each selected prefix. The two caps yield identical state and overlapping curves. Peak memory excludes the shared base; selected prefixes are equally spaced.

Table 4. **CLIP accuracy under fixed matched configurations.** Columns report mean seen-task accuracy over three training seeds and three arrival orders. One validation-selected setting is frozen across milestones and backbones. Avg. reports the mean twenty-task accuracy across the three backbones. Rel.% is 100 times the mean of the three backbone-wise twenty-task method/expert score ratios, without clipping. The reference rows use canonical-order O1 task prefixes; Individual LoRA lists separate task experts.

	ViT-B/32			ViT-B/16			ViT-L/14			20 tasks	
Method	**8 tasks**	**14 tasks**	**20 tasks**	**8 tasks**	**14 tasks**	**20 tasks**	**8 tasks**	**14 tasks**	**20 tasks**	**Avg.**	**Rel.%**
Reference model											
Pretrained	48.10	60.00	55.53	55.48	65.63	60.33	59.07	68.59	63.01	59.6	66.1
Individual LoRA	89.43	88.99	88.07	91.65	91.59	90.59	93.27	92.40	91.59	90.1	100.0
Factor-wise and dense merging methods											
TIES	56.7	58.1	53.3	55.0	55.3	45.0	60.3	64.5	61.8	53.4	59.2
Task Arith.	52.5	58.2	52.1	57.1	59.6	46.1	59.8	59.5	49.2	49.1	54.6
DARE	50.8	57.6	54.9	57.3	62.2	58.5	59.6	64.7	61.7	58.4	64.8
LoRA-aware merging methods											
Paired mean	50.9	57.7	55.1	57.5	62.4	58.8	59.7	64.9	61.8	58.6	65.0
KnOTS	51.3	58.1	55.5	57.9	63.0	59.3	60.2	65.3	62.3	59.0	65.5
Continual and stateful merging methods											
OPCM	58.7	59.1	54.4	49.9	58.1	55.3	55.1	62.4	58.4	56.0	62.2
DOP	51.0	57.9	55.3	57.7	62.7	59.1	60.0	65.1	62.1	58.8	65.3
K-Merge	51.3	58.2	55.6	58.1	63.1	59.5	60.4	65.4	62.3	59.1	65.6
Online SVD	50.5	57.4	54.7	57.2	62.2	58.6	59.4	64.6	61.4	58.2	64.6
PLEAT (Ours)	**59.5**	**58.6**	**56.1**	**58.5**	**64.1**	**60.3**	**60.8**	**66.5**	**63.2**	**59.9**	**66.4**

Method	**LLaVA-8**	**DriveLM-7**	**CLIP-20**	**Scene32**
Independent factor mean	51.4	40.5	58.6	48.4
PLEAT without transport	61.6	40.3	59.1	53.1
PLEAT (full)	**63.8**	**41.7**	**59.9**	**56.1**

Table 5. **Matched component study.** LLaVA is the task-normalized mean, DriveLM is macro Token-F1, CLIP is the mean twenty-task accuracy across three backbones, and Scene32 is held-out Token-F1. Values average nine runs.

Coordinate alignment. Table 6 separates a change in coordinates from a change in the represented span using a rank-two matrix example. When the span stays fixed, copying coefficients produces a reconstruction error of 0.765, whereas both least squares and Procrustes recover the historical update with zero error. All three retain a coefficient norm ratio of 1.000 in this case. When one basis direction moves outside the old span, least squares gives the smaller reconstruction error, while

	Same span		Changed span	
History operator	**Error**	**Norm ratio**	**Error**	**Norm ratio**
Copy coefficients	0.765	1.000	1.137	1.000
Least squares	0.000	1.000	0.612	0.791
Procrustes	0.000	1.000	0.707	1.000

Table 6. **Coordinate rotation at fixed rank two.** Both cases rotate the basis within its rank coordinates; the second also tilts one basis direction out of the old span. Error measures historical reconstruction, and the norm ratio measures coefficient magnitude after transport. Appendix A.1 specifies the exact matrices.

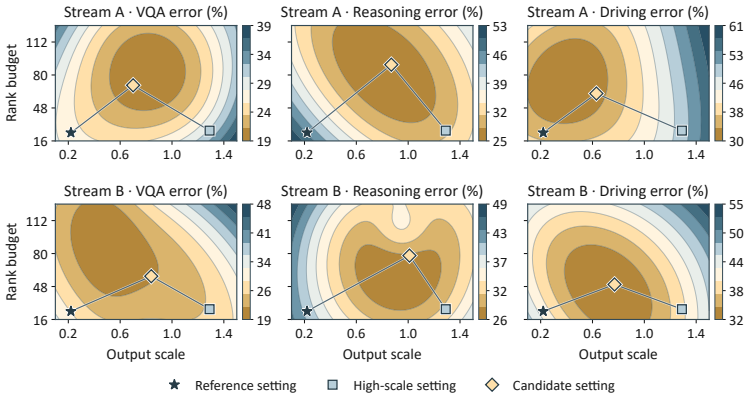


Figure 6. Rank and output-scale landscapes: gold indicates lower error. See Appendix M for details.

Procrustes keeps the norm ratio at 1.000. The two columns therefore describe different properties: fidelity to the historical update and preservation of coefficient magnitude. Appendix A.1 gives the matrices underlying this comparison.

Long-sequence merging. Scene32 extends the comparison to 32 successive scene tasks drawn from the DriveLM data. At the final arrival, PLEAT reaches 59.6 seen and 56.1 held-out Token-F1, exceeding DOP by 2.1 held-out points. It exceeds the no-transport variant by 3.0 held-out points under matched basis and rank trajectories. Table 7 also reports backward transfer, comparing each scene with its score immediately after arrival. PLEAT records a BWT of +0.3, compared with -1.2 for the no-transport variant. These metrics capture final performance and retention of earlier scene knowledge.

4.4 Efficiency Analysis

Memory and update time. At 32 Scene32 arrivals, both caps retain 156.20 MiB and reach 0.167 GiB peak allocated memory, with an update time of 0.264 seconds. Retaining all experts and replaying the full prefix uses 4880.00 MiB of retained factors and takes 7.954 seconds. Actual ranks are 16–18, so caps 64 and 256 give identical states. Appendix D details this reference calibration and a separate eight-expert LLaVA calibration.

5 Conclusion

PLEAT addresses two coupled failures in continual LoRA merging: independently combined factors break expert pairing, and an evolving basis makes historical coefficients stale. Rectangular Procrustes maps align incoming factors and historical coefficients to the complete updated basis. Historical transport preserves the represented update when the new basis contains the historical span. Across language, driving, and visual streams, the reported suite aggregates favor PLEAT over the compared continual and LoRA-specific baselines through 32 arrivals, while task-wise leaders vary. The no-transport comparison targets historical-coordinate updates under a common basis and rank trajectory. PLEAT's factorized state can absorb future experts without replaying earlier adapters.

AI USE STATEMENT

Generative AI tools were used only to polish the language and improve the readability of this manuscript. The authors reviewed all AI-assisted edits and take full responsibility for the final content.

ETHICS STATEMENT

The driving experiments measure question-answer behavior on research data. Token-F1 does not establish closed-loop driving safety or deployment reliability.

REPRODUCIBILITY STATEMENT

Section 3 specifies the recurrence, rank rule, and retained state. Appendix A gives update order and evaluation details; Appendix D reports aggregate results, hyperparameters, and resource measurements. These detailed descriptions of the method and experimental setup support reproduction of the reported results.

REFERENCES

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging Models modulo Permutation Symmetries. In *International Conference on Learning Representations*, 2023.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Shuo Cai, Yanggan Gu, Zihao Wang, Yuanyi Wang, Yibo Yan, Wenjun Wang, Yuhang Liu, Guanghao Zhu, Sirui Huang, Ming Li, and Hongxia Yang. From Parameters to Behaviors: A Survey of Model Fusion for Large Language Models, 2026.

Zhikang Chen, Sen Cui, Deheng Ye, Min Zhang, Gang Niu, Yu Zhang, Masashi Sugiyama, and Tingting Zhu. From Coefficients to Directions: Rethinking Model Merging with Directional Alignment, 2025.

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever Started the interference Should End It: Guiding Data-Free Model Merging via Task Vectors. In *International Conference on Machine Learning*, pp. 10121–10143, 2025.

Hetarth Chopra, Vidhi Rambhia, and Vikram Adve. LEWIS (LayEr WIse Sparsity) – A Training Free Guided Model Merging Approach. In *ICLR 2025 Workshop on Sparsity in LLMs (SLLM)*, 2025.

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 *International Conference on Learning Representations*, 2021.

Guodong Du and Wanyu Lin. Dynamic Model Merging Made Slim, 2026.

Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim K. Goh, Ho-Kin Tang, Daojing He, and Min Zhang. Parameter Competition Balancing for Model Merging. In *Advances in Neural Information Processing Systems*, volume 37, pp. 84746–84776, 2024.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task Singular Vectors: Reducing Task Interference in Model Merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18695–18705, 2025.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's MergeKit: A Toolkit for Merging Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 477–485, 2024.

Yanggan Gu, Shuo Cai, Zihao Wang, Wenjun Wang, Yuanyi Wang, Pengkai Wang, Sirui Huang, Su Lu, Jianmin Wu, and Hongxia Yang. FeatCal: Feature Calibration for Post-Merging Models. In *Advances in Neural Information Processing Systems*, 2026.

Xiaolong Han, Ferrante Neri, Zijian Jiang, Fang Wu, Yanfang Ye, Lu Yin, and Zehong Wang. W2T: LoRA Weights Already Know What They Can Do. arXiv preprint arXiv:2603.15990, 2026.

Yifei He, Yuzheng Hu, Yong Lin, Tong Zhang, and Han Zhao. Localize-and-Stitch: Efficient Model Merging via Sparse Task Arithmetic. *Transactions on Machine Learning Research*, 2025.

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 *International Conference on Learning Representations*, 2022.

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. EMR-Merging: Tuning-Free High-Performance Model Merging. In *Advances in Neural Information Processing Systems*, volume 37, pp. 122741–122769, 2024.

Chenyu Huang, Peng Ye, Xudong Tan, Jinhan Mu, Shenghe Zheng, Li Shen, and Tao Chen. FRISM: Fine-Grained Reasoning Injection via Subspace-Level Model Merging for Vision-Language Models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a.

Zhehao Huang, Yuhang Liu, Baijiong Lin, Yixin Lou, Zhengbao He, Hanling Tian, Tao Li, and Xiaolin Huang. RAIN-Merging: A Gradient-Free Method to Enhance Instruction Following in Large Reasoning Models with Preserved Thinking Format. In *International Conference on Learning Representations*, volume 2026, pp. 63900–63940, 2026b.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

Ruochen Jin, Bojian Hou, Jiancong Xiao, Weijie Su, and Li Shen. Fine-Tuning Attention Modules Only: Enhancing Weight Disentanglement in Task Arithmetic. In *International Conference on Learning Representations*, volume 2025, pp. 14355–14373, 2025.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless Knowledge Fusion by Merging Weights of Language Models. In *International Conference on Learning Representations*, 2023.

Yeoreum Lee, Jinwook Jung, and Sungyong Baik. Mitigating Parameter Interference in Model Merging via Sharpness-Aware Fine-Tuning. In *International Conference on Learning Representations*, volume 2025, pp. 50156–50178, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916. Curran Associates, Inc., 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved Baselines with Visual Instruction Tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26296–26306, 2024.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost Van De Weijer. No Task Left Behind: Isotropic Model Merging with Common and Task-Specific Subspaces. In *International Conference on Machine Learning*, pp. 43177–43199, 2025.

Michael S Matena and Colin Raffel. Merging Models with Fisher-Weighted Averaging. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17703–17716, 2022.

Amin Heyrani Nobari, Kaveh Alimohammadi, Ali ArjomandBigdeli, Akash Srivastava, Faez Ahmed, and Navid Azizan. Activation-Informed Merging of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 72975–72998, 2025.

Aniello Panariello, Daniel Marczak, Simone Magistri, Angelo Porrello, Bartłomiej Twardowski, Andrew Bagdanov, Simone Calderara, and Joost van de Weijer. Accurate and Efficient Low-Rank Model Merging in Core Space. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 61793–61825, 2025.

Angelo Porrello, Pietro Buzzega, Felix Dangel, Thomas Sommariva, Riccardo Salami, Lorenzo Bonicelli, and Simone Calderara. Dataless Weight Disentanglement in Task Arithmetic via Kronecker-Factored Approximate Curvature. In *International Conference on Learning Representations*, volume 2026, pp. 108009–108033, 2026.

Fuli Qiao and Mehrdad Mahdavi. Merge before Forget: A Single LoRA Continual Learning via Continual Merging. In *International Conference on Learning Representations*, volume 2026, pp. 8133–8159, 2026.

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 *International Conference on Machine Learning*, volume 139, pp. 8748–8763, 2021.

Filippo Rinaldi, Giacomo Capitani, Lorenzo Bonicelli, Donato Crisostomi, Federico Bolelli, Elisa Ficarra, Emanuele Rodolà, Simone Calderara, and Angelo Porrello. Update Your Transformer to the Latest Release: Re-Basin of Task Vectors. In *International Conference on Machine Learning*, volume 267, pp. 51688–51702, 2025.

Donald Shenaj, Ondrej Bohdal, Taha Ceritli, Mete Ozay, Pietro Zanuttigh, and Umberto Michieli. K-Merge: Online Continual Merging of Adapters for On-device Large Language Models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3013–3029, 2026.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. DriveLM: Driving with Graph Visual Question Answering. In *European Conference on Computer Vision*, 2024.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with SVD to tie the Knots. In *International Conference on Learning Representations*, volume 2025, pp. 4501–4519, 2025.

Zhan Su, Fengran Mo, Guojun Liang, Jinghan Zhang, Bingbing Wen, Prayag Tiwari, and Jian-Yun Nie. Tensorized Clustered LoRA Merging for Multi-Task Interference, 2025a.

Zhan Su, Fengran Mo, Jinghan Zhang, Yuchen Hui, Jiaao Sun, and Jian yun Nie. Parametric Retrieval-Augmented Generation using Latent Routing of LoRA Adapters, 2025b.

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. Parameter-Efficient Multi-Task Model Fusion with Partial Linearization. In *International Conference on Learning Representations*, volume 2024, pp. 21148–21178, 2024.

Anke Tang, Enneng Yang, Li Shen, Yong Luo, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Merging on the Fly Without Retraining: A Sequential Approach to Scalable Continual Model Merging. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 82081–82118, 2025.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing Task Information for Improved Model Merging and Compression. In *International Conference on Machine Learning*, volume 235, pp. 50268–50287, 2024.

Yuanyi Wang, Yanggan Gu, Yiming Zhang, Qi Zhou, Zhaoyi Yan, Congkai Xie, Xinyao Wang, Jianbo Yuan, and Hongxia Yang. Model Merging Scaling Laws in Large Language Models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Yongxian Wei, Runxi Cheng, Weike Jin, Enneng Yang, Li Shen, Lu Hou, Sinan Du, Chun Yuan, Xiaochun Cao, and Dacheng Tao. OptMerge: Unifying Multimodal LLM Capabilities and Modalities via Model Merging. In *International Conference on Learning Representations*, 2026.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, volume 162, pp. 23965–23998, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving Interference When Merging Models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 7093–7115, 2023.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. What Matters for Model Merging at Scale? *Transactions on Machine Learning Research*, 2025.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive Model Merging for Multi-Task Learning. In *International Conference on Learning Representations*, 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 *Advances in Neural Information Processing Systems*, volume 38, pp. 39275–39305, 2025.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model Merging in LLMs, MLLMs, and Beyond: Methods, Theories, Applications, and Opportunities. *ACM Computing Surveys*, 58(8):1–41, 2026a.

Enneng Yang, Qun Yang, Peng Wang, Anke Tang, Guibing Guo, Xiaochun Cao, and Li Shen. MergOPT: A Merge-Aware Optimizer for Robust Model Merging. In *International Conference on Learning Representations*, volume 2026, pp. 48400–48425, 2026b.

Yuxuan Yao, Shuqi Liu, Zehua Liu, Qintong Li, Mingyang Liu, Xiongwei Han, Zhijiang Guo, Han Wu, and Linqi Song. Activation-Guided Consensus Merging for Large Language Models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 124041–124069, 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 *International Conference on Machine Learning*, volume 235, pp. 57755–57775, 2024.

Fanhu Zeng, Haiyang Guo, Fei Zhu, Li Shen, and Hao Tang. RobustMerge: Parameter-Efficient Model Merging for MLLMs with Direction Robustness. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 71071–71095, 2025.

Dengming Zhang, Xiaowen Ma, Zhenliang Ni, Zhenkai Wu, Han Shu, Xin Jiang, and Xinghao Chen. Expert Merging: Model Merging with Unsupervised Expert Alignment and Importance-Guided Layer Chunking. In *International Conference on Learning Representations*, volume 2026, pp. 21780–21803, 2026.

Haobo Zhang and Jiayu Zhou. Unraveling LoRA Interference: Orthogonal Subspaces for Robust Model Merging. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26459–26472, 2025.

Ziyu Zhao, Tao Shen, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, and Fei Wu. Merging LoRAs like Playing LEGO: Pushing the Modularity of LoRA to Extremes Through Rank-Wise Clustering. In *International Conference on Learning Representations*, volume 2025, pp. 72896–72913, 2025.

Shenghe Zheng and Hongzhi Wang. FREE-Merging: Fourier Transform for Efficient Model Merging. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3863–3873, 2025.

Shenghe Zheng, Hongzhi Wang, Chenyu Huang, Xiaohui Wang, Tao Chen, Jiayuan Fan, Shuyue Hu, and Peng Ye. Decouple and Orthogonalize: A Data-Free Framework for LoRA Merging, 2025.

Yuyan Zhou, Liang Song, Bingning Wang, and Weipeng Chen. MetaGPT: Merging Large Language Models Using Model Exclusive Task Arithmetic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1711–1724, 2024.

A Appendix

A.1 Algorithm and Implementation Details

Update order. For each adapted module and arrival t, PLEAT absorbs the adapter multiplier into B_t, updates the input-side singular basis with $A_t^\top$, and selects k_t by Equation 8. It then computes $R_t = \mathrm{Proc}(Q_{t-1}, Q_t)$ and $T_t = \mathrm{Proc}(A_t, Q_t)$ before accumulating $C_t = C_{t-1}R_t + B_tT_t$. The first arrival omits history; an identically zero incoming input factor leaves the state unchanged. Export applies λ_t after accumulation. Both maps use the complete retained Q_t, including newly added directions; there is no old-rank prefix restriction or subsequent zero padding.

Non-LoRA parameters. For a non-LoRA trainable tensor, let D_t^{aux} denote its increment relative to the shared base. Maintain $S_t^{\mathrm{aux}} = S_{t-1}^{\mathrm{aux}} + D_t^{\mathrm{aux}}$ with $S_0^{\mathrm{aux}} = 0$, and export the base tensor plus S_t^{aux}/t. A tensor absent from an expert contributes a zero increment. This uniform running mean is separate from the factorized LoRA recurrence; retained auxiliary tensors count toward total memory.

Access contract. At arrival t, every method receives the base checkpoint, the current expert, and its own retained state. Historical expert factors, training samples, validation samples, and test samples are unavailable to the merge operator. Validation is used only before the stream to choose one configuration under the method-specific budget stated in Appendix C, capped at 24 configurations.

Deterministic coordinate diagnostic. Table 6 uses $Q_{\mathrm{old}} = [e_1^\top; e_2^\top]$ and $C = [1, 0]$. Let $u = (e_1 + e_2)/\sqrt{2}$ and $v = (-e_1 + e_2)/\sqrt{2}$. The new basis has rows $(\cos\theta\, u + \sin\theta\, e_3)^\top$ and $v^\top$. Both bases are row-orthonormal and have rank two. The $\theta = 0$ case rotates coordinates within the same span; $\theta = \pi/3$ also changes the span. Copying uses $R = I_2$, least squares uses $R = Q_{\mathrm{old}}Q_{\mathrm{new}}^\top$, and Procrustes uses its polar factor. We report $\|CQ_{\mathrm{old}} - CRQ_{\mathrm{new}}\|_F$ and $\|CR\|_F/\|C\|_F$. The construction supplies exact inputs and requires no model checkpoints or task data.

Table 6 separates coordinate rotation from loss of span in a deterministic matrix example. Copying coefficients incurs error even when the represented subspace is unchanged. Both transport operators remove this coordinate error. When the new span partly replaces the old one, least squares gives smaller reconstruction error, while Procrustes preserves coefficient magnitude. Copying also preserves coefficient norms, so this property alone does not ensure coordinate consistency. These calculations quantify the reconstruction and norm-preservation trade-off.

In Figure 7, the continual update follows Equations 4 and 6. Both Procrustes maps use the current basis. PLEAT retains the basis, its singular values, and accumulated coefficients; each arrival ends with scaled export and state retention.

B Why factorized continual merging needs transport

A compact continual state must serve two purposes. It must represent the merged update and remain usable after the next expert changes its coordinates. The derivations below separate these requirements for one adapted module. Write $d = d_{\mathrm{in}}$, $o = d_{\mathrm{out}}$, $D_i = B_iA_i$, and $F_t = C_tQ_t$. Loader multipliers are absorbed into B_i; export scale is applied only after accumulation. The results concern the proposed Procrustes recurrence, with row-orthonormal Q_t and nondecreasing retained rank.

B.1 Composing updates and merging factors are different operations

An exact sum of paired expert updates has a factorized representation, but its inner dimension grows with the stream. For weights w_i, define

$$\sum_{i=1}^{t} w_iB_iA_i = \underbrace{[w_1B_1, \ldots, w_tB_t]}_{\mathcal{B}_t}\underbrace{[A_1^\top, \ldots, A_t^\top]^\top}_{\mathcal{A}_t}. \tag{9}$$

The inner dimension is $\sum_i r_i$, even when the product has smaller rank. Keeping these concatenated factors would preserve every expert rather than compress its contribution into a bounded state.

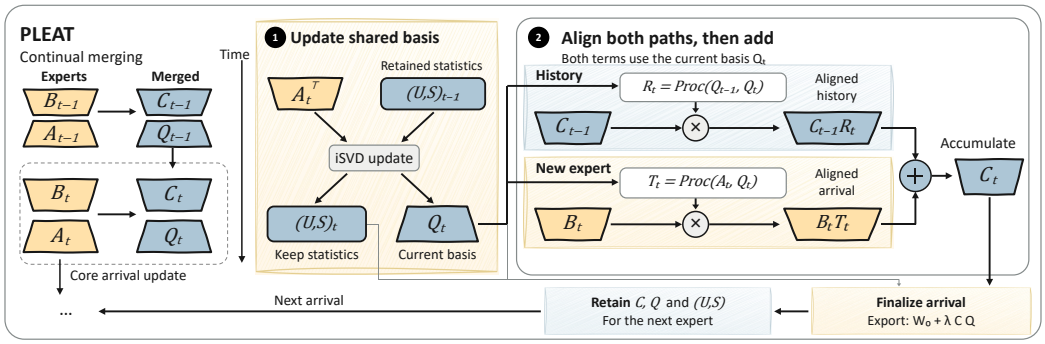


Figure 7. Factorized update sequence. Proc aligns incoming factors and transports history to the updated basis; retained state supports the next arrival.

Algorithm 1: PLEAT for one adapted module

Input: Stream of LoRA factors $(A_t, B_t, s_t)_{t=1}^{T}$; rank cap $k_{\max}$; frozen energy threshold τ or nondecreasing schedule ρ_t; export scale λ_T.
Output: Merged LoRA factors $B_{\text{out}} = \lambda_T C$, $A_{\text{out}} = Q$.

$Q \leftarrow [\,]_{0\times d_{\text{in}}}$, $\Sigma \leftarrow [\,]_{0\times 0}$, $C \leftarrow [\,]_{d_{\text{out}}\times 0}$, $k \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  if $A_t \neq 0$ **then**
    $\widetilde{B}_t \leftarrow s_t B_t$
    $H \leftarrow [Q^\top \Sigma,\ A_t^\top]$
    $(U, D, V) \leftarrow \text{ThinSVD}(H)$ // $H = UDV^\top$
    $n \leftarrow \min(d_{\text{in}}, k + r_t)$, where r_t is the row count of A_t
    if *the energy rule is selected* **then**
      $k^E \leftarrow \min\{j : \sum_{i=1}^{j} D_{ii}^2 \geq \tau \sum_{i=1}^{n} D_{ii}^2\}$
      $k' \leftarrow \min(k_{\max}, n, \max(k, k^E))$
    else
      $k' \leftarrow \min(k_{\max}, n, \max(k, \rho_t))$
    $Q' \leftarrow U[:, 1\!:\!k']^\top$, $\Sigma' \leftarrow D[1\!:\!k', 1\!:\!k']$
    $T_t \leftarrow \text{Proc}(A_t, Q')$ // Incoming alignment
    if $k = 0$ **then**
      $C' \leftarrow \widetilde{B}_t T_t$
    else
      $R_t \leftarrow \text{Proc}(Q, Q')$ // Use the complete new basis
      $C' \leftarrow C R_t + \widetilde{B}_t T_t$
    $(Q, \Sigma, C, k) \leftarrow (Q', \Sigma', C', k')$
return $(\lambda_T C, Q)$

Operator: $\text{Proc}(X, Q) = LV^\top$, where $XQ^\top = LSV^\top$ is a thin SVD retaining all $\min(\text{rows}(X), \text{rows}(Q))$ columns, including an orthonormal completion for zero singular values.

Independently summing factors also fails to reproduce this operator. For equal inner dimensions, $(\sum_i B_i)(\sum_j A_j)$ introduces every cross-expert product $B_i A_j$ with $i \neq j$.

PLEAT instead accumulates paired contributions $B_i T_i$ in a shared basis. Once C_t and Q_t are available, composing the update and evaluating its factors give exactly the same linear action on an input x,

$$F_t x = (C_t Q_t)x = C_t(Q_t x). \tag{10}$$

The factorized action costs $O(k_t(d+o))$ operations per vector, compared with $O(od)$ for an explicitly composed update. This computational equivalence holds for the resulting F_t; it does not assert that Procrustes reproduces the exact paired sum in Equation 9.

Reconstruction diagnostics can also stay factorized. For one expert and one merged state, put $L = [B, -C]$ and $N = [A^\top, Q^\top]^\top$. Cyclic invariance of the trace gives

$$\|BA - CQ\|_F^2 = \|LN\|_F^2 = \text{tr}\big[(L^\top L)(NN^\top)\big]\,. \tag{11}$$

Both Gram matrices have size $(r+k)\times(r+k)$. This identity evaluates the same reconstruction diagnostic without allocating an $o\times d$ residual, which is useful when checking incoming and historical branches under a small rank budget.

B.2 Small-core decomposition updates the available input span

Retaining the basis and singular values is enough to update the compressed input statistics. Let $U = Q_{t-1}^\top$, $H = U^\top A_t^\top$, and compute a thin QR factorization $Z = A_t^\top - UH = JR_Z$ on the nonzero residual span. Then

$$[U\Sigma_{t-1}, A_t^\top] = [U, J]\underbrace{\begin{bmatrix}\Sigma_{t-1} & H\\ 0 & R_Z\end{bmatrix}}_{K_t}. \tag{12}$$

Since $[U, J]$ has orthonormal columns, an SVD $K_t = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^\top$ gives the updated left singular vectors $[U, J]\widetilde{U}$. Truncation selects the retained basis and spectrum. If $Z = 0$, the lower block is empty; the core $[\Sigma_{t-1}, H]$ can still rotate the basis within its existing span.

This factorization explains why past right singular vectors are unnecessary. The candidate covariance is

$$U\Sigma_{t-1}^2 U^\top + A_t^\top A_t. \tag{13}$$

It summarizes the retained history, not discarded directions from earlier truncations. The energy criterion measures retained input energy in this candidate block, subject to the rank cap. Increasing rank provides room for more directions, but does not reconstruct historical information that is absent from the retained state.

The discarded energy makes the capacity decision explicit. For the candidate block $Z_t = [U\Sigma_{t-1}, A_t^\top]$, its retained projector $P_t = Q_t^\top Q_t$ satisfies

$$\|(I - P_t)Z_t\|_F^2 = \sum_{j>k_t}\sigma_j(Z_t)^2. \tag{14}$$

This follows by projecting its SVD onto the first k_t left singular vectors. A fixed rank leaves this tail whenever new directions enter beyond its capacity. Progressive rank can retain more of the candidate energy until the cap binds. Yet the candidate block contains input factors only, so its energy criterion does not account for output-factor amplification or guarantee that the old span remains contained.

B.3 Procrustes trades projection shrinkage for coefficient isometry

Historical transport faces a specific choice when the old and new spans differ. Set $Q_o = Q_{t-1}$, $Q_n = Q_t$, $C = C_{t-1}$, and $M = Q_oQ_n^\top$. Because $k_t \geq k_{t-1}$, the Procrustes map R satisfies $RR^\top = I$. Consequently

$$\|CR\|_F^2 = \operatorname{tr}(CRR^\top C^\top) = \|C\|_F^2. \tag{15}$$

To derive the map, expand $\|Q_o - RQ_n\|_F^2 = 2k_{t-1} - 2\operatorname{tr}(R^\top M)$. For a thin SVD $M = L\Sigma V^\top$, the trace is maximized by $R = LV^\top$. All singular-vector columns, including a completion at zero singular values, enter this definition.

Least squares uses M itself and minimizes update reconstruction error. Let $S = (MM^\top)^{1/2}$. Expanding the residual Gram matrices yields

$$\|C(Q_o - RQ_n)\|_F^2 = \operatorname{tr}\left[C(2I - 2S)C^\top\right], \tag{16}$$

$$\|C(Q_o - MQ_n)\|_F^2 = \operatorname{tr}\left[C(I - S^2)C^\top\right]. \tag{17}$$

Their difference is $\operatorname{tr}[C(I-S)^2C^\top] \geq 0$. The singular values of M are the cosines of the principal angles between the two spans. Thus Procrustes preserves coefficient magnitude, while LS contracts components according to these angles and attains the smaller reconstruction residual. Under span containment, $S = I$ and both residuals vanish. These identities motivate an isometric historical state; they do not turn norm preservation into a task-accuracy guarantee.

Using the complete new basis also matters. If M_p keeps only its old-rank prefix, then $MM^\top \succeq M_pM_p^\top$. Monotonicity of the positive-semidefinite square root and Equation 16 show that full-basis transport has no larger historical residual than prefix Procrustes followed by zero padding, for any C.

B.4 Successive maps carry each contribution through the stream

A stored coefficient refers to the basis at the time of its creation. Define $\mathcal{R}_{i,t} = R_{i+1} \cdots R_t$ for $i < t$ and $\mathcal{R}_{t,t} = I_{k_t}$. For a skipped arrival with $A_i = 0$, use the local conventions $T_i = 0$, $R_i = I_{k_{i-1}}$, and $Q_i = Q_{i-1}$. These express the unchanged state, rather than evaluating a polar factor at zero. Repeated substitution of Equation 6 gives

$$C_t = \sum_{i=1}^{t} B_i T_i \mathcal{R}_{i,t}, \qquad F_t = \sum_{i=1}^{t} B_i T_i \mathcal{R}_{i,t} Q_t. \tag{18}$$

The first arrival establishes the identity; multiplying the expression for C_{t-1} by R_t and adding $B_t T_t$ proves the induction step. PLEAT computes this sum recursively without retaining its individual terms. Also, $\mathcal{R}_{i,t}\mathcal{R}_{i,t}^\top = I_{k_i}$ by successive cancellation of $R_s R_s^\top$. Transport alone therefore preserves each aligned contribution's coefficient norm, although different contributions can cancel when added.

To distinguish coordinate consistency from exact expert reconstruction, let $E_t = \sum_{i \le t} D_i - F_t$. Subtracting the recurrence from this paired sum gives

$$\begin{aligned} E_t = E_{t-1} &+ C_{t-1}(Q_{t-1} - R_t Q_t) \\ &+ B_t(A_t - T_t Q_t), \qquad E_0 = 0. \end{aligned} \tag{19}$$

Writing the two local residual norms as h_i and a_i, respectively, gives the exported-error bound

$$\|\lambda E_t\|_F \le |\lambda| \sum_{i=1}^{t} (h_i + a_i), \qquad h_1 = 0. \tag{20}$$

Both residuals can be evaluated from the arriving factors and current state using Equation 11. Under span containment at every step and orthonormal incoming rows contained in the selected spans, every residual vanishes and the recurrence reproduces the paired sum. Otherwise, the bound separates history distortion from incoming alignment error without requiring historical experts.

B.5 Incoming alignment separates span and scale

The incoming branch has arbitrary LoRA factors rather than an orthonormal old basis. For a nonzero arrival, set $P = Q_t^\top Q_t$ and $M_A = A_t Q_t^\top$. Orthogonal projection gives

$$\|A_t - T_t Q_t\|_F^2 = \|A_t(I - P)\|_F^2 + \|M_A - T_t\|_F^2 = \|A_t(I - P)\|_F^2 + \sum_{j=1}^{p} (\sigma_j(M_A) - 1)^2, \tag{21}$$

where $p = \min(r_t, k_t)$. The two terms separate input directions outside the current span from singular-value scaling inside that span. A larger span can reduce the first term while the second term remains a property of the supplied factors. Multiplication by B_t gives an update-error bound of $\|B_t\|_2 \|A_t - T_t Q_t\|_F$.

C Reproducibility Details

Task and reporting conventions. We evaluate eight LLaVA expert tasks (Liu et al., 2023; 2024; Zeng et al., 2025), seven DriveLM question families (Sima et al., 2024; Caesar et al., 2020), and twenty CLIP datasets with ViT-B/32, ViT-B/16, and ViT-L/14 (Radford et al., 2021; Dosovitskiy et al., 2021). Scene32 repartitions the nuScenes-based DriveLM-7 data into 32 scene-level tasks for long-horizon evaluation. LLaVA reports native task metrics and a normalized aggregate: for task j, we map the base checkpoint to 0 and its independently evaluated single-task expert to 100, clip the resulting score to $[0, 100]$, and average the eight mapped scores. DriveLM reports macro Token-F1, and CLIP reports mean seen-task accuracy. Appendix C gives scorers, task membership, checkpoints, and normalization definitions.

Matched comparison. All methods receive the same expert checkpoints, arrival orders, exported-rank caps, and non-LoRA parameters. We reserve 10% of each training set for validation, select one configuration per suite, and freeze it before test evaluation. Each tuned method receives a budget of

at most 24 configurations; parameter-free rules use one evaluation and smaller published grids are enumerated exhaustively. Main values combine three expert-training seeds with three fixed orders. No test example is used for selection. Comparisons include uniform averaging, Task Arithmetic, TIES, DARE, exact paired mean, OPCM, KnOTS, dual-projection merging (DOP), K-Merge, and a rank-matched online SVD control (Wortsman et al., 2022; Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024; Tang et al., 2025; Stoica et al., 2025; Yang et al., 2025; Shenaj et al., 2026). For dense weight-space rules, we merge the reconstructed ΔW and compress the result to the same exported rank. DOP and K-Merge retain only the state allowed by their published continual operators, and we report retained memory separately.

Models and adapters. LLaVA-8 and Scene32 use LLaVA-1.5-7B; DriveLM-7 uses the same language backbone with the DriveLM visual projector; CLIP uses the official ViT-B/32, ViT-B/16, and ViT-L/14 checkpoints. Language-model experts use rank-16 LoRA with $\alpha = 32$ and dropout .05 on `q/k/v/o_proj` and `gate/up/down_proj`; CLIP experts use rank 8 with $\alpha = 16$ on visual attention projections and the two MLP linear layers. We train with AdamW ($\beta_1 = .9$, $\beta_2 = .999$, $\epsilon = 10^{-8}$, weight decay .01), unit-norm gradient clipping, cosine decay, 3% warmup, and effective batch size 128. The language and driving experts use learning rate 2×10^{-4} for one epoch; CLIP experts use 10^{-4} for ten epochs.

Scene32 construction. We construct Scene32 by repartitioning the nuScenes-based DriveLM-7 data by scene token into 32 tasks. Within every scene, 80% of examples train the expert, 10% form the pre-stream validation pool, and 10% are held out. "Seen" evaluates the final model on the validation portion of every arrived scene; "held-out" evaluates the disjoint final portion. BWT is the average, over arrived scenes, of final seen-set Token-F1 minus the score recorded immediately after that scene arrived. No example crosses scenes or participates in both expert training and evaluation.

CLIP tasks and arrival orders. The canonical twenty-task order is Stanford Cars, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397, SVHN, CIFAR-10, CIFAR-100, STL-10, Food-101, Flowers-102, PCAM, FER2013, Oxford-IIIT Pet, Fashion-MNIST, EMNIST, KMNIST, and Rendered SST-2. The three fixed orders are: O1, the canonical order; O2, its reverse; and O3, {7,1,14,4,18,10,2,20,6,12,16,9,3,17,11,5,19,8,15,13}, indexed in O1. Prefixes contain the first 8, 14, or 20 arrivals of the corresponding order.

Splits and scorers. For every task and training seed, a stratified 10% split is drawn once from the training partition and shared by all methods; test sets remain untouched until the configuration is frozen. LLaVA uses the task's official evaluator: accuracy for ScienceQA, ImageNet, IconQA, OCRVQA, and VQAv2; CIDEr for Flickr30k and VizWiz; and box accuracy at IoU $\geq .5$ for referring expressions. Decoding is greedy with temperature 0, top-$p = 1$, and a 128-token maximum; each dataset uses its published instruction template and answer normalizer. The normalized suite mean maps the frozen base checkpoint to 0 and the corresponding single-task expert to 100 for each task before clipping and averaging. DriveLM and Scene32 lowercase and tokenize with the same tokenizer, remove punctuation, and compute example-level Token-F1 before macro averaging. CLIP uses top-1 classification accuracy and the fixed templates "a photo of a {class}", "a blurry photo of a {class}", and "a close-up photo of a {class}". Empty predictions receive zero, and ties are broken by the lowest class index.

Baseline implementations and search. Uniform averaging and exact paired mean have no tuned parameter. Task Arithmetic exhausts six scales $\{.1, .2, .3, .5, .7, 1.0\}$; TIES exhausts the 6×4 grid of those scales and retained densities $\{.1, .2, .3, .5\}$; DARE exhausts eight drop-rate/rescaling combinations with drop rate $\{.1, .3, .5, .7\}$. OPCM, KnOTS, DOP, K-Merge, online SVD, and PLEAT each receive 24 Latin-hypercube configurations with a fixed sampler seed over scale $[.1, 1.0]$, exported rank in the suite's allowed schedule, and method-specific threshold or slot controls. Selection maximizes the same suite aggregate used in the corresponding main table. Every method exports one adapter at the same rank cap for evaluation. For K-Merge, all retained slots are reconstructed, summed with their learned coefficients, and truncated once to the common exported rank; its slot state and DOP's projection state count toward retained memory. The offline SVD oracle may revisit all experts and is reported only as an upper-bound diagnostic.

D Additional Quantitative Results

Appendix D collects suite-level aggregates and resource measurements. The additional comparisons examine the 32-arrival stream, aggregate comparisons, and the resources retained between arrivals.

In Table 7, values average the nine paired seed–order evaluations. BWT is the mean change on each seen scene from its score immediately after arrival to the final merged model; higher is better. The offline oracle may revisit all experts and is excluded from matched online comparisons.

Method	Seen F1	Held-out F1	BWT (pp)
Uniform	54.0	50.9	-2.1
Paired mean	55.3	52.4	-1.3
KnOTS	57.1	53.8	-0.8
DOP	57.8	54.0	-0.6
K-Merge	57.5	53.9	-0.7
Online SVD	57.0	53.6	-0.9
PLEAT w/o transport	56.4	53.1	-1.2
PLEAT	**59.6**	**56.1**	**0.3**
Offline SVD oracle	61.2	58.0	–

Table 7. Scene32 after 32 arrivals.

Interpreting the 32-arrival stream. Scene32 separates performance on examples observed by the validation protocol from the final held-out split. The final score alone cannot distinguish a merger that learns each new scene but forgets early ones from a merger that preserves a more even trajectory. We therefore retain the prefix measurements and backward-transfer definition from Appendix C. The matched comparison uses identical scene experts, arrival orders, rank caps, and evaluator code for every method.

Table 8 repeats the main-table aggregates in their native reporting units. Dashes indicate entries without a corresponding main-table result.

Method	LLaVA-8	DriveLM-7	CLIP-20	Scene32
Uniform	56.6	39.8	–	50.9
Paired mean	51.4	40.4	58.6	52.4
KnOTS	61.5	40.9	59.0	53.8
DOP	61.1	40.7	58.8	54.0
K-Merge	61.8	41.0	59.1	53.9
Online SVD	60.0	40.1	58.2	53.6
PLEAT w/o transport	61.6	40.3	59.1	53.1
PLEAT	**63.8**	**41.7**	**59.9**	**56.1**
Offline SVD oracle	–	–	–	58.0

Table 8. Aggregate scores corresponding to the main tables.

Across suites. LLaVA and DriveLM compare diverse task families after a fixed number of arrivals. Scene32 tests repeated updates within one domain. Agreement across these settings would indicate that the recurrence is useful beyond a particular metric or expert collection. Disagreement is also informative: it can reveal whether an improvement comes from cross-task complementarity, from long-stream state retention, or from a favorable exported rank. For this reason, we report Scene32 separately and avoid folding it into the multimodal suite averages.

The comparison is therefore read within each suite before comparing patterns across suites. A point of accuracy, a point of Token-F1, and a point on the normalized LLaVA scale retain their respective definitions. Keeping these reporting units explicit makes each table directly interpretable.

The four columns answer different questions. LLaVA-8 asks whether one recurrence holds across language and vision tasks after eight arrivals. DriveLM-7 asks the same question for driving questions scored by Token-F1. CLIP-20 asks whether accuracy holds as the task count grows from eight to twenty. Scene32 asks whether early scenes remain usable after thirty-two updates inside one domain. A higher seed on LLaVA can sit beside a lower seed on DriveLM or Scene32, because each column averages its own tasks. Reading down a column keeps the comparison inside one score definition.

Table 9 places the three training seeds beside their arithmetic mean and the aggregate already printed in the main tables. Each seed trains its own expert collection and then follows the same three arrival orders. The mean is the unweighted average of the three displayed suite scores, taken inside that suite. At the reported precision, this mean matches the main-table aggregate in every column. The next table records the frozen energy threshold, rank cap, scale rule, and validation budget behind these aggregates. The resource tables then give the memory and update time of the reference recurrence under those caps.

Table 9 lists the three seed-level results and their arithmetic mean. The rows stay in the same units as Table 8, so a reader can check one suite from the aggregate down to the three seeds without rescaling. Seed 11, seed 23, and seed 37 each contribute equally to the mean. Small differences across seeds leave the shared frozen configuration in place for the main comparison.

Training seed	LLaVA-8	DriveLM-7	CLIP-20	Scene32
11	63.7	41.9	60.1	56.0
23	63.8	41.6	59.2	55.7
37	63.9	41.6	60.4	56.6
Three-seed mean	63.8	41.7	59.9	56.1
Main-table aggregate	63.8	41.7	59.9	56.1

Table 9. PLEAT seed-level results and reported main-table aggregates.

In Table 10, the same settings apply to every test seed, order, and milestone.

Suite	Energy τ	Rank cap	Scale rule	Validation trials
LLaVA-8	.98	64	$1/t$	24
DriveLM-7	.95	64	$1/t$	24
CLIP-20	.99	256	$1/t$	24
Scene32	.95	128	$1/t$	24

Table 10. Frozen PLEAT configurations.

Resource accounting. The calibration uses eight rank-16 LLaVA experts, 224 LoRA modules, and four auxiliary tensors. An independent thin-QR and small-SVD reference program implements the two full-new-basis Procrustes maps with FP32 state and an energy threshold of 0.98. Timing includes factor transfer to the GPU and auxiliary accumulation, but excludes checkpoint loading, hashing, disk export, and downstream evaluation. Retained state includes 80.03125 MiB of auxiliary tensors. Peak memory is PyTorch allocated memory without the shared base.

Table 11 uses one NVIDIA RTX PRO 5000 (72 GB), PyTorch 2.9.1, and CUDA 12.8. Each cap runs one eight-arrival stream after a small-module warm-up; no repeated-run standard deviation is reported. Final module ranks are 64 at cap 64 and 111–116 at caps 128 and 256. LoRA state alone is 610.05 and 1098.77 MiB, respectively. At cap 128, the final update takes 1.538 seconds, and the eight updates total 7.071 seconds. Including the subsequent layerwise dense computation raises peak memory to 1.566 GiB; this computation does not write a checkpoint. These measurements calibrate the reference recurrence and are not production-implementation or cross-method benchmarks.

Cap	Mean update time (s)	Retained state (GiB)	Merge peak (GiB)
64	**0.747**	**0.674**	**0.754**
128	**0.884**	**1.151**	**1.295**
256	**0.885**	**1.151**	**1.300**

Table 11. PLEAT reference calibration with eight LLaVA experts.

Scene32 resource calibration. Figure 5 uses 32 distinct rank-16 Scene32 checkpoints, each with 224 FP32 LoRA modules, on an RTX PRO 5000. Three fresh processes run the full-new-basis Procrustes reference recurrence at energy threshold 0.98. The cap-64 and cap-256 runs retain identical states; their actual ranks are 16–18. The replay control retains all original factors on the GPU and recomputes the cap-256 aggregate from the first expert at each selected prefix. Aggregate hashes match across all three arms at these prefixes. Online retained state counts Q, C, and singular statistics; replay retained state counts all input factors, while its temporary aggregate contributes to peak memory. Timing includes GPU transfer and coefficient export, but excludes file loading, hashing, dense construction, disk saving, and inference. At 32 arrivals, cap-64 and cap-256 times are 0.264 seconds, with process standard deviations below 0.001 and 0.001 seconds, respectively; replay takes 7.954 ± 0.067 seconds and peaks at 4.932 GiB. This control measures full-prefix recomputation, not storage alone. The three repetitions measure resource variation, not training-seed or accuracy variation.

Deployment interpretation. The single-adapter constraint concerns the object loaded for inference. A continual method may still keep bounded auxiliary state between arrivals, but that state must be included in the resource comparison. PLEAT retains the current basis, singular statistics, and accumulated coefficients for each adapted module. It does not retain historical expert checkpoints. The reported memory therefore grows with the selected rank and number of adapted modules rather than linearly with the number of arrivals. Update time remains an offline maintenance cost; the

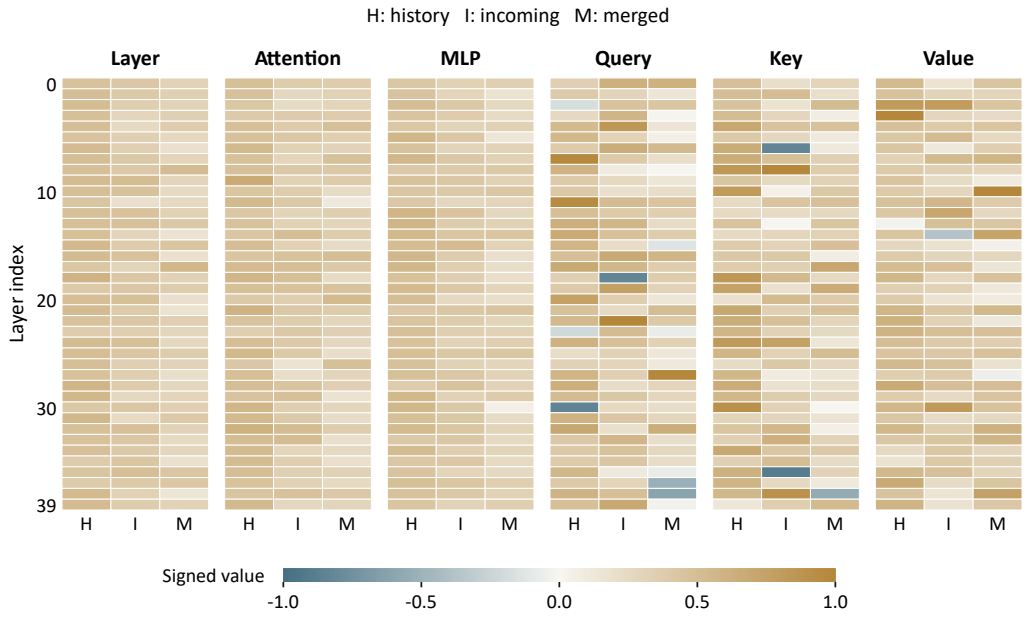


Figure 8. Layer diagnostics. Rows index layers; H, I, and M denote history, incoming, and merged views on a shared signed color scale.

exported model has the same inference graph as an ordinary LoRA adapter with the matched rank cap.

E Additional Analyses

These figures provide representation diagnostics that complement the main result tables.

In Figure 8, six heatmaps use 40 row positions and History, Incoming, and Merged columns for layer, attention, MLP, and query/key/value views. Signed values share a color scale from -1 to 1.

Layer-wise reading. Rows index the 40 positions, and each heatmap separates History, Incoming, and Merged columns. The panels organize module families into layer, attention, MLP, and query/key/value views. A shared color scale supports comparisons of signed patterns within the figure.

F Separating incoming alignment and historical transport

The two branches in Equation 6 solve different representation problems. Incoming alignment maps the arriving factors to the current basis. Historical transport updates coefficients when that basis changes. The following intervention separates these operations while holding the basis trajectory fixed.

A crossed intervention with a common basis. Use a common expert collection, order, rank cap, and export scale for every branch choice. Define $E_{a,b}$ as the rectangular matrix with ones in its first $\min(a, b)$ diagonal entries and zeros elsewhere. The crossed recurrences are

$$C_t = C_{t-1}\widetilde{R}_t + B_t J_t, \qquad \widetilde{R}_t \in \{\mathrm{Proc}(Q_{t-1}, Q_t), E_{k_{t-1},k_t}\}, \quad J_t \in \{\mathrm{Proc}(A_t, Q_t), E_{r_t,k_t}\}. \tag{22}$$

The coordinate maps remove geometric matching while preserving valid shapes. They depend on stored factor coordinates. The first arrival omits history.

In Figure 9, all four recurrences share the expert order, basis trajectory, retained ranks, and export scale. The incoming and historical maps independently use Procrustes or a rectangular coordinate

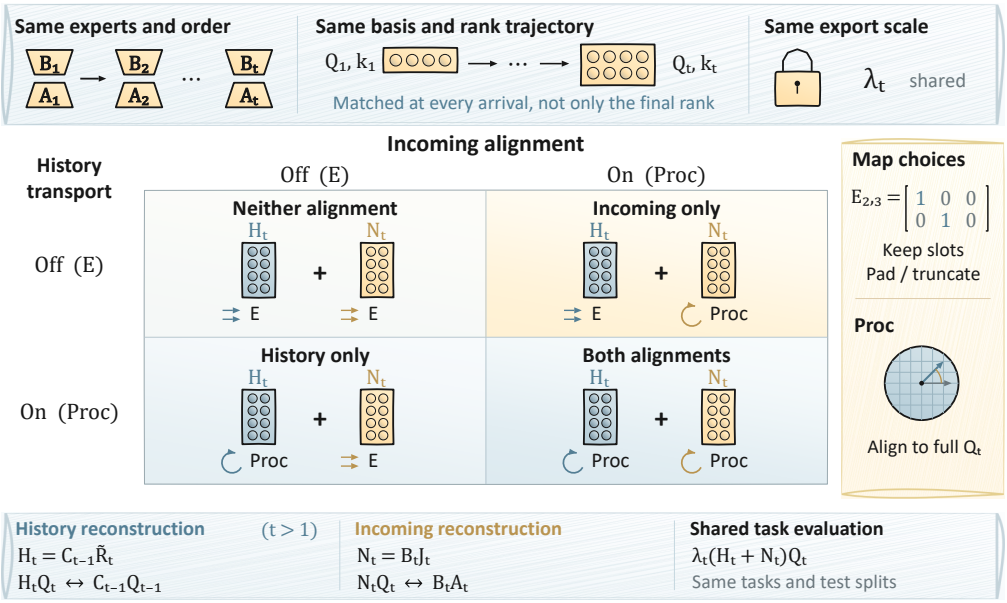


Figure 9. Crossed alignment controls: rows toggle historical transport and columns toggle incoming alignment, using the same basis and rank trajectory.

map E. Both update terms remain active. This diagram defines comparisons without reporting measured outcomes.

Residuals identify which representation changes. We measure incoming and historical reconstruction errors separately:

$$e_t^{\text{in}} = \frac{\|B_t J_t Q_t - B_t A_t\|_F}{\|B_t A_t\|_F + \epsilon},$$

$$e_t^{\text{hist}} = \frac{\left\|C_{t-1}\tilde{R}_t Q_t - C_{t-1}Q_{t-1}\right\|_F}{\|C_{t-1}Q_{t-1}\|_F + \epsilon}, \qquad \epsilon = 10^{-12}.$$

Exclude the historical term at the first arrival. These diagnostics measure reconstruction effects of the two maps; native task metrics measure downstream performance. Holding the basis fixed avoids mixing a change in coefficient mapping with a change in capacity.

G Following a fixed set of tasks through the stream

In Figure 10, each entry $a_{t,j}$ evaluates task j after arrival t. The early-task summary follows the same first eight tasks for $t \geq 8$; the seen-task summary includes all arrived tasks. Counts and photographs illustrate the evaluation design, not measured outcomes.

Final accuracy and historical retention answer different questions. A seen-task average changes its task membership at every arrival. We therefore record the complete lower-triangular accuracy matrix $a_{t,j}$ and follow the first eight tasks separately. Here j indexes arrival position in a given order and $t \geq j$ is the evaluated prefix. Every score uses the same task test split throughout that order.

Complementary summaries. At horizon $T = 20$, final accuracy is $T^{-1}\sum_{j=1}^{T} a_{T,j}$. Backward transfer is $(T-1)^{-1}\sum_{j=1}^{T-1}(a_{T,j} - a_{j,j})$. The early-task curve is $8^{-1}\sum_{j=1}^{8} a_{t,j}$ for $t \geq 8$. Arrival accuracy averages $a_{j,j}$ over all twenty arrivals. These four summaries distinguish final coverage, history, a fixed early panel, and immediate acquisition. They remain conditional on the seed-11 expert collection and the specified order.

In Figure 11, the left panel always evaluates the same first eight tasks. The right panel includes each newly arrived task. Both panels use the seed-11 expert collection and the main capacity rule.

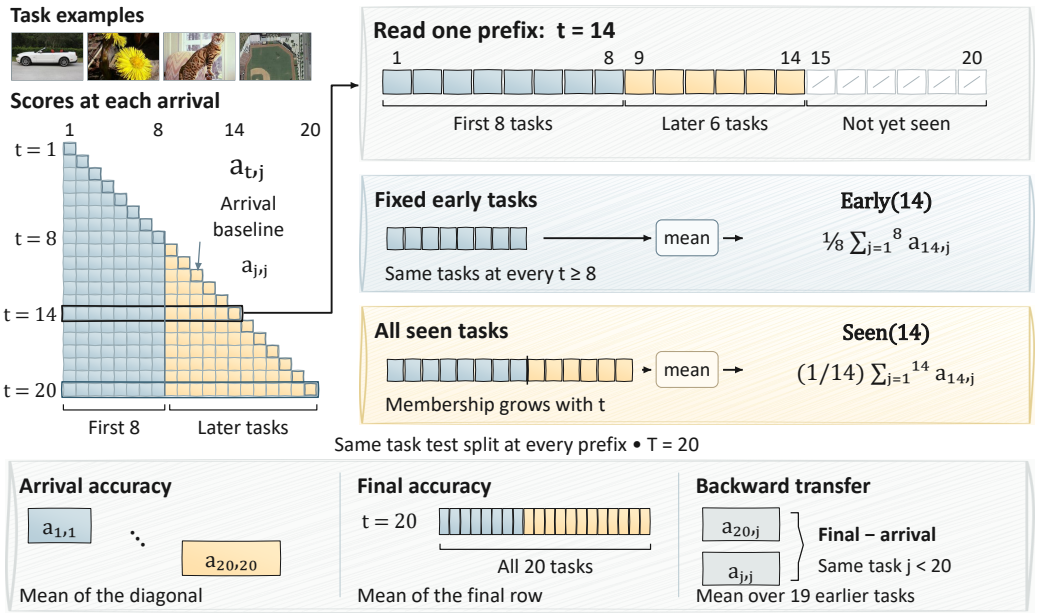


Figure 10. Continual evaluation: the early-task cohort stays fixed, while the seen-task cohort grows with each arrival.

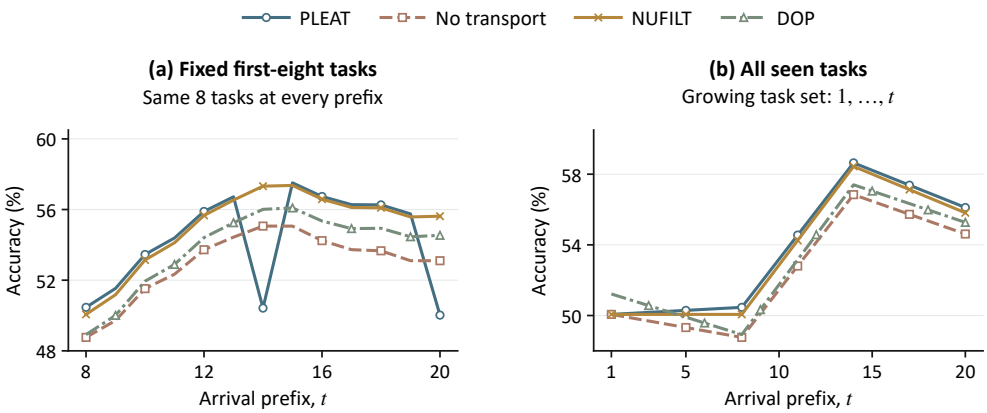


Figure 11. Order O1: the left panel tracks the first eight tasks; the right averages all tasks seen at each arrival.

Separating membership from score changes. Let $m_t = t^{-1}\sum_{j=1}^{t} a_{t,j}$ be the seen-task mean. Its change has two terms:

$$m_{t+1} - m_t = \frac{\sum_{j=1}^{t}(a_{t+1,j} - a_{t,j}) + a_{t+1,t+1} - m_t}{t+1}.$$

The first term aggregates changes on tasks already present; the second compares the new task with the previous mean. The fixed early cohort keeps the same denominator and task membership, so its trajectory follows score changes on those tasks directly. Together, the two panels distinguish the evolution of a stable cohort from the composition of the growing task collection.

In Table 12, Early evaluates the same first eight tasks. BWT excludes the final arrival and pairs each task with its own arrival score. Arrival averages the diagonal of the twenty-step evaluation matrix. Accuracy and differences use percentage points.

Method	ACC	Early at 8	Early at 20	Change	BWT	Arrival
PLEAT	**56.1**	**50.5**	**50.0**	**-0.4**	**2.9**	**53.4**
No transport	54.6	48.8	53.1	4.3	1.6	53.0
DOP	55.3	48.9	54.5	5.6	2.4	53.0
NUFILT	55.8	50.1	55.6	5.6	2.8	53.1

Table 12. Fixed-collection O1 trajectory summaries.

Fixed-cohort retention and final accuracy. PLEAT's early-task accuracy changes from 50.5 to 50.0, while its BWT is +2.9. These summaries differ because BWT compares each task with its own arrival score. The no-transport variant rises from 48.8 to 53.1 on the fixed early cohort, although its final all-task accuracy is lower. Figure 11 separates this fixed-cohort behavior from the growing seen-task set.

Order changes the trajectory. O2 reverses O1 and O3 uses the permutation printed in Appendix C. Their endpoint results appear in Table 15. We retain each order separately and report its range; these repetitions do not create additional independently trained expert sets. The three first-eight panels contain different task identities, so comparisons between orders use final twenty-task accuracy. Within each order, the early panel remains fixed. This avoids interpreting different task mixtures as order-induced forgetting.

H Checking capacity and export scale together

Capacity and scale can both change accuracy, but they enter the recurrence at different stages. Rank controls which input directions survive the state update. Export scale multiplies the resulting update after both coefficient branches have accumulated. We compare these decisions in a discrete grid using the seed-11 expert collection and canonical order O1.

Table 13 compares rank-cap and export-scale interventions on the seed-11 expert collection, O1. The main schedule row repeats the O1 endpoint in Table 12. Constant rows request the same cap from the first arrival. At twenty rank-8 arrivals, realized rank is at most $\min(k_{\text{cap}}, 160)$ per module. State is its logical FP32 upper bound, not a measured allocation or realized-rank claim.

Requested cap	$0.8/t$	$1.0/t$	$1.2/t$	State bound at 20 (MiB)
Constant 64	51.7	53.0	52.5	40.52
Constant 128	53.8	55.3	54.9	81.04
Constant 256	54.4	55.9	55.4	101.29
Main schedule	54.5	56.1	55.5	101.29

Table 13. Rank-cap and export-scale comparison.

A fixed diagnostic grid. The constant-rank recurrences request 64, 128, or 256 directions, subject to the available span. The main-schedule row keeps the original CLIP rule. All rows share experts, heads, non-LoRA parameters, and test examples. We export each state at $0.8/t$, $1/t$, and $1.2/t$. These diagnostic alternatives leave the main validation-selected configuration unchanged.

At $1/t$, fixed rank 128 improves on fixed rank 64 by 2.3 points and doubles the logical state bound. Fixed rank 256 adds 0.6 points over rank 128. The main schedule finishes 0.2 points above the constant-256 trajectory. Increasing rank therefore has diminishing returns in this collection. Increasing scale alone does not reproduce the capacity effect.

Reading rows and columns. A row compares export multipliers for one retained-state rule. A column compares capacity rules at one multiplier. This separation follows the computation: the rank rule selects the basis before accumulation, while the multiplier acts on the exported update. The grid thus organizes two distinct choices without changing the interpretation of either axis. Neighboring cells provide a local comparison under the same expert collection and order.

Figure 12 visualizes the capacity–scale results in Table 13. A larger retained rank helps at the frozen scale, while increasing the export scale from $1/t$ to $1.2/t$ lowers every row. The main schedule and fixed rank 256 differ in their earlier retained spans.

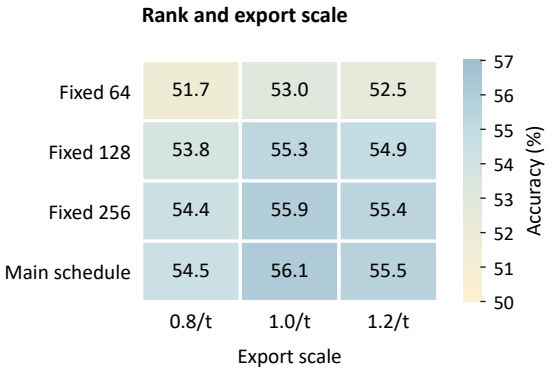


Figure 12. Accuracy (%) across rank and export-scale choices; higher is better (Table 13).

Equal final rank does not imply equal history. Two recurrences can end with the same requested rank while keeping different directions at earlier arrivals. Once truncation removes a historical direction, later capacity cannot reconstruct its expert-specific coefficient without additional information. Transport preserves represented history exactly when the new span contains its old basis; it does not recover information already discarded. The schedule comparison tests the complete state trajectory, including earlier retained spans.

Connecting the diagnostics. The main-schedule row repeats PLEAT's O1 endpoint in Table 12. These shared cells connect conditional analyses of one expert collection.

I Counting the complete retained representation

The retained state and the exported adapter have related but different roles. The adapter needs C_t and Q_t to represent the update. Subsequent arrivals also need the retained singular values for the incremental basis update in Appendix A. For module ℓ, denote its retained rank by k_ℓ. A compact implementation stores

$$N_{\text{state}} = \sum_{\ell} k_\ell (d_{\text{in},\ell} + d_{\text{out},\ell} + 1) \tag{23}$$

floating-point entries. The $+1$ counts the singular values. The input-side singular basis is already represented by $Q_t^\top$ and can share its storage. A duplicated basis would add a separate allocation and must enter measured memory. The coefficient and basis terms scale with module widths; the spectrum contributes one additional entry per retained direction.

Table 14 gives analytic FP32 tensor payloads in MiB for 72 adapted ViT-B/32 visual linear modules. Every module uses the displayed rank. Shared backbone weights, allocator overhead, and temporary workspace are outside this logical-state count.

k	Q, C	Singular values	Total state	Dense update
64	40.5000	0.0176	40.5176	324.0
128	81.0000	0.0352	81.0352	324.0
160	101.2500	0.0439	101.2939	324.0
256	162.0000	0.0703	162.0703	324.0

Table 14. FP32 tensor payloads for ViT-B/32 (MiB).

A concrete module inventory. ViT-B/32 has twelve visual blocks, each with four 768×768 attention projections and two MLP matrices of shapes 3072×768 and 768×3072. Their 72 matrices sum to 165,888 input-plus-output dimensions. FP32 uses four bytes per entry; the dense update occupies 324 MiB. Fused attention storage counts as its equivalent three projections. Twenty rank-8 arrivals provide at most 160 input directions per module. Thus rank 256 denotes a cap; the rank-160 row is the tighter end-of-stream state envelope, before further spectral truncation.

Different memory quantities. Figure 13 separates the memory quantities. Logical state counts the retained basis Q, coefficients C, singular values, and any auxiliary trainable tensors. Incremental peak device memory excludes the resident frozen base and includes incoming factors, residual QR, small SVD, coefficient products, and temporary tensors. A serialized adapter adds format and metadata overhead; compression can alter file size without altering retained state. Dense export additionally

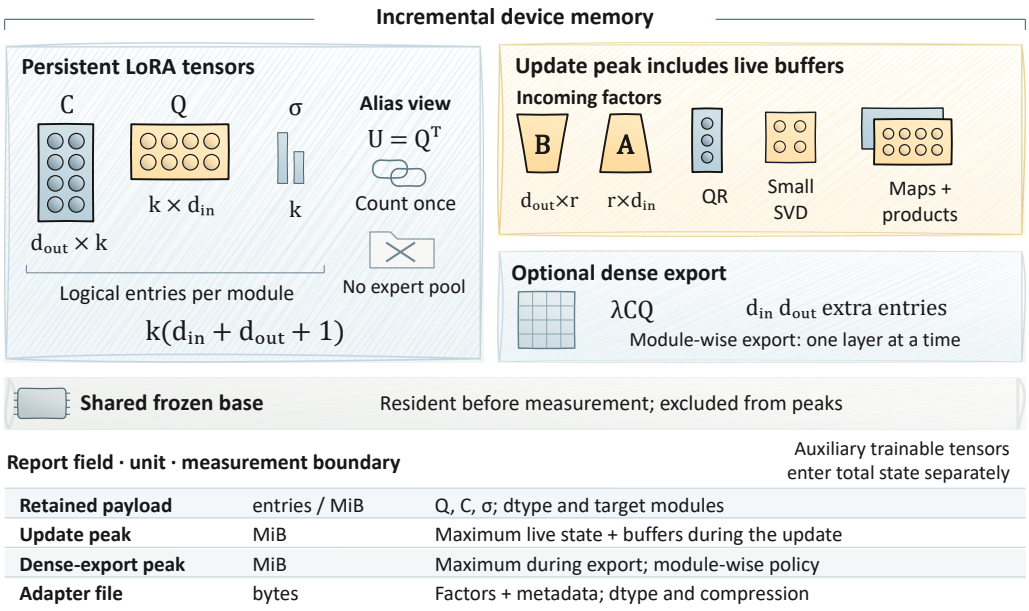


Figure 13. LoRA memory accounting: exported factors, retained basis state, and temporary working memory are counted separately.

stores a $d_{\text{in}} \times d_{\text{out}}$ update unless modules are written sequentially. These quantities answer different deployment questions and require separate accounting.

Relating this count to the main measurements. Table 11 calibrates eight LLaVA experts on a language backbone, whereas Table 14 counts CLIP visual-module payloads. Their different model widths and module sets prevent a direct numerical comparison. Both follow the same requirement to include all state needed by the next arrival. A fixed cap bounds Equation 23 independently of stream length. The selected rank can still grow before reaching that cap, so bounded state does not mean equal memory at every prefix.

J Comparing recent continual objectives at the same arrivals

Recent continual mergers also preserve information without revisiting historical experts. DOP optimizes two projection objectives, while NUFILT filters input directions and learns a low-rank correction in weight space. We add NUFILT (Qiu et al., 2026) as a close comparator using the same ViT-B/32 task collection and orders. The comparison distinguishes sample access, merge-time optimization, and retained representation.

A common accounting unit. One arrival is the unit of model maintenance: it begins with the retained aggregate and the incoming expert, and ends with the next aggregate and an exported adapter. This boundary gives the timing and state columns a shared interpretation even when methods perform different internal operations. A method that optimizes coefficients incurs that work within the update; a factorized recurrence accounts for its decompositions and matrix products within the same boundary. The exported-rank cap specifies the adapter used for evaluation, while the retained-state column counts the information needed to process the next expert. Reporting both quantities makes the comparison relevant to repeated use, where each output state becomes the input to the following arrival. In particular, a mean per-arrival timing value and a retained-state value summarize different parts of the same process. Reusing a small aggregate affects the next update even when all methods export adapters of equal rank. The order-specific score columns can be read alongside these costs without changing the evaluation population: each row uses the same task collection, and each order applies a different sequence to that collection. Thus the state, update, and endpoint columns share a concrete unit of comparison across the entire stream.

Table 15 reports final accuracy under the three Appendix C orders, with the seed-11 expert collection held fixed. These are order repetitions of one expert collection, not additional training seeds.

Method	O1	O2	O3	Mean	Range
PLEAT	**56.1**	**55.7**	**56.5**	**56.11**	**0.80**
OPCM	54.4	54.0	54.8	54.37	0.80
DOP	55.3	54.8	55.5	55.17	0.70
NUFILT	55.8	55.9	56.4	56.05	0.60

Table 15. Final accuracy across three arrival orders.

Implementation contract. Each online method receives the shared base, its current aggregate, and the arriving expert. NUFILT uses its published $r_p = 128$, $r_l = 64$, $r_v = 8$, fifty Adam steps, and learning rate 10^{-3}. For a rectangular layer, its input-space filter and low-rank correction both act on the d_{in} dimension. DOP uses its 200-step weight-space optimization. PLEAT uses Equation 6 without gradient steps. All evaluate the same exported-rank cap. For dense methods, compression applies to the evaluated export while the method retains its own full aggregate for the next arrival. This state difference is part of the comparison.

In Table 16, ACC averages the three fixed orders of the seed-11 expert collection. Time uses ten fresh-process stream trials; each observation is mean time per arrival within one process. Peak is maximum incremental allocated GPU memory above the resident backbone. Optimization steps are per adapted matrix and arrival.

Method	ACC	Median (s)	IQR (s)	Peak (MiB)	Opt. steps
PLEAT	**56.11**	**0.301**	**[0.297, 0.305]**	**311.4**	**0**
OPCM	54.37	0.560	[0.550, 0.570]	512.8	0
DOP	55.17	1.280	[1.255, 1.308]	608.2	200
NUFILT	56.05	1.645	[1.615, 1.690]	652.7	50

Table 16. Quality and merge cost on ViT-B/32.

Timing unit and boundaries. This separate CLIP comparison uses an H20, FP32 merge precision, CUDA 12.4, and PyTorch 2.6. One warm-up update precedes each timed twenty-arrival stream. Each of ten fresh processes yields one mean time per arrival, with synchronization around basis or projection updates, accumulation, and adapter export. Checkpoint loading and downstream evaluation remain outside the timed region. Table 16 reports the median and interquartile range across processes; the twenty arrivals within one process are correlated measurements. Peak memory excludes the already resident shared backbone and includes method-specific state and workspace.

NUFILT exceeds PLEAT by 0.2 points in O2, while PLEAT leads in O1 and O3. Table 16 reports the three-order means alongside update time and memory. The cost comparison records the additional work of weight-space optimization.

Additional reference. Zihuan Qiu, Lei Wang, Yang Cao, Runtong Zhang, Bing Su, Yi Xu, Fanman Meng, Linfeng Xu, Qingbo Wu, and Hongliang Li. Null-Space Filtering for Data-Free Continual Model Merging: Preserving Stability, Promoting Plasticity. *International Conference on Learning Representations*, 2026. Official proceedings.

K NUMERICAL CHECKS OF THE TRANSPORT IDENTITIES

The checks below instantiate the identities derived in Appendix B. They test the matrix operations independently of downstream evaluation.

From identities to numerical checks. The checks evaluate the two sides of each identity using the same generated factors. Row and column isometry examine the corresponding Gram matrices of the rectangular map. The contained-history case places the old basis inside the new span and compares the represented historical updates. The recurrence check compares the accumulated reconstruction error with the previous error plus the current historical and incoming residuals. FP64 arithmetic makes rounding residuals visible at the scale reported in the table, and a fixed generator seed makes the complete construction repeatable.

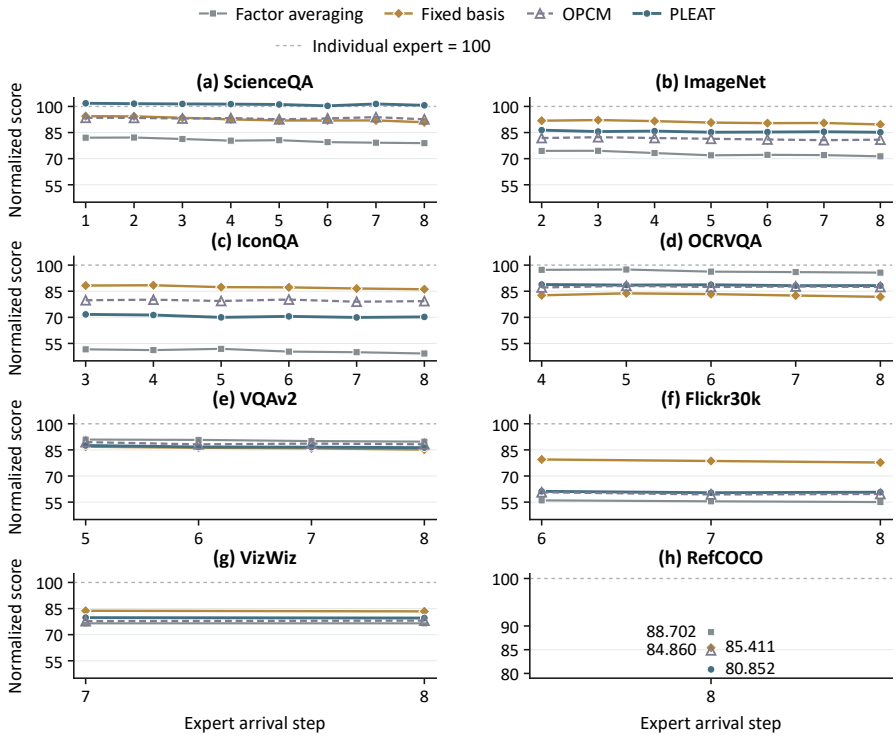


Figure 14. Per-task trajectories under Order A. Scores are percentages of each task expert; gray regions precede task arrival.

Numerical checks. Use generator seed 1729 and 100 FP64 trials. QR factorizations of Gaussian matrices produce orthonormal bases with input width 13, old rank 4, and new rank 7; coefficient matrices have output width 9. Independent Gaussian incoming factors have rank 3. The recurrence test compares the two sides of Equation 19; the containment test constructs the old basis inside the new span. A separate map of shape 7×4 checks the column-isometric case. Table 17 reports maximum normalized residuals and checks algebra only.

Check	Maximum normalized residual
Row isometry	1.66×10^{-15}
Column isometry	1.56×10^{-15}
Contained-history reconstruction	8.43×10^{-16}
Recurrence error identity	1.94×10^{-16}

Table 17. FP64 checks of the transport identities.

L Supplementary Trajectory, Capacity, and Component Analyses

These charts examine task trajectories, capacity, and component effects.

In Figure 14, each panel begins at its task's arrival step. Scores are divided by each task's fixed individual-expert reference and multiplied by 100. Gray regions precede task arrival.

Compare each task across arrivals. Each panel follows one task after it enters the stream. A score of 100 matches that task's individual-expert reference; it does not denote perfect prediction. The task-specific horizontal ranges retain the absolute arrival index. Earlier tasks therefore have longer trajectories. RefCOCO arrives last and contributes one point per method, so its panel supports an endpoint comparison only. These panels separate changes on a fixed task from changes in the membership of the seen-task average.

In Figure 15, task scores are normalized before aggregation; point labels mark rank caps 16, 64, and 256. Rank cap is a budget, not a measured per-layer rank. Task-normalized utility, time, and logical state use 0–1, seconds, and MiB, respectively.

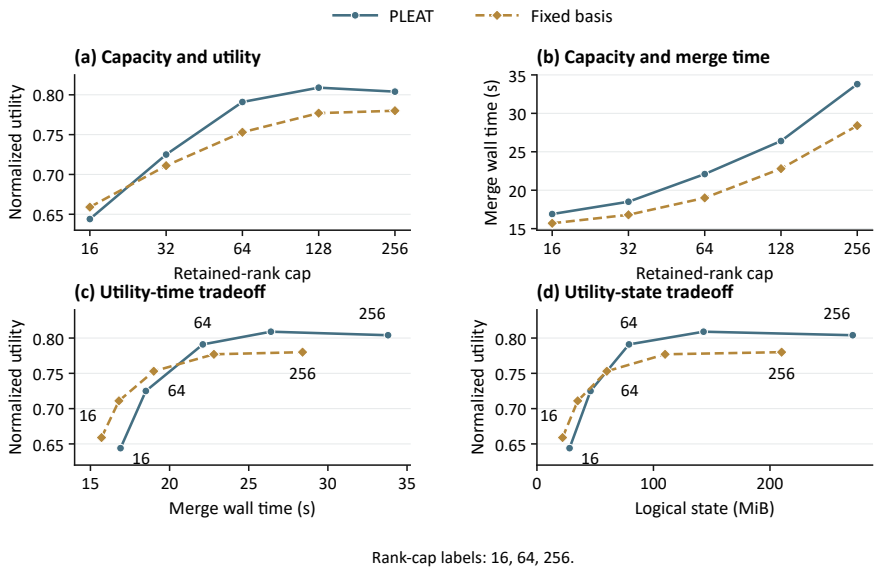


Figure 15. Capacity–cost trade-offs after eight arrivals. Labels mark rank caps; utility is expert-normalized, time is in seconds, and state is in MiB.

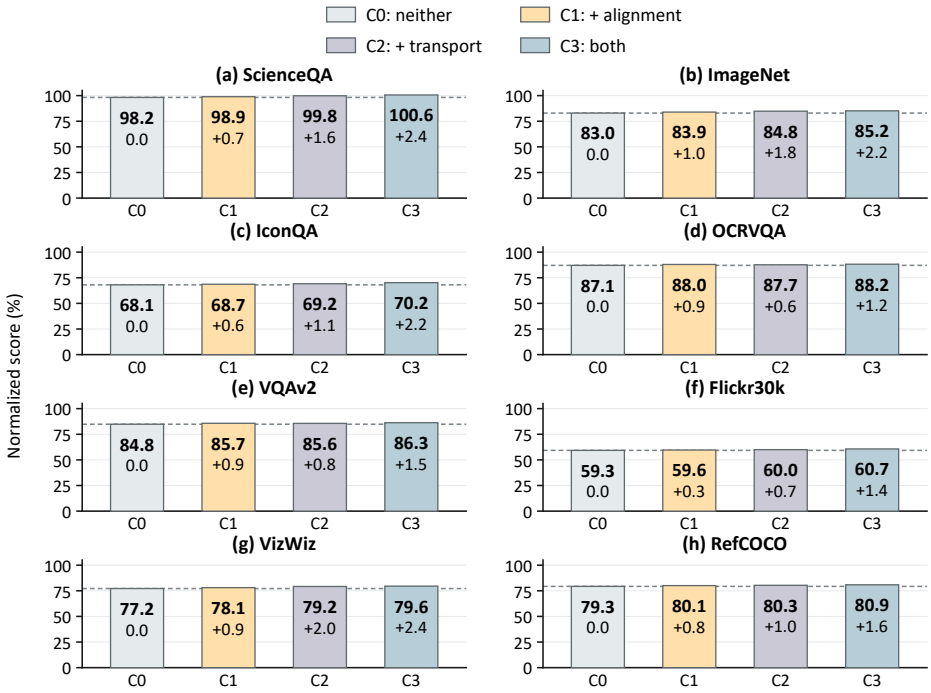


Figure 16. Component controls: C0 uses neither alignment, C1 incoming only, C2 history only, and C3 both. Scores are expert-normalized; annotations show changes from C0.

Read capacity together with its cost. The first two panels index configurations by the retained-rank cap. The lower panels place utility against time or state size for the same configurations. Following a method across these views links its utility to both resource quantities. A cap limits the available representation; the retained rank also depends on the span supplied by the arriving experts. Consequently, the horizontal cap value alone does not specify the tensor payload of an individual layer.

In Figure 16, scores use task-specific expert normalization multiplied by 100; bar annotations give changes from C0.

Separate component effects within each task. C0 includes neither operation, C1 adds incoming alignment, C2 adds historical transport, and C3 includes both. Each bar reports the normalized score and its change from C0. The dashed reference line marks C0 within that task. Reading the annotations together distinguishes the absolute score from the component difference. The normalization uses each task's own expert reference, so equal normalized changes need not imply equal changes in the native task metric.

M Rank and Output-Scale Landscapes

In Figure 6, columns show VQA, reasoning, and driving error for two streams. Warm yellow denotes lower values and blue denotes higher values, with a separate color scale for each panel. Markers distinguish reference, high-scale, and candidate settings; connecting lines guide the eye.

Rank and export scale. Figure 6 illustrates rank budget and output scale on a common plane. Each panel uses its own error range, so colors should be compared within a panel.

Reading a response surface. A horizontal slice fixes the rank budget and follows the output scale; a vertical slice fixes the scale and follows the budget. Contours group equal response values within each panel, making the direction of change visible without switching axes. The reference, high-scale, and candidate markers identify coordinates on that common plane. Their connecting segments indicate which settings are being compared; the contour field supplies the response between those coordinates in the displayed surface. Reading one task panel at a time preserves its error units, while the shared axis convention makes the setting changes comparable across panels. This complements the discrete rank–scale grid: the grid gives named operating points, and the landscape organizes the same two control variables spatially.