Flexible and Evolving Model Merging via Task Interference Modeling

Anonymous authors
Paper under double-blind review

Abstract

LLM model merging aims to build a single multi-task model from multiple existing fine-tuned models, but every practical method faces a tension between accuracy and storage. Monolithic merges are storage-cheap but suffer from inter-task interference; expert routers avoid interference but store N independent model weights. We argue that a principled middle ground exists: partition N tasks into K groups and keep one expert per group, trading accuracy against memory. The central problem then becomes grouping tasks to minimize intra-group interference, and we make two observations. First, we reveal that pairwise interference analysis is insufficient to characterize multi-task interactions: two tasks that exhibit only mild interference in pairwise merges may suffer severe degradation when a third task is introduced. Second, interference modeling should not be static: as more empirical evidence about task compatibilities accumulates, our understanding of task interference can *evolve* and improve. From these insights we build a two-stage **Task Interference Modeling (TIM)** approach. Stage 1 builds a task interference graph from pairwise and triple-merging analysis and produces an initial grouping via spectral clustering. Stage 2 *evolves* the partition by iteratively mining random clustering trials and applying correction moves drawn from three complementary strategies, guaranteeing monotonic improvement as more trial evidence is leveraged. We evaluate TIM on three vision benchmarks of increasing size (8, 14, and 20 tasks) and on six NLI tasks. TIM substantially outperforms similarity-based and random grouping baselines, recovering most of the gap between monolithic merging and per-task expert performance.

1 Introduction

Large pre-trained models serve as the foundation for an ever-growing ecosystem of task-specific fine-tuned variants. A single backbone can be adapted to hundreds of downstream tasks, each producing its own set of weights. Deploying all of them independently, however, is impractical: storage and serving costs scale linearly with the number of tasks, and the overhead quickly becomes prohibitive. Model merging (Ilharco et al., 2023) offers an elegant alternative: instead of keeping N separate models, combine their parameters in weight space to obtain one multi-task model, without any additional training or data.

A rich line of work has developed increasingly sophisticated merging operators—from task arithmetic and sign consensus (Yadav et al., 2023) to sparsification (Yu et al., 2024), SVD-based subspace alignment (Gargiulo et al., 2025; Panariello et al., 2025), and test-time adaptation (Yang et al., 2024)—that progressively narrow the accuracy gap between a single merged model and the full set of per-task experts. Yet a fundamental limit remains: when task updates conflict in parameter space, the merged model degrades on individual tasks. This is the well-known problem of *task interference*, and no monolithic approach has been able to fully resolve it. At the other extreme, router-based methods (Lu et al., 2024; Chen et al., 2025) sidestep interference entirely by keeping task parameters separate and routing inputs to the correct expert, but they pay the full cost of storing all N models.

Both extremes are wasteful in different regimes. A monolithic merge uses too little storage and loses too much accuracy; a router uses too much storage for negligible gains on compatible tasks. In this paper, we propose a *flexible* approach that utilizes $K < N$ experts,

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

where tasks with minimal mutual interference share the same expert. By partitioning N tasks into K groups of compatible tasks, our approach enables a user-controllable trade-off between average merging accuracy and storage overhead. The central question is: *how can we comprehensively model task interference so as to identify groups of compatible tasks?*

The naive approach is to measure pairwise interference—merge every pair, record accuracy drops, and group tasks with low pairwise conflict. This approach implicitly assumes that interference composes: if task A interferes less with B than with C, then A will also fare better in a group with B than in a group with C, regardless of what other tasks join. Our experiments show this assumption is wrong. We empirically reveal an *inconsistent interference accumulation* effect: two tasks that interfere mildly in pairwise merges can collapse when a third task is added, and conversely, tasks with strong pairwise conflicts can actually stabilize each other in triple merges. Pairwise information alone is therefore insufficient; any principled grouping method must account for higher-order interactions.

Furthermore, this interference modeling is static and therefore inevitably suboptimal. We observe, however, that random clusterings of these tasks provide a cheap, unbiased probe of the compatibility landscape: for *some* tasks, a single random trial may accidentally produce a better grouping than the static modeling scheme. The key insight is that we do not need random search to find the global optimum on its own. Instead, we can treat interference modeling as an *evolving process*: we start from a structural prior (Stage 1) and then iteratively *evolve* our understanding of which tasks are compatible by incorporating empirical evidence from random clustering trials. Each round of refinement adds new information about per-task compatibilities that the static graph missed, and the grouping improves monotonically as a result.

We present **Task Interference Modeling (TIM)**, a two-stage approach for *flexible* and *evolving* model merging that operationalizes both insights. **Stage 1** builds a task interference graph where edges are weighted by aggregated pairwise and triple-merging outcomes, so that each edge captures how two tasks interact *in the context of the full task set* rather than in isolation. Spectral clustering on this graph yields an initial grouping of N tasks into K groups—a static structural prior. **Stage 2** then *evolves* this grouping by iteratively mining random clustering trials for empirical evidence. From the subset of random trials that beat the current partition's average accuracy, we compute per-task improvement headroom, identify the most underperforming tasks, and apply count-preserving correction moves drawn from three complementary strategies. Critically, the interference model evolves alongside the partition: as more trial evidence accumulates and the grouping improves, the reference pool of informative trials re-computes, the headroom estimates sharpen, and subsequent corrections become increasingly targeted. Each accepted move strictly improves average accuracy, and the process converges when no further improvement is found.

We evaluate TIM on three vision-task suites of increasing size (8, 14, and 20 tasks on OpenCLIP ViT-B/32) following the protocol of Gargiulo et al. (2025), and on six NLI tasks with Llama-3-8B LoRA adapters. At every storage budget K, TIM substantially outperforms both random grouping and cosine-similarity-based grouping. The framework is agnostic to the base merging operator and works with TA, TSV, and Core-Space alike. To our knowledge, this is the first study to comprehensively model task interference.

2 Related Work

Monolithic Merging Methods. Training-free model merging combines fine-tuned task vectors without additional training (Frankle et al., 2020; Entezari et al., 2022). Task arithmetic (Ilharco et al., 2023) established the basic recipe; TIES-Merging (Yadav et al., 2023), DARE (Yu et al., 2024), TSV (Gargiulo et al., 2025), and Core-Space (Panariello et al., 2025) progressively reduce interference via sparsification, sign consensus, and SVD-based projection. All produce a single merged model with fixed storage cost, which is the limitation TIM addresses.

Router-Based Methods. To avoid interference entirely, router-based approaches keep all N task vectors and route at inference (Lu et al., 2024; Chen et al., 2025; Muqeeth et al., 2024), paying $N\times$ storage. TIM sits between monolithic merging and routers: $K < N$ experts, no extra training.

Task Interference. Multi-task learning has long studied interference via gradient conflicts (Sener and Koltun, 2018; Yu et al., 2020; Standley et al., 2020); in the merging setting, the closest signal is the Singular Task Interference score (Fifty et al., 2021; Gargiulo et al., 2025). However, STI essentially measures the correlation/orthogonality among tasks; consequently, a non-zero STI score cannot distinguish between interfering and compatible task relationships. In contrast, our approach explicitly models interference and compatibility.

3 Preliminaries

We first establish notation, then formalize the flexible merging problem, and finally present our two-stage solution TIM. Consider a pre-trained base model with parameters θ_{base}. Fine-tuning on task t yields task-specific parameters θ_t, and we define the *task vector* as the difference:

$$\tau_t = \theta_t - \theta_{\text{base}}. \tag{1}$$

Given N task vectors, a standard merging operator $\mathcal{M}$ combines them into a single update, and the merged model is:

$$\theta_{\text{merged}} = \theta_{\text{base}} + \lambda \cdot \mathcal{M}(\tau_1, \ldots, \tau_N), \tag{2}$$

where λ is a scaling coefficient. Examples of $\mathcal{M}$ include TA, TIES, DARE, and TSV.

3.1 Problem Formulation

Router-based merging methods keep each task vector as a separate expert and route inputs to the correct one, eliminating interference entirely at the cost of $N\times$ storage. Our goal is to reduce this storage overhead by grouping tasks: instead of N experts, we maintain $K < N$ experts, each serving a group of compatible tasks. The user chooses K to match their memory budget, yielding a controllable accuracy–storage trade-off.

Formally, let $\mathcal{T} = \{1, \ldots, N\}$ be the set of task indices and let $a_t(\theta)$ denote the accuracy of model θ on the validation set of task t. Given a partition $\mathcal{C} = \{G_1, \ldots, G_K\}$ of $\mathcal{T}$ into K disjoint groups, we merge the task vectors within each group using operator $\mathcal{M}$ to obtain expert k:

$$\theta^k_{\text{expert}} \;=\; \theta_{\text{base}} + \lambda_k \cdot \mathcal{M}(\{\tau_t : t \in G_k\}), \quad k = 1, \ldots, K. \tag{3}$$

Task $t \in G_k$ is evaluated on expert k, contributing accuracy $a_t(\theta^k_{\text{expert}})$.

The average per-task accuracy under partition $\mathcal{C}$ is:

$$\bar{a}(\mathcal{C}) \;\triangleq\; \frac{1}{N} \sum_{k=1}^{K} \sum_{t \in G_k} a_t\big(\theta^k_{\text{expert}}\big)\,. \tag{4}$$

At the boundaries, K=1 recovers *monolithic merging* methods and K=N recovers the *router-based merging* methods. The storage cost scales as K/N relative to the full expert set. Our goal is to find the partition that maximizes average accuracy at fixed K:

$$\begin{aligned} \mathcal{C}^\star \;&=\; \arg\max_{\mathcal{C}:|\mathcal{C}|=K} \bar{a}(\mathcal{C}) \\ &= \arg\max_{\mathcal{C}:|\mathcal{C}|=K} \frac{1}{N} \sum_{k=1}^{K} \sum_{t \in G_k} a_t\big(\theta^k_{\text{expert}}\big)\,. \end{aligned} \tag{5}$$

This is a combinatorial optimization over the Stirling-number-sized space of partitions. For N=20 and K=5 there are more than 10^8 possibilities, so exhaustive search is impossible.

4 Our Approach

We propose TIM, a two-stage approach to achieve flexible and evolving LLM model merging. **Stage 1** constructs a task interference graph using both pairwise and triple-merging

outcomes, then applies spectral clustering to produce an initial high-quality grouping (a *structural prior*). **Stage 2** then evolves this grouping by mining empirical evidence from random clustering trials, using iterative count-preserving corrections that become increasingly targeted as the interference model improves. Figure 1 illustrates the overall pipeline.

4.1 Stage 1: Building Task Interference Graph

Our first stage consists of three steps: (1) Task Interference Analysis. We measure interference between pairs of tasks through both pairwise and triple-merging experiments. (2) Graph Construction. We build a directed task interference graph with edge weights that aggregate pairwise and triple-merging evidence. (3) Spectral Clustering. We group tasks into K groups with minimal intra-group interference.

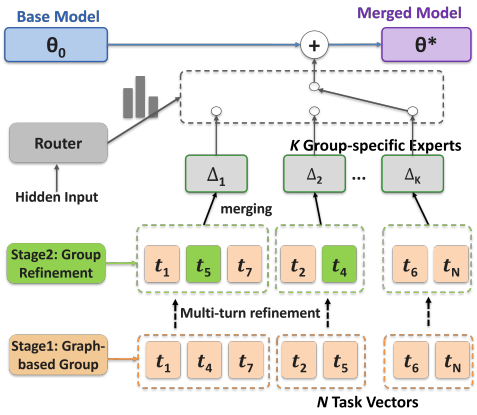


Figure 1: Overview of the TIM framework. Stage 1 builds a task interference graph from pairwise and triple-merging experiments and applies spectral clustering. Stage 2 refines the resulting partition by mining random trials for empirical evidence.

4.1.1 Pairwise and Triple Interference Analysis

Pairwise measurement. For each ordered pair of tasks (i, j), we merge the two task vectors with equal weight and evaluate the resulting model on task i's validation set:

$$\theta_{i\oplus j}^{(i)} = \theta_{\text{base}} + \frac{1}{2} \cdot (\tau_i + \tau_j), \tag{6}$$

yielding an accuracy $\text{acc}_{i|i,j}$ on $\mathcal{D}_i^{\text{val}}$.

This quantity directly reflects how compatible task j is with task i from i's perspective: the higher the accuracy, the less j interferes with i. Since interference is generally asymmetric, both directions are computed independently.

Triplet enrichment. Pairwise scores alone are insufficient because they cannot capture how a third task changes the interaction between two others. Two tasks that are benign in isolation can collapse catastrophically when a third task is added; conversely, two tasks with strong pairwise conflicts can stabilize each other in a larger group. We therefore enrich each pairwise measurement with the full set of triple-merging contexts.

For any triple (s, x, k) of distinct tasks, we merge all three task vectors and evaluate on the source task s:

$$\theta_{s\oplus x\oplus k}^{(s)} = \theta_{\text{base}} + \frac{1}{3} \cdot (\tau_s + \tau_x + \tau_k), \tag{7}$$

$$\text{acc}_{s|s,x,k} = \text{Acc}(\theta_{s\oplus x\oplus k}^{(s)}, \mathcal{D}_s^{\text{val}}). \tag{8}$$

For a fixed source s, we arrange all triple-merging results into a matrix $\mathbf{M}^{(s)}$ where each row corresponds to a third task k and each column to a target task x. The diagonal entry (no third task) is the pairwise accuracy. The final directed edge weight from target x to source s combines the pairwise signal with the aggregated triple-context signal:

$$\mathbf{A}_{s,x} = \alpha \cdot \underbrace{\text{acc}_{s|s,x}}_{\text{pairwise}} + \beta \cdot \frac{1}{N-2} \underbrace{\sum_{k\neq s,x} \text{acc}_{s|s,x,k}}_{\text{triple context}}, \tag{9}$$

where α and β balance the two contributions. Intuitively, the triple-merging sum answers the question: "how does x behave when paired with s *across the whole task set*?" rather than "how does x behave when paired with s in isolation?" The resulting adjacency matrix $\mathbf{A}$ is directed and asymmetric.

One could continue this hierarchy to 4-way merges or beyond. In practice, however, we find that moving from pairwise to triple yields large gains while moving from triple to quartet adds little at disproportionately higher cost, so we stop at order 3.

4.1.2 Graph Construction and Spectral Grouping

We view $\mathbf{A}$ as a directed compatibility graph: each node is a task, and the weight of edge $s \to x$ indicates how well task x fits with task s in context. Since standard spectral clustering requires an undirected affinity matrix, we symmetrize by averaging each edge with its reverse. Given the user-specified budget K, we compute the normalized graph Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$ (where $\mathbf{D}$ is the degree matrix), extract the K eigenvectors corresponding to the smallest eigenvalues to form a spectral embedding, and run K-means on this embedding to obtain the initial partition. This grouping serves as the *structural prior* that Stage 2 will further refine.

4.2 Stage 2: Evolving Refinement via Random-Trial Mining

The spectral grouping of Stage 1 produces a high-quality initial partition, but it is static: the graph is built once from a fixed set of merge experiments, and it cannot improve beyond what those experiments reveal. A single poorly weighted edge can propagate into a suboptimal assignment, and there is no mechanism to correct it short of re-running the entire graph construction. Empirically, even after Stage 1, a few tasks end up in groups where they underperform relative to what some other grouping would have achieved.

Our core idea is to make the interference analysis *evolving* rather than static. Instead of treating the grouping as a one-shot output, we treat it as a starting point that is iteratively refined as more empirical evidence about task compatibilities is incorporated. The agent of this evolution is a cheap batch of random clusterings—each one a uniformly random partition of N tasks into K groups, evaluated with the same per-group merging operator $\mathcal{M}$. Most random trials are worse than the structural prior, but a handful are better. Those better-than-prior trials are informative: they contain empirical evidence about which tasks benefit from being co-grouped and which do not, evidence that the static graph may have missed. We mine this evidence to iteratively evolve the partition.

Concretely, we draw R=50 random partitions per (N, K) configuration, evaluate each one with the per-group merging operator, and cache both per-task accuracies and group assignments. All trials are generated and evaluated upfront, before any refinement begins, so the cost is a one-time batch. The *evolving* nature of Stage 2 lies in *how* this evidence is used: the reference pool of informative trials, the per-task headroom estimates, and the set of correction targets all re-compute after every accepted move, so the refinement becomes increasingly targeted as our model of interference improves.

Reference Pool and Headroom Estimation Starting from the Stage 1 structural prior $\mathbf{g}^{(0)}$, we maintain a *reference pool* of random trials whose average accuracy exceeds that of the current partition. For each task, we estimate its *improvement headroom* as the best accuracy it attains in the pool minus its current accuracy, and weight it with a momentum term that favors tasks already improving in the previous round. The top tasks by weighted headroom become correction targets for each refinement round.

Count-Preserving Correction Moves Each correction move is a *swap*: a target task is moved to a different group and one task from the destination group is moved back, preserving the exact number of groups K. We design three complementary swap strategies—overlap matching, best-partner chasing, and weakest-out eviction—that exploit random-trial evidence along different axes (group structure, per-partner accuracy, and weakest-link removal). The first candidate that improves average accuracy is accepted, and the process repeats until convergence.

Because every accepted move strictly increases $\bar{a}(\mathbf{g})$, the process is monotonically improving and must converge in finite steps. The final partition $\mathbf{g}^{(\text{final})}$ satisfies $\bar{a}(\mathbf{g}^{(\text{final})}) \geq \bar{a}(\mathbf{g}^{(0)})$ by construction. Moreover, whenever the reference pool is non-empty, the final accuracy is at

least as high as the minimum pool accuracy observed: the evolving refinement can never do worse than the best random trial it started with, and it typically does substantially better by combining evidence across multiple trials.

4.2.1 Grouped Model Merging

After the two-stage grouping procedure yields the final partition $\mathbf{g}^{(\text{final})}$ with K groups, we merge the task vectors within each group using any standard merging algorithm $\mathcal{M}$ to obtain K experts:

$$\theta_k^{\text{expert}} = \theta_{\text{base}} + \lambda \mathcal{M}(\{\tau_i : i \in G_k\}), \quad k = 1, \ldots, K, \tag{10}$$

where G_k is the set of tasks assigned to group k.

Our framework is agnostic to the specific choice of $\mathcal{M}$. In our experiments, we incorporate several state-of-the-art merging algorithms, including Task Arithmetic (TA), TIES-Merging, DARE, TSV, and Core-Space Merging.

At inference time, each input is routed to the expert corresponding to its task. For multi-task inference where the task identity is unknown, a lightweight task classifier can be used, similar to router-based approaches.

5 Experiments

We evaluate TIM across both vision and NLP benchmarks, with four questions in mind: (1) How does TIM compare to existing merging methods when evaluated at matched storage budgets? (2) How does the accuracy–storage trade-off behave as we vary K across the full range from 1 to N? (3) What are the individual contributions of the triple-merging signal (Stage 1) and the evolving refinement (Stage 2)? (4) Does TIM work consistently across different base merging operators and modalities?

5.1 Experimental Setup

Vision benchmarks. Following Gargiulo et al. (2025), we evaluate on three vision-task suites of increasing size (8, 14, and 20 tasks), all built on OpenCLIP ViT-B/32. We use the publicly released full fine-tuned checkpoints from Gargiulo et al. (2025).

NLP benchmarks. For language experiments we follow the KnOTS protocol (Stoica et al., 2025) on six NLI datasets. The base model is Llama-3-8B.

Evaluation protocol. We report both absolute top-1 accuracy (averaged across all tasks in the suite) and normalized accuracy (per-task accuracy divided by the corresponding single-task expert accuracy, then averaged). Normalized accuracy provides a comparable recovery-rate metric across suites of different difficulty. The scaling coefficient λ is tuned on held-out validation data.

Baselines. We compare against a broad set of training-free merging methods: Task Arithmetic (TA) (Ilharco et al., 2023), TIES-Merging (Yadav et al., 2023), DARE (Yu et al., 2024), Consensus TA (Wang et al., 2024), TSV (Gargiulo et al., 2025), and Core-Space Merging (Panariello et al., 2025). We also include the zeroshot base model as a lower bound and the set of independent per-task experts as an upper bound. For storage-aware comparisons we additionally evaluate two grouping baselines: *Random Grouping* (random partition into K groups) and *Similarity Grouping* (single-linkage clustering on cosine similarity of flattened task vectors).

5.2 Main Results: Vision Tasks

Table 1 reports absolute and normalized accuracy of every baseline on the 8-, 14-, and 20-task suites. We instantiate TIM with two storage budgets, $K{=}N/2$ and $K{=}N/4$, to illustrate the trade-off.

Several patterns emerge from Table 1. First, methods cluster into two well-separated regimes. Monolithic merging methods such as TA and Core-Space use only $1/N$ of the storage but lose 25 to 35 normalized accuracy points. Router-based methods like Twin-Merging recover nearly all per-task expert accuracy but pay the full $N\times$ storage cost. TIM sits squarely between these two extremes, approaching router-level accuracy at a fraction of the storage. Second, the gap between TIM and the per-task expert upper bound shrinks as N grows: on 20 tasks, TIM with K=10 (50% storage) already reaches 98.57% normalized accuracy. This suggests storage-aware grouping becomes increasingly valuable as the number of tasks grows. Third, TIM outperforms both random and similarity-based grouping at every budget on every suite, confirming that both the higher-order interference graph and the evolving refinement contribute information beyond naive baselines.

Table 1: Main merging results across all three vision-task suites ($N = 8, 14, 20$). For each suite we report *absolute* accuracy (%, "Abs.") and *normalized* accuracy (%, "Norm.").

	$N=8$ **tasks**		$N=14$ **tasks**		$N=20$ **tasks**	
Method	**Abs.**	**Norm.**	**Abs.**	**Norm.**	**Abs.**	**Norm.**
Experts	86.53	100.00	90.32	100.00	90.91	100.00
TA	50.47	62.89	60.79	67.28	58.09	64.71
TIES	50.98	63.52	60.52	66.46	52.68	58.03
DARE	49.76	62.22	60.88	67.37	57.65	64.06
Conse. TA	49.80	62.20	60.25	66.71	57.32	63.01
TSV	53.05	65.73	76.62	84.30	73.39	80.72
Core-Space	53.78	64.44	76.29	83.95	73.25	80.45
Twin	73.12	89.21	88.51	98.80	88.09	98.50
Storage-aware ($K=N/2$):						
Random	89.30	96.29	87.16	96.20	87.94	96.55
Similarity	89.28	96.48	86.34	95.71	88.85	97.56
TIM	**91.39**	**98.47**	**89.22**	**98.54**	**89.72**	**98.57**
Storage-aware ($K=N/4$):						
Random	86.87	93.85	85.17	94.04	85.39	93.73
Similarity	85.30	92.34	83.38	92.13	85.22	93.46
TIM	**87.85**	**94.62**	**87.34**	**96.43**	**87.35**	**95.92**
Zeroshot	48.26	53.59	57.21	63.69	56.10	62.41

5.3 NLP Tasks Results

To test whether TIM transfers across modalities, we evaluate on the KnOTS six-task NLI benchmark (Stoica et al., 2025) with Llama-3-8B LoRA adapters. We use the same per-group merging pipeline as in the vision experiments, with TSV as the base operator.

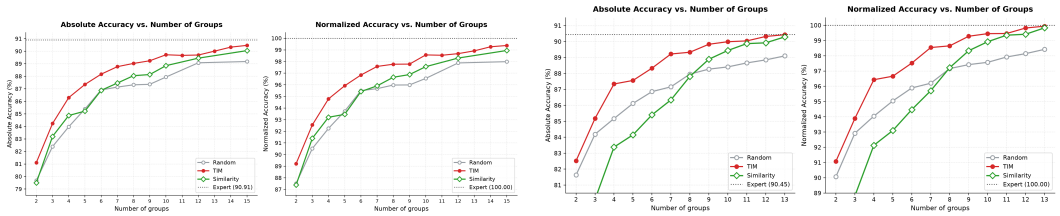
Results are shown in Table 2. At K=1 (all six adapters merged into one), interference is severe: Task Arithmetic loses nearly 19 normalized accuracy points, and even the strongest single-operator baseline (Core-Space) drops by more than 12 points. At K=2, TIM groups the six tasks into two experts and recovers a large portion of the lost accuracy, reaching 96.31% normalized accuracy and outperforming both random and similarity-based grouping by clear margins (+1.99 and +3.96 points, respectively). At K=3 (50% storage), TIM closes most of the remaining gap, achieving 98.28% normalized accuracy—within 1.72 points of the per-task expert upper bound. The relative pattern is consistent with what we observe on vision—higher-order interference matters, and evolving refinement adds value—suggesting that TIM's core insights are modality-independent.

Table 2: NLP results on the 6-task NLI suite. Abs. and norm. accuracy averaged across six datasets.

Method	**Abs.**	**Norm.**
Experts (upper bound)	92.53	100.00
Task Arithmetic	75.08	81.14
TSV	79.50	85.89
Core-Space	80.92	87.39
Storage-aware ($K=3$):		
Random	88.66	95.74
Similarity	89.08	96.19
TIM	**91.00**	**98.28**
Storage-aware ($K=2$):		
Random	87.34	94.32
Similarity	85.50	92.35
TIM	**90.10**	**96.31**

5.4 The Full Accuracy–Storage Trade-off Curve

A core motivation for TIM is that the storage budget should be tunable, not a binary choice between "one model" and "N models." Figure 2 plots the complete accuracy–storage trade-



(a) 20-task suite (b) 14-task suite

Figure 2: Accuracy–storage trade-off on the 20-task and 14-task suites. TIM outperforms both random and similarity grouping at every budget.

Table 3: Effect of the triple-merging term (β) on abs accuracy (%). Pairwise (β=0) uses only pairwise edge weights; full Stage 1 (β=1) uses Eq. (9).

	$N=14$		$N=20$	
Ablation	$K=7$	$K=4$	$K=10$	$K=5$
β=0	96.20	94.04	97.67	94.98
β=1	98.30	95.86	98.36	95.69
Δ	2.1	1.82	0.69	0.71

Table 4: Contribution of Stage 2 evolving refinement. "Stage 1 only" = spectral clustering with full edge weights. "TIM (full)" = Stage 1 + Stage 2 random-trial refinement. Numbers are normalized accuracy (%).

	$N=14$		$N=20$	
Ablation	$K=7$	$K=4$	$K=10$	$K=5$
Stage 1 only	98.30	95.86	98.36	95.69
TIM (full)	98.54	96.43	98.57	95.92
Δ	0.24	0.57	0.21	0.23

off on the 20-task and 14-task suites by sweeping K from small to large, comparing TIM against random grouping and cosine-similarity grouping.

Two findings stand out. First, TIM consistently beats both baselines at every K on every suite. The gap is widest at lower storage budgets ($K/N \lesssim 0.5$), where each grouping decision has a large impact and the baselines' cruder strategies do the most damage. As K increases toward N, all curves converge to the expert upper bound, and the gaps shrink. Second, similarity-based grouping is less reliable: it can perform comparably to random in some regimes and actually fall below it in others. TIM's curve, by contrast, is monotonic and smooth, reflecting the more principled grouping strategy.

5.5 Ablation: Higher-Order Signal and Evolving Refinement

We now dissect the contributions of the two key components of TIM: the *triple-merging term* in Stage 1 and the *evolving refinement* of Stage 2. Table 3 (left) reports the effect of adding the triple-merging signal, and Table 4 (right) reports the additional gain from Stage 2 refinement.

Table 3 shows that triple-merging improves accuracy at *every* (N, K) configuration tested, with gains ranging from +0.69 (at $(N{=}20, K{=}10)$) to +2.10 (at $(N{=}14, K{=}7)$). The improvement is substantially larger at N=14 ($\sim$+2 points) than at N=20 ($\sim$+0.7 points), suggesting that the pairwise-only baseline leaves more headroom for higher-order correction at moderate N. Within each N, the gain is comparable across the two K values, indicating that the triple-merging benefit is robust to the storage budget.

Table 4 shows that the evolving refinement contributes additional gains at *every* (N, K) configuration tested. The largest gain is +0.57 at $(N{=}14, K{=}4)$, with smaller but still positive contributions at the other configurations (+0.21 to +0.24). The refinement helps most at tighter storage budgets (smaller K, fewer groups), where each mis-grouping carries higher cost, and its contribution shrinks as K grows.

5.6 Why Pairwise Analysis Fails: Inconsistent Interference Accumulation

The ablation in §5.5 establishes that the triple term improves grouping quality. But *why* is pairwise information insufficient? In this section we substantiate the answer: task interference does *not* compose linearly as more tasks are added. This is the core empirical phenomenon that motivates TIM's higher-order design.

Table 5: Distribution of the accumulation gap $\Delta_{s;x,k}$ (Eq. (11)) over all 336 ordered triples on the 8-task suite.

Statistic	**Value**
Mean Δ	-0.033
Std of Δ	0.127
Min Δ	-0.360
Max Δ	$+0.251$
Collapse ($\Delta < -0.1$)	17.3%
Rescue ($\Delta > +0.1$)	8.6%
Corr($\hat{a}^{(2)}$, $a^{(3)}$)	0.659

Table 6: Concrete counter-examples of pairwise reasoning. *Case A*: pairwise predicts compatible, triple collapses. *Case B*: pairwise predicts interference, triple rescues. Numbers are absolute accuracy on task s.

Case	s	x	k	$\hat{a}^{(2)}$	$a^{(3)}$	Δ
Case A (collapse)						
A1	EuroSAT	DTD	SUN397	0.500	0.235	-0.265
A2	MNIST	GTSRB	SUN397	0.772	0.515	-0.257
Case B (rescue)						
B1	SUN397	Cars	RESISC45	0.470	0.550	$+0.080$
B2	SUN397	Cars	DTD	0.447	0.520	$+0.073$

For every ordered triple (s, x, k) of distinct tasks, we compute three quantities (all on task s's validation set, at $\lambda = 1$): (i) the pairwise accuracy of s merged with x, $a^{(2)}_{s,x}$; (ii) the triple accuracy of s merged with both x and k, $a^{(3)}_{s,x,k}$; and (iii) a pairwise-only baseline prediction $\hat{a}^{(2)}_{s;x,k} = \frac{1}{2}(a^{(2)}_{s,x} + a^{(2)}_{s,k})$, which is what one would guess for the triple accuracy by linearly averaging the two relevant pairwise scores.

We call the difference between actual and predicted triple accuracy the *accumulation gap*:

$$\Delta_{s;x,k} \triangleq a^{(3)}_{s,x,k} - \hat{a}^{(2)}_{s;x,k}. \tag{11}$$

If interference composed additively, Δ would be close to zero across the board. It is not.

Tables 5 and 6 summarize the distribution and provide concrete examples.

Three patterns emerge from the distribution (Table 5). First, the correlation between the pairwise prediction and the actual triple accuracy is only 0.659—far from 1, confirming that pairwise information is a poor predictor. Second, the gap goes both ways: there are both collapse cases ($\Delta \ll 0$) and rescue cases ($\Delta \gg 0$). Third, the magnitudes are substantial: the worst collapse loses 36 percentage points and the strongest rescue gains 25 percentage points relative to the pairwise baseline.

Table 6 gives concrete examples of both failure modes. Case A shows pairs that look benign in isolation but collapse catastrophically when a third task joins. Case B shows the opposite: pairs with strong pairwise conflicts can be partially rescued by a third task. These are not edge cases with tiny gaps—the differences exceed 25 points in the worst collapse example.

These examples directly undermine the implicit assumption behind any pairwise-based grouping scheme: that tasks which interfere little in pairs will continue to interfere little in larger groups. The existence of both collapse and rescue cases means that pairwise rankings can be wrong in either direction, and that the magnitude of the error can be large. This is precisely why TIM builds edge weights from triple-merging evidence: each edge then reflects how two tasks interact *in the context of the broader task set*, not in isolation. The accuracy gains documented in §5.5 follow directly from this design choice.

6 Conclusion

We have introduced TIM, a two-stage training-free model merging framework that finds task groupings by mining and *evolving* interference patterns. Stage 1 combines pairwise and triple-merging evidence to capture higher-order task interactions. Stage 2 iteratively evolves the grouping via random-trial mining, applying count-preserving correction moves that monotonically improve accuracy. TIM occupies the middle ground between monolithic merging and router-based merging, with a smooth accuracy–storage trade-off controlled by K. It is modular across merging operators and modality-agnostic, outperforming prior training-free methods at every storage budget on three vision suites and one NLP suite. To our knowledge, this is the first study to comprehensively model task interference.

7 AI DISCLOSURE

In this work, we used generative AI tools for "design or provide feedback on research experiments, implement methods". We have not used generative AI tools for "Generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results". Additionally, we used generative AI tools for "suggest experimental parameters, create or edit software code, draft parts of a research paper, edit a research paper to improve readability". We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Chen, Z., Zhou, Z., Zhang, B., Zhang, W., Sun, X., and Yan, J. (2025). Se-merging: A self-enhanced approach for dynamic model merging. Accepted at IJCNN 2025.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. (2022). The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations (ICLR)*.

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. (2021). Efficiently identifying task groupings for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. (2020). Linear mode connectivity and the lottery ticket hypothesis. In *International conference on machine learning. PMLR, 2020: 3259-3269*.

Gargiulo, A. A., Crisostomi, D., Bucarelli, M. S., Scardapane, S., Silvestri, F., and Rodolà, E. (2025). Task singular vectors: Reducing task interference in model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. (2023). Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Lu, Z., Fan, C., Wei, W., Qu, X., Chen, D., and Cheng, Y. (2024). Twin-merging: Dynamic integration of modular expertise in model merging.

Muqeeth, M., Liu, H., and Raffel, C. (2024). Soft merging of experts with adaptive routing. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Panariello, A., Marczak, D., Magistri, S., Porrello, A., Twardowski, B., Bagdanov, A. D., Calderara, S., and van de Weijer, J. (2025). Accurate and efficient low-rank model merging in core space. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sener, O. and Koltun, V. (2018). Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. (2020). Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning (ICML)*.

Stoica, G., Ramesh, P., Ecsedi, B., Choshen, L., and Hoffman, J. (2025). Model merging with svd to tie the knots. In *International Conference on Learning Representations (ICLR)*.

Wang, K., Dimitriadis, N., Ortiz-Jimenez, G., Fleuret, F., and Frossard, P. (2024). Localizing task information for improved model merging and compression. In *Proceedings of the 41st International Conference on Machine Learning (PMLR)*.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. (2023). Ties-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yang, E., Wang, Z., Shen, L., Liu, S., Guo, G., Wang, X., and Tao, D. (2024). Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. (2024). Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning (PMLR)*.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. (2020). Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.