Separating Common and Unique Directions for Model Merging

Shiting Wang[1,2,3] **Yingjie Zou**[1,2,4] **Ke Li**[1,2,5 *]
[1]Earlham Institute, United Kingdom
[2]Living Systems Institute, University of Exeter, United Kingdom
[3]Department of Computer Science, University of Exeter, United Kingdom
[4]Department of Engineering, University of Exeter, United Kingdom
[5]School of Computing Sciences, University of East Anglia, United Kingdom
stwang0106@gmail.com, yz1033@exeter.ac.uk, ke.li@earlham.ac.uk

Abstract

Model merging combines multiple fine-tuned models without additional training, and arithmetic methods have emerged as a simple yet effective way to merge models without data. However, existing arithmetic methods treat task vectors in parameter space and apply a single element-wise rule to directions that can differ in mergeability. In a shared directional basis, we find a consistent two-regime structure: a low-dispersion common bulk shared across tasks and a high-dispersion task-dominated tail that carries disproportionate energy. These findings motivate **C**ommon–**U**nique **S**ubspace **D**ecomposition (CUSD), a data-free pre-merging method that separates task vectors into common and unique directional components, merges the common components with a downstream arithmetic operator, and retains unique residuals separately. Experiments across 11 base models, four model families, and four arithmetic-based merging operators show that CUSD improves 42 of 44 model–operator combinations and achieves up to 22.5% improvement over the corresponding baselines. Our code is publicly available at https://github.com/stwang666/CUSD

1 Introduction

Model merging aims to combine K task-specific checkpoints, fine-tuned from a common base model, into a single multi-task model without access to the original task data or joint retraining [1–8]. This setting is increasingly relevant when training data cannot be shared, checkpoints are produced by different owners, or serving one model per task is too expensive [9, 10]. Arithmetic-based merging methods are a central paradigm in this setting because they require only model weights. They represent each task by a task vector $\tau_i = \theta_i - \theta_{\text{base}}$ [11] and construct a merged update using non-iterative operations such as addition, sign selection, sparsification, or rescaling [11–15]. Despite substantial progress [16, 17], arithmetic merging often remains below the corresponding task-specific models, indicating that residual inter-task interference is not fully resolved by current rules.

We identify one source of this interference: current arithmetic methods are largely coordinate-wise and uniform. They decide how to add, trim, drop, or rescale scalar entries of task vectors, but they do not first determine whether those entries belong to update directions that are consistently shared across tasks or dominated by only a few tasks. This distinction matters because parameter updates are not merely collections of independent coordinates. In matrix-structured layers, a direction represents a coordinated update pattern across many parameter entries. A direction used comparably by all

*Corresponding author

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

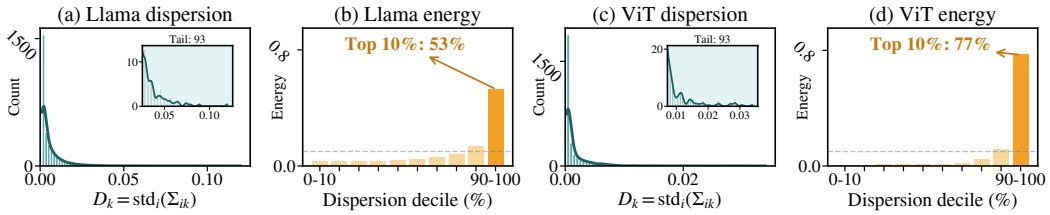


Figure 1: **Direction-level heterogeneity in task vectors.** Motivating examples from representative matrix layers of Llama-3.2-3B (`mlp.gate_proj`) and ViT-B/16 (`mlp.c_proj`). (a, c) Distribution of cross-task dispersion scores over shared directions. (b, d) Share of total directional energy in each dispersion-ranked decile.

tasks is a candidate for aggregation, whereas a direction dominated by one task may need to be preserved, damped, or separated from the shared merge. Applying the same arithmetic rule to both types of directions can either suppress task-dominated signal or inject task-specific interference into the common update.

To expose this structure, we analyze task-vector matrices in a shared directional basis. We use energy to denote the update magnitude carried by task vectors along a shared direction. For each shared direction, we measure the task-wise energy and its dispersion across tasks. Figure 1 shows two motivating examples from Llama-3.2-3B [18] and ViT-B/16 [19]. In both cases, most directions form a low-dispersion bulk, while a small high-dispersion tail carries a disproportionate fraction of the total directional energy. In the shown layers, the highest-dispersion decile accounts for 53% and 77% of total energy, respectively. Thus, the high-dispersion tail is not negligible noise: it contains substantial update magnitude, but its energy is unevenly distributed across tasks. This observation motivates separating directions by their cross-task energy dispersion before applying an arithmetic merging rule. Appendix A.1 provides additional cross-architecture visualizations showing that the same two-regime pattern appears broadly across layers, model families, and task groups.

We propose **C**ommon–**U**nique **S**ubspace **D**ecomposition (CUSD), a data-free pre-merging framework for arithmetic model merging. For each matrix-structured layer, CUSD constructs a shared orthogonal directional basis from the task-vector matrices using a regularized Higher-Order Generalized Singular Value Decomposition derived (HO-GSVD-derived) construction [20, 21]. In this shared basis, CUSD computes a cross-task dispersion score for each direction and partitions directions into two sets: common directions, whose energy is comparable across tasks, and unique directions, whose energy is dominated by one or a few tasks. We use "unique" in this operational sense: these directions are task-dominated residuals, not assumed to encode semantically exclusive features.

Given this partition, the common components C_i and unique residuals R_i can be obtained from K task vectors. It then applies the downstream arithmetic merging operator $\mathcal{F}$ only to the C_i and controls the R_i with a separate scalar coefficient β to get merged task vector τ_m:

$$\tau_m = \mathcal{F}(C_1, \ldots, C_K) + \beta \sum_{i=1}^{K} R_i,$$

This design makes CUSD a pre-merging module rather than a replacement for existing arithmetic operators. The same decomposition depends only on the task vectors and can therefore be computed once and reused across different arithmetic merging rules [11, 14, 15]. Compared with per-task SVD methods [22, 23], CUSD avoids comparing directions chosen independently for each task; compared with post-merge subspace corrections [24], it separates task-dominated directions before the merging rule has mixed them; and compared with data-dependent routing or modular approaches [25], it produces a single static merged model using weights alone. Our main contributions are summarized as follows:

- We identify and quantify a direction-level two-regime structure in task-vector matrices: a low-dispersion bulk of broadly shared directions and a high-dispersion, task-dominated tail that carries disproportionate energy.
- We introduce CUSD, a data-free pre-merging method that uses a HO-GSVD-derived construction to obtain a shared directional basis, separates common components from unique residuals, applies any downstream arithmetic merging operator only to the common components, and controls the unique residuals through a single coefficient β.

- We evaluate CUSD across 11 base models, four model families, and four arithmetic-based merging operators, showing that CUSD consistently improves data-free arithmetic merging across the evaluated settings.

2 Related Work

Current model merging research falls into two categories: data-free methods that rely only on model parameters or task vectors, and data-dependent methods that require additional data during merging.

2.1 Data-Free Model Merging

Arithmetic-based model merging. These methods apply uniform element-wise rules to model weights or task vectors $\{\tau_i\}$ without data or iterative optimization. Task Arithmetic (TA) [11] merges via scaled addition $\theta_{\text{merged}} = \theta_{\text{base}} + \lambda \sum_i \tau_i$; Model Soups [12] averages checkpoints. TIES-Merging [13] resolves sign conflicts through trimming, majority vote, and disjoint averaging. DARE [14] randomly prunes delta parameters with rescaling. Consensus Merging [15] identifies and suppresses catastrophic and selfish parameters based on cross-task agreement patterns. Additional methods in this class include PCB-Merging [26] and Breadcrumbs [27]. LiNeS [17] applies layer-increasing scalar scaling to the merged weights to prevent catastrophic forgetting as a post-merge arithmetic correction. All these operators decide which per-parameter values to retain, drop, or rescale; none examines the directional geometry across task vectors.

SVD-based model merging. A separate line of data-free methods moves from element-wise rules to subspace operations. TSV-Merge [23] performs per-task SVD [22] and orthogonalizes the retained singular vectors via a Procrustes transformation. KnOTS [28] uses joint SVD to align concatenated LoRA [29] updates before applying a merging operator. Iso-Merging [16] decomposes the merged TA output into common and task-specific subspaces with isotropic scaling. The common subspace is defined by the top-k singular vectors of the summed task matrices, while task-specific subspaces are extracted from each task's residual after projecting out the common subspace. Subspace Boosting [24] diagnoses rank collapse in the merged output and mitigates it by boosting smaller singular values, using HO-GSVD [20, 21] as an interpretability tool. These post-merge methods tie their decomposition to a specific merge operator, so the result cannot be reused across methods. Moreover, they repair the merged output after interference has already occurred rather than preventing it. CUSD decomposes each task vector into common and unique subspaces before any merging operator is applied. Because the decomposition depends only on $\{\tau_i\}$, it pairs with any arithmetic-based method and gives common and unique components independent hyperparameters.

Optimization-based model merging. Unlike the non-iterative operators above, methods in this class exploit intrinsic properties of task vectors to construct optimization objectives solved iteratively. AWD [30] proves that inter-task interference is minimized when task vectors are mutually orthogonal, and proposes adaptive weight disentanglement to push them toward orthogonality. DOGE [31] formulates merging as a constrained optimization problem, interprets task vectors as accumulated gradients, and solves it via adaptive projected gradient descent. WUDI-Merging [32] shows that a linear layer's task vectors span an approximate linear subspace of the corresponding inputs, and uses this property to construct a data-free loss that minimizes the interference upper bound. DOP [33] combines data-space approximation with orthogonal projection and designs a data-free loss balancing stability and plasticity for continual merging. Each method optimizes toward a specific structural prior, such as orthogonality (AWD), gradient alignment (DOGE), or subspace preservation (WUDI, DOP), and its effectiveness depends on how well that prior matches the actual task-vector geometry.

2.2 Data-Dependent Model Merging

A second category of methods uses data-derived signals, including Fisher information [34], activation statistics, or unlabeled test samples, to guide the merging process. Fisher Merging [35] weights parameters by diagonal Fisher information estimated from labeled training data. RegMean [36] minimizes a linear regression reconstruction loss using activation inner-product matrices. ZipIt [37] computes activation statistics on unlabeled data to align and fuse redundant features across networks. AdaMerging [38] learns task-wise or layer-wise merging coefficients via entropy minimization on

unlabeled test samples. Representation Surgery [39] learns a linear transformation to repair merged representations using unsupervised data. Twin-Merging [25] decomposes each task vector via per-task SVD into shared and exclusive components, then trains a test-time router on task data to dynamically weight the exclusive components at inference. These methods require Fisher statistics, activations, or unlabeled samples beyond model weights alone and address a different setting from the data-free problem considered here.

3 Method

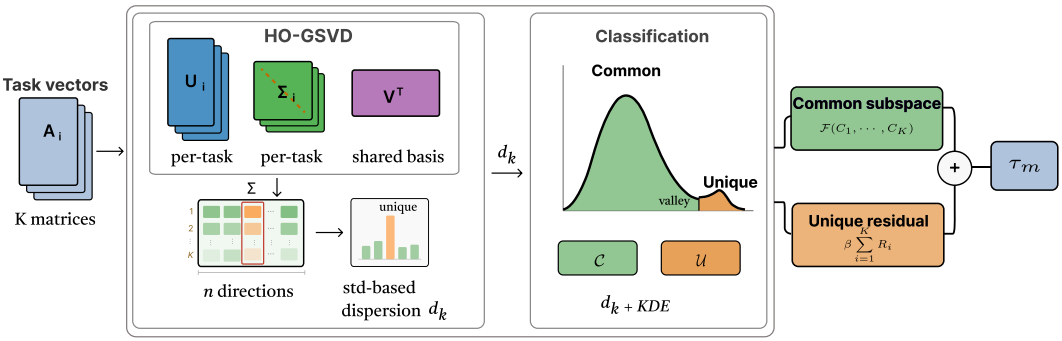


Figure 2: CUSD pipeline overview. Task vectors A_i are projected into an HO-GSVD-derived shared right basis V and directional energies Σ. A std-based dispersion score with a KDE-based threshold partitions directions into common ($\mathcal{C}$) and unique ($\mathcal{U}$). CUSD merges common components with $\mathcal{F}$ and retains unique residuals through $\beta \sum_{i=1}^{K} R_i$.

3.1 Preliminaries

Given a pre-trained base model θ_{base} and K task-specific models $\theta_1, \ldots, \theta_K$ fine-tuned on tasks $1, \ldots, K$, the task vector for task i is $\tau_i = \theta_i - \theta_{\text{base}}$ [11]. Arithmetic-based merging can be written as

$$\tau_m = \mathcal{F}(\tau_1, \ldots, \tau_K), \qquad \theta_{\text{merged}} = \theta_{\text{base}} + \tau_m, \tag{1}$$

where τ_m denotes the merged task vector, $\mathcal{F}$ denotes a downstream merging operator and θ_{merged} indicates the merged model. This work targets data-free arithmetic-based operators, which instantiate $\mathcal{F}$ as non-iterative per-parameter rules applied to $\{\tau_i\}$.

CUSD does not change this formulation; instead, it pre-processes the task vectors before $\mathcal{F}$ is applied. These operators act on scalar entries of $\{\tau_i\}$ and therefore do not expose the matrix-level directional structure of a task update. CUSD instead projects each matrix-structured layer into a shared directional basis, where directions with comparable cross-task energy can be separated from directions dominated by one or a few tasks (Figure 2). For each retained layer ℓ with matrix-structured weights, we denote the corresponding per-task matrices by $A_1^{(\ell)}, \ldots, A_K^{(\ell)} \in \mathbb{R}^{m \times n}$; all subsequent operations are applied independently per layer, and we suppress ℓ for clarity.

3.2 HO-GSVD-Derived Shared Basis

A standard SVD decomposes a single matrix as $A = U\Sigma V^\top$. Applied separately to K task matrices, it produces unrelated right bases. Classical HO-GSVD instead seeks a shared right factor and is commonly written as

$$A_i = U_i \Sigma_i V^\top, \qquad i \in [K],$$

which motivates comparing task matrices in a common right basis. CUSD uses the regularized and symmetrized Gram-quotient construction in Appendix A.2 to obtain an orthonormal shared basis $V = [v_1, \ldots, v_n]$, which is required for Eq. 4.

For each shared direction, CUSD records the directional energy $\sigma_{ik} = \|A_i v_k\|_2$ and collects these values into $\Sigma = [\sigma_{ik}] \in \mathbb{R}^{K \times n}$.

Algorithm 1 CUSD: Common–Unique Subspace Decomposition

Require: $\tau_1, \ldots, \tau_K$ (task vectors); $\mathcal{F}(\cdot)$ (any arithmetic-based method); $\beta \in [0, 1]$
Ensure: θ_{merged} (merged model)
1: **for** each retained layer ℓ with matrix-structured weights **do**
2: $V, \Sigma \leftarrow \text{SharedBasis}\left(A_1^{(\ell)}, \ldots, A_K^{(\ell)}\right)$ {Alg. 2}
3: $\mathcal{C}, \mathcal{U} \leftarrow \text{Classify}(\Sigma)$ {std-based dispersion + KDE; §3.3}
4: $P_{\mathcal{C}} \leftarrow V_{\mathcal{C}} V_{\mathcal{C}}^{\top}$; $P_{\mathcal{U}} \leftarrow I - P_{\mathcal{C}}$ {Orthogonal projectors}
5: **for** $i = 1, \ldots, K$ **do**
6: $C_i \leftarrow A_i P_{\mathcal{C}}$; $R_i \leftarrow A_i P_{\mathcal{U}}$ {Prop. 2}
7: **end for**
8: $\tau_m^{(\ell)} \leftarrow \mathcal{F}(C_1, \ldots, C_K) + \beta \sum_{i=1}^{K} R_i$ {Eq. 5}
9: **end for**
10: **return** $\theta_{\text{merged}} \leftarrow \theta_{\text{base}} + \tau_m$

Because all tasks are evaluated along the same directions, these energies are directly comparable across tasks. This shared-basis construction is computed once for each retained matrix-structured layer and reused across merging methods.

3.3 The CUSD Algorithm

Std-based dispersion score. Given the matrix Σ, CUSD measures the cross-task imbalance of each shared direction. For direction k, the std-based dispersion score is

$$d_k = \text{Std}_{i \in [K]}(\tilde{\sigma}_{ik}), \qquad \tilde{\sigma}_{ik} = \begin{cases} \sigma_{ik}, & \text{absolute scoring,} \\ \frac{\sigma_{ik}}{K^{-1} \sum_i \sigma_{ik}}, & \text{relative scoring.} \end{cases} \tag{2}$$

Directions with comparable cross-task energy receive small d_k, whereas directions dominated by one or a few tasks receive large d_k. Relative scoring removes the per-direction mean energy scale and measures relative cross-task imbalance. We compare the two scoring choices empirically in Section 4.3.

KDE. The threshold separating common from unique is determined automatically from the distribution of $\{d_k\}_{k=1}^{n}$ via kernel density estimation (KDE) with Scott's bandwidth [40] (full details appear in Appendix A.3). Let δ denote the detected valley. CUSD then sets

$$\mathcal{C} = \{k : d_k < \delta\}, \qquad \mathcal{U} = \{k : d_k \geq \delta\}. \tag{3}$$

Subspace merging. Given the partition $(\mathcal{C}, \mathcal{U})$, let $V_{\mathcal{C}}$ and $V_{\mathcal{U}}$ denote the columns of V indexed by $\mathcal{C}$ and $\mathcal{U}$, respectively. Define

$$P_{\mathcal{C}} = V_{\mathcal{C}} V_{\mathcal{C}}^{\top}, \qquad P_{\mathcal{U}} = V_{\mathcal{U}} V_{\mathcal{U}}^{\top} = I - P_{\mathcal{C}}, \qquad C_i = A_i P_{\mathcal{C}}, \qquad R_i = A_i P_{\mathcal{U}}. \tag{4}$$

Because $P_{\mathcal{C}}$ and $P_{\mathcal{U}}$ are complementary orthogonal projectors, this gives an exact orthogonal decomposition: $C_i + R_i = A_i$ and $\langle C_i, R_i \rangle_F = 0$; proof in Appendix A.4. The common component C_i contains directions that are consistently emphasized across tasks, whereas the unique residual R_i contains directions whose energy is concentrated in one or a few tasks and therefore reflects task-specific variation. CUSD therefore applies the merging operator $\mathcal{F}$ only to $\{C_i\}$, while the unique residuals $\{R_i\}$ are summed and scaled by $\beta \in [0, 1]$. The resulting layer-wise merged task vector is

$$\tau_m = \mathcal{F}(C_1, \ldots, C_K) + \beta \sum_{i=1}^{K} R_i. \tag{5}$$

Setting $\beta = 0$ discards all unique directions; $\beta = 1$ retains them in full. Algorithm 1 summarizes the complete per-layer CUSD procedure. The decomposition (lines 2–5) depends only on the task vectors and the classification rule, not on $\mathcal{F}$. This pre-processing is computed once and reused with any arithmetic-based merging method.

Overall, CUSD augments an arithmetic-based merging operator with three components: a shared-basis decomposition, a std-based common–unique partition, and a β-controlled unique residual.

4 Experiments

4.1 Experimental Setup

Models and tasks. We evaluate CUSD on 11 base models covering vision encoders, text encoders, and LLMs: CLIP ViT-B/32, ViT-B/16, and ViT-L/14 [19, 41]; RoBERTa-Base and RoBERTa-Large [42]; and MergeBench Llama and Gemma models [18, 43, 44]. The evaluation follows prior model- merging benchmarks on eight image-classification datasets [32, 38], eight GLUE NLU tasks [32, 45], and MergeBench code, math, and multilingual LLM tasks [43]. Full model and dataset details are provided in Appendix A.5.

Baselines and implementation. We pair CUSD with four representative data-free arithmetic-based merging methods: Task Arithmetic [11], TIES-Merging [13], DARE [14], and Consensus Merging [15]. For each method, we report the plain merge and the corresponding CUSD-preprocessed merge; TSV-Merge [23] and Iso- Merging [16] are included in the subspace-based comparison. We also evaluate CUSD preprocessing with WUDI-Merging [32] and ACE-Merging [7]. CUSD is applied to matrix-structured linear-layer task vectors. Unless otherwise specified, baseline hyperparameters follow the original papers or MergeBench defaults and are kept fixed between plain and CUSD-preprocessed runs. Additional implementation details are provided in Appendix A.5, and Section 4.3 reports the scoring mode and β sensitivity used in the main experiments.

4.2 Main Results

Overall performance. Table 1 reports average normalized performance across all model families. Parenthesized values denote changes from the corresponding plain merge, and boldface marks the best score in each model column. CUSD improves 42 of the 44 model–operator combinations, with one tie and one small decrease. The largest gain is 22.5 normalized points on RoBERTa-Large, and all ViT and RoBERTa settings improve. On LLMs, CUSD improves 22 of 24 settings. Detailed task-level results in Appendix A.6 show that these gains often reflect an improved multi-task trade-off: small drops on already strong tasks are accompanied by larger gains on other tasks, consistent with separating unique directions before arithmetic aggregation.

Table 1: **Overall performance across all models.** We report average normalized scores.

Method	with CUSD	ViT-B/32	ViT-B/16	ViT-L/14	RoBERTa-B	RoBERTa-L	Llama-3B	Llama-8B	Llama-3B-Inst	Llama-8B-Inst	Gemma-2b	Gemma-9b
Fine-tuned	—	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Task Arithmetic	×	71.8	76.8	87.6	53.5	58.3	69.6	93.7	73.5	82.5	94.2	92.0
	✓	85.0 (+13.2)	88.9 (+12.1)	93.0 (+5.4)	68.7 (+15.1)	**70.0** (+11.7)	69.6 (0.0)	**94.1** (+0.4)	75.8 (+2.3)	86.5 (+4.0)	95.7 (+1.5)	94.8 (+2.8)
TIES-Merging	×	74.0	78.5	88.5	51.1	47.3	69.2	93.7	72.7	82.6	94.1	91.1
	✓	**87.0** (+13.0)	**90.3** (+11.8)	**94.8** (+6.3)	**69.3** (+18.2)	69.8 (+22.5)	69.7 (+0.5)	**94.1** (+0.4)	75.5 (+2.8)	86.2 (+3.6)	**97.3** (+3.2)	95.3 (+4.2)
DARE	×	71.7	76.6	87.5	54.0	58.8	68.9	**94.1**	73.4	83.1	94.1	91.2
	✓	84.8 (+13.2)	88.8 (+12.2)	92.9 (+5.4)	68.3 (+14.3)	69.5 (+10.7)	70.3 (+1.4)	93.8 (-0.3)	75.4 (+2.0)	86.4 (+3.3)	96.2 (+2.1)	94.8 (+3.6)
Consensus	×	73.2	78.0	88.0	55.2	61.9	67.4	93.8	76.2	86.2	88.8	94.6
	✓	85.0 (+11.8)	88.8 (+10.8)	92.8 (+4.9)	67.3 (+12.1)	**70.0** (+8.1)	**70.6** (+3.2)	93.9 (+0.1)	**76.6** (+0.4)	**87.5** (+1.3)	93.1 (+4.3)	**95.5** (+0.9)

Results on encoder models. On vision encoders, CUSD improves all four operators on all three ViT models, with average gains of about 12 points on ViT-B/32 and ViT-B/16. Appendix Tables 8–11 show a trade-off pattern: with TIES-Merging, CUSD reduces the already strong MNIST score by only 0.1 points while improving three weaker tasks by 8.5–10.7 points. On text encoders, CUSD improves every operator on both RoBERTa models, and the detailed GLUE tables show that STS-B changes from negative plain-merge scores to positive CUSD scores across operators. These trends support the view that common–unique separation improves the multi-task balance in encoder merging.

Results on LLMs. On generative LLMs, CUSD improves most settings. The strongest gains appear on the Llama Instruct and Gemma models, while Llama-3.1-8B leaves little headroom because plain merges already reach normalized averages around 94; the only negative LLM change is a 0.3-point decrease for DARE on this model. The role of partial unique- residual scaling in LLMs is examined in the β ablation (Section 4.3) and analysis (Section 4.4). Full task-level LLM results are provided in Appendix Tables 13–18.

Comparison with other merging methods. Table 2 compares CUSD+TA with TSV-M, Iso-C, and Iso-CTS across four model families. For LLMs, the SVD baselines report the best-observed scores

Table 2: Comparison with subspace-based merging methods. Average normalized scores are reported.

Method	ViT B/16	RoBERTa Base	Llama 3.2-3B	Gemma 2-2b
Fine-tuned	100.0	100.0	100.0	100.0
TSV-M	93.4	45.9	64.5	96.1
Iso-C	94.5	42.9	65.7	**97.7**
Iso-CTS	**95.6**	42.8	64.5	97.2
CUSD+TA	88.9	**68.7**	**69.6**	95.7

Table 3: Effect of CUSD preprocessing on WUDI-Merging and ACE-Merging. Average normalized scores are reported.

Method	ViT B/16	RoBERTa Base	Llama 3.2-3B	Gemma 2-2b
Fine-tuned	100.0	100.0	100.0	100.0
WUDI	**95.6**	69.3	65.6	93.6
CUSD+WUDI	94.0	**80.7**	**66.9**	**94.9**
ACE	**96.4**	**85.6**	61.9	93.7
CUSD+ACE	94.7	81.1	**63.3**	**98.6**

from post-hoc scaling searches on the test sets. Iso-CTS performs best on ViT-B/16 and Iso-C on Gemma-2-2b, whereas CUSD+TA achieves the highest scores on RoBERTa-Base and Llama-3.2-3B.

We further evaluate CUSD preprocessing with WUDI-Merging and ACE-Merging (Table 3). CUSD improves WUDI in three of four settings and ACE on both LLMs, while encoder results are mixed.

4.3 Ablation Study

Dispersion scoring choice. We compare absolute and relative scoring (Eq. 2). Relative scoring performs better in six of the eight LLM model–operator settings, while absolute scoring performs better with Task Arithmetic on both models (Table 4). These results support relative scoring as the preferred choice for the evaluated LLM settings. Section 4.4 examines how cross-task directional-energy patterns motivate the choice between absolute and relative scoring.

Table 4: Absolute vs. relative dispersion scoring on decoder LLMs. We report normalized average performance; green values denote gains over the corresponding plain merge.

Method	**Llama-3.2-3B**				**Gemma-2-2b**			
	w/o CUSD	**CUSD** (s_{abs})	**CUSD** (s_{rel})	s_{rel} - s_{abs}	**w/o CUSD**	**CUSD** (s_{abs})	**CUSD** (s_{rel})	s_{rel} - s_{abs}
Task Arithmetic	69.6	**70.0**(+0.4)	69.6(+0.0)	-0.4	94.2	**96.9**(+2.7)	95.7(+1.5)	-1.2
TIES-Merging	69.2	69.5(+0.3)	**69.7**(+0.5)	0.2	94.1	95.9(+1.8)	**97.3**(+3.2)	1.4
DARE	68.9	69.6(+0.7)	**70.3**(+1.4)	0.7	94.1	95.8(+1.7)	**96.2**(+2.1)	0.4
Consensus	67.4	70.0(+2.6)	**70.6**(+3.2)	0.6	88.8	91.7(+2.9)	**93.1**(+4.3)	1.4

Direction selection. To examine whether the choice of unique directions matters, we compare CUSD with Random, Reversed (lowest dispersion), and Top-energy selection. In the matched controls, each method selects the same number of unique directions as CUSD in every layer. We also test Top-energy with fixed selection fractions. CUSD outperforms all controls on both backbones (Table 5); Top-energy is more competitive than Random and Reversed, consistent with the energy concentration of CUSD's unique directions.

Common–unique merging design. We apply $\mathcal{F}$ to common components only (Common, Eq. 5), unique components only (Unique), or both separately (Both). Common performs best in all four reported settings (Table 6), supporting common aggregation with separately controlled unique-residual retention.

Unique-residual coefficient. The coefficient β controls the retention strength of the unique residual. We sweep $\beta \in \{0.0, 0.1, \ldots, 1.0\}$ to examine how sensitive CUSD is to this residual contribution. Figure 3 shows that encoders favor small β, whereas decoder LLMs benefit from moderate residual retention, typically around $\beta \in [0.2, 0.5]$. Section 4.4 analyzes this difference. Since per-operator sweeps in Figure 13 show consistent trends across downstream merge operators, Figure 3 reports the average over the four methods.

We further examine how β interacts with the downstream hyperparameter λ, using Task Arithmetic as a case study. For CUSD+TA, $\tau_m = \lambda \sum_i C_i + \beta \sum_i R_i$, which reduces to plain TA when $\beta = \lambda$.

Table 5: Direction-selection ablation for CUSD+TA. Average accuracy (%) is reported; Random shows mean ± standard deviation over three seeds.

Model	Random	Reversed	Top-energy (matched)	Top-energy (0.2%)	Top-energy (0.5%)	Top-energy (1.0%)	CUSD
ViT-B/32	59.20 ± 0.08	59.57	73.78	74.75	73.68	70.97	**76.48**
ViT-B/16	65.64 ± 1.17	65.09	81.07	80.84	80.34	78.97	**81.74**

Table 19 reports a post-hoc joint sweep on ViT-B/32. CUSD+TA outperforms plain TA at the same λ in every tested configuration with $\lambda \geq 0.3$ and $\beta < \lambda$. These results show that reducing the unique-residual contribution relative to the common component can improve merging performance in this setting.

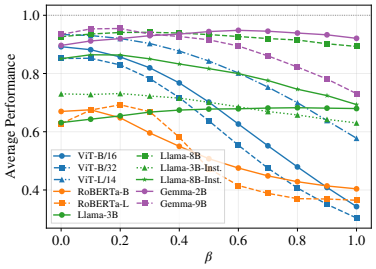


Figure 3: Sensitivity to the unique-residual retention coefficient β across all 11 models.

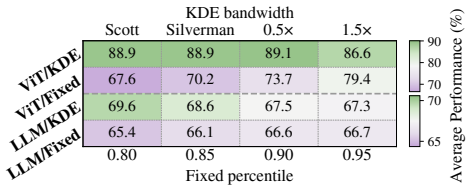


Figure 4: Average performance under KDE-based thresholds and fixed percentile thresholds. For KDE rows, columns denote bandwidth choices; for fixed-threshold rows, columns denote percentile cutoffs.

Adaptive KDE thresholding. CUSD uses KDE valley detection to determine the common–unique partition from the dispersion-score distribution, avoiding a fixed global cutoff. Figure 4 compares this adaptive threshold with fixed percentile thresholds. KDE thresholding consistently outperforms fixed thresholds on both ViT and LLM settings, with the largest gap on ViT where fixed percentiles are sensitive to the chosen cutoff. The result is also stable across standard KDE bandwidth choices: Scott's and Silverman's rules give nearly identical performance, and moderate bandwidth changes preserve the overall advantage of KDE. This supports using a data-driven layer-adaptive partition rather than tuning a fixed percentile threshold.

Table 6: Merging-design ablation. Average normalized scores are reported; Common denotes CUSD.

Model	Operator	Common	Unique	Both
ViT-B/16	TIES	**90.3**	73.0	72.4
ViT-B/16	DARE	**88.8**	76.4	76.3
Gemma-2-2b	TIES	**97.3**	94.2	95.2
Gemma-2-2b	DARE	**96.2**	93.6	94.2

Table 7: One-time CUSD decomposition cost.

Model	Tasks	Layers	Total time	Time per layer	GPU peak
ViT-B/32	8	48	16.5s	0.343s	1.1 GB
ViT-B/16	8	48	16.3s	0.340s	1.1 GB
ViT-L/14	8	96	48.5s	0.505s	1.9 GB
Llama-3B	3	196	67.5s	0.345s	5.0 GB
Llama-8B	3	224	217.2s	0.970s	15.4 GB

4.4 Analysis

What CUSD separates. The common–unique partition gives an operational view of the two-regime structure observed in Figure 1 and further visualized in Appendix A.1: directions assigned to the common subspace form a broad low-dispersion bulk, whereas directions assigned to the unique residual are often fewer but carry substantially higher per-direction energy. Figure 5 confirms this pattern across the evaluated model families. Unique directions consistently have higher per-direction energy than common directions, especially in encoder models. This shows that CUSD does not merely discard weak or low-energy directions; it separates an energy-concentrated residual subspace whose directions are unevenly supported across tasks. As summarized in Figure 6, this concentration

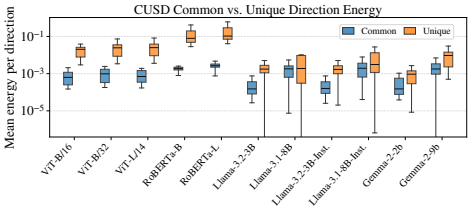


Figure 5: Layer-wise mean energy per direction in common and unique subspaces.

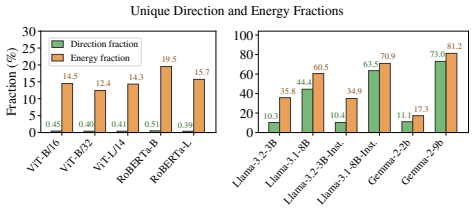


Figure 6: Fractions of directions and total energy assigned to the unique subspace.

is particularly strong in encoders: fewer than 1% of directions account for roughly 12–20$\%$ of the directional energy.

Understanding dispersion scoring. For decoder LLMs, small absolute dispersion does not necessarily mean that a direction is uniformly balanced across tasks. If such directions were simply shared evenly by all tasks, mean normalization would keep their relative dispersion small. Instead, Figure 8 shows that the low-dispersion mass under s_{abs} is substantially reduced under s_{rel}. Together with the task-imbalanced energy patterns in Figure 7, this indicates that many low-absolute-dispersion directions are low or moderate energy directions that are nevertheless dominated by specific tasks. Relative scoring removes this raw-scale suppression and exposes these directions as high-dispersion.

The encoder case is different. In ViT, the heatmap in Figure 7 shows more balanced directional energies across tasks, except for a small high-energy region with visible task dominance. In this regime, s_{abs} preserves the energy-concentration signal, whereas s_{rel} gives low-energy directions the same scale as high-energy ones and can amplify small relative fluctuations. The ViT distributions in Figure 8 are consistent with this effect: relative scoring introduces a longer high-dispersion tail despite the comparatively balanced heatmap. This can weaken the separation between high-energy task-dominated directions and low-energy variations. This supports our main-experiment choice of using s_{abs} for the evaluated encoder settings and s_{rel} for the evaluated LLM settings. Appendix Figures 11 and 12 provide additional layer heatmaps.

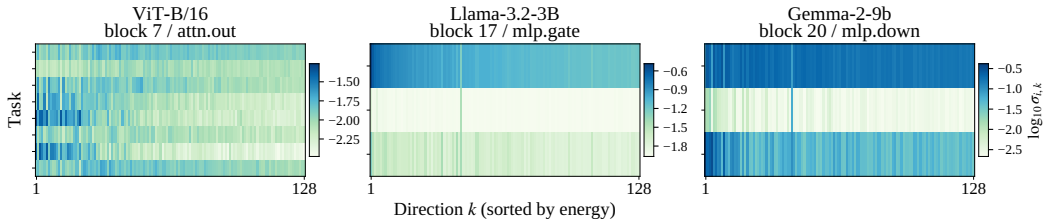
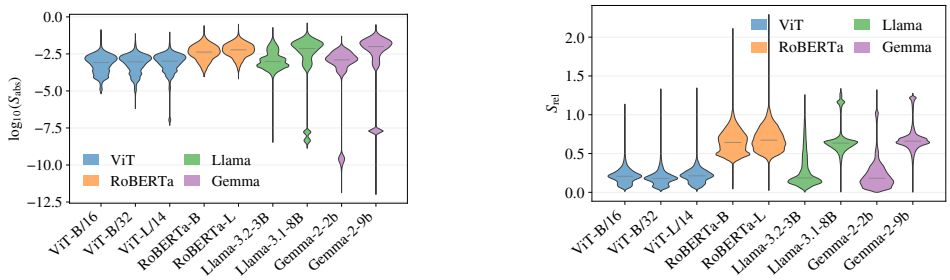


Figure 7: Per-task directional singular values for representative ViT, Llama, and Gemma layers. Each panel shows the HO-GSVD singular values σ_{ik} of task vectors along the top shared directions.

Unique residual scale across model families. Figure 6 shows that the unique subspace differs across the evaluated model families and task groups. In encoders, the unique subspace is extremely sparse: it contains fewer than 1% of directions but carries roughly 12–20$\%$ of the directional energy. This creates a concentrated residual component, so increasing β can quickly reintroduce task-specific interference; this explains why encoder performance in Figure 3 is sensitive to β and why $\beta = 0$ is a stable default. In decoder LLMs, the unique subspace is much larger and carries a substantial fraction of the energy, from 17.3% to 81.2%. Removing this broader residual component entirely can discard task-specific generative information, while moderate residual scaling preserves part of it without directly merging it into the common component. This explains why LLMs prefer nonzero β and remain less sensitive over a moderate range of β values.

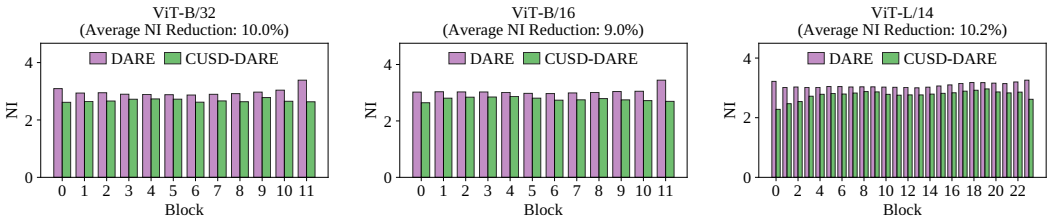
Interference reduction. To examine whether CUSD reduces interference inside the merged update, we measure a normalized interference proxy (NI), defined in Appendix A.9. Figure 9 compares DARE and CUSD-DARE on the three ViT backbones. CUSD consistently lowers NI across blocks, with



(a) Absolute dispersion across all blocks/directions. (b) Relative dispersion across all blocks/directions.

Figure 8: Distribution of dispersion scores under absolute and relative scoring. Mean normalization changes the dispersion distribution, exposing scale-suppressed task imbalance in the evaluated LLM settings while amplifying low-energy relative fluctuations in the evaluated encoder settings.

average reductions of 10.0%, 9.0%, and 10.2% on ViT-B/32, ViT-B/16, and ViT-L/14, respectively. This supports the proposed mechanism: separating common and unique directions before merging reduces the parameter-space interference that arises when arithmetic rules are applied directly to task-dominated directions. The same trend holds for other downstream merge operators, as reported in Appendix A.9.



(a) ViT-B/32. (b) ViT-B/16. (c) ViT-L/14.

Figure 9: **Per-block interference reduction.** DARE and CUSD-DARE are compared on ViT models using the normalized interference proxy (NI); lower values indicate less deviation from the task-specific updates.

Computation cost. CUSD adds a one-time shared basis decomposition whose common components and unique residuals are reused across merge operators. Table 7 shows that ViT decompositions take 16.3–48.5s with at most 1.9GB peak GPU memory, while Llama-3.1-8B takes 217.2s with 15.4GB peak memory. The cost is therefore practical and amortized across downstream arithmetic methods.

5 Conclusion

This work studies data-free arithmetic merging from a direction-level perspective. In an HO-GSVD-derived shared basis, task-vector matrices exhibit broadly common directions and high-dispersion unique directions whose energy is concentrated in only a subset of tasks. CUSD uses this observation to decompose task vectors before arithmetic aggregation, merge only the common components, and control the unique residuals separately. Since the decomposition depends only on model weights, the same common components and unique residuals can be reused across downstream arithmetic merge operators.

Across vision encoders, text encoders, and decoder LLMs, CUSD improves most model–operator combinations and reduces parameter-space interference. The ablations further show that effective unique-residual scaling varies across the evaluated model families and task groups, supporting the separate control of common aggregation and unique residuals. These results support separating common and task-dominated directions before arithmetic merging.

Acknowledgments and Disclosure of Funding

This work was supported by the UKRI Future Leaders Fellowship under grants MR/S017062/1 and MR/X011135/1; the National Natural Science Foundation of China under grants 62376056 and 62076056; Isambard AI Open Access (AIOA - 01 - 2026-06-04 - EvoRNA: Predictive Scaling of Evolution-Aware and Multimodal RNA Foundation Models for Crop Resilience); Modal Labs through a research grant of compute credits; the Royal Society Faraday Discovery Fellowship (FDF/S2/251014); BBSRC Transformative Research Technologies (UKRI1875); the Royal Society International Exchanges Award (IES/R3/243136); the Kan Tong Po Fellowship (KTP/R1/231017); the Amazon Research Award; and the Alan Turing Fellowship.

References

[1] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):1–41, 2026.

[2] Mingyang Song and Mao Zheng. Model merging in the era of large language models: Methods, applications, and future directions. *arXiv preprint arXiv:2603.09938*, 2026.

[3] Yuxuan Yao, LIU Shuqi, Zehua Liu, Qintong Li, LIU Mingyang, Xiongwei Han, Zhijiang Guo, Han Wu, and Linqi Song. Activation-guided consensus merging for large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[4] Yuanyi Wang, Yanggan Gu, Yiming Zhang, Qi Zhou, Zhaoyi Yan, Congkai Xie, Xinyao Wang, Jianbo Yuan, and Hongxia Yang. Model merging scaling laws in large language models. *arXiv preprint arXiv:2509.24244*, 2025.

[5] Amin Heyrani Nobari, Kaveh Alim, Ali ArjomandBigdeli, Akash Srivastava, Faez Ahmed, and Navid Azizan. Activation-informed merging of large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[6] Marawan Gamal Abdel Hameed, Derek Tam, Pascal Jr Tikeng Notsawo, Colin Raffel, and Guillaume Rabusseau. Model merging via data-free covariance estimation. *arXiv preprint arXiv:2604.01329*, 2026.

[7] Bo Xu, Haotian Wu, Hehai Lin, Weiquan Huang, Beier Zhu, Yao Shu, and Chengwei Qin. Ace-merging: Data-free model merging with adaptive covariance estimation. *arXiv preprint arXiv:2603.02945*, 2026.

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

[9] Karsten Roth, Vishaal Udandarao, Sebastian Dziadzio, Ameya Prabhu, Mehdi Cherti, Oriol Vinyals, Olivier Hénaff, Samuel Albanie, Matthias Bethge, and Zeynep Akata. A practitioner's guide to continual multimodal pretraining. *arXiv preprint arXiv:2408.14471*, 2024.

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

[11] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

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

[13] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

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

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

[16] Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D Bagdanov, and Joost Van De Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. *arXiv preprint arXiv:2502.04959*, 2025.

[17] Ke Wang, Nikolaos Dimitriadis, Alessandro Favero, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Lines: Post-training layer scaling prevents forgetting and enhances model merging. In *The Thirteenth International Conference on Learning Representations*, 2025.

[18] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[20] Sri Priya Ponnapalli, Michael A Saunders, Charles F Van Loan, and Orly Alter. A higher-order generalized singular value decomposition for comparison of global mrna expression from multiple organisms. *PloS one*, 6(12):e28072, 2011.

[21] Idris Kempf, Paul J Goulart, and Stephen R Duncan. A higher-order generalized singular value decomposition for rank-deficient matrices. *SIAM Journal on Matrix Analysis and Applications*, 44(3):1047–1072, 2023.

[22] Virginia Klema and Alan Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

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

[24] Ronald Skorobogat, Karsten Roth, and Mariana-Iuliana Georgescu. Subspace-boosted model merging. *arXiv preprint arXiv:2506.16506*, 2025.

[25] Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. *Advances in Neural Information Processing Systems*, 37:78905–78935, 2024.

[26] Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim K Goh, Ho-Kin Tang, Daojing He, et al. Parameter competition balancing for model merging. *Advances in Neural Information Processing Systems*, 37:84746–84776, 2024.

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

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

[29] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

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

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

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

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

[34] Jorma J Rissanen. Fisher information and stochastic complexity. *IEEE transactions on information theory*, 42(1):40–47, 2002.

[35] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

[36] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

[37] George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. *arXiv preprint arXiv:2305.03053*, 2023.

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

[39] Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In *Proceedings of the 41st International Conference on Machine Learning*, pages 56332–56356, 2024.

[40] David W Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015.

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[42] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[43] Yifei He, Siqi Zeng, Yuzheng Hu, Rui Yang, Tong Zhang, and Han Zhao. Mergebench: A benchmark for merging domain-specialized llms. *arXiv preprint arXiv:2505.10833*, 2025.

[44] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[45] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 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, 2018.

[46] Shi-Ting Wang, Jin-Hua Zheng, Yuan Liu, Juan Zou, and Sheng-Xiang Yang. An extended fuzzy decision variables framework for solving large-scale multiobjective optimization problems. *Information Sciences*, 643:119221, 2023.

[47] Yingjie Zou, Yuan Liu, Juan Zou, Shengxiang Yang, and Jinhua Zheng. An evolutionary algorithm based on dynamic sparse grouping for sparse large scale multiobjective optimization. *Information Sciences*, 631:449–467, 2023.

[48] Tianmeng Hu, Yongzheng Cui, Biao Luo, and Ke Li. RIDER: 3D RNA inverse design with reinforcement learning-guided diffusion. In *Proc. International Conference on Learning Representations*, volume 2026, pages 79321–79346, 2026.

[49] Tianmeng Hu, Biao Luo, and Ke Li. CocoRNA: Collective RNA design with cooperative multi-agent reinforcement learning. In *Proc. International Conference on Machine Learning*, volume 306, pages 45495–45519, 2026.

[50] Wanli Ma, Tianmeng Hu, and Ke Li. Reflex: Rna ensemble generation via flexibility-calibrated stochastic bridge. *arXiv preprint arXiv:2610.26261*, 2026.

[51] Tianmeng Hu, Biao Luo, and Ke Li. TANGO: RNA topology and geometry co-design. *arXiv preprint arXiv:2610.20277*, 2026. URL `https://arxiv.org/abs/2610.20277v1`.

A Appendix

A.1 Additional Directional-Structure Evidence

Figure 10 extends the motivation analysis in Figure 1 to all evaluated model families, including ViT, RoBERTa, Llama, and Gemma. Each panel shows the task-vector directional energy distribution in the HO-GSVD-derived shared basis for a representative layer. Across architectures and task groups, the distributions consistently contain a low-dispersion bulk together with a high-dispersion tail of task-dominated directions. This recurring pattern supports the central premise of CUSD: arithmetic merging should first separate directions that are broadly common across tasks from directions whose energy is concentrated in one or a few task-specific models.

A.2 HO-GSVD-Derived Shared-Basis Details

This appendix gives the regularized HO-GSVD-derived shared-basis construction used to compute V and Σ for each matrix-structured layer. The construction follows the pairwise Gram-quotient idea of HO-GSVD, with ridge regularization and symmetrization for neural-network task-vector matrices. The symmetrization step produces an orthonormal basis suitable for CUSD's projection step, so CUSD does not rely on an exact classical HO-GSVD factorization. Algorithm 2 summarizes the implemented procedure.

Let $A_1, \ldots, A_K$ denote the task-vector matrices for one matrix-structured layer, where $A_i \in \mathbb{R}^{m\times n}$ and all tasks share the same column dimension n. For a fixed $\pi > 0$, define

$$G_i = A_i^\top A_i + \pi I_n, \qquad i = 1, \ldots, K. \tag{6}$$

Each G_i is symmetric positive definite, so all quotient terms below are well defined. We form the regularized pairwise quotient average

$$\widetilde{S}_\pi = \frac{1}{K(K-1)} \sum_{1\le i<j\le K} \left(G_j^{-1} G_i + G_i^{-1} G_j\right). \tag{7}$$

In practice, the products $G_j^{-1}G_i$ are computed by solving linear systems rather than explicitly forming matrix inverses. Since $\widetilde{S}_\pi$ is not guaranteed to be symmetric, CUSD uses

$$S_\pi = \frac{1}{2}\left(\widetilde{S}_\pi + \widetilde{S}_\pi^\top\right) \tag{8}$$

and computes the orthogonal eigendecomposition

$$S_\pi = V \Lambda V^\top, \qquad V^\top V = I_n. \tag{9}$$

The columns of $V = [v_1, \ldots, v_n]$ are sorted by descending eigenvalue. The symmetrization step preserves the pairwise quotient signal while producing an orthonormal basis for the projection step in CUSD.

For each task i and shared direction v_k, CUSD records the raw directional energy

$$\sigma_{ik} = \|A_i v_k\|_2, \qquad \Sigma = [\sigma_{ik}] \in \mathbb{R}^{K\times n}. \tag{10}$$

These raw directional energies form the matrix Σ used by CUSD for dispersion scoring.

Proposition 1 (Well-defined shared-basis construction)**.** *For any $\pi > 0$, the CUSD shared-basis construction in Algorithm 2 is well defined and returns an orthonormal basis V.*

Proof. For each task matrix A_i, the matrix $G_i = A_i^\top A_i + \pi I_n$ is symmetric positive definite because $\pi > 0$. Hence all linear systems involving G_i in Eq. 7 are well defined. The symmetrized quotient matrix $S_\pi = (\widetilde{S}_\pi + \widetilde{S}_\pi^\top)/2$ is real symmetric. By the spectral theorem, it admits an eigendecomposition $S_\pi = V\Lambda V^\top$ with $V^\top V = I_n$. Thus the construction returns an orthonormal shared basis. □

The orthogonality of V is the key property used in the lossless decomposition proof in Appendix A.4.

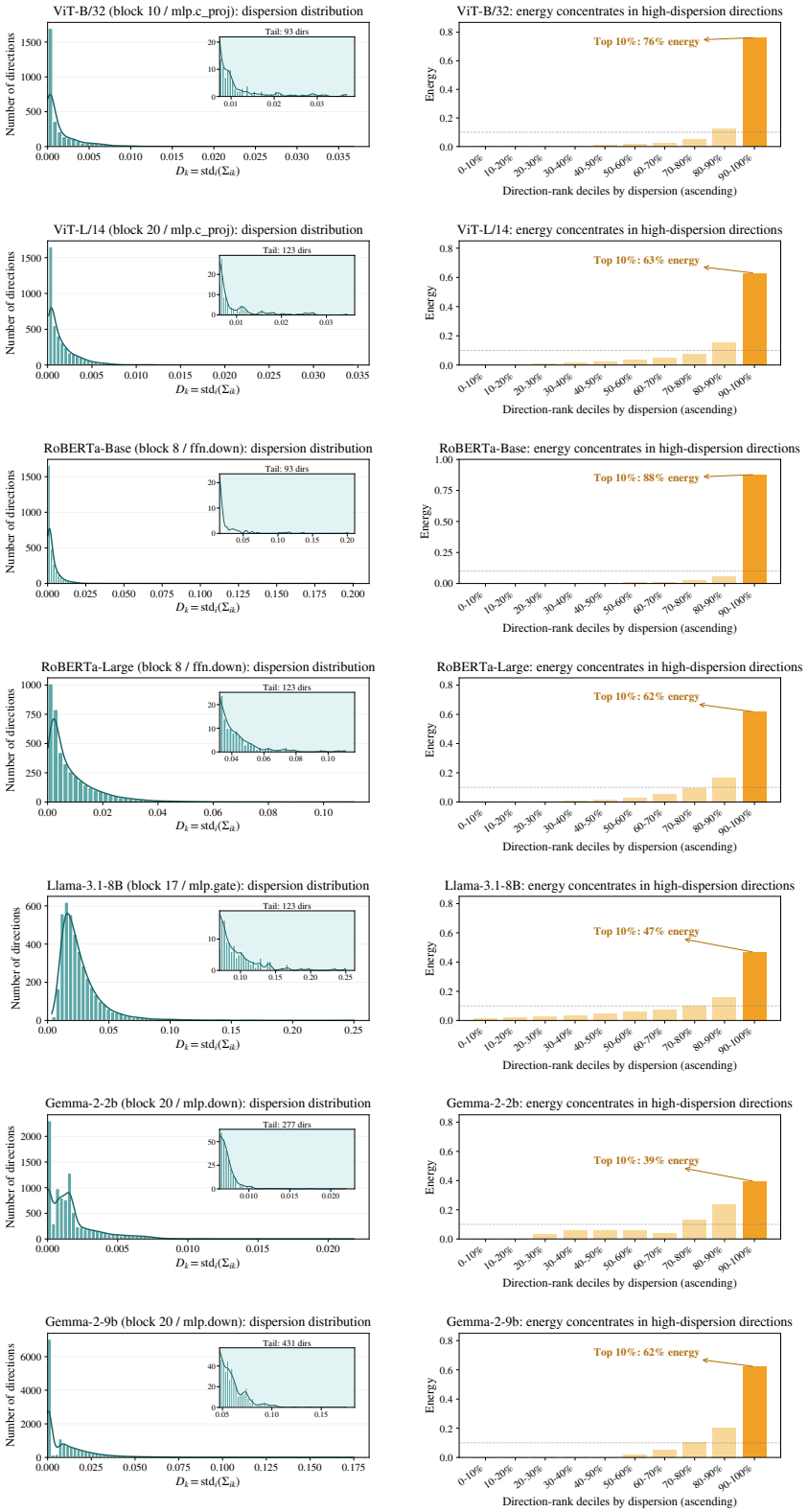


Figure 10: **Direction-level heterogeneity in task vectors.** Directional energy structure across all evaluated model families.

Algorithm 2 Regularized HO-GSVD-Derived Shared-Basis Construction

Require: $A_1, \ldots, A_K$ with $A_i \in \mathbb{R}^{m_i \times n}$; $K \geq 2$; $\pi = 10^{-3}$
Ensure: $V \in \mathbb{R}^{n \times n}$ (orthogonal shared basis); $\Sigma \in \mathbb{R}^{K \times n}$ (directional-energy magnitudes)
1: **Regularized Grams:** $G_i \leftarrow A_i^\top A_i + \pi I_n$ for $i = 1, \ldots, K$
2: **Pairwise quotient average:** $\widetilde{S}_\pi \leftarrow \dfrac{1}{K(K-1)} \displaystyle\sum_{1 \leq i < j \leq K} (\mathrm{solve}(G_j, G_i) + \mathrm{solve}(G_i, G_j))$
3: **Symmetrization:** $S_\pi \leftarrow \frac{1}{2}\left(\widetilde{S}_\pi + \widetilde{S}_\pi^\top\right)$
4: **Orthogonal eigendecomposition:** $S_\pi = V \Lambda V^\top$; sort columns of V by descending eigenvalue
5: **Magnitude extraction:** $\sigma_{ik} \leftarrow \|A_i v_k\|_2$ for all $i \in [K]$ and $k \in [n]$
6: **return** $V,\ \Sigma = [\sigma_{ik}]$

A.3 KDE Classification Details

CUSD selects the common–unique threshold from the empirical distribution of dispersion scores using kernel density estimation (KDE) [40]. This provides a layer-adaptive threshold without manually fixing a global cutoff.

Density Estimation Framework Given a set of n dispersion scores $\mathcal{D} = \{d_k\}_{k=1}^n$, we construct the density estimate $\hat{f}(d)$ using a Gaussian kernel:

$$\hat{f}(d) = \frac{1}{nh} \sum_{k=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(d - d_k)^2}{2h^2}\right). \tag{11}$$

The bandwidth h is set by Scott's rule for univariate data:

$$h = \hat{\sigma} n^{-1/5}, \tag{12}$$

where $\hat{\sigma}$ denotes the sample standard deviation of $\mathcal{D}$. We evaluate $\hat{f}(d)$ over a uniform grid of $N = 1{,}000$ points covering $[\min_k d_k, \max_k d_k]$.

Threshold Selection and Decision Rule The algorithm identifies candidate thresholds by locating local minima of the estimated density. Let $\mathcal{V}$ be the set of grid points corresponding to these local minima. If $\mathcal{V}$ is nonempty, CUSD selects the valley closest to the midpoint of the score range:

$$\delta = \arg\min_{v \in \mathcal{V}} \left| v - \frac{\min_k d_k + \max_k d_k}{2} \right|. \tag{13}$$

The final partition is

$$\mathcal{C} = \{k : d_k < \delta\}, \qquad \mathcal{U} = \{k : d_k \geq \delta\}.$$

If no valid valley is found, CUSD applies a conservative no-valley policy and assigns all directions in that layer to the common set $\mathcal{C}$.

A.4 Lossless Orthogonal Decomposition

Proposition 2. *Let $P_\mathcal{C} = V_\mathcal{C} V_\mathcal{C}^\top$ be the orthogonal projector onto the common directions, and let $P_\mathcal{U} = I - P_\mathcal{C}$. Set $C_i = A_i P_\mathcal{C}$ and $R_i = A_i P_\mathcal{U}$. Then for every task i: (i) $C_i + R_i = A_i$; (ii) $\langle C_i, R_i \rangle_F = 0$.*

Proof. The claim uses only the fact that the CUSD basis V is orthonormal. Let $V_\mathcal{C}$ contain the columns of V indexed by $\mathcal{C}$. Since $V^\top V = I$, the selected columns also satisfy $V_\mathcal{C}^\top V_\mathcal{C} = I$. Hence

$$P_\mathcal{C}^\top = (V_\mathcal{C} V_\mathcal{C}^\top)^\top = P_\mathcal{C}, \qquad P_\mathcal{C}^2 = V_\mathcal{C}(V_\mathcal{C}^\top V_\mathcal{C}) V_\mathcal{C}^\top = P_\mathcal{C}.$$

Thus $P_\mathcal{C}$ is the orthogonal projector onto the span of the common directions. Since $P_\mathcal{U} = I - P_\mathcal{C}$, we also have

$$P_\mathcal{U} + P_\mathcal{C} = I, \qquad P_\mathcal{U} P_\mathcal{C} = (I - P_\mathcal{C}) P_\mathcal{C} = P_\mathcal{C} - P_\mathcal{C}^2 = 0.$$

Therefore,

$$C_i + R_i = A_i P_{\mathcal{C}} + A_i P_{\mathcal{U}} = A_i (P_{\mathcal{C}} + P_{\mathcal{U}}) = A_i.$$

For orthogonality under the Frobenius inner product,

$$\langle C_i, R_i \rangle_F = \operatorname{tr}(C_i^\top R_i) = \operatorname{tr}\big(P_{\mathcal{C}} A_i^\top A_i P_{\mathcal{U}}\big)\,.$$

Using cyclic invariance of the trace,

$$\operatorname{tr}\big(P_{\mathcal{C}} A_i^\top A_i P_{\mathcal{U}}\big) = \operatorname{tr}\big(P_{\mathcal{U}} P_{\mathcal{C}} A_i^\top A_i\big) = 0,$$

because $P_{\mathcal{U}} P_{\mathcal{C}} = 0$. Hence $\langle C_i, R_i \rangle_F = 0$ for every task i. □

A.5 Experimental Setup

Models. We evaluate CUSD on 11 base models from four model families (ViT, RoBERTa, Llama, and Gemma), covering vision encoders, text encoders, and decoder-only LLMs. For vision experiments, we use CLIP ViT-B/32, ViT-B/16, and ViT-L/14 models [19, 41]. For Natural Language Understanding (NLU) experiments, we use RoBERTa-Base and RoBERTa-Large [42]. For generative LLM experiments, we use the MergeBench fine-tuned models [43] based on Llama-3.2-3B, Llama-3.1-8B, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Gemma-2-2b, and Gemma-2-9b [18, 44].

Datasets. For vision tasks, we use eight image classification datasets following prior work [32, 38]: Cars, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397, and SVHN. For NLU tasks, we follow prior model-merging evaluations [32] and use the eight GLUE tasks [45]: CoLA, SST-2, MRPC, STS-B, QQP, MNLI, QNLI, and RTE. For generative language tasks, we follow MergeBench [43] and evaluate code generation with Humaneval+ and MBPP+, mathematical reasoning with GSM8k, and multilingual understanding with M-MMLU, M-ARC, and M-Hellaswag.

Baselines. We evaluate CUSD with four representative data-free arithmetic-based merging methods: Task Arithmetic [11], TIES-Merging [13], DARE [14], and Consensus Merging [15]. For DARE and Consensus Merging, we use their Task Arithmetic instantiations. For each method, we report the plain merged model and the corresponding CUSD-preprocessed variant. We also include TSV-Merge [23] and Iso-Merging [16] for comparison with subspace-based merging methods. Fine-tuned task-specific models are included as reference upper bounds where available.

Implementation details. CUSD operates on matrix-structured linear-layer task vectors. We use absolute scoring for encoder models (ViT and RoBERTa) and relative scoring for decoder LLMs. Section 4.3 compares the dispersion scoring modes and analyzes sensitivity to the unique-residual coefficient β. Figure 3 provides empirical guidance on suitable β ranges for the evaluated model families. Given fixed hyperparameters, CUSD uses only model weights during decomposition and merging. Baseline hyperparameters follow the original papers or MergeBench defaults and are kept identical between plain and CUSD-preprocessed runs.

A.6 Main Results Details

Metrics. We report normalized scores relative to the corresponding fine-tuned task-specific references. For ViT and RoBERTa results in Table 1, task scores are first averaged in their original metric scale and the resulting average is normalized by the corresponding fine-tuned average. For LLMs, following [43], each benchmark score is normalized by the corresponding fine-tuned benchmark score before averaging.

Results on vision tasks. Tables 8–10 provide task-level results for the three ViT models used in the vision experiments. Each table reports the plain arithmetic merge and the corresponding CUSD merge on the eight image classification tasks. CUSD improves all four downstream operators on the three CLIP ViT backbones. The gains are largest on ViT-B/32 and ViT-B/16, where the average improvement across operators is about 12 normalized points, and remain positive on ViT-L/14, with an average gain of about 5 normalized points. The task-level results show a clear trade-off. For example, on ViT-L/14 with TIES-Merging, CUSD slightly reduces the already strong MNIST accuracy by 0.1 points, but improves EuroSAT, RESISC45, and DTD by 10.7, 8.5, and 9.2 points, respectively, raising the average task accuracy by 5.9 points. This pattern is consistent with CUSD

reducing the influence of task-dominated directions that would otherwise bias the arithmetic merge, thereby improving the overall multi-task balance. TIES-Merging with CUSD achieves the best normalized average on all three ViT models. Detailed task-level results are provided in Appendix Tables 8–10.

Table 8: Multi-task performance of merged ViT-L/14 models on eight tasks.

Method	with CUSD	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Avg.
Fine-tuned	—	92.5	84.2	91.0	99.2	99.7	99.7	83.6	98.1	93.5
Task Arithmetic	×	80.5	64.6	75.6	86.7	99.0	86.1	72.9	89.7	81.9
	✓	87.2 (+6.7)	71.9 (+7.2)	87.7 (+12.0)	92.6 (+6.0)	98.7 (-0.3)	94.5 (+8.4)	76.1 (+3.2)	87.2 (-2.5)	**87.0** (+5.1)
TIES-Merging	×	81.1	65.3	77.8	87.5	99.1	87.2	73.3	90.5	82.7
	✓	87.7 (+6.6)	74.5 (+9.2)	88.4 (+10.7)	95.0 (+7.5)	99.0 (-0.1)	95.7 (+8.5)	77.2 (+3.9)	91.2 (+0.8)	**88.6** (+5.9)
DARE	×	80.4	64.0	75.6	86.7	99.0	86.1	72.8	89.7	81.8
	✓	87.1 (+6.7)	71.6 (+7.6)	87.3 (+11.8)	92.7 (+6.0)	98.7 (-0.3)	94.4 (+8.3)	76.0 (+3.3)	87.2 (-2.6)	**86.9** (+5.1)
Consensus	×	80.8	65.0	76.7	87.0	99.0	86.6	73.0	89.7	82.2
	✓	87.2 (+6.4)	71.4 (+6.4)	87.6 (+10.9)	92.4 (+5.4)	98.7 (-0.4)	94.3 (+7.7)	75.9 (+2.9)	86.8 (-2.9)	**86.8** (+4.6)

Table 9: Multi-task performance of merged ViT-B/16 models on eight tasks.

Method	with CUSD	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Avg.
Fine-tuned	—	86.5	82.1	90.2	99.2	99.8	99.7	80.6	97.9	92.0
Task Arithmetic	×	59.0	51.4	64.9	76.3	98.5	71.3	54.3	89.2	70.6
	✓	75.6 (+16.5)	62.1 (+10.7)	85.7 (+20.8)	87.1 (+10.9)	97.9 (-0.5)	89.2 (+17.9)	71.6 (+17.3)	84.7 (-4.5)	**81.7** (+11.1)
TIES-Merging	×	60.7	52.3	68.5	77.1	98.8	73.4	56.6	90.2	72.2
	✓	74.7 (+14.0)	65.2 (+12.9)	86.8 (+18.3)	90.3 (+13.3)	98.4 (-0.4)	90.2 (+16.8)	70.7 (+14.1)	88.4 (-1.9)	**83.1** (+10.9)
DARE	×	59.2	51.3	64.8	75.6	98.4	71.3	54.2	89.2	70.5
	✓	75.3 (+16.1)	62.1 (+10.8)	85.6 (+20.8)	87.1 (+11.4)	97.8 (-0.6)	89.2 (+18.0)	71.6 (+17.4)	84.9 (-4.3)	**81.7** (+11.2)
Consensus	×	60.8	52.1	67.1	77.1	98.6	72.9	56.2	89.5	71.8
	✓	75.7 (+15.0)	62.1 (+9.9)	85.7 (+18.5)	86.7 (+9.6)	97.9 (-0.7)	89.5 (+16.6)	71.7 (+15.5)	84.4 (-5.1)	**81.7** (+9.9)

Table 10: Multi-task performance of merged ViT-B/32 models on eight tasks.

Method	with CUSD	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Avg.
Fine-tuned	—	78.0	79.4	88.9	98.7	99.7	99.7	78.3	97.5	90.0
Task Arithmetic	×	46.4	48.5	64.2	71.4	97.2	64.1	46.2	79.3	64.7
	✓	65.1 (+18.8)	60.4 (+12.0)	81.9 (+17.7)	79.8 (+8.5)	96.5 (-0.7)	83.4 (+19.3)	65.5 (+19.4)	79.3 (0.0)	**76.5** (+11.9)
TIES-Merging	×	48.8	49.7	66.9	72.8	97.7	66.7	49.0	81.3	66.6
	✓	65.6 (+16.8)	63.7 (+14.0)	83.8 (+16.9)	83.7 (+10.9)	96.8 (-0.9)	87.1 (+20.4)	66.1 (+17.1)	79.8 (-1.5)	**78.3** (+11.7)
DARE	×	45.9	48.6	64.2	71.4	97.2	63.7	45.9	79.1	64.5
	✓	64.6 (+18.7)	60.3 (+11.7)	81.8 (+17.6)	79.9 (+8.5)	96.5 (-0.7)	83.2 (+19.5)	65.5 (+19.6)	79.1 (-0.1)	**76.4** (+11.9)
Consensus	×	48.2	49.6	65.7	72.2	97.3	65.7	48.4	80.0	65.9
	✓	65.3 (+17.1)	60.5 (+10.9)	82.0 (+16.3)	79.6 (+7.4)	96.4 (-0.9)	83.5 (+17.8)	65.8 (+17.4)	79.2 (-0.8)	**76.5** (+10.7)

Results on NLU tasks. Tables 11 and 12 provide task-level GLUE results for RoBERTa-Base and RoBERTa-Large. Each table reports the plain merge and the corresponding CUSD merge for all four downstream arithmetic operators. CUSD also improves every merging operator on both RoBERTa-Base and RoBERTa-Large. The effect is especially strong for operators whose plain merges suffer from severe degradation, such as TIES-Merging on RoBERTa-Large, where the normalized average increases from 47.3 to 69.8. A striking example is STS-B, the semantic textual similarity task: all plain merges yield negative normalized scores, whereas CUSD turns them positive across every operator in both RoBERTa-Base and RoBERTa-Large. These results indicate that common–unique separation is not limited to vision encoders and can also recover lost performance in discriminative language models. Detailed GLUE results are reported in Appendix Tables 12 and 11.

Table 11: Multi-task performance of merged RoBERTa-Base models on the 8-task GLUE benchmark.

Method	with CUSD	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg.
Fine-tuned	—	55.1	87.5	88.0	92.4	89.4	67.5	93.8	88.3	82.8
Task Arithmetic	×	0.0	64.6	79.5	75.5	71.3	49.1	71.1	-56.8	44.3
	✓	6.6 (+6.6)	63.6 (-1.0)	78.2 (-1.3)	74.1 (-1.4)	37.6 (-33.7)	50.9 (+1.8)	87.7 (+16.6)	55.9 (+112.6)	**56.8** (+12.5)
TIES-Merging	×	0.0	64.9	79.0	73.2	76.4	46.2	65.9	-67.2	42.3
	✓	6.6 (+6.6)	62.1 (-2.8)	77.9 (-1.1)	63.6 (-9.6)	36.6 (-39.8)	53.4 (+7.2)	90.3 (+24.3)	68.4 (+135.6)	**57.4** (+15.1)
DARE	×	4.6	64.6	79.4	75.6	71.7	48.7	71.3	-58.6	44.7
	✓	6.6 (+1.9)	63.0 (-1.6)	78.1 (-1.3)	73.6 (-2.1)	37.4 (-34.3)	50.9 (+2.2)	87.7 (+16.4)	54.8 (+113.3)	**56.5** (+11.8)
Consensus	×	4.6	66.9	78.0	78.6	71.8	46.6	75.5	-56.3	45.7
	✓	6.6 (+1.9)	59.4 (-7.5)	77.3 (-0.7)	66.6 (-12.0)	33.3 (-38.5)	53.1 (+6.5)	87.7 (+12.3)	61.6 (+117.9)	**55.7** (+10.0)

Table 12: Multi-task performance of merged RoBERTa-Large models on the 8-task GLUE benchmark.

Method	with CUSD	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg.
Fine-tuned	—	59.1	90.5	89.0	94.5	90.5	70.0	96.6	91.7	85.2
Task Arithmetic	×	0.2	75.7	48.6	69.5	82.1	58.8	85.1	-22.4	49.7
	✓	9.1 (+8.8)	72.2 (-3.4)	27.9 (-20.8)	83.5 (+14.0)	59.5 (-22.7)	70.8 (+11.9)	92.7 (+7.6)	62.0 (+84.3)	**59.7** (+10.0)
TIES-Merging	×	-1.4	71.0	56.3	53.3	84.6	54.5	72.1	-67.9	40.3
	✓	5.7 (+7.1)	74.5 (+3.5)	29.1 (-27.2)	85.3 (+32.0)	58.6 (-26.0)	70.0 (+15.5)	92.6 (+20.4)	60.3 (+128.2)	**59.5** (+19.2)
DARE	×	2.4	75.6	48.8	70.2	82.1	59.2	85.7	-23.2	50.1
	✓	9.6 (+7.2)	71.7 (-3.9)	26.9 (-21.9)	82.9 (+12.7)	59.1 (-23.0)	69.7 (+10.5)	92.4 (+6.8)	61.3 (+84.5)	**59.2** (+9.1)
Consensus	×	2.4	76.3	40.1	76.8	80.3	57.8	89.1	-0.9	52.7
	✓	9.9 (+7.5)	71.7 (-4.6)	27.9 (-12.2)	83.4 (+6.6)	59.9 (-20.4)	70.8 (+13.0)	92.7 (+3.6)	61.1 (+62.1)	**59.7** (+6.9)

Results on generative LLMs. Tables 13–18 provide task-level results for the six LLMs: Llama-3.2-3B, Llama-3.1-8B, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, Gemma-2-2B, and Gemma-2-9B. Scores are normalized by the corresponding fine-tuned task-specific models before averaging. On generative LLMs, CUSD improves most settings, although the margins are more model-dependent than on encoder models. The strongest gains appear on Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, and the Gemma models. Llama-3.1-8B is the least responsive case: plain arithmetic merges already reach normalized averages around 94, leaving little headroom, and the only negative change is a small 0.3-point decrease for DARE. In the evaluated LLM settings, partial unique-residual scaling is often beneficial; this behavior is examined in the β ablation (Section 4.3) and the analysis of unique directions across models (Section 4.4). Full task-level LLM results are provided in the Appendix Tables 13–18.

Table 13: Performance of merged Llama-3.2-3B models. The task-specific checkpoints are fine-tuned for coding, math, and multilingual tasks; normalized scores and their average are reported.

Method	with CUSD	Humaneval+	MBPP+	GSM8k	M-MMLU	M-ARC	M-HellaSwag	Avg.
Fine-tuned	—	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Task Arithmetic	×	45.7	59.4	39.2	87.0	93.4	93.0	**69.6**
	✓	45.7 (0.0)	52.8 (-6.6)	45.8 (+6.6)	86.0 (-1.0)	94.0 (+0.6)	93.2 (+0.2)	**69.6** (0.0)
TIES-Merging	×	46.1	58.5	37.9	87.2	93.1	92.6	69.2
	✓	45.2 (-0.9)	53.3 (-5.2)	47.1 (+9.2)	85.8 (-1.4)	94.0 (+0.9)	93.0 (+0.4)	**69.7** (+0.5)
DARE	×	47.3	53.8	38.8	87.2	93.3	93.0	68.9
	✓	47.9 (+0.6)	57.1 (+3.3)	42.4 (+3.6)	86.8 (-0.4)	94.2 (+0.9)	93.2 (+0.2)	**70.3** (+1.4)
Consensus	×	47.9	59.9	24.4	87.8	91.5	92.8	67.4
	✓	52.1 (+4.2)	51.9 (-8.0)	47.1 (+22.7)	85.7 (-2.1)	93.9 (+2.4)	93.1 (+0.3)	**70.6** (+3.2)

Table 14: Performance of merged Llama-3.1-8B models. The task-specific checkpoints are fine-tuned for coding, math, and multilingual tasks; normalized scores and their average are reported.

Method	with CUSD	Humaneval+	MBPP+	GSM8k	M-MMLU	M-ARC	M-HellaSwag	Avg.
Fine-tuned	—	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Task Arithmetic	×	82.5	100.0	93.7	95.5	89.7	100.8	93.7
	✓	82.6 (+0.1)	102.4 (+2.4)	91.3 (-2.4)	96.1 (+0.6)	90.9 (+1.2)	101.3 (+0.5)	**94.1** (+0.4)
TIES-Merging	×	82.6	100.8	94.2	95.2	88.8	100.4	93.7
	✓	83.0 (+0.4)	104.0 (+3.2)	88.2 (-6.0)	97.1 (+1.9)	91.2 (+2.4)	101.2 (+0.8)	**94.1** (+0.4)
DARE	×	84.6	101.2	92.7	95.1	90.3	100.7	**94.1**
	✓	83.2 (-1.4)	100.4 (-0.8)	91.2 (-1.5)	96.0 (+0.9)	91.0 (+0.7)	101.2 (+0.5)	93.8 (-0.3)
Consensus	×	83.8	99.2	89.5	97.1	92.3	101.1	93.8
	✓	83.4 (-0.4)	100.8 (+1.6)	91.9 (+2.4)	96.2 (-0.9)	90.4 (-1.9)	100.9 (-0.2)	**93.9** (+0.1)

Table 15: Performance of merged Llama-3.2-3B-Instruct models. The task-specific checkpoints are fine-tuned for coding, math, and multilingual tasks; normalized scores and their average are reported.

Method	with CUSD	Humaneval+	MBPP+	GSM8k	M-MMLU	M-ARC	M-HellaSwag	Avg.
Fine-tuned	—	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Task Arithmetic	×	55.5	46.5	64.2	85.5	96.2	93.3	73.5
	✓	61.3 (+5.8)	54.0 (+7.5)	62.4 (-1.8)	86.4 (+0.9)	97.2 (+1.0)	93.6 (+0.3)	**75.8** (+2.3)
TIES-Merging	×	53.8	44.3	63.9	85.1	95.9	93.4	72.7
	✓	58.0 (+4.2)	57.5 (+13.2)	61.1 (-2.8)	86.4 (+1.3)	96.4 (+0.5)	93.6 (+0.2)	**75.5** (+2.8)
DARE	×	51.7	50.0	64.6	85.0	95.9	93.3	73.4
	✓	57.6 (+5.9)	56.6 (+6.6)	62.4 (-2.2)	85.9 (+0.9)	96.3 (+0.4)	93.4 (+0.1)	**75.4** (+2.0)
Consensus	×	57.8	59.3	63.8	86.3	96.5	93.8	76.2
	✓	60.2 (+2.4)	60.2 (+0.9)	62.5 (-1.3)	86.5 (+0.2)	96.4 (-0.1)	93.7 (-0.1)	**76.6** (+0.4)

Table 16: Performance of merged Llama-3.1-8B-Instruct models. The task-specific checkpoints are fine-tuned for coding, math, and multilingual tasks; normalized scores and their average are reported.

Method	with CUSD	Humaneval+	MBPP+	GSM8k	M-MMLU	M-ARC	M-HellaSwag	Avg.
Fine-tuned	—	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Task Arithmetic	×	55.5	87.9	87.6	84.2	87.3	92.4	82.5
	✓	70.7 (+15.2)	91.4 (+3.5)	83.8 (-3.8)	88.3 (+4.1)	90.7 (+3.4)	94.0 (+1.6)	**86.5** (+4.0)
TIES-Merging	×	56.9	87.9	87.8	84.5	86.7	92.1	82.6
	✓	68.8 (+11.9)	92.6 (+4.7)	85.2 (-2.6)	87.3 (+2.8)	89.6 (+2.9)	93.8 (+1.7)	**86.2** (+3.6)
DARE	×	59.7	87.5	87.9	83.8	87.3	92.3	83.1
	✓	71.7 (+12.0)	91.8 (+4.3)	84.9 (-3.0)	86.8 (+3.0)	89.5 (+2.2)	93.7 (+1.4)	**86.4** (+3.3)
Consensus	×	68.6	93.4	85.5	86.6	89.2	93.7	86.2
	✓	73.7 (+5.1)	94.9 (+1.5)	82.7 (-2.8)	88.7 (+2.1)	91.2 (+2.0)	94.0 (+0.3)	**87.5** (+1.3)

Table 17: Performance of merged Gemma-2-2b models. The task-specific checkpoints are fine-tuned for coding, math, and multilingual tasks; normalized scores and their average are reported.

Method	with CUSD	Humaneval+	MBPP+	GSM8k	M-MMLU	M-ARC	M-HellaSwag	Avg.
Fine-tuned	—	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Task Arithmetic	×	80.6	94.9	75.0	106.8	104.0	104.0	94.2
	✓	83.7 (+3.1)	98.0 (+3.1)	76.4 (+1.4)	107.4 (+0.6)	104.5 (+0.5)	104.1 (+0.1)	**95.7** (+1.5)
TIES-Merging	×	80.4	90.3	79.5	107.1	103.3	103.8	94.1
	✓	90.8 (+10.4)	94.4 (+4.1)	83.2 (+3.7)	107.6 (+0.5)	104.0 (+0.7)	103.9 (+0.1)	**97.3** (+3.2)
DARE	×	80.6	91.8	76.7	107.6	104.0	104.2	94.1
	✓	87.3 (+6.7)	94.9 (+3.1)	78.6 (+1.9)	108.1 (+0.5)	104.1 (+0.1)	104.1 (-0.1)	**96.2** (+2.1)
Consensus	×	60.5	94.9	68.8	103.2	102.7	102.8	88.8
	✓	77.5 (+17.0)	90.3 (-4.6)	75.8 (+7.0)	107.2 (+4.0)	104.3 (+1.6)	103.7 (+0.9)	**93.1** (+4.3)

Table 18: Performance of merged Gemma-2-9b models. The task-specific checkpoints are fine-tuned for coding, math, and multilingual tasks;normalized scores and their average are reported.

Method	with CUSD	Humaneval+	MBPP+	GSM8k	M-MMLU	M-ARC	M-HellaSwag	Avg.
Fine-tuned	—	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Task Arithmetic	×	73.5	102.8	105.0	96.3	83.4	90.7	92.0
	✓	72.4 (-1.1)	106.8 (+4.0)	101.3 (-3.7)	99.5 (+3.2)	90.7 (+7.3)	98.1 (+7.4)	**94.8** (+2.8)
TIES-Merging	×	71.5	102.0	105.0	96.2	81.8	90.3	91.1
	✓	71.5 (0.0)	109.2 (+7.2)	99.6 (-5.4)	100.3 (+4.1)	91.5 (+9.7)	99.6 (+9.3)	**95.3** (+4.2)
DARE	×	72.8	99.6	105.5	96.0	82.9	90.3	91.2
	✓	72.1 (-0.7)	106.8 (+7.2)	101.7 (-3.8)	99.5 (+3.5)	90.7 (+7.8)	98.2 (+7.9)	**94.8** (+3.6)
Consensus	×	80.7	105.2	100.2	99.3	88.6	93.7	94.6
	✓	74.4 (-6.3)	107.6 (+2.4)	100.6 (+0.4)	100.3 (+1.0)	91.7 (+3.1)	98.5 (+4.8)	**95.5** (+0.9)

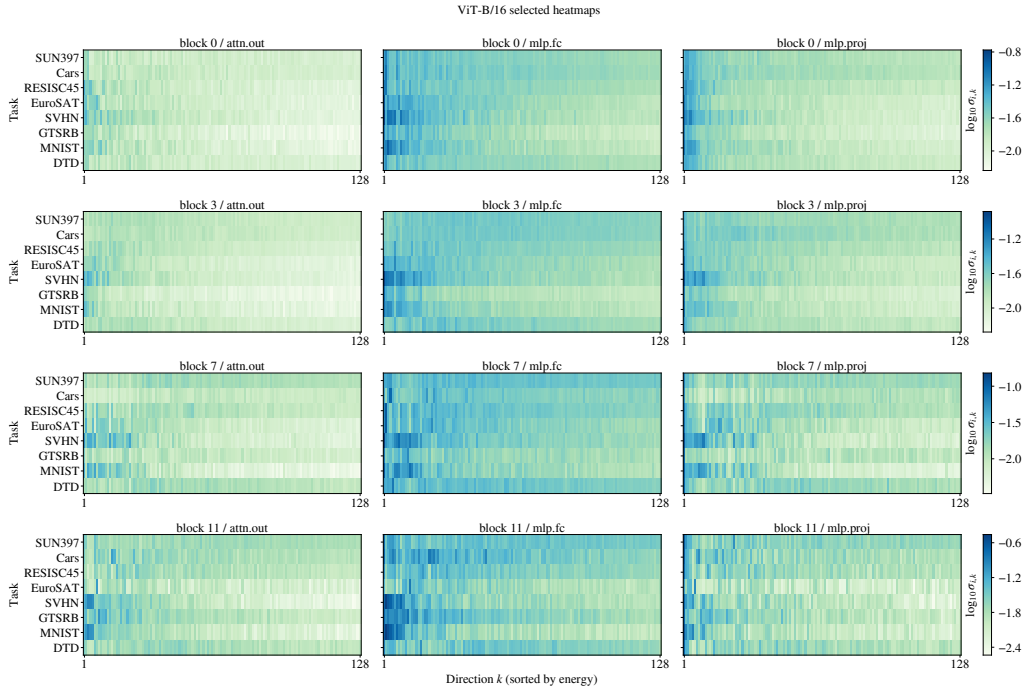


Figure 11: **ViT-B/16 directional singular-value heatmaps.** Each panel shows $\log_{10} \sigma_{i,k}$ for one layer component, with directions sorted by energy.

A.7 Dispersion Scoring Analysis

Figure 11 and Figure 12 provide additional layer heatmaps for the representative ViT and Llama settings used in the dispersion-scoring analysis.

A.8 Additional Hyperparameter Sensitivity Analysis

Figure 13 reports β sensitivity separately for each downstream merging method. This complements the averaged sweep in Figure 3 and checks whether the observed trend is consistent across Task Arithmetic, TIES-Merging, DARE, and Consensus Merging.

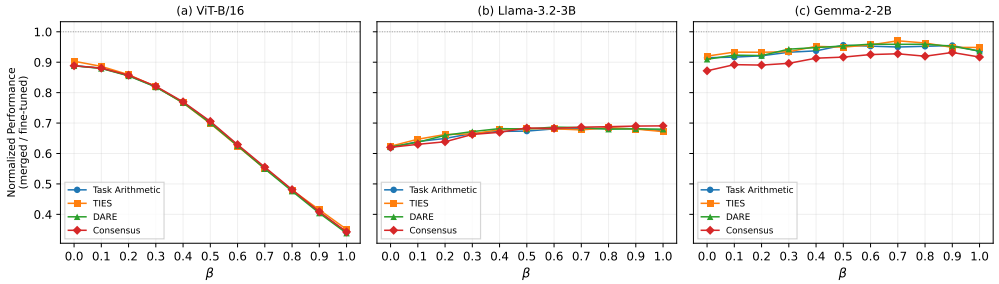


Figure 13: Per-method sensitivity to β across downstream arithmetic-based merging operators.

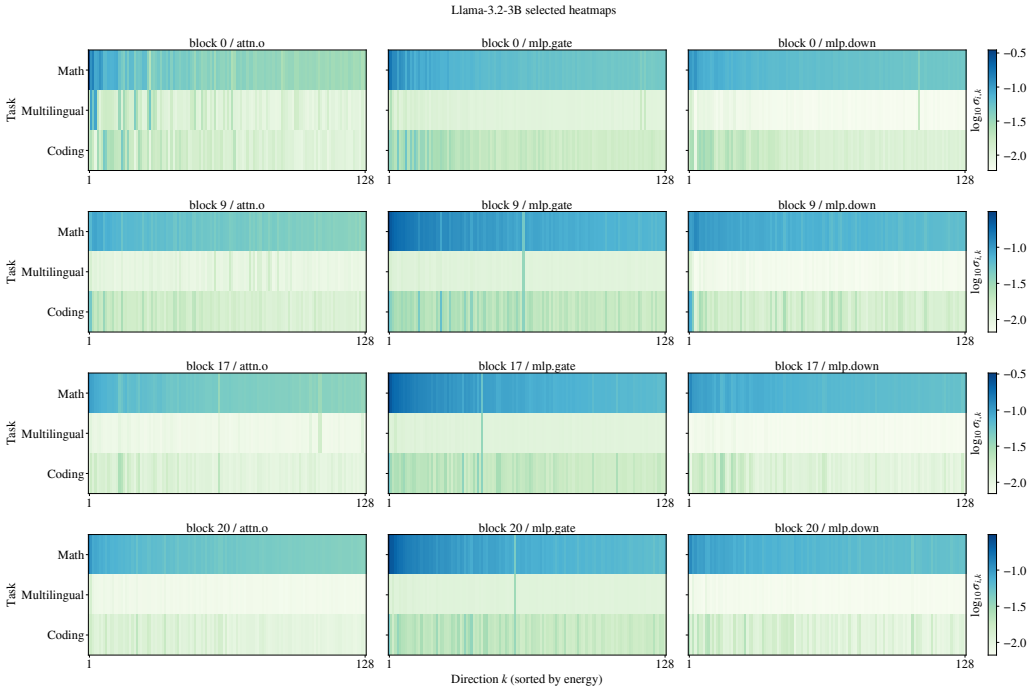


Figure 12: **Llama-3.2-3B directional singular-value heatmaps.** Each panel shows $\log_{10} \sigma_{i,k}$ for one layer component, with directions sorted by energy.

Table 19: Joint sensitivity to the Task Arithmetic scale λ and unique-residual coefficient β on ViT-B/32. We report average accuracy (%) from a post-hoc sweep on the test sets. The final column reports plain TA at the same λ; bold entries outperform this baseline. † denotes the hyperparameter configuration used in the original submission.

$\lambda \backslash \beta$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	Plain TA
0.1	57.43	61.74	**63.33**	**62.22**	59.15	53.95	47.22	39.87	33.62	28.21	23.91	61.74
0.2	66.62	69.07	69.36	67.35	62.75	56.56	49.11	41.37	34.97	29.31	24.91	69.36
0.3	**72.49**	**73.87**	**72.94**	70.05	64.59	57.76	49.88	42.29	35.91	30.36	25.85	70.05
0.4	**76.48**†	**76.50**	**74.65**	**70.48**	64.65	57.43	49.84	42.52	36.31	31.18	26.94	64.65†
0.5	**77.64**	**77.02**	**74.15**	**69.70**	**63.45**	56.27	49.03	42.36	36.58	31.94	27.80	56.27
0.6	**77.33**	**75.82**	**72.47**	**67.53**	**61.25**	**54.54**	47.97	41.74	36.49	32.35	28.43	47.97
0.7	**75.53**	**73.54**	**69.82**	**64.63**	**58.90**	**52.56**	**46.45**	40.91	36.36	32.47	28.86	40.91
0.8	**72.48**	**70.22**	**66.47**	**61.61**	**56.11**	**50.44**	**45.04**	**40.05**	36.13	32.33	28.99	36.13
0.9	**68.78**	**66.37**	**62.71**	**58.20**	**53.25**	**48.21**	**43.42**	**39.15**	**35.57**	32.20	29.06	32.20
1.0	**64.46**	**62.11**	**58.83**	**54.94**	**50.45**	**46.04**	**41.90**	**38.28**	**34.92**	**31.87**	28.98	28.98

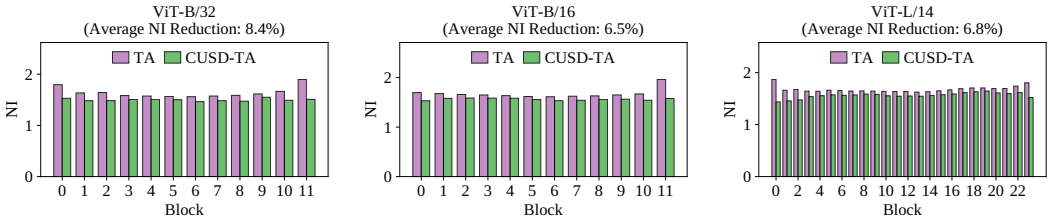
A.9 Additional Per-Block Interference Analysis

For a merged task vector $\tau_m^{(\ell)}$ at layer ℓ and task-specific task vectors $\{\tau_i^{(\ell)}\}_{i=1}^K$, we measure a normalized interference proxy (NI) as

$$\mathrm{NI} = \frac{1}{K} \sum_{i=1}^{K} \frac{\|\tau_m - \tau_i\|_F^2}{\|\tau_i\|_F^2}. \tag{14}$$

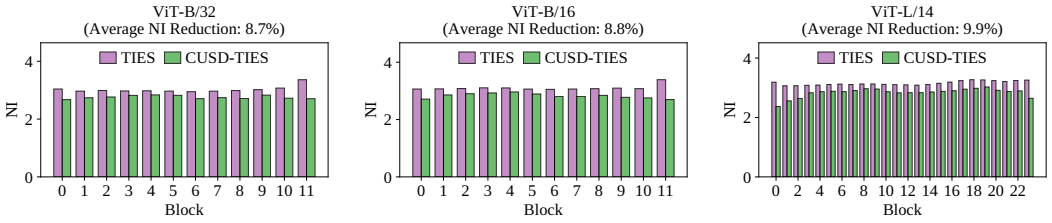
For the per-block plots, we aggregate the decomposed matrix-structured layers in each block. We use this normalized distance as a parameter-space proxy for interference: a larger value means that the

merged update deviates more from the task-specific updates. Figure 14, Figure 15, and Figure 16 provide the per-block interference analysis for the remaining merging methods on ViT models.



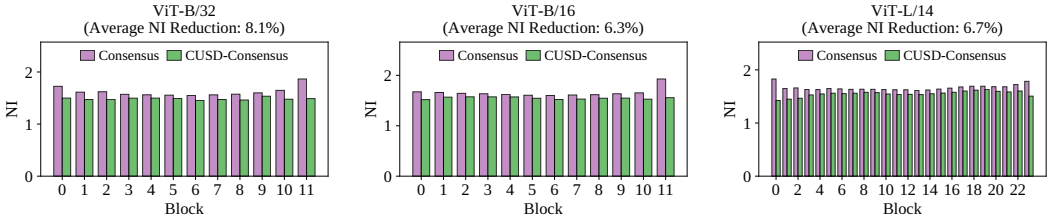
(a) ViT-B/32. (b) ViT-B/16. (c) ViT-L/14.

Figure 14: Per-block interference analysis for Task Arithmetic on ViT models.



(a) ViT-B/32. (b) ViT-B/16. (c) ViT-L/14.

Figure 15: Per-block interference analysis for TIES-Merging on ViT models.



(a) ViT-B/32. (b) ViT-B/16. (c) ViT-L/14.

Figure 16: Per-block interference analysis for Consensus Merging on ViT models.

A.10 Limitations

We acknowledge the potential limitations of our work. Performance gains vary across methods and models, and SVD-based baselines remain stronger in some evaluated settings. CUSD uses only model weights during decomposition and merging, but selecting β can require validation data. CUSD is evaluated in the standard homogeneous-merging setting, where all task-specific checkpoints are fine-tuned from the same base architecture. It does not address heterogeneous merging across different architectures. Our evaluation also focuses on dense models at the ViT, RoBERTa, Llama, and Gemma scales considered in this work; substantially larger checkpoints, such as 70B-class LLMs, and Mixture-of-Experts architectures are left for future work. Future work could also explore multiobjective optimization for balancing common aggregation and unique retention [46, 47]. We further plan to investigate merging compatible models for RNA design [48–51].

A.11 Broader Impacts

Positive Societal Impacts By improving data-free arithmetic model merging, CUSD has the potential for several positive societal impacts. Because CUSD uses only model weights during

decomposition and merging, it can support privacy-sensitive and federated settings where checkpoints are produced by different owners and the underlying data cannot be shared, e.g., when fine-tuned models from hospitals, financial institutions, or independent research groups must be combined into a single multi-task model without disclosing proprietary or personal data. By more reliably merging task-specific checkpoints into one multi-task model, CUSD reduces the need to deploy and serve a separate fine-tuned model per task, which lowers storage, memory, inference compute, and energy consumption, and is particularly beneficial in resource-constrained environments such as edge devices and small organizations. Because CUSD operates as a pre-merging module that augments existing arithmetic operators (Task Arithmetic, TIES, DARE, Consensus) rather than replacing them, practitioners can adopt it without retraining or redesigning their pipelines, which broadens access to high-quality multi-task model assembly across the ViT, RoBERTa, Llama, and Gemma families considered in this work and similar architectures. Together, these properties can accelerate research and deployment of capable multi-task models in domains such as scientific discovery, medical decision support, multilingual NLP, and educational tools, and can make advanced AI capabilities available to a broader range of users and institutions.

Negative Societal Impacts Despite these benefits, the advances presented also carry potential negative societal impacts. Improved data-free merging makes it easier to combine the capabilities of independently distributed checkpoints, which could be leveraged for malicious purposes, such as merging a safety-aligned model with a checkpoint fine-tuned for harmful or unsafe behavior in order to circumvent alignment, or assembling specialized capabilities for generating disinformation, sophisticated phishing content, or deceptive media at scale. Because CUSD merges models without inspecting their training data, biases, unsafe behaviors, or memorized content present in any constituent checkpoint can be inherited and aggregated in the merged model, potentially leading to unfair or discriminatory outcomes when these models are deployed in decision-making pipelines. The reduced cost of obtaining strong multi-task models from publicly available checkpoints might also lower the barrier to repurposing released models in ways that their original developers did not intend, and may contribute to the concentration of capabilities in the hands of actors with the largest pools of high-quality fine-tuned weights. Mitigation strategies—such as releasing merging tools alongside responsible-use guidelines, evaluating merged models for safety, fairness, and bias before deployment, developing detectors for content produced by merged models, and applying safety filters or alignment procedures after merging—are important to address these risks.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The assertions made in the abstract and introduction are fully supported by our methodology and empirical findings.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Please refer to Appendix A.10 to see our related discussion.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: Please refer to Appendix A.4 to see our theoretical proof.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We conduct detailed descriptions of our experimental setup in Section 4. The source code is publicly available at `https://github.com/stwang666/CUSD`

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The source code is publicly available at `https://github.com/stwang666/CUSD`

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Please refer to Section 4 for detailed empirical settings.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper does not report error bars, but experiments were conducted across multiple model architectures and sizes to validate the robustness of the method.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We report the time consumption of our proposed method and computed resources used in the experimental setup section.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conducted in this paper fully conforms to the NeurIPS Code of Ethics in every respect.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss both the potential positive and negative societal impacts of our work in Appendix A.11.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The methods primarily discussed in our paper do not involve data or models with a high risk of misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have appropriately cited the papers relevant to our work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The source code is publicly available at `https://github.com/stwang666/CUSD`

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core methodology of this paper does not involve LLMs as an important, original, or non-standard component. LLMs from the Llama and Gemma families appear only as experimental subjects on which the proposed merging method is evaluated.

    Guidelines:

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