Robust and Efficient Continual Model Merging via Global Singular Subspace Separation and Restoration

Juncheol Shin[1] **Byeongho Yu**[2] **Semin Kim**[1] **Eunhyeok Park**[1,2]
[1]Graduate School of Artificial Intelligence
[2]Department of Computer Science & Engineering
POSTECH, South Korea
{jchshin,bhyu418,smkim25,eh.park}@postech.ac.kr

Abstract

Continual model merging incrementally integrates task-specific models without revisiting past models or data. However, repeated merging entangles past task components and distorts their singular geometry, making balanced task-wise allocation increasingly difficult. We propose **Global Singular Subspace Separation and Restoration (GS3R)**, which augments the merged update with a compact restoration state. At each merge, GS3R restores the retained singular geometry, separates task-wise subspaces, and globally reallocates the available rank across tasks fairly. To reduce merging-time memory overhead, we further introduce **Pre-merge Restoration and Flushing (PRF)**. Experiments on vision, NLP, and multimodal benchmarks show that GS3R provides a strong accuracy–efficiency trade-off.

1 Introduction

Recent advances in machine learning have enabled effective personalization of foundation models via task-specific fine-tuning, leading to the widespread availability of specialized models across domains. In this context, *model merging* has emerged as an attractive paradigm for constructing a single multi-task model from multiple task-specific models, enabling efficient reuse without additional retraining. In particular, prior methods based on singular value decomposition (SVD) or auxiliary optimization [1, 2, 3, 4, 5] achieve strong performance while incurring no inference-time overhead. However, standard model merging methods typically assume simultaneous access to all task-specific models, which is hard to satisfy in practice where models often arrive sequentially.

Recent studies have begun to address this challenge through *continual model merging (CMM)*, which incrementally integrates task-specific models without retraining or storing all previous models [6, 7, 8]. Despite their progress, treating this accumulated update as a single task creates two coupled problems. First, past task components become entangled, so the incoming task competes against all previous tasks as a single aggregate, leading to imbalanced capacity allocation. Second, repeated conflict-resolution operations progressively distort the singular subspaces of earlier tasks. Together, these effects cause *task balance collapse*, where information from early tasks is increasingly underrepresented as merging proceeds.

In this work, we propose *Global Singular Subspace Separation and Restoration (GS3R)*[1], a CMM framework built on a simple principle: the representation used for inference need not be the only information preserved for future merging. GS3R propagates the merged update together with a compact restoration state that preserves the singular geometry removed during orthogonalization. When a new task arrives, GS3R restores the retained pre-orthogonalization geometry, separates the recovered components into task-wise subspaces, and performs global rank allocation across all

[1]Our code is available at https://github.com/ewsn1593/GS3R.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

observed tasks before orthogonalizing them again for reconstruction. This restores task-wise access to past information without storing previous task models or explicit task-specific subspaces.

To make this process practical, we further introduce *Pre-merge Restoration and Flushing (PRF)*, which restores the accumulated subspace before loading the incoming task and immediately releases the restoration state. GS3R therefore introduces no inference-time overhead, while its rank budget provides a controllable trade-off between additional storage and merging accuracy.

Our contributions are summarized as follows:

- **Recoverable merging state.** GS3R preserves compact restoration information that recovers past singular geometry without storing previous task models.
- **Global task-wise merging.** Restored subspaces are separated by task, enabling balanced capacity allocation across all observed tasks.
- **Resource-efficient realization.** PRF and rank-controlled variants reduce memory and storage costs, with optional optimization providing further gains.

2 Related Work

2.1 Model Merging

Existing model merging methods can be broadly categorized into weight-space aggregation, conflict-aware task vector editing, optimization-based merging, and low-rank or SVD-based merging. A prominent line of work focuses on combining fine-tuned weights or their differences from the pre-trained model, while mitigating performance degradation caused by conflicts among multiple task updates [9, 10, 11, 12, 13, 14, 15, 16]. More recently, refined optimization-based strategies [5, 17] and approaches that handle core task components via SVD have been extensively studied [1, 2, 3, 4, 18]. Optimization-based methods can achieve strong merging quality, but often require iterative procedures or auxiliary objectives, which increase merging-time computation. In contrast, SVD-based approaches provide structured low-rank representations of task updates and are often efficient at inference time. However, these approaches rely on the availability of complete task-specific information during merging, which is often impractical in real-world scenarios.

2.2 Continual Model Merging

To enable model merging in more practical settings, such as sequential adaptation to new tasks or reduced storage overhead, CMM has been introduced. Closely related to continual learning [19, 20, 21, 22, 23], CMM aims to build a unified multi-task model by incrementally incorporating new tasks. Unlike standard continual learning, where a model is updated using task data or gradients, CMM operates directly in weight space and typically assumes no access to previous task data. In CMM, task-specific models arrive sequentially, and the merger must update a unified model without revisiting all previous task models or training data. This naturally leads to a core challenge: updates from later tasks tend to overshadow those from earlier ones, causing important information to be gradually lost. To mitigate this issue, methods such as OPCM [6], DOP [7], and NUFILT [8] reduce interference in sequential updates through SVD-based orthogonal projection, auxiliary optimization, null-space filtering, or their combinations. In contrast to prior methods that primarily resolve interference locally at each step, our method explicitly maintains balance across singular components accumulated throughout the continual process, while avoiding the need to store all previous task models.

3 Preliminaries and Challenges of Continual Model Merging

3.1 Problem Definition

Consider a pre-trained model with weights θ_{pre} and a set of task-specific fine-tuned models $\{\theta_1, \ldots, \theta_N\}$ obtained from N different tasks. Standard model merging assumes that these models are available simultaneously and seeks to construct a single model that performs well across all tasks without jointly retraining on their original data.

Definition 3.1 (Model Merging)**.** *Given task-specific models $\{\theta_1, \ldots, \theta_N\}$ and a shared pre-trained model θ_{pre}, model merging produces merged weights θ_m through a merge operator $\mathcal{M}$:*

$$\theta_m = \mathcal{M}(\theta_1, \ldots, \theta_N; \theta_{\text{pre}}). \tag{1}$$

A common representation is the *task vector* $\tau_i = \theta_i - \theta_{\text{pre}}$, which captures the parameter displacement induced by fine-tuning. For example, Task Arithmetic [24] simply aggregates task vectors:

$$\theta_m = \theta_{\text{pre}} + \lambda \sum_{i=1}^{N} \tau_i, \tag{2}$$

where λ is a scaling coefficient. Such simple aggregation can work surprisingly well when task updates are sufficiently compatible, but interference between tasks often calls for more structured merging.

Several recent methods therefore operate on matrix-valued task updates in their singular subspaces [2, 3, 4]. For a task matrix τ_i, the decomposition $\tau_i = U_i \Sigma_i V_i^\top$ represents the update as a collection of singular components, allowing a merge operator to identify important directions and resolve conflicts at the component level. In particular, SVD-based approaches often concatenate singular vectors from multiple tasks and orthogonalize or project them before reconstructing the merged update [3, 4]. When all tasks are available simultaneously, such conflict resolution is performed once. In a continual setting, however, its output becomes the input to subsequent merges, fundamentally changing the problem.

Continual model merging (CMM) considers this sequential regime, where task-specific models arrive one at a time and past models cannot be revisited.

Definition 3.2 (Continual Model Merging)**.** *At step t, CMM combines the previously merged model θ_m^{t-1} with the newly arrived task model θ_t and optionally maintains a compact auxiliary state s^{t-1}:*

$$(\theta_m^t, s^t) = \tilde{\mathcal{M}}(\theta_m^{t-1}, s^{t-1}, \theta_t; \theta_{\text{pre}}), \qquad t \geq 2. \tag{3}$$

Previous task-specific models and their training data are not revisited.

This sequential setting fundamentally differs from standard model merging. At step t, the merger has access only to the incoming task and the previously merged model, rather than the individual models or data from tasks $1, \ldots, t-1$. Past information must therefore survive through a representation that is repeatedly reused and modified as new tasks arrive.

3.2 Task Balance Collapse

Over time, this can progressively bias the merged representation toward later tasks, a phenomenon we refer to as **Task Balance Collapse**. We identify two closely related causes: entanglement of past task information and accumulated distortion from repeated conflict resolution.

Entanglement of past task information. In standard model merging, all task updates are available together, allowing the merge operator to compare and allocate capacity across tasks directly. In CMM, however, tasks $1, \ldots, t-1$ are compressed into the previous merged update τ_m^{t-1}, where their individual identities and singular components are no longer directly accessible. The next merge must therefore treat this entangled representation as a single operand against the newly arrived task. As the number of tasks grows, this mismatch between the objects being merged and the underlying task structure makes balanced, task-wise capacity allocation increasingly difficult.

Accumulated distortion from repeated conflict resolution. This loss of task-wise access is further compounded by the transformations used to reduce interference. SVD-based merging commonly orthogonalizes or projects singular subspaces before reconstructing the merged update. While standard merging applies such conflict resolution once to the full set of tasks, CMM repeatedly applies it to a representation that has already been modified by previous merges. Consequently, directions inherited from earlier tasks undergo more transformations and gradually drift from their original geometry. At the same time, joint orthogonalization obscures the geometric relations that distinguish singular components from different tasks. Thus, the representation becomes not only increasingly distorted, but also harder to disentangle at the next merge.

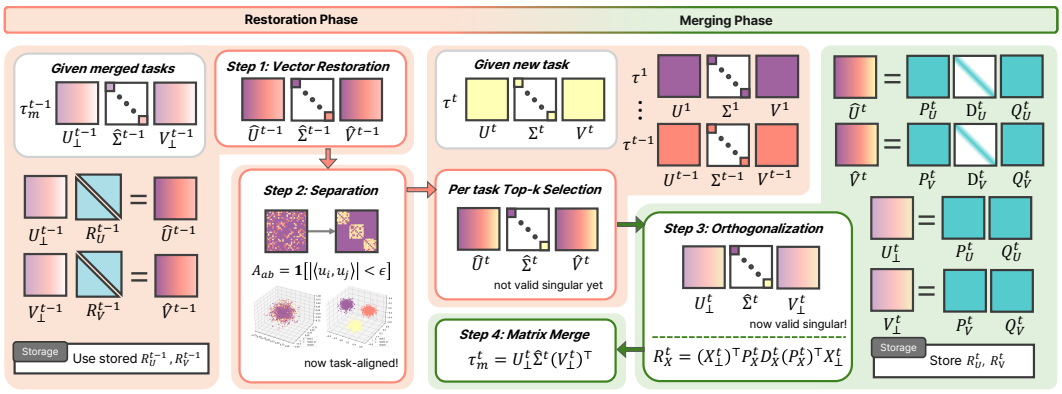


Figure 1: Overall process of GS3R. Each loop restores the task-wise geometry from the previous state, performs global task-wise merging, and stores restoration metadata for the next step.

Together, these effects reveal a central requirement of CMM: the merger must preserve not only an effective model for the current tasks, but also enough structure to recover the task-wise geometry needed by future merges.

4 Proposed Method: GS3R

Building on this requirement, we propose **Global Singular Subspace Separation and Restoration (GS3R)**, which reformulates continual SVD-based merging by changing the state propagated across merge steps. Rather than retaining only the merged update, GS3R propagates a recoverable state

$$\mathcal{S}^t = \left(\tau_m^t,\, R_U^t,\, R_V^t\right), \qquad \tau_m^t = U_\perp^t \hat{\Sigma}^t (V_\perp^t)^\top, \tag{4}$$

where τ_m^t is the update used for inference, while R_U^t and R_V^t are compact metadata that preserve the geometry needed by the next merge. At each step, GS3R restores this geometry, recovers task-wise subspaces, allocates the rank budget across all observed tasks, and re-orthogonalizes the selected components for conflict-reduced reconstruction. The resulting update is used for inference, while the restoration state is carried forward to the next step. All operations are applied independently to each matrix-valued parameter, and we omit the layer index for clarity.

At step t, let $\tau^t \in \mathbb{R}^{m\times n}$ denote the incoming task matrix and $r \le d = \min(m,n)$ the rank budget. We first decompose

$$\tau^t = U_t \Sigma_t V_t^\top, \tag{5}$$

exposing the incoming task as singular components to be merged with the recovered components of previous tasks.

Step 1: Vector Restoration

At the beginning of step t, the previous merged update has the orthogonalized form

$$\tau_m^{t-1} = U_\perp^{t-1} \hat{\Sigma}^{t-1} (V_\perp^{t-1})^\top. \tag{6}$$

$U_\perp^{t-1}$ and $V_\perp^{t-1}$ are valid orthogonal bases but no longer the original concatenated singular-vector bases used before the previous orthogonalization. In Figure 1, this corresponds to the state stored under "Given merged tasks." GS3R therefore first restores the pre-orthogonalized bases using the restoration matrices stored at the previous step:

$$\hat{U}^{t-1} = U_\perp^{t-1} R_U^{t-1}, \qquad \hat{V}^{t-1} = V_\perp^{t-1} R_V^{t-1}. \tag{7}$$

This reverses the stretch removed by the previous orthogonalization, so that the columns of $\hat{U}^{t-1}$ and $\hat{V}^{t-1}$ again carry the retained singular directions in pre-orthogonalized form. Crucially, this restoration makes task-wise separation possible by recovering the orthogonality structure needed to identify which retained components originated from the same task.

Step 2: Global Singular Subspace Separation and Top-k Selection

With this geometry restored, GS3R next recovers the hidden task-wise structure of the accumulated update. The restored $\hat{U}^{t-1}$ and $\hat{V}^{t-1}$ contain singular directions from tasks $1, \ldots, t-1$, but with interleaved columns whose task identities are not stored explicitly. GS3R infers their task-wise grouping from a basic SVD property: singular vectors from the same task are mutually orthogonal by construction, whereas those from different tasks need not be. Mutual orthogonality therefore provides a natural cue for recovering task groups. Visualization and further analysis of inter- and intra-task orthogonality are provided in Appendix C.1.

We build an affinity graph over the columns of $\hat{U}^{t-1}$. For two restored left singular vectors $\hat{u}_a$ and $\hat{u}_b$, we define $A_{ab} = \mathbf{1}\left[|\langle \hat{u}_a, \hat{u}_b \rangle| < \epsilon\right]$ with a small tolerance ϵ. We then apply spectral clustering [25] and determine the cluster count using the eigengap heuristic. Details and empirical validation of the spectral clustering procedure are provided in Appendix C.2. The cluster assignment is propagated to the paired columns of $\hat{V}^{t-1}$ and to the corresponding entries of $\hat{\Sigma}^{t-1}$, yielding task-wise components:

$$\{U_i, \Sigma_i, V_i\}_{i=1}^{t-1}, \qquad \hat{U}^{t-1}\hat{\Sigma}^{t-1}(\hat{V}^{t-1})^\top = \sum_{i=1}^{t-1} U_i \Sigma_i V_i^\top. \tag{8}$$

The next merge therefore operates on task-wise singular subspaces rather than on "one previous merged task plus one new task," removing the granularity mismatch identified in Section 3.2.

Combined with Eq. (5), we now have disentangled singular components for all tasks $i = 1, \ldots, t$. GS3R then performs global, task-balanced selection under the rank budget r. With $k = \lfloor r/t \rfloor$ and $\rho = r - tk$, we take the top-k components from every task and assign the remaining ρ slots to the tasks with the largest next singular values. Letting $\mathcal{I}_i$ denote the selected indices for task i, the selected components are

$$\begin{aligned} \hat{U}^t &= [\, U_1^{(\mathcal{I}_1)}, \ldots, U_t^{(\mathcal{I}_t)} \,], \qquad \hat{\Sigma}^t = \operatorname{diag}\left(\Sigma_1^{(\mathcal{I}_1)}, \ldots, \Sigma_t^{(\mathcal{I}_t)}\right), \\ \hat{V}^t &= [\, V_1^{(\mathcal{I}_1)}, \ldots, V_t^{(\mathcal{I}_t)} \,]. \end{aligned} \tag{9}$$

This corresponds to the "Per-task Top-k Selection" block in Figure 1. Capacity is thus allocated at the task granularity: singular values prioritize directions within each task, while task-wise separation prevents the incoming task from competing against the entire past as a single aggregate.

Step 3: Orthogonalization and Restoration-State Update

The matrices $\hat{U}^t$ and $\hat{V}^t$ are not yet valid singular vector bases since vectors from different tasks may be correlated. Direct use of $\hat{U}^t\hat{\Sigma}^t(\hat{V}^t)^\top$ therefore still carries cross-task interference. Following SVD-based conflict-resolution methods [3, 4], GS3R orthogonalizes the concatenated bases via compact SVDs,

$$\hat{U}^t = P_U^t D_U^t (Q_U^t)^\top, \qquad \hat{V}^t = P_V^t D_V^t (Q_V^t)^\top, \tag{10}$$

and replaces them with their polar factors:

$$U_\perp^t = P_U^t (Q_U^t)^\top, \qquad V_\perp^t = P_V^t (Q_V^t)^\top. \tag{11}$$

The polar factor is the closest orthogonal basis to the original concatenation in Frobenius norm. Thus, Eq. (11) can be interpreted as the least-distorting orthogonalization of the selected task subspaces, resolving cross-task correlations while keeping the selected singular values $\hat{\Sigma}^t$ unchanged.

However, the same orthogonalization that resolves cross-task interference would recreate the accumulated distortion identified in Section 3.2 if the removed geometry were discarded. GS3R therefore retains the discarded stretch as compact restoration matrices. For $X \in \{U, V\}$,

$$R_X^t = Q_X^t D_X^t (Q_X^t)^\top. \tag{12}$$

A direct substitution gives $\hat{X}^t = X_\perp^t R_X^t$, so R_U^t and R_V^t make the pre-orthogonalized bases exactly recoverable at the next merge. The equivalent construction used in Figure 1 and the detailed proof are provided in Appendix A.

The restoration matrices are compact. If GS3R keeps rank r, then $R_U^t, R_V^t \in \mathbb{R}^{r \times r}$. They are symmetric positive semidefinite, requiring only the upper triangular entries to be stored. The resulting storage overhead is $\mathcal{O}(r^2)$ per layer, independent of the number of merged tasks. Importantly, these matrices are used only during merging and introduce no inference-time overhead.

Step 4: Merged Matrix Construction

Finally, GS3R constructs the merged update at step t using the orthogonalized bases and selected singular values:

$$\tau_m^t = U_\perp^t \hat{\Sigma}^t (V_\perp^t)^\top. \tag{13}$$

Together with the newly computed restoration matrices, this returns the algorithm to the state in Eq. (4). The merged update τ_m^t is used for inference, while (R_U^t, R_V^t) are retained for the next merge. This closes the continual loop: *restore* $\rightarrow$ *separate* $\rightarrow$ *select* $\rightarrow$ *orthogonalize* $\rightarrow$ *merge and store*.

4.1 Pre-merge Restoration and Flushing

The restoration matrices are compact, but peak memory can still become a practical bottleneck if the system loads the previous merged model, the new task model, and the restoration matrices simultaneously. We therefore introduce **Pre-merge Restoration and Flushing (PRF)**, an implementation strategy that changes only the execution order, not the mathematical output.

PRF exploits the fact that R_U^{t-1} and R_V^{t-1} are needed only for Eq. (7). When a merge request arrives, GS3R first loads the previous merged model and the restoration matrices, restores $\hat{U}^{t-1}$ and $\hat{V}^{t-1}$, and immediately flushes the restoration matrices from memory. Only after this restoration phase does it load the new task model. The restoration matrices and new task weights therefore never need to coexist in memory, preserving the benefit of restoration while limiting the peak memory overhead.

4.2 Resource-Aware Variants

Truncated GS3R. The rank budget r controls the storage–accuracy trade-off. Full GS3R sets $r = d$, while truncated GS3R uses $r < d$ and retains only the most informative singular directions. This reduces additional storage from $\mathcal{O}(d^2)$ to $\mathcal{O}(r^2)$ per layer, while leaving the rest of the pipeline unchanged within the retained rank-r subspace.

Auxiliary optimization. GS3R is data-free and optimization-free by default, but can also serve as a structured initialization for optimization-based merging when extra compute is available. Since Step 2 recovers task-wise components in factorized form $\tau_i = U_i \Sigma_i V_i^\top$, we initialize the optional optimization from the pre-orthogonalized reconstruction $\tilde{\tau}^t = \hat{U}^t \hat{\Sigma}^t (\hat{V}^t)^\top$, which retains the recovered task geometry, rather than Eq. (13), and minimize the auxiliary objective from [5]:

$$\mathcal{L}(\tau) = \sum_i \frac{\left\| \left(\tau - U_i \Sigma_i V_i^\top\right) \left(U_i \Sigma_i V_i^\top\right)^\top \right\|_F^2}{\left\| U_i \Sigma_i V_i^\top \right\|_F^2}. \tag{14}$$

Letting the optimized update be τ_{opt}^t, we store the residual

$$\Delta^t = \tau_{\text{opt}}^t - \tau_m^t, \tag{15}$$

to preserve the GS3R loop invariant. Before the next merge, GS3R subtracts Δ^t to recover τ_m^t in Eq. (13) and then applies restoration as usual. Auxiliary optimization therefore remains a modular add-on without changing the core GS3R pipeline.

5 Experimental Results

We evaluate GS3R from three perspectives: whether recovering task-wise singular structure improves continual merging performance, how the rank-controlled restoration state trades accuracy for storage and merging cost, and how each component contributes to the overall framework. We consider vision, NLP, and multimodal benchmarks. For vision, we use CLIP [26]-based ViT [27] models fine-tuned on 20 downstream tasks and, following prior work [6, 7], evaluate continual merging over suites of 8, 14, and 20 tasks. For NLP, we use Flan-T5-Base [28] and continually merge models fine-tuned on 8 GLUE tasks [29]. To assess scalability and generality beyond these settings, we further merge LLaVA-1.5-7B [30] models on four multimodal benchmarks: IconQA [31], ImageNet

Table 1: Comparison of continual model merging performance for ViT models on vision tasks. We report the mean and standard deviation of accuracy (ACC) and backward transfer (BWT). STO denotes the relative storage overhead against the merged model. The horizontal line separates optimization-free (top) and optimization-based (bottom) methods. Best results within each group are indicated in bold, and second-best results are underlined. "C." stands for Continual.

	Method	ViT-B/32			ViT-B/16			ViT-L/14		
		8 tasks	14 tasks	20 tasks	8 tasks	14 tasks	20 tasks	8 tasks	14 tasks	20 tasks
	Pre-Trained	48.1	56.9	55.6	55.4	62.0	59.8	64.9	69.1	65.6
	Fine-Tuned	90.4	89.3	89.8	92.4	91.3	91.6	94.3	93.4	93.5
	C. Fine-Tuned	79.8	67.4	62.6	82.9	72.2	68.2	90.0	70.9	77.7
ACC↑	Average (SWA)	66.3±0.0	65.4±0.0	61.1±0.0	72.3±0.0	69.7±0.0	64.8±0.0	80.0±0.0	77.5±0.0	71.1±0.0
	C. Task Arithmetic	67.5±0.0	66.5±0.0	60.6±0.0	77.1±0.0	70.9±0.0	64.2±0.0	82.1±0.0	77.9±0.0	70.3±0.0
	C. Ties-Merging	49.0±10.2	66.2±0.6	59.9±0.7	66.8±3.7	70.5±0.8	63.0±1.6	64.3±7.0	78.0±0.6	68.3±0.9
	OPCM	75.5±0.5	71.9±0.3	65.7±0.2	81.8±0.3	77.1±0.5	70.3±0.2	87.0±0.4	83.5±0.2	76.0±0.2
	GS3R ($r = 256$)	81.8±0.2	76.5±0.3	70.9±0.5	86.7±0.1	80.3±0.2	76.1±0.5	89.4±0.4	86.4±0.1	81.7±0.3
	GS3R	**82.7**±0.2	**78.7**±0.3	**72.7**±0.6	**86.9**±0.1	**81.8**±0.3	**77.7**±0.5	**90.3**±0.5	**88.0**±0.3	**82.7**±0.7
	DOP	78.3±1.6	73.0±1.2	68.5±1.6	84.5±0.8	76.2±1.2	72.2±1.8	88.3±0.7	85.5±0.8	80.7±0.9
	NUFILT	83.6±0.2	78.0±0.2	71.0±0.9	87.3±0.1	83.1±0.3	78.1±0.9	91.6±0.1	89.2±0.1	84.7±0.8
	GS3R + Opt.	**84.7**±0.3	**80.0**±0.4	**73.2**±1.4	**89.2**±0.1	**83.8**±0.3	**79.4**±0.7	**92.2**±0.3	**89.8**±0.3	**85.2**±1.5
BWT↑	Average (SWA)	-11.5±2.2	-8.0±1.3	-7.1±2.1	-9.7±1.5	-7.1±1.4	-7.3±1.7	-7.3±1.4	-5.8±1.0	-6.4±1.5
	C. Task Arithmetic	-9.6±1.5	-1.3±0.3	-3.4±0.4	-4.2±1.0	-1.3±0.4	-3.6±0.4	-7.1±0.8	-1.8±0.3	-3.3±0.3
	C. Ties-Merging	-15.3±8.0	**1.9**±0.6	**-1.5**±0.7	-5.5±0.4	**1.4**±0.7	**-1.5**±1.2	-13.0±5.7	**1.1**±0.4	**-2.9**±1.0
	OPCM	-6.3±1.1	-6.0±1.0	-7.8±1.5	-4.8±0.7	-5.1±1.4	-6.3±2.2	-2.6±1.0	-4.3±0.7	-6.5±1.8
	GS3R ($r = 256$)	-4.9±1.2	-6.4±1.2	-9.3±1.8	-3.1±0.6	-5.7±1.1	-7.5±1.3	-2.7±0.9	-3.5±0.8	-6.1±1.0
	GS3R	**-4.3**±0.9	-5.2±0.9	-8.8±1.7	**-3.0**±0.5	-4.6±1.1	-6.9±1.4	**-1.9**±1.0	-2.7±0.8	-6.3±1.5
	DOP	-6.1±2.3	-8.8±2.1	-12.5±2.8	-3.9±1.0	-7.5±2.2	-11.4±2.8	-3.2±1.2	-4.6±1.0	-8.4±1.4
	NUFILT	**-2.7**±0.7	-5.7±0.9	**-8.9**±2.3	**-1.6**±0.5	**-3.5**±0.6	**-7.1**±1.9	**-1.1**±0.3	**-2.0**±0.3	**-4.6**±0.7
	GS3R + Opt.	-3.4±0.8	**-5.0**±1.0	-10.3±2.0	-2.0±0.3	-4.1±0.7	-7.6±1.2	-1.2±0.5	-2.1±0.6	-7.3±1.2
STO↓	GS3R ($r = 256$)	1.05× (+ 19.0 MB)			1.06× (+ 19.1 MB)			1.03× (+ 38.2 MB)		
	GS3R	1.49× (+ 170.1 MB)			1.50× (+ 170.2 MB)			1.50× (+ 604.8 MB)		
	GS3R + Opt.	2.46× (+ 510.0 MB)			2.49× (+ 510.6 MB)			2.50× (+ 1813.8 MB)		

captioning [32], ScienceQA [33], and VizWiz-Caption [34], which span diagram reasoning, natural image captioning, scientific question answering, and real-world low-vision scenarios.

We compare against Simple Weight Averaging (SWA) [35], Continual Task Arithmetic [24], Continual TIES-Merging [9], and three recent CMM methods: OPCM [6], DOP [7], and NUFILT [8]. All experiments use publicly available checkpoints [36, 37]. Results are averaged over 10 random task orders for vision and NLP benchmarks and over 6 task orders (alphabetical and 5 random) for multimodal benchmarks. Vision and NLP experiments run on an NVIDIA RTX 3090 GPU, and multimodal experiments on an NVIDIA A100-SXM4-80GB GPU. Further details on experimental settings are provided in Appendix B.

5.1 Continual Model Merging on Vision Tasks

For vision tasks, we evaluate GS3R under three settings: full-rank GS3R, truncated GS3R with $r = 256$, and full-rank GS3R with auxiliary optimization. Table 1 summarizes the results across three ViT backbones and task sequences of increasing length.

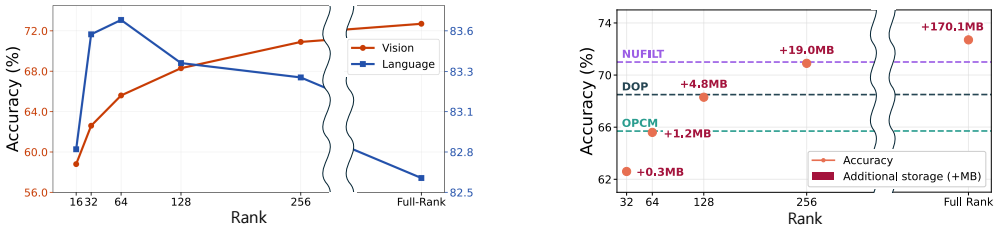
Without auxiliary optimization, GS3R consistently outperforms OPCM and DOP across all backbone and task-count settings. Its advantage is particularly visible on ViT-B/32 as the continual sequence grows: GS3R improves over NUFILT from 78.0 to 78.7 on 14 tasks and from 71.0 to 72.7 on 20 tasks, consistent with its goal of preserving balanced task-wise capacity as more tasks accumulate. Truncating the retained subspace to $r = 256$ preserves most of this performance while substantially reducing the auxiliary storage. When additional computation is available, auxiliary optimization further improves GS3R and achieves the highest accuracy across all nine vision settings. These results show that the recovered task-wise subspaces provide a strong optimization-free merging state while remaining compatible with further refinement.

5.2 Continual Model Merging on NLP Tasks

Table 2 evaluates GS3R on Flan-T5-Base continually merged over eight GLUE tasks. With a rank budget of $r = 64$, GS3R reaches an average accuracy of 83.7 without any optimization and achieves

Table 2: Continual model merging results for the Flan-T5-Base model on GLUE tasks. We report the mean and standard deviation of accuracy and backward transfer.

Method	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST2	STSB	ACC(↑)	BWT(↑)	STO(↓)
Pre-Trained	69.1	56.5	76.2	88.4	82.1	80.1	91.2	62.2	75.7	-	-
Fine-Tuned	75.0	83.4	87.5	91.5	85.4	85.9	93.6	88.7	86.4	-	-
OPCM	69.7	72.9	78.8	90.2	83.8	82.2	92.3	74.7	80.6±0.3	-2.1±0.6	1.00×
GS3R	70.5	79.5	78.8	90.6	**84.3**	**83.0**	92.9	81.3	82.6±0.1	-1.9±0.6	1.525×
GS3R ($r = 64$)	**71.7**	**81.1**	**80.0**	**90.7**	84.2	82.2	**93.1**	**86.4**	**83.7**±0.1	**-1.2**±0.4	1.004×
DOP	72.6	80.9	**81.0**	90.4	83.4	81.0	92.9	86.1	83.6±0.1	**-1.3**±0.3	1.00×
NUFILT	**72.7**	**83.0**	80.3	**90.9**	**84.0**	80.6	**93.3**	**86.9**	**84.0**±0.2	-1.4±0.5	1.00×
GS3R ($r = 64$) + Opt.	72.0	**83.0**	80.5	90.8	**84.0**	**82.0**	93.2	86.4	**84.0**±0.1	-1.9±0.2	1.804×



(a) Accuracy on Vision and NLP tasks

(b) Storage trade-off

Figure 2: Accuracy and storage trade-off with varying rank r. (a) shows accuracy variation with varying r on vision and NLP tasks. (b) shows that combining GS3R with truncated SVD yields a favorable trade-off. $r = 256$ preserves strong performance while requiring only 0.05× more storage.

the best BWT of -1.2 among all methods. This surpasses OPCM by 3.1 points and matches DOP while coming within 0.3 of NUFILT, even though both rely on auxiliary optimization. The low BWT is notable because it indicates that the merged model retains earlier tasks almost intact as new ones are added, which is precisely the property the restoration state is designed to preserve. Adding auxiliary optimization raises the average accuracy to 84.0, on par with NUFILT. This demonstrates that the same recoverable state formulation extends beyond vision models without a task-specific redesign.

Interestingly, the truncated variant performs better than full-rank GS3R on GLUE, unlike the monotonic trend observed on the vision benchmark. Figure 2(a) shows that NLP performance peaks at an intermediate rank rather than at full rank. Our analysis in Appendix C.3 attributes this behavior to the substantially lower effective rank of several Flan-T5 task matrices. Once the informative directions are already retained, additional components contribute little to the task while still participating in joint orthogonalization, where they can increase the distortion introduced during merging. Truncation therefore serves not only as a storage mechanism, but also as an effective filter when the task updates are intrinsically low-rank.

5.3 Continual Model Merging on Multimodal Tasks with Large-Scale Models

Table 3 reports results for continually merging LLaVA-1.5-7B over the four multimodal tasks. As in the vision experiments, we evaluate full-rank GS3R, truncated GS3R with $r = 256$, and full-rank GS3R with auxiliary optimization. Compared to the previous benchmarks, this setting involves a billion-scale backbone and open-ended generation tasks rather than classification. Nonetheless, the same recoverable-state formulation transfers without modification.

Under the original scorers, all GS3R variants outperform every baseline, with optimization-free GS3R ($r = 256$) reaching 69.5 against 66.6 for NUFILT. However, part of this margin stems from IconQA, the only task scored by exact string match, where merged models often append extra tokens to the correct answer. We therefore also report ACC†, which scores IconQA with a format-tolerant rule for all methods and thus measures answer correctness more directly (Appendix C.4). Under this protocol, GS3R without any optimization attains 72.1, within 0.6 of NUFILT, and truncating to $r = 256$ yields 72.7, matching NUFILT with only 0.2% additional storage and a standard deviation of just 0.1 across task orders. With auxiliary optimization, GS3R reaches 73.5, surpassing NUFILT by 0.8 points and attaining the best score among optimization-based methods on all four tasks. GS3R

Table 3: Continual model merging results of LLaVA-1.5-7B on multimodal tasks. ACC† scores IconQA with a format-tolerant rule applied identically to all methods, with the corresponding IconQA accuracy in parentheses. Detailed explanations are provided in Appendix C.4.

Method	IconQA	Image	SciQA	VizWiz	ACC(↑)	ACC†(↑)	STO(↓)
Pre-Trained	14.1	40.9	61.7	41.0	39.4	–	–
Fine-Tuned	75.5	96.0	83.7	64.8	80.0	–	–
C. Task Arithmetic	55.7 (55.9)	75.4	77.2	46.7	63.8±0.0	63.8±0.0	1.00×
C. Ties-Merging	**57.6** (63.7)	84.1	79.8	44.6	66.5±3.9	68.1±1.6	1.00×
OPCM	54.6 (63.1)	83.1	80.9	45.6	66.0±0.7	68.2±0.8	1.00×
GS3R (r=256)	56.2 (69.3)	**93.8**	80.9	47.0	**69.5**±0.2	**72.7**±0.1	1.002×
GS3R	56.9 (68.2)	90.6	**82.1**	**47.4**	69.2±0.8	72.1±0.6	1.535×
NUFILT	45.3 (69.6)	94.0	80.8	46.3	66.6±5.9	72.7±1.0	1.00×
GS3R + Opt.	**58.7** (71.2)	**94.2**	**81.7**	**46.7**	**70.3**±1.1	**73.5**±1.0	2.455×

therefore matches or surpasses NUFILT under both protocols, without any gradient computation in the optimization-free case. Overall, the recovered task-wise singular structure remains an effective and scalable merging state beyond vision and NLP, extending to large-scale multimodal models.

5.4 Storage Accuracy Trade-off

The rank budget r directly controls the size of the recoverable state carried between merge steps. We therefore examine how aggressively this state can be truncated while preserving merging quality. Figure 2(b) reports the resulting accuracy–storage trade-off for ViT-B/32 over 20 vision tasks.

Full-rank GS3R requires 170.1 MB of additional storage and achieves 72.7% accuracy. Reducing the budget to $r = 256$ lowers the additional storage to only 19.0 MB, an $8.9\times$ reduction, while retaining 70.9% accuracy, a drop of 1.8 points. Thus, most of the benefit of the recoverable state can be retained with only a small fraction of its full-rank storage. This makes the rank budget a direct and controllable mechanism for trading merging accuracy against deployment storage.

5.5 Merging Overhead Comparison

A useful continual merger must also keep the cost of each update practical. Table 4 therefore compares the final accuracy after merging 20 tasks on ViT-B/32 together with the peak GPU memory and elapsed time of a single merge, measured on an NVIDIA RTX 3090 GPU. With PRF, GS3R reaches 72.7% accuracy using 180.0 MB of peak GPU memory and 17.6 seconds per merge. This is only a modest increase over the lightweight OPCM baseline, while providing a 7.0 point accuracy gain.

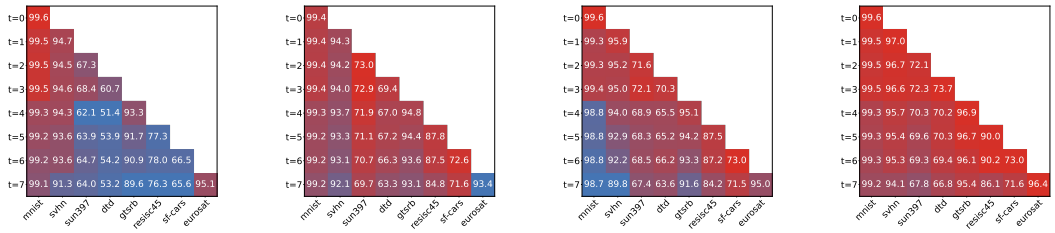
Compared with the stronger NUFILT baseline, GS3R improves accuracy from 71.0% to 72.7%, while reducing peak GPU memory from 2449.3 MB to 180.0 MB and merge time from 26.8 to 17.6 seconds. These results show that the additional restoration state does not translate into a large runtime or memory burden when combined with PRF. Auxiliary optimization can further raise the accuracy to 73.2%, at the expected cost of higher memory consumption and longer merging time, providing an additional accuracy–compute operating point when resources permit.

Method	Accuracy (%)	GPU Mem. (MB)	Time (s)
OPCM	65.7±0.2	134.2	14.6
DOP	68.5±1.6	259.4	100.8
NUFILT	71.0±0.9	2449.3	26.8
GS3R	72.7±0.6	180.0	17.6
GS3R + Opt.	73.2±1.4	584.1	60.6

Table 4: Merging overhead comparison on ViT-B/32, measured on an NVIDIA RTX 3090 GPU. Our method achieves high accuracy with low latency and minimal GPU memory overhead.

G	R	T	O	ViT-B/32			
				8 tasks	14 tasks	20 tasks	+ Storage
✗	✗	✗	✗	67.8	64.9	55.6	0 MB
✓	✗	✗	✗	67.8	65.0	55.6	0 MB
✗	✓	✗	✗	71.8	70.2	61.7	170.1 MB
✓	✓	✗	✗	82.7	78.7	72.7	170.1 MB
✓	✓	✓	✗	81.8	76.5	70.9	19.0 MB
✓	✓	✗	✓	84.7	80.0	73.2	510.0 MB

Table 5: Ablation study on components of GS3R. **G**: Global singular subspace separation, **R**: singular vector Restoration, **T**: Truncated SVD, **O**: auxiliary Optimization.



(a) DOP (b) NUFILT (c) GS3R (d) GS3R + Opt.

Figure 3: Accuracy trajectories during continual merging of ViT-B/32 over 8 vision tasks. Rows correspond to merge steps and columns to tasks, showing how previously merged tasks evolve as new tasks arrive. Colors are independently normalized per task.

5.6 Ablation Study

We ablate the key components of GS3R on ViT-B/32 to understand how restoration and task-wise separation interact. The results are summarized in Table 5. Applying global singular subspace separation alone provides essentially no improvement over the baseline, because the accumulated bases remain orthogonalized and no longer expose the task-wise geometry required for reliable separation. In contrast, restoration alone already yields substantial gains, improving accuracy by 4.0, 5.3, and 6.1 points on the 8-, 14-, and 20-task settings, respectively. However, without subsequent task-wise separation and balanced rank allocation, the restored components still compete as an entangled aggregate.

Combining restoration with global separation produces a much larger improvement, reaching 82.7%, 78.7%, and 72.7% across the three settings. This confirms the intended dependency between the two components: restoration recovers the geometric structure needed to identify retained task components, while separation converts that structure into task-wise capacity allocation. The remaining variants characterize the resource trade-off of this core pipeline. Truncating the retained subspace to $r = 256$ reduces the storage from 170.1 MB to 19.0 MB while preserving most of the accuracy, whereas auxiliary optimization further improves performance at the cost of additional storage and computation.

5.7 Qualitative Analysis of Merging Behavior

To visualize how task balance evolves during continual merging, we track the accuracy of each observed task after every merge step on the 8-task ViT-B/32 setting. Figure 3 reveals a clear difference in merging dynamics. DOP exhibits pronounced degradation on several earlier tasks as new tasks are incorporated, illustrating the task balance collapse discussed in Section 3.2. In contrast, NUFILT and GS3R both mitigate this degradation and better preserve the balance and accuracy of previously merged tasks. This behavior is consistent with GS3R explicitly recovering and reallocating capacity across retained task-wise components at each merge step. With auxiliary optimization, GS3R further improves the final task accuracies while maintaining this balanced merging behavior.

Additional t-SNE [38] visualizations in Appendix C.8 provide a complementary representation-level view of the merged models. For a task merged early in a 20-task sequence, GS3R preserves well-separated and class-consistent clusters, whereas prior methods lose much of this structure.

6 Conclusion

We identified two coupled causes of task balance collapse in continual model merging: the entanglement of past task components and the accumulated distortion of their singular geometry. To address both, we proposed *Global Singular Subspace Separation and Restoration (GS3R)*, which preserves a compact recoverable state alongside the merged update. This enables past task subspaces to be restored, separated, and globally reallocated at each merge without storing previous task models. Together with rank-controlled truncation and Pre-merge Restoration and Flushing (PRF), GS3R provides a practical accuracy–resource trade-off across vision, NLP, and multimodal benchmarks. More broadly, our results highlight that continual merging should preserve not only the current merged model, but also the structure required for future merges.

Acknowledgments and Disclosure of Funding

This work was supported by IITP and NRF grant funded by the Korea government (MSIT) (RS-2019-II191906, RS-2024-00415602, RS-2026-25490269).

References

[1] Aniello Panariello, Daniel Marczak, Simone Magistri, Angelo Porrello, Bartłomiej Twardowski, Andrew D Bagdanov, Simone Calderara, and Joost van de Weijer. Accurate and efficient low-rank model merging in core space. *arXiv preprint arXiv:2509.17786*, 2025.

[2] 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, pages 4501–4519, 2025.

[3] Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18695–18705, 2025.

[4] 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 *Forty-second International Conference on Machine Learning*, 2025.

[5] 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 *Forty-second International Conference on Machine Learning*, 2025.

[6] 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

[8] 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. In *The Fourteenth International Conference on Learning Representations*, 2026.

[9] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 7093–7115. Curran Associates, Inc., 2023.

[10] Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. EMR-Merging: Tuning-free high-performance model merging. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 122741–122769. Curran Associates, Inc., 2024. doi: 10.52202/079017-3900.

[11] Wenju Sun, Qingyong Li, Yangliao Geng, and Boyang Li. CAT merging: A training-free approach for resolving conflicts in model merging. In *Forty-second International Conference on Machine Learning*, 2025.

[12] Shenghe Zheng and Hongzhi Wang. Free-merging: Fourier transform for efficient model merging. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3863–3873, 2025.

[13] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023.

[14] Wenju Sun, Qingyong Li, Wen Wang, Yangliao Geng, and Boyang Li. Task arithmetic in trust region: A training-free model merging approach to navigate knowledge conflicts. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 5178–5187, 2025.

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

[16] 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*, volume 2024, pages 22743–22763, 2024.

[17] Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 56332–56356. PMLR, 21–27 Jul 2024.

[18] Jiho Choi, Donggyun Kim, Chanhyuk Lee, and Seunghoon Hong. Revisiting weight averaging for model merging. *arXiv preprint arXiv:2412.12153*, 2024.

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

[20] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[21] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR, 06–11 Aug 2017.

[22] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8218–8227, 2021.

[23] James Seale Smith, Lazar Valkov, Shaunak Halbe, Vyshnavi Gutta, Rogerio Feris, Zsolt Kira, and Leonid Karlinsky. Adaptive memory replay for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3605–3615, June 2024.

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

[25] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[26] 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 Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021.

[27] 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.

[28] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob

Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[29] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

[30] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[31] Pan Lu, Liang Qiu, Jiaqi Chen, Tony Xia, Yizhou Zhao, Wei Zhang, Zhou Yu, Xiaodan Liang, and Song-Chun Zhu. Iconqa: A new benchmark for abstract diagram understanding and visual language reasoning. *arXiv preprint arXiv:2110.13214*, 2021.

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[33] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35: 2507–2521, 2022.

[34] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3608–3617. IEEE, 2018.

[35] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[36] Anke Tang, Li Shen, Yong Luo, Enneng Yang, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Fusionbench: A unified library and comprehensive benchmark for deep model fusion, 2025.

[37] Fanhu Zeng, Haiyang Guo, Fei Zhu, Li Shen, and Hao Tang. Robustmerge: Parameter-efficient model merging for mllms with direction robustness. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, volume 38, pages 71071–71095, 2025.

[38] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[39] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010. doi: 10.1109/CVPR.2010.5539970.

[40] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. doi: 10.1109/ICCVW.2013.77.

[41] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017. doi: 10.1109/JPROC.2017.2675998.

[42] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207, 2018. doi: 10.1109/IGARSS.2018.8519248.

[43] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, 2011.

[44] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32: 323–332, 2012.

[45] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[46] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[47] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

[48] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pages 210–218. Springer, 2018.

[49] Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *International conference on neural information processing*, pages 117–124. Springer, 2013.

[50] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[51] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

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

[53] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pages 446–461. Springer, 2014.

[54] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[55] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[56] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018.

[57] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

A Theoretical Proof of Restoration

In Section 4, we explain the construction of restoration matrices. Here, we present the detailed restoration process and prove that the restoration matrices indeed recover the damage induced by orthogonalization (i.e., $\hat{U}^t = U_\perp^t R_U^t$ and $\hat{V}^t = V_\perp^t R_V^t$). We explain only the restoration of $\hat{U}^t$, as the exact same procedure is applied to $\hat{V}^t$.

Proof. Let the SVD of $\hat{U}^t$ be:

$$\hat{U}^t = P_U^t D_U^t (Q_U^t)^\top, \qquad U_\perp^t = P_U^t (Q_U^t)^\top. \tag{16}$$

We define the restoration matrix as follows:

$$R_U^t = Q_U^t D_U^t (Q_U^t)^\top. \tag{17}$$

Equivalently, using only the orthogonalized basis and the SVD factors, this can be expressed as:

$$R_U^t = (U_\perp^t)^\top P_U^t D_U^t (P_U^t)^\top U_\perp^t. \tag{18}$$

Then the restoration guarantee follows directly as $(Q_U^t)^\top Q_U^t = I$:

$$U_\perp^t R_U^t = P_U^t (Q_U^t)^\top Q_U^t D_U^t (Q_U^t)^\top = P_U^t D_U^t (Q_U^t)^\top = \hat{U}^t. \tag{19}$$

□

B Experimental Details

We provide detailed experimental settings for merging vision, NLP, and multimodal tasks, including configurations for auxiliary optimization.

B.1 Vision Task Orders

The 8-task setting includes SUN397 [39], Cars [40], RESISC45 [41], EuroSAT [42], SVHN [43], GTSRB [44], MNIST [45], and DTD [46]. The 14-task setting further includes Flowers102 [47], PCAM [48], FER2013 [49], OxfordIIITPet [50], STL10 [51], and CIFAR100 [52]. The 20-task setting additionally includes CIFAR10 [52], Food101 [53], FashionMNIST [54], EMNIST [55], KMNIST [56], and RenderedSST2 [26, 57]. Task orders are generated using 10 seeds (40–49):

8-task orders

- MNIST, SVHN, SUN397, DTD, GTSRB, RESISC45, Cars, EuroSAT
- Cars, GTSRB, MNIST, SUN397, EuroSAT, RESISC45, DTD, SVHN
- EuroSAT, DTD, RESISC45, SUN397, SVHN, MNIST, GTSRB, Cars
- GTSRB, DTD, RESISC45, SUN397, MNIST, SVHN, EuroSAT, Cars
- SVHN, DTD, MNIST, GTSRB, Cars, SUN397, RESISC45, EuroSAT
- MNIST, DTD, SVHN, RESISC45, GTSRB, SUN397, EuroSAT, Cars
- MNIST, RESISC45, DTD, SUN397, Cars, SVHN, GTSRB, EuroSAT
- Cars, GTSRB, RESISC45, DTD, SUN397, SVHN, EuroSAT, MNIST
- MNIST, SUN397, RESISC45, SVHN, Cars, DTD, GTSRB, EuroSAT
- SVHN, MNIST, EuroSAT, Cars, RESISC45, DTD, GTSRB, SUN397

14-task orders

- OxfordIIITPet, STL10, PCAM, Cars, Flowers102, SUN397, MNIST, DTD, SVHN, GTSRB, RESISC45, EuroSAT, CIFAR100, FER2013
- RESISC45, FER2013, PCAM, Cars, DTD, CIFAR100, GTSRB, EuroSAT, SUN397, MNIST, STL10, Flowers102, SVHN, OxfordIIITPet

- EuroSAT, GTSRB, RESISC45, SVHN, PCAM, CIFAR100, DTD, SUN397, MNIST, STL10, Flowers102, FER2013, Cars, OxfordIIITPet
- Flowers102, CIFAR100, GTSRB, RESISC45, MNIST, EuroSAT, SVHN, SUN397, FER2013, PCAM, DTD, Cars, STL10, OxfordIIITPet
- Flowers102, MNIST, GTSRB, FER2013, RESISC45, STL10, DTD, PCAM, OxfordIIITPet, Cars, CIFAR100, SUN397, SVHN, EuroSAT
- RESISC45, Flowers102, EuroSAT, STL10, OxfordIIITPet, GTSRB, MNIST, PCAM, FER2013, SVHN, CIFAR100, SUN397, DTD, Cars
- Cars, SVHN, CIFAR100, GTSRB, STL10, MNIST, DTD, SUN397, OxfordIIITPet, PCAM, RESISC45, Flowers102, FER2013, EuroSAT
- Cars, GTSRB, SVHN, EuroSAT, RESISC45, CIFAR100, DTD, FER2013, OxfordIIITPet, SUN397, PCAM, STL10, MNIST, Flowers102
- FER2013, SUN397, CIFAR100, SVHN, DTD, GTSRB, STL10, Cars, PCAM, RESISC45, MNIST, OxfordIIITPet, EuroSAT, Flowers102
- CIFAR100, MNIST, DTD, Cars, PCAM, GTSRB, EuroSAT, RESISC45, SVHN, OxfordIIITPet, FER2013, SUN397, STL10, Flowers102

20-task orders

- STL10, KMNIST, GTSRB, FER2013, Cars, PCAM, RenderedSST2, DTD, EuroSAT, Food101, RESISC45, EMNIST, FashionMNIST, SUN397, CIFAR100, CIFAR10, Flowers102, OxfordIIITPet, SVHN, MNIST
- MNIST, CIFAR10, CIFAR100, OxfordIIITPet, SVHN, KMNIST, Cars, GTSRB, RenderedSST2, RESISC45, DTD, Food101, FER2013, EuroSAT, FashionMNIST, SUN397, STL10, Flowers102, EMNIST, PCAM
- KMNIST, FashionMNIST, FER2013, DTD, SVHN, MNIST, GTSRB, OxfordIIITPet, CIFAR10, PCAM, Flowers102, STL10, RenderedSST2, Food101, SUN397, Cars, CIFAR100, RESISC45, EMNIST, EuroSAT
- Cars, OxfordIIITPet, SVHN, RenderedSST2, EMNIST, DTD, STL10, FashionMNIST, RESISC45, FER2013, MNIST, Flowers102, KMNIST, GTSRB, PCAM, CIFAR100, SUN397, Food101, CIFAR10, EuroSAT
- Food101, STL10, CIFAR100, CIFAR10, OxfordIIITPet, GTSRB, FashionMNIST, FER2013, MNIST, SVHN, Flowers102, RESISC45, Cars, EMNIST, KMNIST, EuroSAT, RenderedSST2, SUN397, PCAM, DTD
- KMNIST, FashionMNIST, MNIST, CIFAR10, Flowers102, GTSRB, DTD, RenderedSST2, CIFAR100, EMNIST, FER2013, OxfordIIITPet, SVHN, Cars, STL10, PCAM, EuroSAT, SUN397, Food101, RESISC45
- MNIST, CIFAR10, CIFAR100, EuroSAT, PCAM, Flowers102, RenderedSST2, EMNIST, RESISC45, OxfordIIITPet, Food101, FER2013, GTSRB, STL10, KMNIST, SUN397, Cars, SVHN, FashionMNIST, DTD
- EMNIST, GTSRB, PCAM, EuroSAT, CIFAR10, RESISC45, MNIST, OxfordIIITPet, Flowers102, DTD, SVHN, Food101, RenderedSST2, SUN397, FER2013, CIFAR100, Cars, KMNIST, STL10, FashionMNIST
- CIFAR10, KMNIST, RenderedSST2, Food101, SUN397, OxfordIIITPet, Flowers102, RESISC45, PCAM, EuroSAT, DTD, Cars, FER2013, EMNIST, STL10, SVHN, GTSRB, CIFAR100, MNIST, FashionMNIST
- EMNIST, GTSRB, MNIST, STL10, PCAM, RESISC45, CIFAR10, Food101, EuroSAT, CIFAR100, RenderedSST2, FER2013, SUN397, OxfordIIITPet, DTD, SVHN, Flowers102, KMNIST, Cars, FashionMNIST

B.2 NLP Task Orders

We conduct NLP experiments on eight tasks from the GLUE benchmark [29], namely CoLA, MNLI, MRPC, QNLI, QQP, RTE, SST2, and STSB. For STSB, we report Spearman's correlation ρ. Task orders are generated using 10 seeds (40–49):

GLUE

- MRPC, RTE, QNLI, MNLI, CoLA, SST2, QQP, STSB
- CoLA, QQP, STSB, QNLI, RTE, MNLI, MRPC, SST2
- QNLI, QQP, SST2, STSB, MRPC, RTE, CoLA, MNLI
- SST2, STSB, QQP, QNLI, MNLI, RTE, MRPC, CoLA
- MRPC, RTE, QNLI, MNLI, CoLA, STSB, QQP, SST2
- RTE, STSB, MNLI, CoLA, MRPC, SST2, QNLI, QQP
- SST2, RTE, MRPC, STSB, QQP, CoLA, QNLI, MNLI
- SST2, MNLI, MRPC, STSB, QQP, QNLI, CoLA, RTE
- SST2, CoLA, QNLI, MRPC, STSB, QQP, MNLI, RTE
- STSB, QQP, RTE, SST2, CoLA, QNLI, MRPC, MNLI

B.3 Multimodal Task Orders

We conduct multimodal experiments on four vision-language tasks, namely IconQA [31], ImageNet [32], ScienceQA [33], and VizWiz [34], using the checkpoints and following the evaluation protocol of RobustMerge [37]. In addition to the alphabetical order, task orders are generated using 5 seeds (40–43, 45). Seed 44 is skipped since it yields the same permutation as seed 40, so that all six orders are distinct:

- IconQA, ImageNet, ScienceQA, VizWiz
- ImageNet, IconQA, ScienceQA, VizWiz
- ScienceQA, IconQA, ImageNet, VizWiz
- ScienceQA, ImageNet, VizWiz, IconQA
- ScienceQA, VizWiz, ImageNet, IconQA
- IconQA, VizWiz, ImageNet, ScienceQA

B.4 Hyperparameters

Global singular subspace separation We use a fixed orthogonality tolerance of $\epsilon = 10^{-4}$ throughout all experiments.

Auxiliary optimization For each layer, we optimize the merged weights using the Adam optimizer with a learning rate of 1×10^{-5} for vision and multimodal tasks and 1×10^{-4} for NLP tasks. For ViT variants and LLaVA, all matrix-shaped weights are optimized, whereas for the Flan-T5 architecture, embedding layers are excluded. We observe that the number of required optimization steps increases with the number of tasks. Accordingly, we perform 300, 400, 800, and 1000 iterations when merging 4, 8, 14, and 20 tasks, respectively.

Multimodal baselines For multimodal tasks, Continual Task Arithmetic and Continual TIES-Merging use a scaling coefficient of $\lambda = 0.3$, with TIES-Merging keeping the top 20% of parameters, and OPCM uses its default setting of $\alpha = 0.5$ in FusionBench [36]. NUFILT is run with its official implementation and default hyperparameters ($r_p = 128$, $r_l = 64$, $r_v = 8$, 50 iterations, learning rate 1×10^{-3}). Only the task order is varied, identically for all methods.

C Additional Analysis and Experimental Results

C.1 Inter- and Intra-task Orthogonality of Singular Vectors

The subspace separation in GS3R is based on the observation that singular vectors within a single task are mutually orthogonal, whereas orthogonality across different tasks does not necessarily hold. To demonstrate this, for each weight matrix, we collect the corresponding singular vectors from every

task and concatenate them according to the shape of the weight. We then compute the Gram matrices of these concatenated singular vectors and present them in Figure 4 and Figure 5.

The Gram matrices from different layers exhibit a clear block-diagonal structure, where each diagonal block corresponds to the singular vectors of an individual task. This pattern confirms that intra-task orthogonality is preserved, while inter-task orthogonality is not. Such a structural property justifies our subspace separation strategy, as it allows GS3R to effectively isolate task-specific subspaces via spectral clustering.

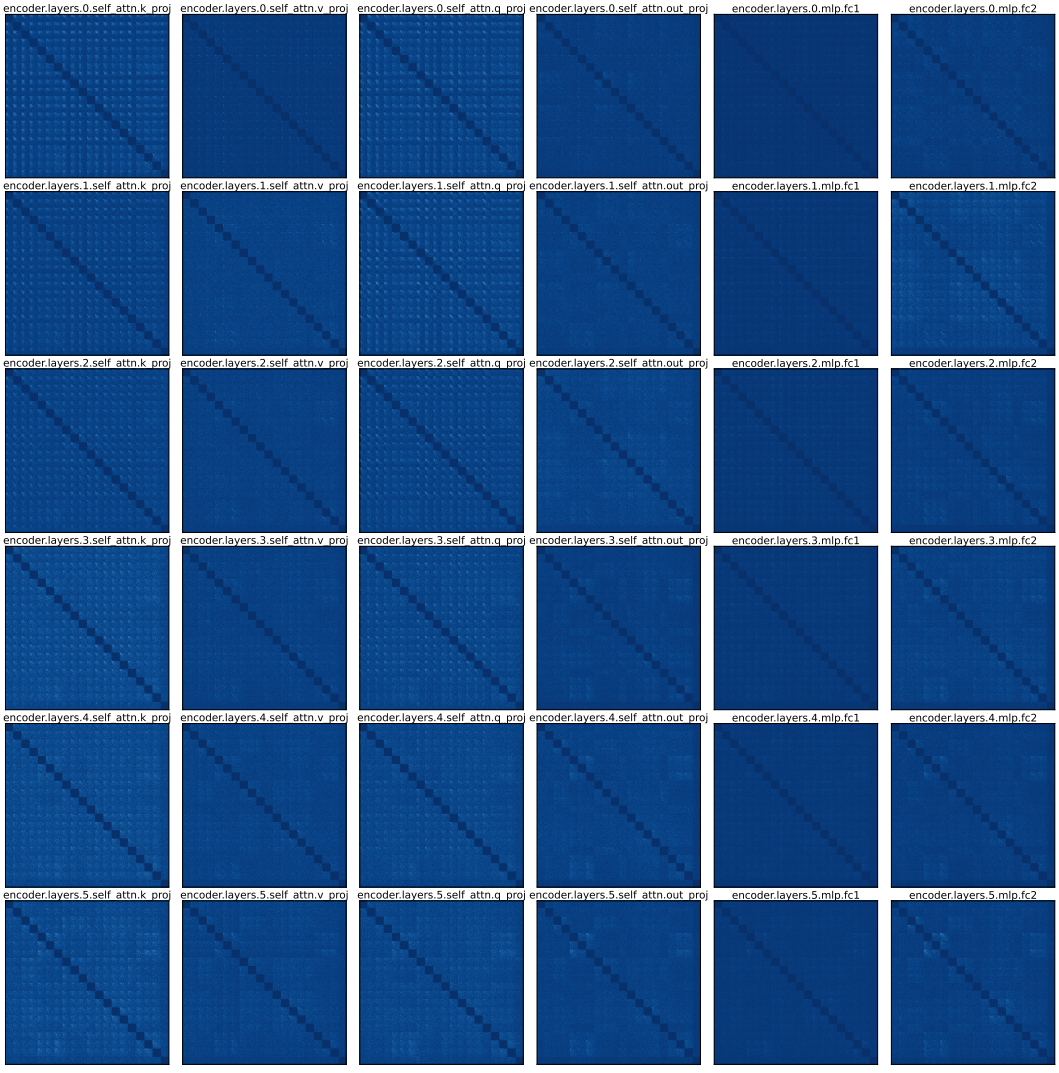


Figure 4: Gram matrix of concatenated singular vectors from U of each task.

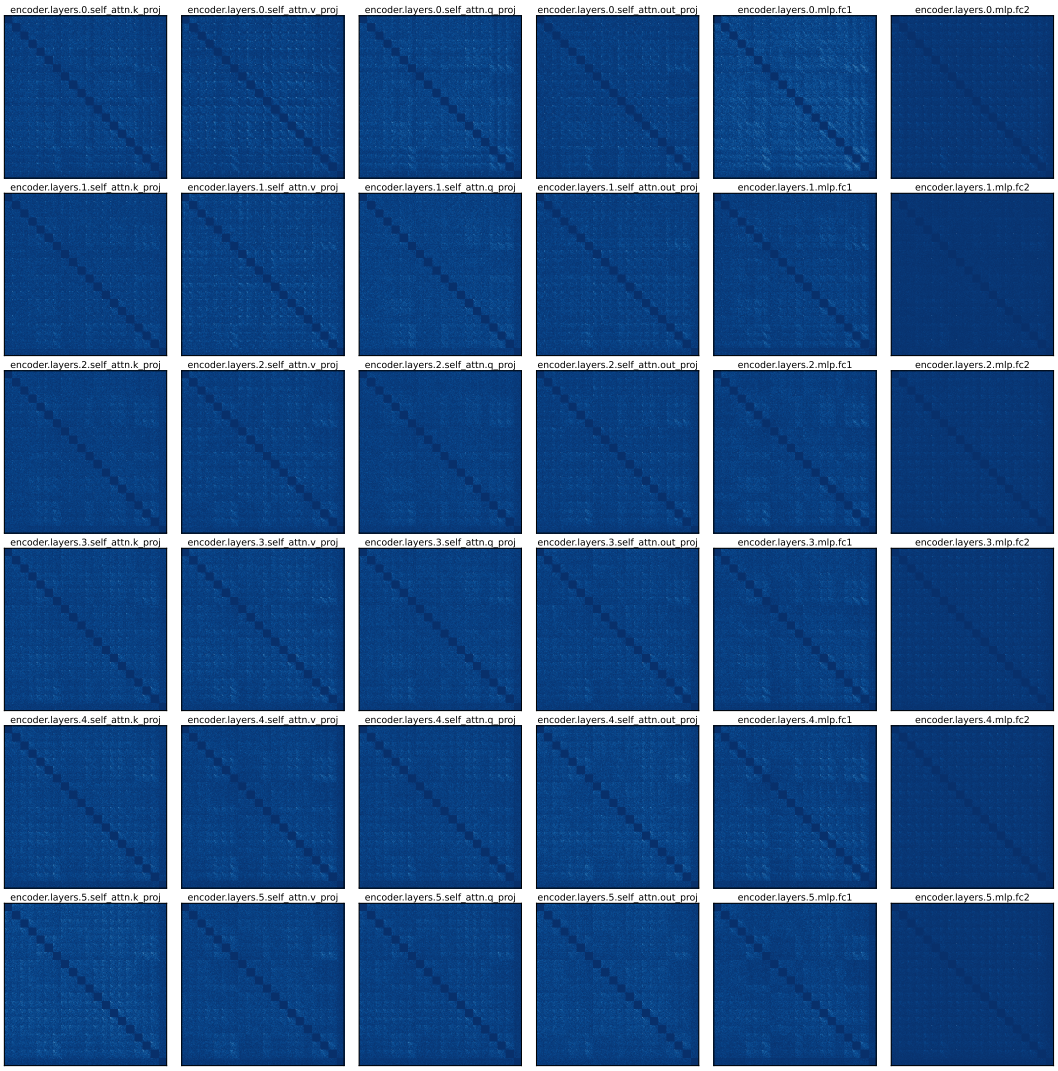


Figure 5: Gram matrix of concatenated singular vectors from V of each task.

Algorithm 1 Subspace Separation via Spectral Clustering

Require: Restored singular vectors $U = [u_1, \ldots, u_N]$, singular values $\sigma_1, \ldots, \sigma_N$, tolerance ϵ

1: **for** $i = 1, \ldots, N$ **do** ▷ Normalize Columns
2: $\quad \bar{u}_i \leftarrow u_i / \|u_i\|$
3: **end for**
4: **for** $i = 1, \ldots, N$ **do** ▷ Build Orthogonality Affinity
5: $\quad$ **for** $j = 1, \ldots, N$ **do**
6: $\quad\quad$ **if** $i \neq j$ **and** $|\langle \bar{u}_i, \bar{u}_j \rangle| < \epsilon$ **then**
7: $\quad\quad\quad A_{ij} \leftarrow 1$
8: $\quad\quad$ **else**
9: $\quad\quad\quad A_{ij} \leftarrow 0$
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: **end for**
13: $D \leftarrow \mathrm{diag}(A \cdot \mathbf{1})$ ▷ Degree Matrix, $\mathbf{1}$ is the all-ones vector
14: $L \leftarrow I - D^{-1/2} A D^{-1/2}$ ▷ Symmetric Normalized Laplacian
15: $(\lambda_1 \leq \cdots \leq \lambda_N)$, $Q \leftarrow \mathrm{eigendecomposition}(L)$
16: $k \leftarrow \arg\max_i (\lambda_{i+1} - \lambda_i)$ ▷ Eigengap Heuristic
17: $Y \leftarrow$ first k columns of Q
18: **for** $i = 1, \ldots, N$ **do** ▷ Row-normalize Embedding
19: $\quad Y_{i,:} \leftarrow Y_{i,:} / \|Y_{i,:}\|$
20: **end for**
21: labels $\leftarrow$ k-means(Y, k)
22: **for** $c = 1, \ldots, k$ **do** ▷ Form Task-wise Groups
23: $\quad G_c \leftarrow \{\, i \mid \text{labels}_i = c \,\}$
24: $\quad$ Sort G_c by σ_i in descending order
25: **end for**
26: **return** $G_1, \ldots, G_k$ and k

C.2 Details and Analysis on Spectral Clustering

In this section, we further describe and analyze the spectral clustering used in the subspace separation step. Specifically, we provide its algorithmic details, validate its reliability, and present an ablation study on the tolerance parameter ϵ.

Algorithmic details Algorithm 1 summarizes the separation step. The key property it exploits is that singular vectors originating from the same task are mutually orthogonal by construction of the SVD, while vectors from different tasks typically carry a small but nonzero correlation. The affinity therefore encodes pairwise orthogonality. Two normalized columns are connected ($A_{ij} = 1$) when their absolute inner product falls below the tolerance ϵ, so each task's components form a densely connected group (lines 4–12). We then form the symmetric normalized Laplacian and compute its eigenvalues in ascending order (lines 13–15). When the graph consists of k well-separated dense groups, the smallest k eigenvalues concentrate near zero and a clear jump appears afterward, so the number of tasks is selected as the position of the largest gap between consecutive eigenvalues (line 16). Note that the number of merged tasks is never supplied as an input; it is inferred entirely from the accumulated structure. Clustering is completed by row-normalizing the first k eigenvectors and running k-means, following standard normalized spectral clustering, and the components of each recovered group are sorted by singular value for the subsequent rank selection (lines 17–25).

Validation of clustering The separation criterion could in principle fail in two ways: components of different tasks may happen to be orthogonal within the tolerance, or related tasks may share dominant singular directions. To measure how often either occurs, we run the separation step on every weight matrix of ViT-B/32 (73 layers) with all 20 task vectors, and compare the recovered grouping against the ground-truth task identity via the correct cluster count ratio and the Adjusted Rand Index (ARI). The 20 task suite is the relevant stress case for the second failure mode, as it contains four MNIST-family datasets, SVHN, and both CIFAR-10 and CIFAR-100. As shown in Table 6, at the default tolerance ($\epsilon = 10^{-4}$) separation is exact in every layer. Notably, shared

Table 6: Reliability of the separation step on ViT-B/32 with 20 tasks under varying tolerance ϵ. We report the fraction of layers (out of 73) with the correct cluster count and the mean Adjusted Rand Index. $\epsilon = 10^{-4}$ is used throughout the paper.

Tolerance ϵ	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
Correct cluster count	0/73	70/73	**73/73**	**73/73**	**73/73**
Mean ARI	0.007	0.959	**1.000**	**1.000**	**1.000**

Table 7: Effect of the tolerance ϵ on merging performance, merging 8 tasks with ViT-B/32 over 10 task orders.

Tolerance ϵ	10^{-2}	5×10^{-3}	10^{-3}	10^{-4}	10^{-5}
Accuracy (%)	81.6±0.6	82.4±0.3	**82.7**±0.2	**82.7**±0.2	**82.7**±0.2

dominant directions produce no misassignment despite the closely related datasets, consistent with the block-diagonal Gram matrices in Appendix C.1. Degradation appears only at $\epsilon = 10^{-2}$, where the tolerance approaches the typical magnitude of cross-task inner products and different tasks begin to be linked in the affinity graph.

Ablation study on tolerance parameter ϵ Table 7 reports the merging performance when sweeping ϵ over three orders of magnitude (8 tasks, ViT-B/32, 10 task orders). Accuracy is flat across the range where the separation is exact and drops only as ϵ approaches the cross-task inner-product scale, the same point where the clustering itself breaks down. The default value thus sits in the middle of a wide plateau rather than at a tuned point.

C.3 Analysis of Full-rank Degradation of NLP Tasks

As reported in Section 5.2, the truncated variant of GS3R outperforms its full-rank counterpart on GLUE, while the ordering is reversed on the vision suite. We first verify that this is not a seed-level fluctuation. The rank sweep with 10 new task orders (Table 8) reproduces the same pattern, with accuracy peaking at rank budget $r = 64$ and falling off toward both ends by margins an order of magnitude larger than the seed variance. The trend is also not monotonic in r, so a single error source cannot account for it. We identify and measure two sources separately below.

Effective rank differs sharply by architecture We measure the energy rank of the task vectors, defined as the minimum rank capturing 90% of $\sum_i \sigma_i^2$, averaged over tasks and layers (Table 9). Encoder updates in Flan-T5 are no more concentrated than the vision suite (197.6 vs. 147.2), but decoder updates saturate at a far smaller rank (26.1). Since decoder matrices form the majority of the merged matrices in Flan-T5, their early saturation pulls the aggregate down (93.0) and governs the behavior under a uniform budget.

Error decomposition To locate where the damage occurs, we decompose the merging error of each task vector $\tau = U\Sigma V^\top$ into two stages, measured task-wise per layer. Truncation first reduces τ to the r/N directions allocated to each of the N tasks under the total budget r, written $\tau' = U'\Sigma'(V')^\top$. Orthogonalization then stacks the truncated bases across tasks and replaces them with their polar factors, holding the singular values fixed. We write $U_\perp$ and $V_\perp$ for the resulting blocks of this task. The two stages are measured as follows:

Table 8: Rank sweep on GLUE with Flan-T5-Base, re-run with 10 new task orders. The same pattern as the original experiment appears: accuracy peaks at $r = 64$ and falls off toward both ends by margins an order of magnitude larger than the seed variance.

r	16	32	64	128	256	Full Rank
Accuracy (%)	82.73±0.11	83.59±0.10	**83.63**±0.06	83.38±0.12	83.30±0.08	82.64±0.17

Table 9: Energy rank of task vectors, defined as the minimum rank capturing 90% of $\sum_i \sigma_i^2$, averaged over tasks and layers.

	GLUE: Flan-T5			Vision suite
	Decoder	Encoder	All	ViT-B/32 (8 tasks)
Energy rank	26.1	197.6	93.0	147.2

Table 10: Error decomposition under varying total rank budget r. E_{trunc}, E_{orth}, and E_{total} are defined in Eq. (20). On GLUE, orthogonalization damage dominates and grows with r, so the total is minimized at an intermediate budget. On the vision suite, truncation dominates and the total decreases monotonically.

	r	16	32	64	128	256
GLUE	E_{trunc}	0.30	0.23	0.18	0.14	0.10
	E_{orth}	0.67	0.72	0.76	0.81	0.87
	E_{total}	0.87	**0.86**	**0.86**	0.88	0.92
Vision	E_{trunc}	0.59	0.50	0.40	0.30	0.21
	E_{orth}	0.47	0.48	0.52	0.58	0.67
	E_{total}	0.93	0.90	0.86	**0.82**	**0.82**

$$E_{\text{trunc}} = 1 - \frac{\|\tau'\|_F}{\|\tau\|_F}, \qquad E_{\text{orth}} = \frac{\|U_\perp \Sigma' V_\perp^\top - \tau'\|_F}{\|\tau'\|_F}, \qquad E_{\text{total}} = \frac{\|U_\perp \Sigma' V_\perp^\top - \tau\|_F}{\|\tau\|_F}. \quad (20)$$

Because Σ' appears identically in both terms of E_{orth}, it isolates the displacement of the singular bases caused by orthogonalization alone. The three quantities are measured independently and are not additive, since their reference points differ.

Orthogonalization dominates on GLUE, truncation on vision Table 10 shows that the two suites sit in opposite regimes. On GLUE, E_{trunc} is already low at the smallest budget and falls only slowly, consistent with the low effective rank of the decoder, while E_{orth} is the larger term throughout and grows with r. The total is therefore minimized at an intermediate budget, matching the optimum in Figure 2(a). On the vision suite the ordering reverses. Truncation dominates and drops steeply as r grows, since the task vectors still carry signal in those directions, so the total decreases monotonically and full rank performs best. The same measurement predicting opposite optima on the two suites, both agreeing with the measured accuracy, supports its validity as a diagnostic.

Why surplus directions are harmful Directions beyond the signal rank are not inert. During orthogonalization they occupy the shared basis space and displace the signal directions of other tasks, which is precisely why E_{orth} grows with r. Truncation acts as a filter on these surplus directions, whereas orthogonalization converts them into signal displacement, which is why full rank underperforms when the effective rank is small.

C.4 Evaluation Protocol for Multimodal Tasks

Official scoring rules The four benchmarks are scored by the evaluation scripts released with RobustMerge [37], and three of them are tolerant to the formatting of generated text by construction. ImageNet counts a prediction as correct when the ground-truth label and the prediction contain each other as substrings, ScienceQA extracts the selected option from the generated sentence with a regular expression before comparison, and VizWiz-Caption averages standard captioning metrics. IconQA is the only exception. Its scorer counts a prediction as correct only when the full generated string exactly matches the ground-truth answer string after case normalization, so any additional character makes an otherwise correct answer count as wrong.

Observed failure mode Inspecting the IconQA outputs of the merged models, we found that the exact-match scorer primarily penalizes formatting drift rather than wrong answers. Merged models

frequently append a trailing period or additional tokens to the answer, such as "4." for the ground truth "4" or "0. paw." for "0", where the selected option is correct but the string comparison fails. The individually fine-tuned checkpoint produces no such outputs, so the drift is introduced by merging rather than inherited from fine-tuning. It appears in every merging method, only marginally for Continual Task Arithmetic. Ignoring the trailing text raises IconQA accuracy by 6 to 13 points for Continual TIES-Merging, OPCM, and the GS3R variants, and by 24 points for NUFILT, whose exact-match score also varies strongly across task orders.

Format-tolerant rule To evaluate IconQA at the same level of tolerance as the other three tasks, we additionally count a prediction as correct if its first token (the leading characters up to whitespace, a period, or a comma), or the full string after stripping a trailing period, matches the ground-truth answer, case-insensitively. The rule is applied identically to all methods and changes only the IconQA column. Table 3 reports both the original (ACC) and the format-tolerant (ACC†) averages, with the format-tolerant IconQA accuracy given in parentheses.

Discussion The two scoring rules measure different things and lead to the same conclusion. The original scorer folds formatting drift into the IconQA score, and under it GS3R outperforms all baselines, with part of this margin reflecting that the merged model of NUFILT drifts further from the expected answer format than ours. The format-tolerant rule isolates whether the correct answer was produced, and under it truncated GS3R matches NUFILT without any optimization while GS3R with optimization remains the best method. GS3R therefore matches or surpasses NUFILT under both protocols, and unlike NUFILT its results remain stable across task orders under exact match. We report both so that the contribution of formatting is visible rather than hidden in the average. How different merging operators affect the output format of generative models is an interesting question in its own right, which we leave to future work.

C.5 Rank Allocation Strategy

In our implementation, GS3R treats all tasks equally and allocates the rank budget uniformly across tasks. However, since all task-specific information is separated in the separation step, different rank allocation strategies can be employed. Beyond the uniform allocation used throughout the paper, we evaluate four alternatives that allocate the rank budget from task-specific properties of the spectrum. Three allocate proportionally to nuclear mass, energy, and Frobenius norm, respectively, while the fourth is a global top-k selection that ignores task boundaries entirely. Table 11 reports the results of merging 8 tasks with ViT-B/32 over 10 task orders.

Table 11: Comparison of rank allocation strategies when merging 8 tasks with ViT-B/32. We report mean and standard deviation of accuracy over 10 task orders.

	Nuclear	Energy	Frobenius	Global Top-k	Ours
Accuracy (%)	82.3±0.3	82.3±0.2	82.5±0.3	<u>82.6</u>±0.2	**82.7**±0.2

None of the alternatives improves on uniform allocation. Since each variant reads intrinsic rank differently from the same spectrum, this indicates that the information needed for a better allocation is not present in the task vectors themselves. Task complexity and importance are properties of the deployment scenario rather than of the weight space, so a policy exploiting them would require a signal from outside the data-free setting. Because the allocation rule only determines which indices are selected in Eq. (9), such a policy can replace ours without modifying the rest of the pipeline, and we leave this as a direction for future work.

C.6 Comparison with Explicit Subspace Storage

One simple yet intuitive baseline not considered in the main paper is to explicitly store the task-wise subspaces. For a layer with task matrix $\tau \in \mathbb{R}^{m\times n}$ and rank budget r, this requires storing $(m+n)r$ entries per task, while GS3R stores only r^2 entries regardless of the model width. The advantage grows as the rank budget decreases, which is precisely the regime of edge systems.

The results are summarized in Table 12. Even at full rank, GS3R uses about $3\times$ less storage, and the gap widens to 9–$69\times$ in the truncated regime, since the metadata size of GS3R depends only on the

Table 12: Comparison with the explicit-storage baseline under varying rank budgets, merging 8 tasks with ViT-B/32. We report the mean accuracy (%) and the additional storage (MB). Results of GS3R are averaged over 10 task orders. The explicit-storage baseline is deterministic given the task order.

	Accuracy (%)		Storage (MB)	
Rank Budget (r)	Explicit	GS3R	Explicit	GS3R
32	72.9	72.7±0.3	21.3	**0.3**
64	76.3	76.0±0.3	42.6	**1.2**
128	79.4	79.1±0.2	85.1	**4.8**
256	82.0	81.8±0.2	170.0	**19.0**
Full	83.1	82.7±0.2	509.8	**170.1**

rank budget and not on the model width. At the same time, the accuracy difference is under 0.5%p at every rank. Since the baseline stores the exact per-task subspaces, it serves as an upper bound where task identity and pre-orthogonalization geometry are given for free. That GS3R recovers this bound within 0.5%p, without storing any task identity, is direct evidence that restoration and separation are practically effective.

C.7 The Case of Repeated Tasks

An interesting corner case of continual merging is when the same task arrives more than once. Since spectral clustering is applied to the previously merged vector and the incoming task is added afterward as a separate group, a task arriving a second time enters as an additional group and receives twice the rank budget. What is less obvious is what happens during orthogonalization once its components appear twice. We provide a theoretical analysis and empirical verification below.

Theoretical analysis Let $\hat{U}$ denote the concatenated left singular vectors of all selected components across tasks where each column is a unit vector. The analysis for $\hat{V}$ is identical. Orthogonalization replaces $\hat{U} = PDQ^\top$ with its polar factor $PQ^\top$. This resets every stretch value in D to one. For a unit direction w, the stretch $\sigma_w^2 = \sum_c \langle w, u_c \rangle^2$ is the total load placed on w by all columns of all tasks, so directions shared by several tasks are suppressed more strongly. On a direction w shared by several tasks we split this sum into the terms coming from the columns of task j and the terms coming from all other columns. Writing $\sigma_j^2 = \sum_{c \in j} \langle w, u_c \rangle^2$ and $\sigma_{\text{rest}}^2 = \sum_{c \notin j} \langle w, u_c \rangle^2$ gives $\sigma_w^2 = \sigma_j^2 + \sigma_{\text{rest}}^2$. The fraction α_j of that direction retained by task j is

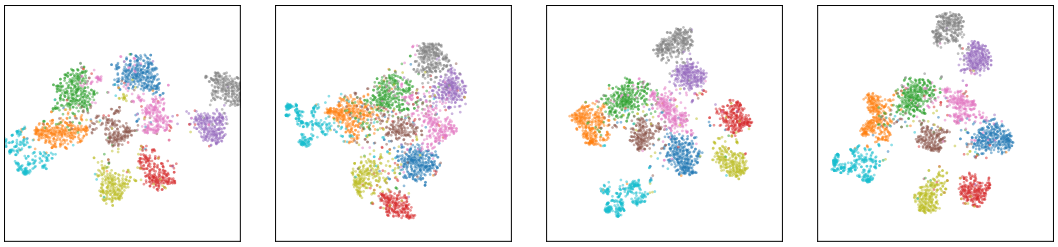
$$\alpha_j = \frac{\sigma_j^2}{\sigma_j^2 + \sigma_{\text{rest}}^2} < 1. \tag{21}$$

If the same task arrives m times, its contribution becomes $m\sigma_j^2$ while σ_{rest}^2 stays the same, and the retained fraction becomes

$$\alpha_j(m) = \frac{m\sigma_j^2}{m\sigma_j^2 + \sigma_{\text{rest}}^2}. \tag{22}$$

This increases with m but never exceeds 1. Repetition therefore only restores what the overlap had discounted and never strengthens the task beyond what it would have on an unshared direction. The shares of the other tasks shrink accordingly, making this a reallocation of capacity rather than a corruption of the merge.

Empirical verification We verify this prediction in the full pipeline. After merging 8 tasks with ViT-B/32, the task appearing fourth in the order is merged once more. Order 1 is MNIST, SVHN, SUN397, DTD, GTSRB, RESISC45, Cars, EuroSAT, and Order 2 is MNIST, SUN397, RESISC45, SVHN, Cars, DTD, GTSRB, EuroSAT, so the repeated task is DTD and SVHN, respectively. As shown in Table 13, the repeated task improves substantially in both orders as predicted. As the results also show, however, this does not necessarily translate into an overall improvement since the



(a) OPCM (b) DOP (c) NUFILT (d) OURS

Figure 6: t-SNE visualization of the representation distributions of EuroSAT extracted from merged models using different methods.

Table 13: Effect of merging a repeated task. After merging 8 tasks with ViT-B/32, the task appearing fourth in each order (DTD for Order 1, SVHN for Order 2) is merged once more. The accuracy of the repeated task is underlined.

	Cars	DTD	Euro	GTS	MNIST	RESIS	SVHN	SUN	AVG
Order 1, original	71.5	<u>63.6</u>	95.0	91.6	98.7	84.2	89.8	67.4	82.7
Order 1, repeated	72.1	<u>68.8</u>	94.7	92.9	98.8	84.9	91.2	67.8	83.9
Order 2, original	70.5	63.2	95.3	92.8	98.7	83.0	<u>91.0</u>	67.1	82.7
Order 2, repeated	68.7	61.1	91.0	90.2	98.4	80.9	<u>95.8</u>	65.9	81.5

shares of the remaining tasks shrink. Filtering component pairs with inner products near one removes this behavior if undesired, though a repeated arrival may itself signal that the task matters more in deployment.

C.8 Additional Qualitative Analysis

We provide t-SNE [38] visualizations of representations of the fourth merged EuroSAT task from a ViT-B/32 model merged over 20 tasks. As shown in Figure 6, prior methods fail to preserve structured representations, whereas our method maintains well-separated, class-consistent clusters. This qualitative difference aligns with the superior performance of the merged model and indicates that our approach better preserves task-specific representations during continual model merging.

D Limitations & Broader Impact

D.1 Limitations

Scalability to longer task sequences Although our experiments include relatively long continual merging settings with up to 20 sequential tasks, we do not explicitly investigate regimes in which the accumulated task complexity substantially exceeds the representational capacity of the base model. As increasingly diverse tasks are merged into a fixed-capacity model, interference between task-specific representations may become unavoidable, potentially leading to performance saturation or degradation even when task balance is preserved. Understanding the relationship among model capacity, task diversity, and the maximum number of effectively mergeable tasks remains an important direction for future work.

Heterogeneous architectures Another limitation is that our experiments primarily focus on supervised fine-tuned checkpoints with relatively aligned architectures. The behavior of the proposed method under heterogeneous settings, such as models trained with different tokenizers, modalities, or optimization objectives, remains underexplored. Investigating robust merging across highly diverse foundation models could further broaden the applicability of continual model merging.

D.2 Broader Impact

This work aims to improve continual model merging while preserving task balance. Nevertheless, misuse of merged models could exacerbate risks such as membership inference attacks and safety leakage. Mitigating these risks requires that progress in model merging be paired with rigorous AI safety research and adherence to ethical guidelines, including the development of robust safeguards against information leakage and the incorporation of differential privacy techniques.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We present the contributions in the abstract and clearly summarize them in the introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the broader impact and limitations in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide a proof of restoration in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide experimental details in Section 5 and Appendix B. Our code is included in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All models and datasets used in our experiments are publicly available on Hugging Face. Our code is publicly released on GitHub, with instructions to reproduce the main experimental results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all detailed experimental settings in Section 5 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our main experimental results report the mean and standard deviation over multiple (10 for vision and NLP, 6 for multimodal) trials.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the computational resources required for the experiments and provide an overhead analysis in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that our work complies with all applicable guidelines.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impact and limitations in Appendix D.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work does not involve the release of pretrained models, scraped datasets, or other components that require special safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide references to all models and data used in the experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not introduce new assets such as datasets or pretrained models. It proposes a novel continual model merging algorithm for existing models.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core methodology does not involve the use of LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.