PRIME: Ultra-Low-Rank Principal–Residual Model Merging

Seungho Lee[1], **Kyungsu Lee**[2], **Zuchi Bazarvaani**[1], **Jeongmin Ahn**[1], **Insuk Seo**[1], **Donghyeon Jeon**[3], **Inho Kang**[3], **Seung-Hoon Na**[1*]
[1]Ulsan National Institute of Science and Technology
{mina9853, sunset, jeongmin0209, insukvomn, nash}@unist.ac.kr
[2]Jeonbuk National University
ksl@jbnu.ac.kr
[3]Naver Corporation
{donghyeon.jeon, once.ihkang}@navercorp.com

Abstract

Model merging has emerged as an effective approach for integrating multiple task-specific fine-tuned models into a single unified model without requiring additional data-intensive training. A central challenge in model merging is to reduce task interference while preserving the task-specific capabilities of the original models. In this work, we propose **PRIME**, an *ultra-low-rank principal–residual model merging* framework that decomposes task vector merging into two complementary stages. First, *ultra-low-rank principal task vector merging* retains only a small fraction of singular vectors, effectively reducing task interference while preserving most of the task-specific performance. Second, *orthogonal residual task vector merging* incorporates the remaining components by projecting them onto the null space of the principal subspace, thereby avoiding interference while recovering additional task-relevant information. Extensive experiments on eight natural language processing tasks demonstrate that PRIME consistently outperforms existing model merging methods, achieving improvements of up to 1.18%p on T5 and 1.9%p on LLaMA-3.2-3B.

1 Introduction

In the era of the pretraining–finetuning paradigm, pretrained models are often adapted to multiple domains and tasks, resulting in a collection of task-specific fine-tuned models (Muqeeth et al., 2024). However, as the number of tasks increases, maintaining separate models becomes increasingly resource-inefficient.

Model merging has emerged as an effective alternative, constructing a single *merged* model without additional data-intensive training by aggregating task vectors, defined as the parameter differences between fine-tuned models and a shared base model (Yang et al., 2024a; Yadav et al., 2024). An effective merging strategy should satisfy two key objectives: (1) *reduction of task interference* (i.e., the *merging gap*) (Xiong et al., 2024), and (2) *retention of task-specific capabilities* (i.e., *task retention*) (Wei et al., 2025).

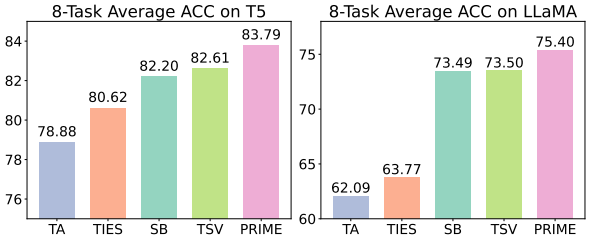

Figure 1: Performance comparison of model merging methods across eight NLP tasks. PRIME consistently achieves the highest average accuracy on both T5 and LLaMA-3.2-3B models. Note that the evaluation tasks differ between T5 and LLaMA.

Prior work has explored pruning-based methods (Yadav et al., 2023; Yu et al., 2024), optimization-based approaches that enforce task arithmetic (Ortiz-Jimenez et al., 2023; Jin et al., 2023; Wei et al., 2025; Xiong et al., 2024; Cheng et al., 2025), and orthogonalization-based methods based on singular value decomposition (SVD) (Gargiulo et al., 2025; Tang et al., 2025b; Choi et al., 2025). Among them, Task Singular Vectors (TSV) (Gargiulo et al., 2025) provide a principled framework for mitigating singular task interference via whitening and subspace decorrelation.

Despite these advances, two fundamental challenges remain unresolved: (1) whether task interference can be further mitigated under an ultra-low-rank regime that retains only a minimal subset of dominant components, and (2) how the discarded components can be systematically exploited to preserve task-specific capabilities that are otherwise degraded by aggressive truncation.

To address these challenges, we propose **PRIME**

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 3415–3436

(**p**rincipal–**r**es**i**dual **me**rging), an ultra-low-rank model merging framework that decomposes merging into two complementary stages:

1. **Ultra-low-rank principal task vector merging** (*ultra-principal merging*) retains only a small subset of dominant singular vectors. This aggressive rank reduction effectively alleviates task interference while preserving most of the task-relevant information.

2. **Orthogonal residual task vector merging** (*orthogonal residual merging*) recovers the remaining information by projecting residual task vectors onto the null space of the principal representation. This enables additional task-specific information to be incorporated without introducing interference.

By explicitly separating interference minimization and retention recovery, PRIME provides a structured solution to the interference–retention trade-off.

Extensive experiments on eight diverse natural language processing (NLP) tasks demonstrate that PRIME consistently and robustly outperforms existing merging methods across different model architectures, achieving performance improvements of up to 1.18%p on T5 (Raffel et al., 2020) and 1.9%p on LLaMA-3.2-3B (Touvron et al., 2023a,b; Dubey et al., 2024). These results highlight the effectiveness and general applicability of PRIME under both encoder–decoder and decoder-only settings.

2 Related Works

Model-parameter-based merging Model merging aims to integrate multiple expert models into a single model, with simple parameter averaging serving as a fundamental baseline. Prior work improves this baseline by introducing more informed aggregation strategies. For example, Fisher-Merging (Matena and Raffel, 2022) and RegMean (Jin et al., 2023) replace isotropic averaging with Fisher-information–weighted and inner-product–based rules, respectively. Other approaches, such as Model Soups (Wortsman et al., 2022) and LoRASoup (Prabhakar et al., 2024), show that weight averaging can even improve generalization. Beyond direct averaging, Model Fusion via Optimal Transport (Singh and Jaggi, 2020) aligns neurons before merging to improve compatibility. However, these approaches often rely on parameter-level aggregation and do not explicitly address structured task interference.

Task-vector-based merging Task Arithmetic (Ilharco et al., 2023) introduces task vectors as parameter differences between fine-tuned and base models. While intuitive, directly composing task vectors suffers from inter-task interference, leading to performance degradation (Xiong et al., 2024). Several works attempt to mitigate this issue at inference time, such as AdaMerging (Yang et al., 2023) and Representation Surgery (Yang et al., 2024b), which introduce task- or layer-specific adaptations. However, these approaches typically require additional tuning or auxiliary components.

Pruning-based conflict mitigation Pruning-based methods reduce interference by selectively removing conflicting parameters. TIES-Merging (Yadav et al., 2023) proposes a trim–elect–merge procedure, while DARE (Yu et al., 2024) aggressively prunes redundant components and rescales the remainder. DELLA (Deep et al., 2024) combines these strategies for improved robustness. Additional approaches, such as Model Breadcrumbs (Davari and Belilovsky, 2024) and Consensus Merging (Wang et al., 2024), further remove outlier weights. While effective, these methods rely on sparsification and may discard useful task-specific information.

SVD- and subspace-based merging Orthogonalization-based methods leverage singular value decomposition (SVD) to reduce interference in a structured manner. Task Singular Vectors (TSV) (Gargiulo et al., 2025) provide a principled framework by decorrelating task-specific singular vectors via whitening, achieving strong empirical performance. Subsequent works explore subspace alignment and decomposition, such as Isotropic-Merging (Marczak et al., 2025), KnOTs (Stoica et al., 2024), and DOGE (Wei et al., 2025). Despite their effectiveness, these methods primarily operate within the retained principal subspace and do not explicitly address how truncated components can be utilized for improving task retention.

LoRA merging Recent works extend merging to lightweight adapter modules such as LoRA (Hu et al., 2022), enabling efficient integration without modifying full model weights (Prabhakar et al., 2025; Zhao et al., 2025; Salami et al., 2025; Tang et al., 2023; Su et al., 2025).

3 Preliminary

Following prior work (Jin et al., 2023; Xiong et al., 2024; Cheng et al., 2025), we focus on linear layers when constructing task vectors. Without loss of generality, we denote the weight matrix of a linear layer as $\boldsymbol{\theta} \in \mathbb{R}^{d' \times d}$.

3.1 Background: Model Merging

Formally, let $\boldsymbol{\theta}_0 \in \mathbb{R}^{d' \times d}$ denote the parameters of a pretrained model, and let $\boldsymbol{\theta}_i$ denote the parameters of the model fine-tuned on the i-th task. The task vector $\boldsymbol{\tau}_i$ for task i is defined as the parameter offset between the fine-tuned and pretrained models (Ilharco et al., 2023): $\boldsymbol{\tau}_i = \boldsymbol{\theta}_i - \boldsymbol{\theta}_0$.

Given n tasks, the merged task vector $\boldsymbol{\tau}_m$ is obtained by applying a merging function $\mathsf{Merge}(\cdot)$ to the set of task vectors $\{\boldsymbol{\tau}_i\}_{i=1}^n$:

$$\boldsymbol{\tau}_m = \mathsf{Merge}(\{\boldsymbol{\tau}_i\}_{i=1}^n; \boldsymbol{\theta}_0)\,.$$

The parameters of the merged model are then given by $\boldsymbol{\theta}_m = \boldsymbol{\theta}_0 + \boldsymbol{\tau}_m$.

Task Arithmetic Following (Ortiz-Jimenez et al., 2023), *Task Arithmetic* defines a simple linear merging rule, denoted by $\mathsf{Merge}_{\mathrm{TA}}$, as

$$\boldsymbol{\tau}_m = \sum_{i=1}^{n} \alpha_i \boldsymbol{\tau}_i,$$

where α_i is the merging coefficient for task i.

Merging Gap For simplicity, we adopt Task Arithmetic as a representative merging method. Following (Xiong et al., 2024; Wei et al., 2025), the *merging gap* measures how well $\mathsf{Merge}_{\mathrm{TA}}$ preserves the Task Arithmetic property for task i:

$$J_i \;=\; \left(\mathcal{L}_i\left(\boldsymbol{\theta}_0 + \sum_{j=1}^{n} \alpha_j \boldsymbol{\tau}_j\right) - \mathcal{L}_i(\boldsymbol{\theta}_0 + \alpha_i \boldsymbol{\tau}_i)\right)^2, \tag{1}$$

Inner-Product Magnitude as Task Interference We quantify task interference using the magnitude of the Frobenius inner product between two task vectors:

$$\rho_{ij} = \left|\langle \boldsymbol{\tau}_i,\, \boldsymbol{\tau}_j \rangle_F\right|. \tag{2}$$

The following theorem establishes a direct connection between this interference measure and the merging gap.

Theorem 1 (Zero Inner-Product Magnitude Implies Zero Merging Gap)**.** *Under the Task Arithmetic merging rule* $\mathsf{Merge}_{\mathrm{TA}}$*, if* $\rho_{ij} = 0$ *for all* $i \neq j$*, then the merging gap for the* i*-th task satisfies* $J_i = 0$*.*

A complete proof is provided in Appendix A.1. This result is closely related to the analysis in (Xiong et al., 2024).

3.2 Trade-off between Task Interference and Task Retention with Respect to Rank r

Rank-r Truncated SVD For a task vector $\boldsymbol{\tau}_i$, let its rank-r truncated SVD approximation be

$$\boldsymbol{\tau}_i^{[:r]} = \boldsymbol{U}_i^{[:r]} \boldsymbol{\Sigma}_i^{[:r]} \left(\boldsymbol{V}_i^{[:r]}\right)^\top, \tag{3}$$

where only the top-r singular values are retained.

In this subsection, we analyze the expected effect of decreasing the rank r in Eq. (3). In particular, reducing r induces an inherent trade-off between task interference and task retention: smaller ranks can suppress interference among task vectors, but may also discard task-specific information. This observation motivates the need for a merging strategy that balances both effects.

Task Interference with Respect to Rank r Given a sufficiently small rank $r \ll \min(d', d)$, truncating task vectors to rank r is expected to reduce task interference. In particular, the magnitude of the Frobenius inner product between two truncated task vectors satisfies

$$\begin{aligned} \left|\left\langle \boldsymbol{\tau}_i^{[:r]},\, \boldsymbol{\tau}_j^{[:r]} \right\rangle_F\right| &= \|\boldsymbol{\tau}_i^{[:r]}\|_F \|\boldsymbol{\tau}_j^{[:r]}\|_F \,|\cos(\theta_{ij})| \\ &\leq \|\boldsymbol{\tau}_i^{[:r]}\|_F \|\boldsymbol{\tau}_j^{[:r]}\|_F. \end{aligned} \tag{4}$$

where $\cos(\theta_{ij})$ denotes the cosine similarity between the two truncated task vectors.

Moreover, since $\|\boldsymbol{\tau}_i^{[:r]}\|_F = \|\boldsymbol{\Sigma}_i^{[:r]}\|_F$, the upper bound in Eq. (4) increases monotonically with respect to r. Therefore, from this upper-bound perspective, smaller values of r are expected to reduce task interference due to stronger low-rank pruning.

Task Retention with Respect to Rank r However, reducing the rank r also increases the approximation error relative to the original task vector, which may degrade task-specific ability. To formalize this effect, analogous to the merging gap, we define the *task retention gap* for task i as follows:

Definition 1 (Task Retention Gap)**.** Let $\boldsymbol{\tau}_i'$ be an approximation of $\boldsymbol{\tau}_i$. The task retention gap for task i is defined as

$$G_i \;=\; \Big(\mathcal{L}_i(\boldsymbol{\theta}_0+\boldsymbol{\tau}_i') - \mathcal{L}_i(\boldsymbol{\theta}_0+\boldsymbol{\tau}_i)\Big)^2. \tag{5}$$

We obtain the following result.

Theorem 2 (Decreasing Rank Increases Task Retention Gap)**.** *Let $\boldsymbol{\tau}_i^{[:r]}$ be the rank-r truncated SVD approximation defined in Eq. (3). Then, as r decreases, the task retention gap in Eq. (5) with $\boldsymbol{\tau}_i' = \boldsymbol{\tau}_i^{[:r]}$ increases monotonically.*

A complete proof is provided in Appendix A.2.

3.3 TSV-Merging

Building on the rank-r truncated SVD formulation introduced in Eq. (3), TSV-Merging consists of two main steps: (i) extracting principal task vectors via low-rank truncation, and (ii) whitening the concatenated task vectors to reduce inter-task correlation.

For each task vector $\boldsymbol{\tau}_i$, TSV-Merging first uses its rank-r approximation $\boldsymbol{\tau}_i^{[:r]}$ as the principal task vector. The corresponding left singular vectors, singular value matrices, and right singular vectors are then concatenated across tasks:

$$\begin{aligned}
\boldsymbol{U}_s &= [\boldsymbol{U}_1^{[:r]} \mid \boldsymbol{U}_2^{[:r]} \mid \cdots \mid \boldsymbol{U}_n^{[:r]}],\\
\boldsymbol{\Sigma}_s &= [\boldsymbol{\Sigma}_1^{[:r]} \mid \boldsymbol{\Sigma}_2^{[:r]} \mid \cdots \mid \boldsymbol{\Sigma}_n^{[:r]}],\\
\boldsymbol{V}_s &= [\boldsymbol{V}_1^{[:r]} \mid \boldsymbol{V}_2^{[:r]} \mid \cdots \mid \boldsymbol{V}_n^{[:r]}].
\end{aligned} \tag{6}$$

Next, a whitening procedure is applied to the concatenated principal singular vectors:

$$\begin{aligned}
\hat{\boldsymbol{U}}_s &= \mathsf{Whitening}(\boldsymbol{U}_s),\\
\hat{\boldsymbol{V}}_s &= \mathsf{Whitening}(\boldsymbol{V}_s).
\end{aligned} \tag{7}$$

Following Gargiulo et al. (2025), for a matrix $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, the whitening operator is defined as

$$\mathsf{Whitening}(\boldsymbol{W}) = \boldsymbol{U}\boldsymbol{V}^\top. \tag{8}$$

Finally, the merged task vector is reconstructed as

$$\boldsymbol{\tau}_m = \hat{\boldsymbol{U}}_s \boldsymbol{\Sigma}_s \hat{\boldsymbol{V}}_s^\top. \tag{9}$$

4 PRIME-Merging

4.1 Adaptive Rank Selection

Based on the analysis in Section 3 and Appendices A.1 and A.2, the merging interference term and retention loss term can be approximated as

$$\begin{aligned}
J_i(r) &\approx \left(\sum_{j\neq i} \left\langle \boldsymbol{\tau}_i^{[:r]}, \boldsymbol{\tau}_j^{[:r]} \right\rangle_F\right)^2,\\
G_i(r) &\approx \left(\|\boldsymbol{\Sigma}_i^{[r:]}\|_F^2\right)^2.
\end{aligned} \tag{10}$$

Here, $J_i(r)$ measures the interference introduced by merging within the retained principal subspace, while $G_i(r)$ quantifies the information loss caused by truncating singular components beyond rank r.

Using these two quantities, we define the total interference–retention objective:

$$F(r) = \sum_i \left(J_i(r) + G_i(r)\right). \tag{11}$$

Substituting Eq. (10), we obtain

$$F(r) \approx \sum_i \left[\left(\sum_{j\neq i} \left\langle \boldsymbol{\tau}_i^{[:r]}, \boldsymbol{\tau}_j^{[:r]} \right\rangle_F\right)^2 + \left(\|\boldsymbol{\Sigma}_i^{[r:]}\|_F^2\right)^2\right]. \tag{12}$$

Although $F(r)$ is not necessarily globally convex, its two components exhibit opposite tendencies with respect to r. As r increases, the retention loss $G_i(r)$ monotonically decreases, while the interference term $J_i(r)$ generally increases as more shared principal directions are retained. This naturally induces a trade-off structure, and empirically $F(r)$ often exhibits a stable local minimum in the ultra-low-rank regime.

We therefore estimate the principal rank as

$$r^* = \arg\min_r F(r). \tag{13}$$

Stable coarse-grained search. To avoid sensitivity to small local fluctuations, we evaluate $F(r)$ over the discrete candidate set

$$\mathcal{S} = \{\, \beta d k \mid k = 1, \ldots, \lfloor 1/\beta \rfloor \,\} \cup \{1\}, \tag{14}$$

where d is the layer dimension and β is the search interval ratio. In all experiments, we use $\beta = 0.01$.

We update the running minimum only when

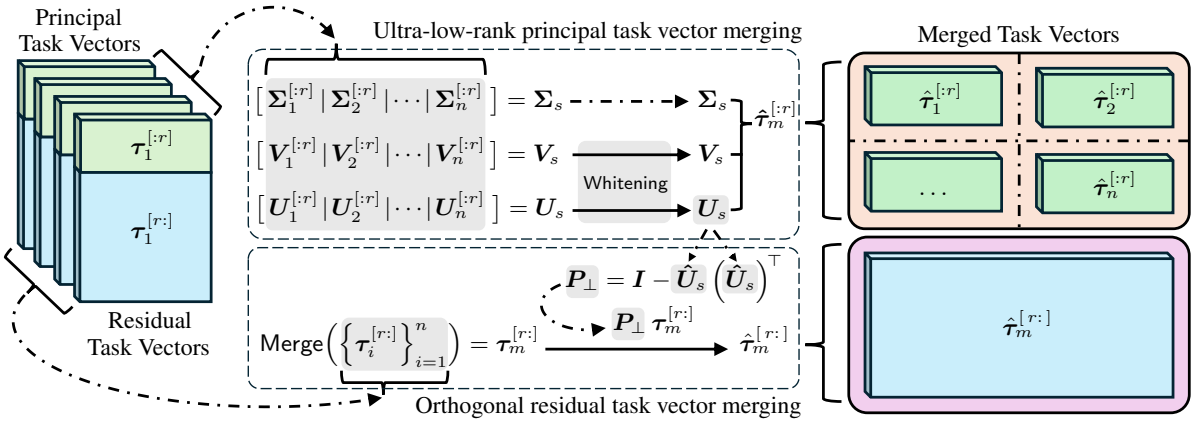

Figure 2: Figure 2: Overview of PRIME-Merging. Each task vector is decomposed into a principal component and a residual component. The principal components are jointly merged after whitening to reduce interference within the shared low-rank subspace. The residual components are then merged in the orthogonal complement of the principal subspace, preserving additional task-specific information without reintroducing conflicts. By combining these two stages, PRIME reduces interference while improving task retention.

$$F(r_{\text{new}}) < F_{\min} + \gamma, \tag{15}$$

where

$$\gamma = \kappa F_{\min}, \tag{16}$$

and $\kappa = 0.025$. This tolerance rule prevents overreaction to negligible numerical differences and improves robustness.

Feasible neighborhood. After obtaining r^*, we additionally consider a nearby feasible region

$$r \in [\max(1, r^* - \beta d),\ r^* + \beta d], \tag{17}$$

which allows lightweight local refinement around the estimated optimum.

Shared rank across layers. For simplicity, we use a single shared rank across layers by minimizing the layer-averaged objective $F(r)$. The resulting adaptive rank selections are reported in Appendix B. For the main experiments, however, we fix the rank to 1% across all settings to ensure a consistent and directly comparable evaluation protocol.

4.2 Ultra-Low-Rank Principal Task Vector Merging

In **PRIME-Merging**, motivated by the analysis in Section 3.2, we employ an *ultra-low-rank principal task vector* with a very small rank r to substantially reduce task interference.

Rather than relying on a fixed heuristic choice, the appropriate ultra-low-rank regime can be estimated through the adaptive rank selection criterion introduced in Section 4.1. In our main experiments, however, we use a fixed rank of approximately 1% of the original dimensions across all models for a consistent and directly comparable evaluation protocol.

Given the ultra-low-rank principal task vectors, we apply TSV using Eqs. (3–9) to obtain the corresponding ultra-low-rank *merged* principal vector, as defined in Eq. (9).

We note that the whitening step in TSV directly reduces task interference. To examine how interference is reduced while preserving task-specific information after whitening, we define the *whitened principal task vectors*, obtained by reconstructing the task-specific components from the whitened left and right singular vectors in Eq. (7), as follows:

$$\hat{\boldsymbol{\tau}}_i^{[:r]} = \hat{\boldsymbol{U}}_s^{[tr:(t+1)r]} \boldsymbol{\Sigma}_s^{[tr:(t+1)r]} \left(\hat{\boldsymbol{V}}_s^{[tr:(t+1)r]} \right)^\top. \tag{18}$$

As discussed in Section 4.1, smaller principal ranks generally reduce cross-task interference, supporting the use of an ultra-low-rank principal stage for conflict mitigation. The discarded components are subsequently handled in the residual stage, enabling complementary recovery of task-specific information.

Figures 3 and 4 illustrate on T5 that the proposed ultra-low-rank principal stage substantially compresses task vectors while preserving task-specific

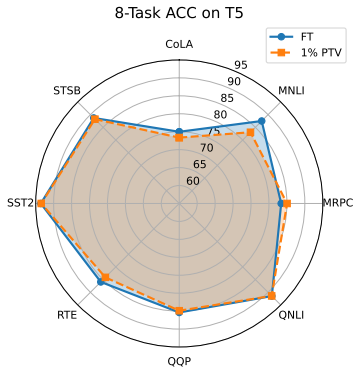

Figure 3: Performance comparison between the original task vectors and the ultra-low-rank principal task vectors when merging eight NLP tasks on T5. Despite aggressive rank truncation, the principal task vectors preserve nearly all task-specific utility, retaining approximately 99% of the original performance on average.

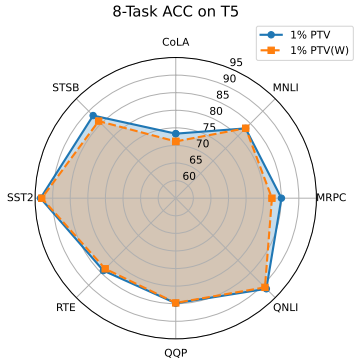

Figure 4: Performance comparison before and after whitening of the ultra-low-rank principal task vectors when merging eight NLP tasks on T5. Whitening preserves task performance while preparing the principal vectors for interference-aware merging, with approximately 99% average performance retention.

performance, and that the subsequent whitening step maintains this utility without introducing noticeable degradation. These observations support the use of a compact principal representation for interference mitigation prior to residual integration. Corresponding results for LLaMA-3.2-1B and LLaMA-3.2-3B are provided in Sections C and D, respectively.

4.3 Orthogonal Residual Task Vector Merging

While the ultra-low-rank principal stage effectively mitigates task interference, aggressive truncation may discard task-specific information. To compensate for this retention loss, PRIME additionally performs *orthogonal residual task vector merging*.

The key idea is to merge the discarded residual components separately and inject them only through directions orthogonal to the principal merged subspace. This allows complementary information recovery while avoiding reintroduction of conflicts already controlled in the principal stage.

Formally, the residual task vector of task i is defined as

$$\boldsymbol{\tau}_i^{[r:]} = \boldsymbol{\tau}_i - \boldsymbol{\tau}_i^{[:r]}. \tag{19}$$

These residual vectors are merged and projected onto the orthogonal complement of the whitened principal subspace:

$$\begin{aligned} \boldsymbol{P}_\perp &= \boldsymbol{I} - \hat{\boldsymbol{U}}_s \left(\hat{\boldsymbol{U}}_s\right)^\top, \\ \hat{\boldsymbol{\tau}}_m^{[r:]} &= \boldsymbol{P}_\perp \left(\mathsf{Merge}\left(\{\boldsymbol{\tau}_i^{[r:]}\}_{i=1}^n\right)\right). \end{aligned} \tag{20}$$

Thus, the residual update is constrained to directions that do not overlap with the retained principal basis. As shown in Appendix A.3– A.4,

$$\boldsymbol{P}_\perp \hat{\boldsymbol{\tau}}_i^{[:r]} = \boldsymbol{0}, \qquad \boldsymbol{P}_\perp \hat{\boldsymbol{U}}_l = \boldsymbol{0}. \tag{21}$$

These properties ensure that the residual component is injected without interfering with the principal merged representation.

While various strategies can be used in $\mathsf{Merge}(\cdot)$, our default choice is simple averaging:

$$\boldsymbol{\tau}_m^{[r:]} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\tau}_i^{[r:]}. \tag{22}$$

Experimental comparisons with alternative residual merging strategies are reported in Table 4.

Finally, PRIME combines the principal merged vector and the orthogonally projected residual merged vector:

$$\hat{\boldsymbol{\tau}}_m = \hat{\boldsymbol{\tau}}_m^{[:r]} + \hat{\boldsymbol{\tau}}_m^{[r:]}. \tag{23}$$

This principal–residual decomposition enables simultaneous interference mitigation and task information recovery within a unified merging framework.

The interference reduction effect of PRIME can be further verified through the cosine similarities among the resulting principal–residual task vectors, reported in Sections E, F, and G for T5, LLaMA-3.2-1B, and LLaMA-3.2-3B, respectively. A complete description of the PRIME procedure is provided in Section H.

Type	Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST2	STSB	Avg.
Original	Base	69.13	56.45	76.23	88.45	82.12	80.14	91.17	62.19	75.74
Individual	FT	74.98	87.50	83.41	91.49	85.37	85.92	93.58	88.70	86.37
Merging	TA	69.13	62.65	79.41	89.80	83.86	81.23	91.74	73.21	78.88
	TIES	71.43	68.49	80.15	90.06	83.07	80.14	91.74	79.85	80.62
	DARE	69.22	63.17	79.66	89.88	83.88	80.87	91.74	75.52	79.24
	DELLA	70.85	48.86	83.33	89.49	83.18	78.70	91.40	82.01	78.48
	Iso-C	69.22	58.04	77.45	88.49	83.08	80.14	91.17	65.07	76.58
	Iso-CTS	69.13	58.66	77.70	88.56	83.20	80.14	91.28	65.65	76.79
	SB	70.28	75.70	79.41	**90.92**	**84.56**	**83.39**	**93.23**	80.07	82.20
	TSV	70.47	78.21	79.41	90.74	84.32	83.75	92.78	81.18	82.61
	PRIME	**71.91**	**82.00**	**81.37**	90.88	83.63	81.23	93.00	**86.26**	**83.79**

Table 1: Performance of merging T5 models fine-tuned on eight NLP tasks. For T5, whose task vectors satisfy $\boldsymbol{\tau}_i \in \mathbb{R}^{d' \times d}$, **PRIME** operates in an ultra-low-rank regime, where the principal rank is determined according to the model dimensions.

Method	Weighted Sum	Prune Ratio	Top Rank Ratio	Common Ratio	Beta
TA	$\frac{1}{n}$	/	/	/	/
TIES	/	80%	/	/	/
DARE	/	80%	/	/	/
DELLA	/	80%	/	/	/
Iso-C	/	/	/	/	/
Iso-CTS	/	/	/	80%	/
SB	/	/	/	/	0.0
TSV	/	/	$\frac{1}{n}$	/	/
PRIME	/	/	1%	/	/

Table 2: Merging hyperparameters for TA (Task Arithmetic), TIES, DARE, DELLA, Iso-C, Iso-CTS, SB (Subspace Boosting), TSV, and PRIME. For methods involving randomness during merging, the random seed was fixed to 42.

5 Experiments

5.1 Settings

Environment All experiments were conducted on NVIDIA RTX A6000 D6 48GB GPUs (NVIDIA Corporation, 2021), which provided sufficient computational resources for model merging and evaluation across all experimental settings.

Models We conduct experiments on T5 (Raffel et al., 2020) and LLaMA-3.2 models with 1B and 3B parameter scales (Touvron et al., 2023a,b; Dubey et al., 2024). Details of the fine-tuned models are provided in Appendix I, J, K, and L.

Datasets The T5 and LLaMA-3.2 models are trained on two different sets of eight tasks. The T5 model uses the GLUE (Wang et al., 2018) task pool, while the LLaMA-3.2 model relies on a custom task pool.

Merging Hyperparameters For TA (Ilharco et al., 2023), we use uniform weighting with $\frac{1}{n}$. For TIES (Yadav et al., 2023), 80% of task vector components are pruned by magnitude; the same setting is used for DARE and DELLA. For Iso-CTS (Marczak et al., 2025), the common subspace ratio is set to 80%. For SB (Anonymous, 2025), the *Beta* parameter is set to 0.0. For TSV (Gargiulo et al., 2025), the retained rank is set to $\frac{1}{n}$ of the full rank.

5.2 Result on T5

We conduct merging experiments using the T5 model, which follows the standard Transformer architecture consisting of both an encoder and a decoder (Vaswani et al., 2017). Given the relatively small model size of T5 compared to recent large-scale language models, we evaluate its performance on the *GLUE* benchmark. GLUE comprises a collection of NLP tasks with moderate difficulty and has been widely used to assess the general language understanding capabilities of encoder–decoder models. This benchmark therefore provides a suitable and well-established evaluation setting for analyzing the effectiveness of our merging approach on T5.

As shown in Table 1, PRIME-Merging achieves superior average performance compared to the baseline approaches. In particular, it yields performance improvements of 1.59%p over SB and 1.18%p over TSV, demonstrating that PRIME-Merging is effective for merging T5 models with an encoder–decoder architecture.

Type	Model	MMLU	HS	AGN	CQA	MNLI	WG	MRPC	WiC	Avg.
Original	Base	30.84	27.19	63.89	51.35	35.45	50.59	63.71	49.79	46.60
Individual	FT	61.37	93.09	95.11	79.03	90.53	85.71	87.19	75.07	83.39
Merging	TA	52.98	63.93	85.74	68.47	48.00	59.83	66.78	51.00	62.09
	TIES	51.92	70.62	85.46	71.42	40.45	73.95	67.94	48.43	63.77
	DARE	53.51	64.53	86.01	68.63	48.00	60.14	66.96	51.00	62.35
	DELLA	51.19	72.79	85.89	71.99	36.81	76.16	69.86	48.43	64.14
	Iso-C	51.25	60.77	82.50	63.47	43.79	50.91	66.78	49.93	58.68
	Iso-CTS	50.69	61.07	82.57	61.43	44.40	50.04	66.55	50.21	58.37
	SB	**55.81**	84.38	**91.63**	75.76	74.41	79.95	71.94	54.00	73.49
	TSV	54.93	83.10	91.18	75.84	72.62	80.74	76.00	53.57	73.50
	PRIME	54.73	**85.52**	91.16	**76.90**	**75.43**	**82.40**	**79.01**	**58.07**	**75.40**

Table 3: Performance of merging LLaMA-3.2-3B models fine-tuned on eight NLP tasks. For LLaMA-3.2-3B, whose task vectors satisfy $\boldsymbol{\tau}_i \in \mathbb{R}^{d' \times d}$, **PRIME** operates in an ultra-low-rank regime, where the principal rank is determined according to the model dimensions.

Type	Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST2	STSB	Avg.
Original	Base	69.13	56.45	76.23	88.45	82.12	80.14	91.17	62.19	75.74
Individual	FT	74.98	87.50	83.41	91.49	85.37	85.92	93.58	88.70	86.37
Merging	PRIME	71.91	82.00	81.37	90.88	83.63	81.23	93.00	86.26	83.79
	PRIME(TIES)	71.81	81.45	79.66	90.77	83.46	81.59	93.23	86.36	83.54
	PRIME(S-SB)	72.67	82.22	81.37	90.88	83.71	81.23	93.58	86.71	84.05

Table 4: This table reports the performance obtained by varying the $\mathsf{Merge}(\cdot)$ operation in Eq. (20) on T5 models. S-SB denotes **Scaled-Subspace Boosting**, which is described in Appendix M and N.

5.3 Result on LLaMA-3.2

We conduct merging experiments using the LLaMA-3.2 model, which adopts a decoder-only transformer architecture. Compared to the T5 model, the LLaMA-3.2-1B and 3B variants are substantially larger, enabling evaluation on more challenging and diverse benchmarks beyond GLUE. Accordingly, we assess their performance on datasets such as MMLU, HellaSwag, and WiC, which are commonly used to evaluate reasoning and generalization capabilities in large language models. For the evaluation of decoder-only models, we follow the standardized evaluation protocol provided by the *lm-evaluation-harness* (Gao et al., 2024).

As shown in Table 3, **PRIME-Merging** achieves superior average performance compared to the baseline approaches. In particular, it outperforms SB by 1.91%p and TSV by 1.90%p. These results indicate that PRIME-Merging is effective not only for the T5 model with an encoder–decoder architecture but also for the LLaMA-3.2 model with a decoder-only architecture. Results for the LLaMA-3.2-1B model are provided in Appendix O.

5.4 Merging Strategies for $\mathsf{Merge}(\cdot)$

To further examine the extensibility of **PRIME-Merging**, we test whether the residual-space operator $\mathsf{Merge}(\cdot)$ can adopt alternative merging strategies beyond the default Task Arithmetic formulation. Since PRIME separates the principal and orthogonal residual subspaces, the residual component naturally accommodates different task-vector merging rules while preserving the low-interference property of the principal representation.

Table 4 summarizes the results on the T5 model. Although not every strategy consistently improves all benchmarks, multiple alternatives remain competitive, and certain choices improve the average score by up to 0.26%p over the default setting. These findings suggest that PRIME does not rely on a single residual merging rule, but benefits from the principal–residual decomposition.

Overall, PRIME-Merging serves as a general and extensible framework that benefits from appropriately designed residual-space merging strategies. Additional experiments on the LLaMA-3.2 models are reported in Appendix O.

5.5 Temporal & Spatial Efficiency

Temporal Efficiency

Method	T5	LLaMA-1B	LLaMA-3B
TSV	1m 49s	5m 27s	29m 44s
PRIME	**1m 31s**	**4m 24s**	**23m 52s**

Table 5: Merging time of TSV and PRIME when combining eight fine-tuned models for each model architecture.

Table 5 reports the time required for model merging across different model architectures. The reported merging time measures the duration from the start of task-vector-based merging over all layers to the construction of the final merged task vector.

Overall, **PRIME-Merging** consistently requires less merging time than TSV, demonstrating improved computational efficiency. This advantage becomes more pronounced as model size increases, from the smallest T5 model to larger-scale models. These results suggest that PRIME scales more favorably with model size, making it particularly effective for larger architectures.

Spatial Efficiency

Method	T5	LLaMA-1B	LLaMA-3B
TSV	1.9GiB	9.7GiB	14.3GiB
PRIME	1.9GiB	9.7GiB	14.3GiB

Table 6: GPU memory usage of TSV and PRIME when combining eight fine-tuned models for each model architecture during Merging.

Table 6 reports the GPU memory usage during the model merging process. The reported memory consumption corresponds to the peak GPU memory usage observed throughout the entire merging procedure.

Overall, **PRIME-Merging** exhibits GPU memory usage comparable to that of TSV, indicating that its improved efficiency and performance are achieved without introducing additional memory overhead.

In the original TSV (Gargiulo et al., 2025), the retained rank is set to $\frac{d}{n}$, such that the principal subspace scales proportionally with the number of tasks. In contrast, PRIME operates in an ultra-low-rank regime with a substantially smaller retained rank, using this compact principal subspace for the main merging process.

Although PRIME introduces an additional residual merging stage that explicitly incorporates the remaining task-vector components while preventing interference with the principal subspace, the associated overhead remains limited in practice. This is because the overall computational cost is still dominated by the SVD operations, making the extra residual processing relatively inexpensive.

As a result, PRIME not only improves merging performance but also reduces overall merging time compared to TSV. Moreover, these gains in efficiency and effectiveness are achieved without a substantial increase in GPU memory usage, demonstrating the practical scalability of our approach.

6 Conclusion

In this paper, we proposed **PRIME**, an ultra-low-rank principal–residual model merging framework that explicitly addresses the trade-off between task interference and task retention. PRIME decomposes task-vector merging into two complementary stages: principal merging in a compact ultra-low-rank subspace to suppress cross-task interference, and orthogonal residual merging to recover task-relevant information removed by aggressive rank truncation without reintroducing major conflicts.

Extensive experiments on eight natural language processing tasks demonstrate that PRIME consistently outperforms existing model merging approaches across both encoder–decoder and decoder-only architectures, achieving up to 1.18%p gains on T5 and 1.9%p gains on LLaMA-3.2-3B. Compared with TSV-Merging, PRIME also achieves shorter merging times while maintaining comparable GPU memory usage.

Although PRIME is instantiated on top of TSV in this work, the proposed principal–residual framework is broadly applicable to other merging methods based on principal subspace representations. Future work includes extending PRIME to alternative merging paradigms, larger-scale generative models, and more adaptive criteria for balancing principal and residual components.

7 Limitations

In this work, we evaluate PRIME-Merging on models with 0.2B (T5), 1B, and 3B parameters, providing a representative range for controlled and reproducible merging experiments across encoder–decoder and decoder-only architectures. While these models are smaller than recent frontier-scale large language models, they enable systematic analysis of interference behavior, merging efficiency, and adaptive rank selection under realistic computational budgets.

Extending PRIME to substantially larger models remains an important direction for future work. Such evaluations would further validate the scalability of the proposed framework, although they require considerably greater computational cost for both large-scale merging and downstream benchmarking.

Acknowledgments

This work was supported by NAVER Corp and Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(No.RS-2020-II201336, Artificial Intelligence graduate school support(UNIST)).

References

Quora question pairs dataset. https://www.kaggle.com/competitions/quora-question-pairs/data. Accessed: 2025-01-31.

Anonymous. 2025. Subspace-boosted model merging. In *Submitted to The Fourteenth International Conference on Learning Representations*. Under review.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. 2025. Whoever started the interference should end it: Guiding data-free model merging via task vectors. *arXiv preprint arXiv:2503.08099*.

Jiho Choi, Donggyun Kim, Chanhyuk Lee, and Seunghoon Hong. 2025. Revisiting weight averaging for model merging. *Preprint*, arXiv:2412.12153.

MohammadReza Davari and Eugene Belilovsky. 2024. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pages 270–287. Springer.

Pala Tej Deep, Rishabh Bhardwaj, and Soujanya Poria. 2024. Della-merging: Reducing interference in model merging through magnitude-based sampling. *Preprint*, arXiv:2406.11617.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 510 others. 2024. The llama 3 herd of models.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. 2025. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18695–18705.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

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

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2023. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*.

Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D. Bagdanov, and Joost van de Weijer. 2025. No task left behind: Isotropic model merging with common and task-specific subspaces. In *Forty-second International Conference on Machine Learning*.

Michael S Matena and Colin Raffel. 2022. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems*.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. 2024. Learning to route among specialized experts for zero-shot generalization. *Preprint*, arXiv:2402.05859.

NVIDIA Corporation. 2021. NVIDIA RTX A6000. https://www.nvidia.com/en-us/products/workstations/rtx-a6000/. Accessed: 2025-03-08.

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

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Akshara Prabhakar, Yuanzhi Li, Karthik Narasimhan, Sham Kakade, Eran Malach, and Samy Jelassi. 2024. Lora soups: Merging loras for practical skill composition tasks. *Preprint*, arXiv:2410.13025.

Akshara Prabhakar, Yuanzhi Li, Karthik Narasimhan, Sham Kakade, Eran Malach, and Samy Jelassi. 2025. LoRA soups: Merging LoRAs for practical skill composition tasks. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 644–655, Abu Dhabi, UAE. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

Riccardo Salami, Pietro Buzzega, Matteo Mosconi, Jacopo Bonato, Luigi Sabetta, and Simone Calderara. 2025. Closed-form merging of parameter-efficient modules for federated continual learning. In *The Thirteenth International Conference on Learning Representations*.

Sidak Pal Singh and Martin Jaggi. 2020. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, pages 22045–22055. Curran Associates, Inc.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. 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, Seattle, Washington, USA. Association for Computational Linguistics.

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. 2024. Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*.

Zhan Su, Fengran Mo, Guojun Liang, Jinghan Zhang, Bingbing Wen, Prayag Tiwari, and Jian-Yun Nie. 2025. Tensorized clustered lora merging for multi-task interference. *arXiv preprint arXiv:2508.03999*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Anke Tang, Li Shen, Yong Luo, Enneng Yang, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. 2025a. Fusionbench: A comprehensive benchmark of deep model fusion. *Journal of Machine Learning Research*.

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. 2023. Parameter efficient multi-task model fusion with partial linearization. *arXiv preprint arXiv:2310.04742*.

Anke Tang, Enneng Yang, Li Shen, Yong Luo, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. 2025b. 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*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur'elien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. 2024. Localizing task information for improved model merging and compression. *arXiv preprint arXiv:2405.07813*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments.

Yongxian Wei, Anke Tang, Li Shen, Zixuan Hu, Chun Yuan, and Xiaochun Cao. 2025. Modeling multi-task model merging as adaptive projective gradient descent. *arXiv preprint arXiv:2501.01230*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *Preprint*, arXiv:2203.05482.

Feng Xiong, Runxi Cheng, Wang Chen, Zhanqiu Zhang, Yiwen Guo, Chun Yuan, and Ruifeng Xu. 2024. Multi-task model merging via adaptive weight disentanglement. *arXiv preprint arXiv:2411.18729*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. 2024. What matters for model merging at scale? *arXiv preprint arXiv:2410.03617*.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024a. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. 2024b. Representation surgery for multi-task model merging. *arXiv preprint arXiv:2402.02705*.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2023. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level convolutional networks for text classification. *Preprint*, arXiv:1509.01626.

Ziyu Zhao, Tao Shen, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, and Fei Wu. 2025. Merging loRAs like playing LEGO: Pushing the modularity of loRA to extremes through rank-wise clustering. In *The Thirteenth International Conference on Learning Representations*.

A Proof of Theoretical Results

A.1 Proof of Theorem 1

Theorem 3 (Zero Inner-Product Magnitude Implies Zero Merging Gap)**.** *Let the merging rule be Task Arithmetic:*

$$\mathsf{Merge}_{\mathrm{TA}}(\{\boldsymbol{\tau}_i\}_{i=1}^n) = \sum_{i=1}^n \alpha_i \boldsymbol{\tau}_i.$$

If

$$\langle \boldsymbol{\tau}_i, \boldsymbol{\tau}_j \rangle_F = 0 \quad \forall i \neq j,$$

then, under the standard first-order approximation of the task loss, the task-specific merging gap satisfies

$$J_i = 0.$$

Proof. Recall that the merging gap for task i is

$$J_i = \left(\mathcal{L}_i \left(\boldsymbol{\theta}_0 + \sum_{j=1}^n \alpha_j \boldsymbol{\tau}_j \right) - \mathcal{L}_i(\boldsymbol{\theta}_0 + \alpha_i \boldsymbol{\tau}_i) \right)^2.$$

Using a first-order Taylor expansion of $\mathcal{L}_i(\cdot)$ around $\boldsymbol{\theta}_0$,

$$\mathcal{L}_i(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) \approx \mathcal{L}_i(\boldsymbol{\theta}_0) + \langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}_0), \boldsymbol{\delta} \rangle_F.$$

Substituting into J_i gives

$$J_i \approx \left(\left\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}_0), \sum_{j \neq i} \alpha_j \boldsymbol{\tau}_j \right\rangle_F \right)^2.$$

Following prior data-free model merging analyses, we use the approximation

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}_0) \approx -\boldsymbol{\tau}_i.$$

Therefore,

$$J_i \approx \left(\left\langle -\boldsymbol{\tau}_i, \sum_{j \neq i} \alpha_j \boldsymbol{\tau}_j \right\rangle_F \right)^2 = \left(\sum_{j \neq i} \alpha_j \langle \boldsymbol{\tau}_i, \boldsymbol{\tau}_j \rangle_F \right)^2.$$

If $\langle \boldsymbol{\tau}_i, \boldsymbol{\tau}_j \rangle_F = 0$ for all $j \neq i$, every cross-term vanishes, and thus

$$J_i = 0.$$

□

A.2 Proof of Theorem 2

Theorem 4 (Decreasing Rank Increases Task Retention Gap)**.** *Let $\boldsymbol{\tau}_i^{[:r]}$ denote the rank-r truncated SVD approximation of $\boldsymbol{\tau}_i$, retaining the top-r singular values.*
Under the same first-order approximation used in Appendix A.1, the task retention gap

$$G_i = \left(\mathcal{L}_i(\boldsymbol{\theta}_0 + \boldsymbol{\tau}_i^{[:r]}) - \mathcal{L}_i(\boldsymbol{\theta}_0 + \boldsymbol{\tau}_i) \right)^2$$

increases monotonically as r decreases.

Proof. Using first-order Taylor expansion around $\boldsymbol{\theta}_0$, Using first-order Taylor expansion around $\boldsymbol{\theta}_0$,

$$\mathcal{L}_i(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) \approx \mathcal{L}_i(\boldsymbol{\theta}_0) + \langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}_0), \boldsymbol{\delta} \rangle_F.$$

Hence,

$$G_i \approx \left(\left\langle \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}_0), \boldsymbol{\tau}_i^{[:r]} - \boldsymbol{\tau}_i \right\rangle_F \right)^2.$$

Using the standard approximation

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}_0) \approx -\boldsymbol{\tau}_i,$$

we obtain

$$G_i \approx \left\langle -\boldsymbol{\tau}_i, \boldsymbol{\tau}_i^{[:r]} - \boldsymbol{\tau}_i \right\rangle_F^2.$$

Let

$$\boldsymbol{\tau}_i = \boldsymbol{U}_i \boldsymbol{\Sigma}_i (\boldsymbol{V}_i)^\top, \quad \boldsymbol{\tau}_i^{[:r]} = \boldsymbol{U}_i^{[:r]} \boldsymbol{\Sigma}_i^{[:r]} \left(\boldsymbol{V}_i^{[:r]} \right)^\top.$$

Define the residual tail:

$$\boldsymbol{R}_r = \boldsymbol{\tau}_i - \boldsymbol{\tau}_i^{[:r]} = \boldsymbol{U}_i^{[r:]} \boldsymbol{\Sigma}_i^{[r:]} \left(\boldsymbol{V}_i^{[r:]} \right)^\top,$$

Then,

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_i^{[:r]} + \boldsymbol{R}_r.$$

By orthogonality of singular components,

$$\langle \boldsymbol{\tau}_i^{[:r]}, \boldsymbol{R}_r \rangle_F = 0.$$

Therefore,

$$\langle \boldsymbol{\tau}_i, \boldsymbol{R}_r \rangle_F = \|\boldsymbol{R}_r\|_F^2.$$

Hence,

$$G_i \approx \|\boldsymbol{R}_r\|_F^4 = \left(\|\boldsymbol{\Sigma}_i^{[r:]}\|_F^2\right)^2.$$

Since the tail energy

$$\|\boldsymbol{\Sigma}_i^{[r:]}\|_F^2$$

monotonically increases as fewer singular values are retained (i.e., as r decreases), it follows that G_i also increases monotonically as r decreases.

□

A.3 Proof on Null-Space Projection of the Whitened Principal Subspace

Proposition 1 (Null-Space Projection of the Whitened Principal Subspace)**.** *Let $\hat{\boldsymbol{U}}_s$ denote the whitened principal basis with orthonormal columns, i.e., $\hat{\boldsymbol{U}}_s^\top \hat{\boldsymbol{U}}_s = \boldsymbol{I}$, and define the orthogonal projection matrix*

$$\boldsymbol{P}_\perp \;=\; \boldsymbol{I} - \hat{\boldsymbol{U}}_s \hat{\boldsymbol{U}}_s^\top.$$

Then, for "whitened" principal task vector $\hat{\boldsymbol{\tau}}_t^{[:r]}$ of Eq. (18) that lies in the column space of $\hat{\boldsymbol{U}}_s$, we have

$$\boldsymbol{P}_\perp \hat{\boldsymbol{\tau}}_t^{[:r]} = \boldsymbol{0}.$$

Proof. By construction, each whitened principal task vector $\hat{\boldsymbol{\tau}}_t^{[:r]}$ lies in the subspace spanned by $\hat{\boldsymbol{U}}_s$. Hence, there exists a coefficient vector $\boldsymbol{c}_t$ such that

$$\hat{\boldsymbol{\tau}}_t^{[:r]} = \hat{\boldsymbol{U}}_s \boldsymbol{c}_t.$$

Applying the null-space projection $\boldsymbol{P}_\perp$ yields

$$\begin{aligned}
\boldsymbol{P}_\perp \hat{\boldsymbol{\tau}}_t^{[:r]} &= \left(\boldsymbol{I} - \hat{\boldsymbol{U}}_s \hat{\boldsymbol{U}}_s^\top\right) \hat{\boldsymbol{U}}_s \boldsymbol{c}_t \\
&= \hat{\boldsymbol{U}}_s \boldsymbol{c}_t - \hat{\boldsymbol{U}}_s \hat{\boldsymbol{U}}_s^\top \hat{\boldsymbol{U}}_s \boldsymbol{c}_t \\
&= \hat{\boldsymbol{U}}_s \boldsymbol{c}_t - \hat{\boldsymbol{U}}_s \left(\hat{\boldsymbol{U}}_s^\top \hat{\boldsymbol{U}}_s\right) \boldsymbol{c}_t \\
&= \hat{\boldsymbol{U}}_s \boldsymbol{c}_t - \hat{\boldsymbol{U}}_s \boldsymbol{c}_t \\
&= \boldsymbol{0},
\end{aligned}$$

where we use the orthonormality condition $\hat{\boldsymbol{U}}_s^\top \hat{\boldsymbol{U}}_s = \boldsymbol{I}$. □

A.4 Proof on Null-Space Projection Induced by the Whitened Basis

Proposition 2 (Null-Space Projection Induced by the Whitened Basis)**.** *Let $\hat{\boldsymbol{U}}_l \in \mathbb{R}^{d\times(r\times T)}$ denote the whitened left singular vectors with orthonormal columns, i.e., $\hat{\boldsymbol{U}}_l^\top \hat{\boldsymbol{U}}_l = \boldsymbol{I}$. Define the projection matrix as*

$$\boldsymbol{P} = \hat{\boldsymbol{U}}_l \hat{\boldsymbol{U}}_l^\top,$$

and the corresponding null-space projection as

$$\boldsymbol{P}_\perp = \boldsymbol{I} - \boldsymbol{P}.$$

Then, the null-space projection completely removes all components lying in the subspace spanned by $\hat{\boldsymbol{U}}_l$, i.e.,

$$\boldsymbol{P}_\perp \hat{\boldsymbol{U}}_l = \boldsymbol{0}.$$

Proof. **Assumption.** Assume that $\hat{\boldsymbol{U}}_l \in \mathbb{R}^{d\times(r\times T)}$ has orthonormal columns, i.e., $\hat{\boldsymbol{U}}_l^\top \hat{\boldsymbol{U}}_l = \boldsymbol{I}$.

Derivation. By definition, the null-space projection matrix is given by

$$\boldsymbol{P}_\perp = \boldsymbol{I} - \hat{\boldsymbol{U}}_l \hat{\boldsymbol{U}}_l^\top.$$

Applying $\boldsymbol{P}_\perp$ to $\hat{\boldsymbol{U}}_l$, we obtain

$$\boldsymbol{P}_\perp \hat{\boldsymbol{U}}_l = (\boldsymbol{I} - \hat{\boldsymbol{U}}_l \hat{\boldsymbol{U}}_l^\top)\hat{\boldsymbol{U}}_l.$$

Argument. Using the orthonormality of $\hat{\boldsymbol{U}}_l$, we have

$$\hat{\boldsymbol{U}}_l \hat{\boldsymbol{U}}_l^\top \hat{\boldsymbol{U}}_l = \hat{\boldsymbol{U}}_l (\hat{\boldsymbol{U}}_l^\top \hat{\boldsymbol{U}}_l) = \hat{\boldsymbol{U}}_l.$$

Therefore,

$$\boldsymbol{P}_\perp \hat{\boldsymbol{U}}_l = \hat{\boldsymbol{U}}_l - \hat{\boldsymbol{U}}_l = \boldsymbol{0}.$$

Conclusion. Thus, the null-space projection $\boldsymbol{P}_\perp$ exactly removes all components lying in the subspace spanned by $\hat{\boldsymbol{U}}_l$. □

r =	(1)*	1%	2%	3%	4%	5%	10%	12.5%
$\sum_i J_i$	1.8752	7.4368	8.4864	9.2128	9.7438	10.1184	11.552	12.0448
$\sum_i G_i$	0.4844	0.1775	0.1327	0.1058	0.0885	0.0774	0.0428	0.0336
$F(r)$	2.3596	7.6143	8.6191	9.3186	9.8323	10.1958	11.5948	12.0784
PRIME	82.12	83.73	83.64	83.65	83.70	83.76	83.14	82.61

Table 7: (T5) Performance of PRIME at r^* selected using the automatic rank selection method based on $F(r)$, along with nearby grid values, for T5. The table also reports the corresponding $\sum_i J_i$, $\sum_i G_i$, and $F(r)$. Note that $F(r)$ increases immediately after $r = 1$, resulting in $r^* = 1$.

r =	(1)	1%	2%*	3%	4%	5%	10%	12.5%
$\sum_i J_i$	0.0000	0.0256	0.0384	0.0573	0.0749	0.0960	0.2048	0.2688
$\sum_i G_i$	0.2075	0.1809	0.1686	0.1592	0.1506	0.1431	0.1110	0.0982
$F(r)$	0.2075	0.2065	0.2070	0.2166	0.2255	0.2391	0.3158	0.3670
PRIME	58.74	64.20	64.25	64.35	64.33	64.31	63.74	63.20

Table 8: (LLaMA-3.2-1B) Performance of PRIME at r^* selected using the automatic rank selection method based on $F(r)$, along with nearby grid values, for LLaMA-3.2-1B. The table also reports the corresponding $\sum_i J_i$, $\sum_i G_i$, and $F(r)$. Note that $F(r)$ exhibits a relatively flat region at small r and begins to increase more clearly beyond approximately 2%, resulting in $r^* = 2\%$.

r =	(1)	1%*	2%	3%	4%	5%	10%	12.5%
$\sum_i J_i$	0.0000	0.032	0.0576	0.0767	0.1003	0.1280	0.2752	0.3648
$\sum_i G_i$	0.1940	0.1649	0.1539	0.1443	0.1358	0.1279	0.0962	0.0835
$F(r)$	0.1940	0.1969	0.2115	0.2211	0.2361	0.2559	0.3714	0.4483
PRIME	66.08	75.40	75.19	75.29	75.03	74.88	74.05	73.56

Table 9: (LLaMA-3.2-3B) Performance of PRIME at r^* selected using the automatic rank selection method based on $F(r)$, along with nearby grid values, for LLaMA-3.2-3B. The table also reports the corresponding $\sum_i J_i$, $\sum_i G_i$, and $F(r)$. Note that $F(r)$ exhibits a relatively flat region at small r and begins to increase more clearly beyond approximately 1%, resulting in $r^* = 1\%$.

B Result of Adaptive Rank Selection

Empirical analysis of adaptive rank selection. Tables 7, 8, and 9 report $\sum_i J_i$, $\sum_i G_i$, the surrogate objective $F(r)$, and PRIME performance across nearby rank candidates. Overall, the selected rank r^* consistently lies in the ultra-low-rank regime and aligns with strong-performing regions.

For T5, $F(r)$ increases immediately after $r = 1$, yielding $r^* = 1$. This suggests that retaining only the dominant singular direction is sufficient to balance interference reduction and task retention. Despite this highly compact setting, PRIME remains competitive, with nearby ranks (1–5%) showing similarly strong performance.

For LLaMA-3.2-1B, $F(r)$ is relatively flat at small ranks and reaches its minimum near $r = 2\%$, after which it increases steadily. PRIME also achieves its best performance in this region.

For LLaMA-3.2-3B, the minimum of $F(r)$ appears near $r = 1\%$, while performance remains stable across neighboring ranks (1–3%), indicating robustness to small rank changes.

These results show that the optimal principal rank depends on model architecture, and that minimizing $F(r)$ provides a practical criterion for selecting an effective ultra-low-rank region.

C Performance of Pincipal Task Vector on LLaMA-3.2-1B

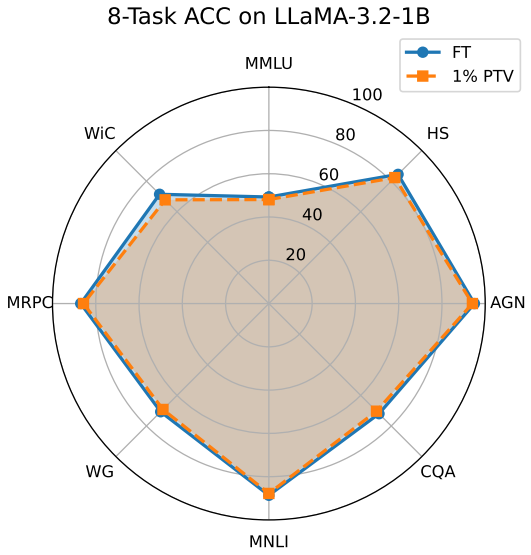

Figure 5: Performance comparison between the original task vectors and the ultra-low-rank principal task vectors when merging eight NLP tasks on LLaMA-3.2-1B. Despite aggressive rank truncation, the principal task vectors preserve nearly all task-specific utility, retaining approximately 98% of the original performance on average.

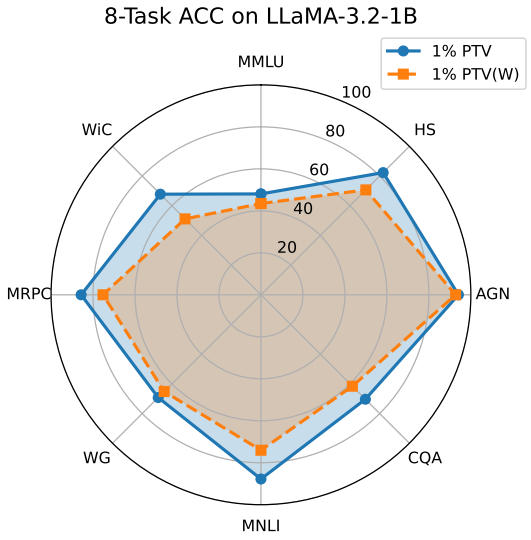

Figure 6: Performance comparison before and after whitening of the ultra-low-rank principal task vectors when merging eight NLP tasks on LLaMA-3.2-1B. Whitening preserves task performance while preparing the principal vectors for interference-aware merging, with approximately 88% average performance retention.

Figure 5 and Figure 6 present the results for the LLaMA-3.2-1B model corresponding to Figure 3 and Figure 4.

D Performance of Pincipal Task Vector on LLaMA-3.2-3B

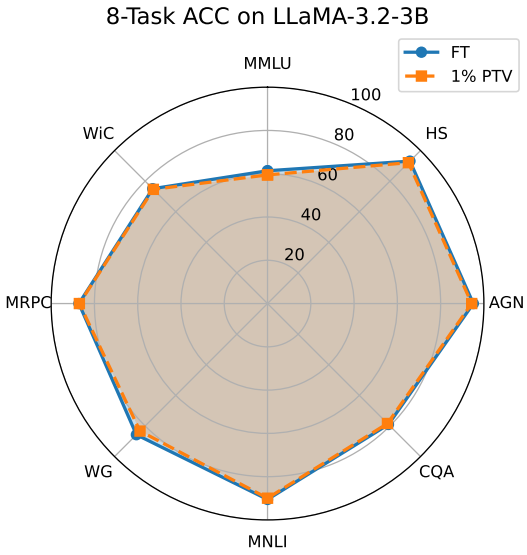

Figure 7: Performance comparison between the original task vectors and the ultra-low-rank principal task vectors when merging eight NLP tasks on LLaMA-3.2-3B. Despite aggressive rank truncation, the principal task vectors preserve nearly all task-specific utility, retaining approximately 99% of the original performance on average.

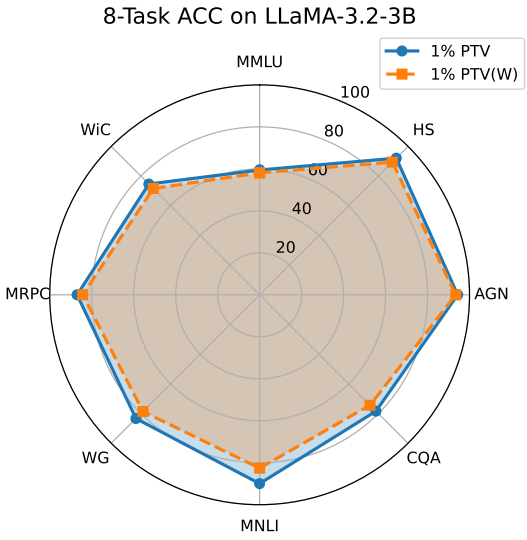

Figure 8: Performance comparison before and after whitening of the ultra-low-rank principal task vectors when merging eight NLP tasks on LLaMA-3.2-3B. Whitening preserves task performance while preparing the principal vectors for interference-aware merging, with approximately 96% average performance retention.

Figure 7 and Figure 8 present the results for the LLaMA-3.2-3B model corresponding to Figure 3 and Figure 4.

E Cosine Similarity on T5

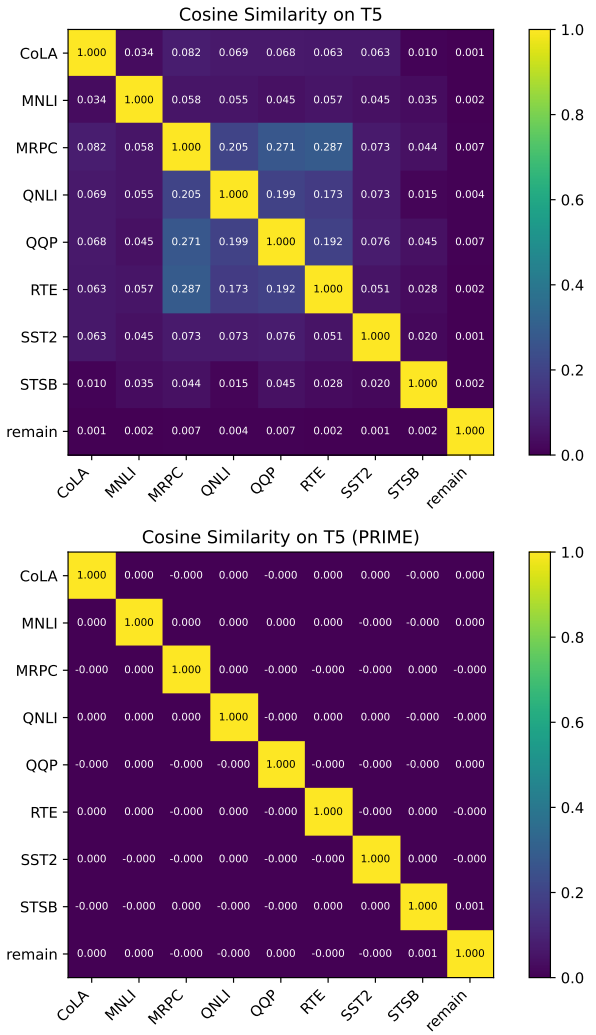

Figure 9: osine similarity among task-specific principal subspaces and the merged residual vector on T5, before and after applying PRIME-Merging. Before projection, several principal subspaces exhibit non-negligible overlap, indicating potential cross-task interference. After PRIME, off-diagonal similarities are nearly eliminated, and the merged residual vector becomes approximately orthogonal to the principal subspaces. These results verify that PRIME effectively suppresses subspace interference while preserving complementary residual information.

Figure 9 shows cosine similarities among principal task subspaces and the merged residual vector on T5. Before PRIME, several task subspaces overlap, indicating potential interference during merging. After PRIME, most off-diagonal similarities are nearly zero, and the residual vector becomes approximately orthogonal to the principal subspaces. This confirms that PRIME reduces subspace interference while preventing residual updates from re-entering the principal space.

F Cosine Similarity on LLaMA-3.2-1B

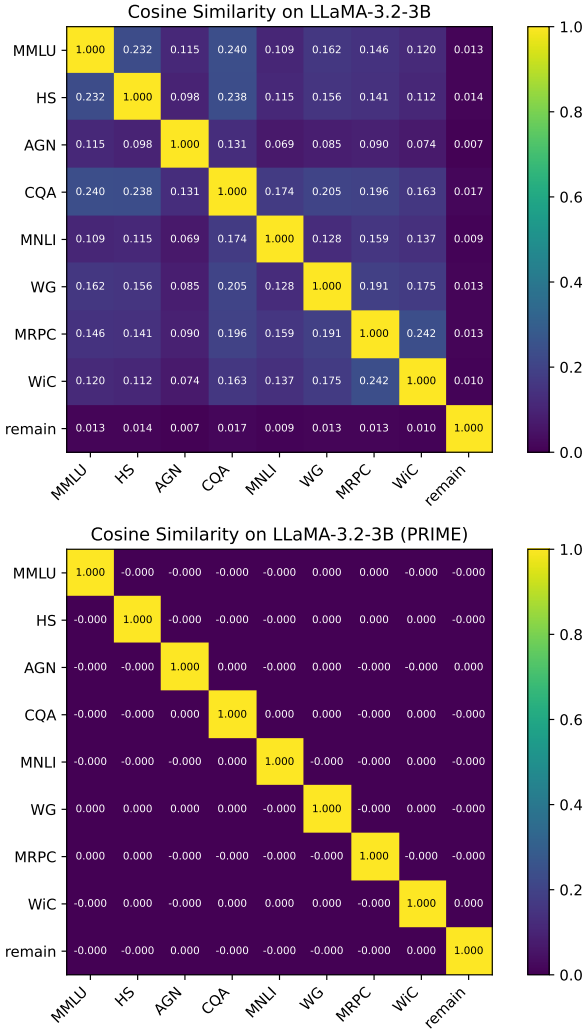

Figure 10: osine similarity among task-specific principal subspaces and the merged residual vector on LLaMA-3.2-1B, before and after applying PRIME-Merging. Before projection, several principal subspaces exhibit non-negligible overlap, indicating potential cross-task interference. After PRIME, off-diagonal similarities are nearly eliminated, and the merged residual vector becomes approximately orthogonal to the principal subspaces. These results verify that PRIME effectively suppresses subspace interference while preserving complementary residual information.

Figure 10 shows cosine similarities among principal task subspaces and the merged residual vector on LLaMA-3.2-1B.

G Cosine Similarity on LLaMA-3.2-3B

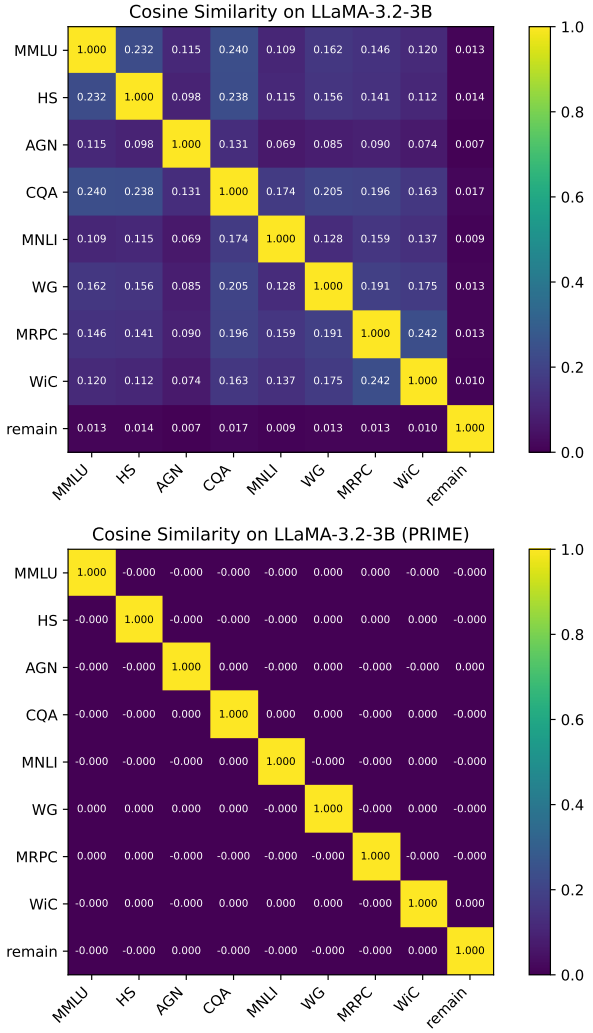

Figure 11: osine similarity among task-specific principal subspaces and the merged residual vector on LLaMA-3.2-3B, before and after applying PRIME-Merging. Before projection, several principal subspaces exhibit non-negligible overlap, indicating potential cross-task interference. After PRIME, off-diagonal similarities are nearly eliminated, and the merged residual vector becomes approximately orthogonal to the principal subspaces. These results verify that PRIME effectively suppresses subspace interference while preserving complementary residual information.

Figure 11 shows cosine similarities among principal task subspaces and the merged residual vector on LLaMA-3.2-3B.

H Cosine Similarity on LLaMA-3.2-3B

Algorithm 1 PRIME-Merging Procedure

Require: $\boldsymbol{\tau}_{t,l}$: Task Vectors
r: Rank

▷ **Identify** Principal and Residual

for $i \in \{1, \cdots, n\}$ **do**

$$\boldsymbol{U}_i, \boldsymbol{\Sigma}_i, \boldsymbol{V}_i^T = \mathrm{SVD}\left(\boldsymbol{\tau}_i\right)$$

$$\boldsymbol{\tau}_i^{[:r]} = \boldsymbol{U}_i^{[:r]} \boldsymbol{\Sigma}_i^{[:r]} \left(\boldsymbol{V}_i^{[:r]}\right)^\top$$

$$\boldsymbol{\tau}_i^{[r:]} = \boldsymbol{\tau}_i - \boldsymbol{\tau}_i^{[:r]}$$

end for

▷ **PRIME-Merging**

$$\boldsymbol{U}_s = [\boldsymbol{U}_1^{[:r]} | \boldsymbol{U}_2^{[:r]} | \ldots | \boldsymbol{U}_n^{[:r]}]$$

$$\boldsymbol{\Sigma}_s = [\boldsymbol{\Sigma}_1^{[:r]} | \boldsymbol{\Sigma}_2^{[:r]} | \ldots | \boldsymbol{\Sigma}_n^{[:r]}]$$

$$\boldsymbol{V}_s = [\boldsymbol{V}_1^{[:r]} | \boldsymbol{V}_2^{[:r]} | \ldots | \boldsymbol{V}_n^{[:r]}]$$

$$\hat{\boldsymbol{U}}_s = \mathrm{Whitening}\left(\boldsymbol{U}_s\right)$$

$$\hat{\boldsymbol{V}}_s = \mathrm{Whitening}\left(\boldsymbol{V}_s\right)$$

▷ **Orthogonal Residual Task Vector Merging**

$$\hat{\boldsymbol{\tau}}_{m,l}^{[:r]} = \hat{\boldsymbol{U}}_s \boldsymbol{\Sigma}_s \left(\hat{\boldsymbol{V}}_s\right)^\top, \quad \boldsymbol{P}_\perp = \boldsymbol{I} - \hat{\boldsymbol{U}}_s \left(\hat{\boldsymbol{U}}_s\right)^\top$$

$$\hat{\boldsymbol{\tau}}_m^{[r:]} = \boldsymbol{P}_\perp \left(\mathrm{Merge}\left(\{\boldsymbol{\tau}_i^{[r:]}\}_{i=1}^n\right)\right)$$

$$\hat{\boldsymbol{\tau}}_m = \hat{\boldsymbol{\tau}}_m^{[:r]} + \hat{\boldsymbol{\tau}}_m^{[r:]}$$

The PRIME-Merging procedure is summarized in Algorithm 1. Given task vectors, PRIME first decomposes each vector into a low-rank principal component and a residual component via truncated SVD.

The overall workflow consists of two stages: **(i)** *ultra-low-rank principal merging*, where compact principal components are jointly whitened and merged to suppress cross-task interference within the shared subspace; and **(ii)** *orthogonal residual merging*, where the remaining residual components are merged and projected onto the orthogonal complement of the principal subspace to recover additional task-specific information without reintroducing conflicts.

The final merged task vector is obtained by combining the merged principal and residual components. Through this structured decomposition, PRIME simultaneously reduces interference and preserves task-relevant information during model merging.

I Training Parameters - LLaMA-3.2-1B

Dataset	Epochs	Batch Size	LR
MMLU	4	512	2×10^{-5}
HS	3	256	
AGN	4	512	
CQA	5	256	
MNLI	1	512	
WG	3	512	
MRPC	5	64	
WiC	5	32	

Table 10: Fine-tuning hyperparameters for the LLaMA-3.2-1B model

The LLaMA-3.2-1B model was fine-tuned on four NVIDIA RTX A6000 GPUs using the Hugging Face Transformers Trainer framework (Wolf et al., 2020). We used the AdamW optimizer, a cosine learning rate schedule, and a warmup ratio of 1%.

J Training Parameters - LLaMA-3.2-3B

Dataset	Epochs	Batch Size	LR
MMLU	4	512	2×10^{-5}
HS	3	256	
AGN	4	512	
CQA	5	256	
MNLI	1	512	
WG	3	512	
MRPC	5	64	
WiC	5	32	

Table 11: Fine-tuning hyperparameters for the LLaMA-3.2-3B model

The LLaMA-3.2-3B model was fine-tuned on four NVIDIA RTX A6000 GPUs using the Hugging Face Transformers Trainer framework (Wolf et al., 2020). We used the AdamW optimizer, a cosine learning rate schedule, and a warmup ratio of 1%.

K Fine-tuning Models of T5

For the T5 model, we use the fine-tuned models provided by *FusionBench* (Tang et al., 2025a). Details of the models fine-tuned on each dataset are as follows:

- CoLA: https://huggingface.co/tanganke/flan-t5-base_glue-cola
- MNLI: https://huggingface.co/tanganke/flan-t5-base_glue-mnli
- MRPC: https://huggingface.co/tanganke/flan-t5-base_glue-mrpc
- QNLI: https://huggingface.co/tanganke/flan-t5-base_glue-qnli
- QQP: https://huggingface.co/tanganke/flan-t5-base_glue-qqp
- RTE: https://huggingface.co/tanganke/flan-t5-base_glue-rte
- SST2: https://huggingface.co/tanganke/flan-t5-base_glue-sst2
- STSB: https://huggingface.co/tanganke/flan-t5-base_glue-stsb

In the merging experiments involving the T5 model, we exclusively use the fine-tuned models provided by *FusionBench*, while implementing the merging and benchmarking code ourselves.

L List of Datasets

- **T5** - GLUE (Wang et al., 2018) Task Fool
  CoLA (Warstadt et al., 2019),
  MNLI (Williams et al., 2018),
  MRPC (Dolan and Brockett, 2005),
  QNLI (Rajpurkar et al., 2016),
  QQP (qqp),
  RTE (Giampiccolo et al., 2007),
  SST2 (Socher et al., 2013),
  STSB (Cer et al., 2017)
- **LLaMA-3.2** - Custom Task Fool
  MMLU (Hendrycks et al., 2021b,a),
  HellaSwag (Zellers et al., 2019),
  AG News (Zhang et al., 2016),
  CommonsenceQA (Talmor et al., 2019),
  MNLI (Williams et al., 2018),
  WinoGrande (Sakaguchi et al., 2019),
  MRPC (Dolan and Brockett, 2005),
  WiC (Pilehvar and Camacho-Collados, 2019)

- MMLU: https://huggingface.co/datasets/cais/mmlu
- HellaSwag: https://huggingface.co/datasets/Rowan/hellaswag
- AGNews: https://huggingface.co/datasets/fancyzhx/ag_news
- CommonsenceQA: https://huggingface.co/datasets/tau/commonsense_qa
- MNLI: https://huggingface.co/datasets/SetFit/mnli
- Winogrande: https://huggingface.co/datasets/XiaHan19/winogrande4MC
- MRPC: https://huggingface.co/datasets/SetFit/mrpc
- WiC: https://huggingface.co/datasets/Deehan1866/WiC

M Beta Zero Subspace Boosting

The formulation of Subspace Boosting (SB) (Anonymous, 2025) is summarized as follows:

$$\mathbf{\Sigma}_m = \mathsf{diag}\left([\sigma_m^{(1)}, \sigma_m^{(2)}, \ldots, \sigma_m^{(d)}]\right) \tag{24}$$

$$\hat{n} = \min\left\{ n \in \{1, \ldots, d\} \;\middle|\; \frac{\sum_{i=1}^{n} \sigma_m^{(i)}}{\sum_{i=1}^{d} \sigma_m^{(i)}} \geq \beta \right\} \tag{25}$$

$$\tilde{\mathbf{\Sigma}}_m = \mathsf{diag}\left([\sigma_m^{(1)}, \ldots, \sigma_m^{(\hat{n}-1)}, \sigma_m^{(\hat{n})}, \ldots, \sigma_m^{(\hat{n})}]\right) \tag{26}$$

As shown in Eq. (25), SB determines an index $\hat{n}$ as a function of the *Beta* parameter β, and flattens all singular values with indices greater than $\hat{n}$ to the value at that index. This operation equalizes the scaling of lower-energy singular directions, preventing the merged representation from being dominated by a small number of high-energy directions. When β is set to 0.0, the resulting index $\hat{n}$ consistently becomes 1, regardless of the original singular value distribution. Under this setting, all singular values are flattened to the largest singular value, and the SB formulation can therefore be simplified as follows:

$$\tilde{\mathbf{\Sigma}}_m = \mathsf{diag}\left([\sigma_m^{(1)}, \sigma_m^{(1)}, \ldots, \sigma_m^{(1)}]\right) \tag{27}$$

The original SB paper also reports results for the setting where the *Beta* parameter β is set to 0.0, demonstrating that this configuration yields competitive performance. Motivated by this observation, we adopt the $\beta = 0.0$ setting in our implementation. To further simplify the overall procedure and reduce unnecessary complexity, we modify the original formulation accordingly, as described above.

N Scaled-Subspace Boosting

In this section, we present **Subspace Boosting (SB)** (Anonymous, 2025) and **Scaled-Subspace Boosting (S-SB)**, a slightly modified variant used in our experiments. We first briefly review the original SB approach (Anonymous, 2025) and then describe the modified version adopted in our method.

To apply SB, we first require a merged task vector, which is computed as

$$\boldsymbol{\tau}_m = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\tau}_i. \tag{28}$$

SB argues that rank collapse may occur when task vectors are merged via weighted summation, and mitigates this issue by modifying the singular values through SVD. Specifically, the merged task vector is decomposed as

$$\boldsymbol{\tau}_m = \boldsymbol{U}_m \mathbf{\Sigma}_m \boldsymbol{V}_m^\top, \tag{29}$$

where

$$\mathbf{\Sigma}_m = \mathsf{diag}\left([\sigma_m^{(1)}, \sigma_m^{(2)}, \ldots, \sigma_m^{(d)}]\right). \tag{30}$$

The SB procedure then constructs a modified singular value matrix by flattening the singular values as

$$\tilde{\mathbf{\Sigma}}_m = \mathsf{diag}\left([\sigma_m^{(1)}, \sigma_m^{(1)}, \ldots, \sigma_m^{(1)}]\right), \tag{31}$$

which yields the boosted merged task vector

$$\tilde{\boldsymbol{\tau}}_m = \boldsymbol{U}_m \tilde{\mathbf{\Sigma}}_m \boldsymbol{V}_m^\top. \tag{32}$$

In the original SB formulation (Anonymous, 2025), this flattening operation is controlled by a *Beta* parameter. In our experiments, we observe that setting *Beta* $= 0.0$ consistently yields strong performance, corresponding to flattening all singular values to the largest singular value $\sigma_m^{(1)}$, as shown in Eq. (31) and discussed in Appendix M.

Scaled-Subspace Boosting (S-SB) S-SB extends SB with *Beta* $= 0.0$ by introducing an additional scaling factor. Specifically, Eq. (31) is modified as

$$\tilde{\mathbf{\Sigma}}_m = \mathsf{diag}\left([\alpha\sigma_m^{(1)}, \alpha\sigma_m^{(1)}, \ldots, \alpha\sigma_m^{(1)}]\right), \tag{33}$$

where α is a scaling factor applied uniformly to the flattened singular values.

This scaling is motivated by the observation that, in PRIME, the residual task vectors have already had their principal subspace components removed, and that projecting them via $\boldsymbol{P}_\perp$ may reduce their overall norm. The scaling factor α compensates for this effect by amplifying the remaining singular values. Based on empirical validation, we set $\alpha = 1.7$ in all experiments.

As shown in Table 4, applying S-SB yields an additional performance improvement of 0.26%p compared to the original PRIME.

O Additional Result

Type	Model	MMLU	HS	AGN	CQA	MNLI	WG	MRPC	WiC	Avg.
Original	Base	27.64	25.05	31.41	21.29	35.44	49.49	66.49	50.00	38.35
Individual	FT	49.25	84.44	94.99	71.99	88.61	70.72	86.96	71.43	77.30
Merging	TA	38.06	38.10	77.39	51.43	36.93	55.96	66.78	50.21	51.86
	TIES	39.27	49.99	76.20	52.33	43.22	60.85	68.81	53.07	55.47
	DARE	38.40	37.46	77.29	52.01	37.73	55.33	67.83	51.14	52.15
	DELLA	37.70	49.23	75.29	48.81	44.82	63.69	70.26	51.86	55.21
	Iso-C	35.86	30.55	73.24	46.85	35.50	50.20	65.91	48.57	48.34
	Iso-CTS	35.77	28.59	75.26	45.86	35.44	49.64	64.23	49.93	48.09
	SB	**44.07**	67.65	**87.28**	64.95	46.64	63.85	69.86	52.93	62.15
	TSV	43.56	70.03	87.13	66.42	48.26	65.35	70.26	54.00	63.13
	PRIME	43.37	**72.81**	87.13	**66.34**	**50.03**	**66.46**	**72.35**	**55.14**	**64.20**

Table 12: Performance of merging LLaMA-3.2-1B models fine-tuned on eight NLP tasks. For LLaMA-3.2-1B, whose task vectors satisfy $\boldsymbol{\tau}_i \in \mathbb{R}^{d' \times d}$, **PRIME** operates in an ultra-low-rank regime, where the principal rank is determined according to the model dimensions..

Type	Model	MMLU	HS	AGN	CQA	MNLI	WG	MRPC	WiC	Avg.
Original	Base	27.64	25.05	31.41	21.29	35.44	49.49	66.49	50.00	38.35
Individual	FT	49.25	84.44	94.99	71.99	88.61	70.72	86.96	71.43	77.30
Merging	PRIME	43.37	72.81	87.13	66.34	50.03	66.46	72.35	55.14	64.20
	PRIME(TIES)	43.14	71.61	86.76	64.62	48.45	67.56	70.90	54.43	63.43
	PRIME(S-SB)	43.64	73.21	87.46	66.26	50.59	67.09	72.35	55.43	64.50

Table 13: This table reports the performance obtained by varying the $\mathsf{Merge}(\cdot)$ operation in Eq. 20 on LLaMA-3.2-1B models. S-SB denotes **Scaled-Subspace Boosting**, which is described in Appendix M and N.

Type	Model	MMLU	HS	AGN	CQA	MNLI	WG	MRPC	WiC	Avg.
Original	Base	30.84	27.19	63.89	51.35	35.45	50.59	63.71	49.79	46.60
Individual	FT	61.37	93.09	95.11	79.03	90.53	85.71	87.19	75.07	83.39
Merging	PRIME	54.73	85.52	91.16	76.90	75.43	82.40	79.01	58.07	75.40
	PRIME(TIES)	54.64	85.74	91.25	76.58	75.57	82.72	77.74	57.79	75.25
	PRIME(S-SB)	54.96	86.14	91.62	76.41	75.93	82.87	78.38	58.86	75.65

Table 14: This table reports the performance obtained by varying the $\mathsf{Merge}(\cdot)$ operation in Eq. 20 on LLaMA-3.2-3B models. S-SB denotes **Scaled-Subspace Boosting**, which is described in Appendix M and N.