Approaching Effective Merging in Model Embedding Space

Wei Chen, Zhenyuan Dong, Qiang Qiu
Purdue University, IN, USA
{chen2732, dong485, qqiu}@purdue.edu

Abstract

With the growing number of fine-tuned variants derived from pre-trained models, model merging emerges as a key challenge: effectively integrating task-specific adaptations while preserving performance. However, naive weight averaging often fails when parameter updates are misaligned, revealing the need for a structured space in which merging becomes principled. In this work, we hypothesize that pre-trained models possess an intrinsic low-rank structure. Specifically, we posit that the transformation from input to output is mediated by a latent bottleneck of low intrinsic dimensionality; we leverage this low-dimensional space to construct a *model embedding space*. This space ensures that fine-tuned updates, *i.e.*, *model embeddings*, of all adaptations remain confined within the intrinsic structure of the pre-trained model. As model embeddings lie in the same low-dimensional space, *model merging* via direct averaging becomes more effective. Moreover, we further introduce an aligned subspace of these model embeddings to improve the efficacy of model merging. Experiments across vision and multimodal tasks, including ViT and multimodal LLMs, demonstrate that our approach enables effective model merging and provides a unified geometric view of adaptation and combination in pre-trained models. The source code is in our GitHub repo: Model-Embedding-Space-Merging.

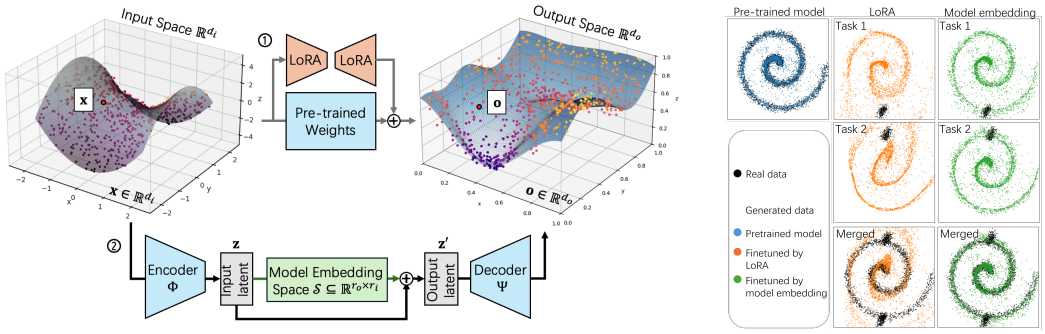
1 Introduction

Large pre-trained models have become the foundation of modern vision, language, and multimodal systems [14, 35, 63, 72]. With the rise of parameter-efficient fine-tuning methods [28, 44, 61], thousands of task-specific variants can now be derived from a single pre-trained checkpoint. This proliferation of fine-tuned models raises a central challenge: *how to fuse multiple adaptations effectively*, *i.e.*, how to integrate their specialized knowledge while preserving the pre-trained model's capabilities without compromising other adaptations [15, 51, 62, 71, 81].

Naive parameter averaging, although simple, often fails because fine-tuned updates are *misaligned*: conventional methods such as LoRA [28] directly modify the weights, but the resulting updates typically lie in task-specific subspaces that differ across fine-tuning tasks [31, 57, 69, 77], as illustrated in Figure 2 (a). Thus, a core difficulty of model merging lies in the absence of a shared space in which these updates can be meaningfully fused.

We address this challenge by analyzing the pre-trained model through its intrinsic structure [19, 30, 38] and constructing a **model embedding space**. Specifically, we interpret the pre-trained layer as an autoencoder-style map: there exist encoding and decoding bases that project the input and output features onto their respective intrinsic low-dimensional subspaces. Fine-tuning within this space ensures that adaptations remain confined to the pre-trained manifold. The fine-tuned updates are referred to as the **model embeddings**.

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



(a) (b)

Figure 1: (a) The pre-trained weights map features from the input space $\mathbb{R}^{d_i}$ to the output space $\mathbb{R}^{d_o}$. (a-1) The most common fine-tuning approach, *i.e.*, LoRA [28], is applied to the *weight space* $\Delta\mathbf{W} \in \mathbb{R}^{d_o \times d_i}$. (a-2) In our framework, we reinterpret the pre-trained model through an autoencoder perspective, where it transforms inputs to outputs via a pathway of low intrinsic dimension $r_i \ll d_i$. Both the encoder Φ and decoder Ψ are derived *solely* from the pre-trained weights. We construct a *model embedding space* $\mathcal{S} \subseteq \mathbb{R}^{r_o \times r_i}$ in which adaptation to downstream tasks is performed only within this space. The detailed case design can be found in Section 2.5. (b) In $\mathcal{S}$, the model adaptation and merging are more effective. While LoRA adapts a low-rank subspace of the weights to downstream data, it risks distorting the original pre-trained distribution. In contrast, our method facilitates updates while preserving the integrity of the pre-trained structure. Consequently, our merged model generates samples that align more closely with real data, whereas the LoRA model tends to exhibit distribution shifts that deviate from the pre-trained knowledge.

This formulation naturally benefits **model merging**. When multiple fine-tuned models are updated in the low-dimensional model embedding space, constructing a merged model by simply averaging the model embeddings becomes more effective, as illustrated in Figure 2 (b). Moreover, to further enhance alignment among updates, we propose to construct an aligned subspace of them before constructing the merged update, which leads to improved model merging performance.

Empirically, we show that model embedding space enables effective model merging. Across vision and multimodal tasks, including vision transformers and multimodal large language models, our approach demonstrates strong adaptability to new tasks and significantly enhances merging stability.

In summary, this work reframes the problem of model merging through the lens of the pre-trained model:

- We construct a **model embedding space** that establishes a shared low-rank basis derived from the pre-trained model.
- Within this space, model merging is more effective, as it preserves the intrinsic structure of the pre-trained representations.
- We further propose fusing model embeddings within an **aligned subspace**, a technique that significantly enhances the efficacy of model merging.

2 Methods

2.1 Overview

Setting. Consider a pre-trained layer $f(\mathbf{x}; \mathbf{W})$ that maps an input feature $\mathbf{x} \in \mathbb{R}^{d_\text{i}}$ to an output feature $\mathbf{o} \in \mathbb{R}^{d_\text{o}}$. We assume the input distribution is supported on a low–intrinsic-dimension manifold $\mathcal{M}_\text{in} \subset \mathbb{R}^{d_\text{i}}$ with $\dim(\mathcal{M}_\text{in}) = r_\text{i} \ll d_\text{i}$; the output distribution is supported on $\mathcal{M}_\text{out} \subset \mathbb{R}^{d_\text{o}}$ with $\dim(\mathcal{M}_\text{out}) = r_\text{o} \ll d_\text{o}$. Figure 1 (a) illustrates this manifold of the input space and output space.

Low intrinsic dimension. To capture the low-intrinsic structure, the pre-trained layer $f(\mathbf{x}; \mathbf{W})$ can be interpreted as an *autoencoder-style* map, with encoding and decoding bases

$$\Phi \in \mathbb{R}^{r_\text{i} \times d_\text{i}}, \qquad \Psi \in \mathbb{R}^{r_\text{o} \times d_\text{o}},$$

which project the input and output features onto their intrinsic low-dimensional subspaces.

Remark 2.1. In this work, we construct Φ and Ψ directly from the pre-trained weights, ensuring zero reconstruction error. Because these weights have been optimized on large-scale data, they encode statistically meaningful structure, allowing the resulting bases to capture the intrinsic structure learned during pre-training.

2.2 Model Embedding Space

Full fine-tuning or unconstrained low-rank updates on $\mathbf{W}$ can rotate channel subspaces and drift off the manifolds $\mathcal{M}_{\text{in}} \to \mathcal{M}_{\text{out}}$. In this work, we introduce a *model embedding space* and constrain downstream fine-tuning to this space. Specifically, the *model embedding space* is the matrix space

$$\mathcal{S} \subseteq \mathbb{R}^{r_{\text{o}} \times r_{\text{i}}}.$$

In this space, the *model embedding* $\Delta\mathbf{S}$ represents the updated parameters when adapting the pre-trained model to the downstream task. It acts as an operator that characterizes the interaction between the r_{i} input latent and the r_{o} output latent during the adaptation. In our method, r_i is set equal to r_o. Given the input latent $\mathbf{z} \in \mathbb{R}^{r_i}$, the output latent $\mathbf{z}' \in \mathbb{R}^{r_o}$ is

$$\mathbf{z}' = \mathbf{z} + \Delta\mathbf{S}\,\mathbf{z}. \tag{1}$$

This process is depicted in Figure 1 (a). Compared to LoRA updates, model embedding operates in a low-dimensional space rather than a low-rank weight space, as $r_i \ll d_i$. The detailed case design can be found in Section 2.5.

2.3 Merging in Model Embedding Space

Given M updates $\{\Delta\mathbf{S}_i\}_{i=1}^{M}$ in the model embedding space, the goal of model merging is to find a fused model embedding $\Delta\hat{\mathbf{S}}$.

Arithmetic average. A simple way to obtain $\Delta\hat{\mathbf{S}}$ is through an arithmetic averaging of all updates, *i.e.*,

$$\Delta\hat{\mathbf{S}} = \frac{1}{M}\sum_{i=1}^{M}\Delta\mathbf{S}_i. \tag{2}$$

Although all updates lie in the same model embedding space, different tasks may still emphasize distinct sub-directions. To address this issue, we propose aligning model embeddings in the subspace before merging.

Aligned subspace merging. Instead of directly averaging $\{\Delta\mathbf{S}_i\}_{i=1}^{M}$, we propose aligned subspace merging,

$$\Delta\hat{\mathbf{S}} = g\left(\{\Delta\mathbf{S}_i\}_{i=1}^{M}\right). \tag{3}$$

We first align the subspace of model embeddings by extracting a common subspace basis that captures information shared across tasks, and then perform merging within this subspace. Table 10 shows the effectiveness of aligned subspace merging on a MNIST masked-image reconstruction toy task.

(1) Aligned subspace. For each model embedding $\Delta\mathbf{S}_i$, we obtain the basis of its row space using an SVD

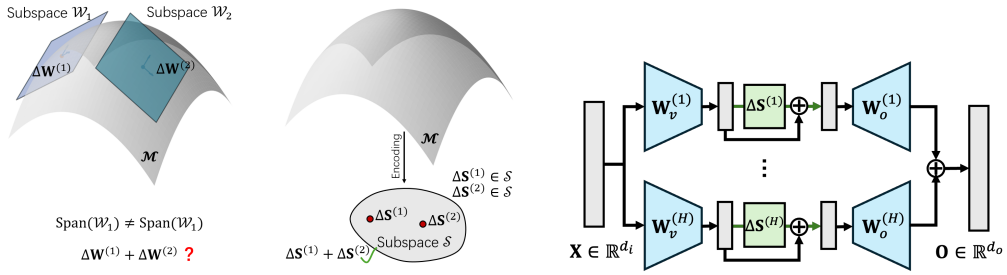
$$\Delta\mathbf{S}_i = U_i\Sigma_i V_i^{\top}, \tag{4}$$

where $V_i \in \mathbb{R}^{r_i \times r_m}$ is the right singular vector basis ($r_m \leq \min(r_i, r_o)$). We then stack these bases and perform a second SVD:

$$\tilde{V} = \begin{bmatrix} V_1^{\top} \\ \vdots \\ V_M^{\top} \end{bmatrix} \in \mathbb{R}^{(Mr_m)\times r_i}, \qquad \tilde{V} = U^{'}\Sigma^{'}V^{'\top}, \tag{5}$$

where $U^{'}$ and $V^{'}$ are singular vectors of stacked basis $\tilde{V}$, and $\Sigma^{'}$ are its singular values. We take the leading r right singular vectors as the aligned subspace basis

$$\mathbf{B} = V^{'}(:, 1{:}r) \in \mathbb{R}^{r_i \times r}. \tag{6}$$



(a) (b) (c)

Figure 2: (a) Conventional fine-tuning methods such as LoRA [28] directly operate in the model *weight space*, but the resulting updates $\Delta\mathbf{W}$ typically reside in task-specific subspaces that differ across fine-tuning tasks. When the updates are not aligned, *i.e.*, they span different subspaces, simply fusing or averaging these weights becomes ill-posed and may degrade performance. (b) In contrast, our method first encodes the weight space into a low-dimensional representation, which we define as *model embedding space* $\mathcal{S}$, and constrains adaptation to occur only within $\mathcal{S}$. This ensures that all updates $\Delta\mathbf{S}$ lie in a common representational space, making direct merging of multiple adaptations meaningful. (c) Illustration of tuning model embedding space in multi-head attention (MHA). Model embeddings are denoted as $\Delta\mathbf{S}^{(1)}, \cdots, \Delta\mathbf{S}^{(H)}$.

By Eckart-Young's theorem [17], $\mathbf{B}$ is the best rank-r approximation of $\tilde{V}$.

(2) Merging in the aligned subspace. With $\mathbf{B}$ fixed, each model embedding is represented in a shared subspace with its coefficients:

$$C_i = \Delta\mathbf{S}_i\mathbf{B} \in \mathbb{R}^{r_o \times r}. \tag{7}$$

In this subspace, the merged coefficients $\bar{C}$ are obtained by minimizing the squared Frobenius distance to $\{C_i\}_{i=1}^M$:

$$\bar{C} = \arg\min_{C \in \mathbb{R}^{r_o \times r}} \sum_{i=1}^{M} \|C - C_i\|_F^2, \tag{8}$$

which has the closed-form solution

$$\bar{C} = \frac{1}{M}\sum_{i=1}^{M} C_i. \tag{9}$$

It leads to the merged update in the model embedding space:

$$\Delta\hat{\mathbf{S}} = \bar{C}\mathbf{B}^\top. \tag{10}$$

2.4 Extend Aligned Subspace Merging to Weight Space

The aligned subspace merging strategy can also be applied in the weight space; for example, LoRA weight updates can be aligned in a shared basis before merging. Given M LoRA updates $\{\mathbf{W}_i^A\}_{i=1}^M$ and $\{\mathbf{W}_i^B\}_{i=1}^M$, we can separately obtain their aligned subspace merging via

$$\begin{aligned} \hat{\mathbf{W}}^A &= g\left(\{\mathbf{W}_i^A\}_{i=1}^M\right), \\ \hat{\mathbf{W}}^B &= g\left(\{\mathbf{W}_i^B\}_{i=1}^M\right), \end{aligned} \tag{11}$$

where $g(\cdot)$ denotes the merging function defined in Eq. (3). This alignment ensures that different LoRA updates are expressed in a shared subspace.

2.5 Case Study: Multi-head Attention

We discuss the model embedding space using architecture-specific examples, such as Multi-Head Attention (MHA). The complete formulations for Multi-layer Perceptron (MLP) are deferred to the Appendix A.

Setting. The multi-head attention layer [72] transforms the input sequence $\mathbf{X} \in \mathbb{R}^{N\times d}$ to the output sequence $\mathbf{O} \in \mathbb{R}^{N\times d}$, where N denotes the sequence length, d is the dimension of input and output features. It allows multiple attention mechanisms to work in parallel, with each head independently learning attention patterns. For H attention heads, each attention head calculates attention maps as $\mathbf{A}^{(h)} = \texttt{softmax}(\mathbf{X}\,\mathbf{W}_q^{(h)}\mathbf{W}_k^{(h)\top}\mathbf{X}^\top)$, where projection matrices $\mathbf{W}_q^{(h)}, \mathbf{W}_k^{(h)} \in \mathbb{R}^{d\times d_h}$, $h = 1, \cdots, H$, $d_h = \frac{d}{H}$. The multi-head attention represents as,

$$\texttt{MHA}(\mathbf{X}) = \sum_{h=1}^{H} \mathbf{A}^{(h)}\mathbf{X}\mathbf{W}_v^{(h)}\mathbf{W}_o^{(h)\top} \tag{12}$$

where projection matrices $\mathbf{W}_v^{(h)}, \mathbf{W}_o^{(h)} \in \mathbb{R}^{d\times d_h}$. MHA naturally exhibits an *encoding/decoding structure*, with a low intrinsic dimension given by d_h.

Encoding process. Attention and per-head outputs:

$$\mathbf{Z}^{(h)} = \mathbf{A}^{(h)}\mathbf{X}\mathbf{W}_v^{(h)} \in \mathbb{R}^{N\times d_h}. \tag{13}$$

Each attention head applies a projection from d to d_h, so it naturally serves as an *encoder* that maps $\mathbf{X}$ into a lower-dimensional representation.

Decoding process. Concatenate heads and apply the fixed output projection:

$$\mathbf{O} = \sum_{h=1}^{H} \mathbf{Z}^{(h)}\mathbf{W}_o^{(h)\top}, \tag{14}$$

where $\mathbf{W}_o^{(h)} \in \mathbb{R}^{d\times d_h}$ serves as a *decoder* that maps $\mathbf{Z}^{(h)}$ back to a higher-dimensional representation.

Model embedding. Given *fixed* pre-trained weights $\mathbf{W}_q^{(h)}$, $\mathbf{W}_k^{(h)}$, $\mathbf{W}_v^{(h)}$, $\mathbf{W}_o^{(h)}$, for each head h, we insert a learnable model embedding $\Delta\mathbf{S}^{(h)} \in \mathbb{R}^{d_h\times d_h}$ to adapt to the downstream task:

$$\begin{aligned} \mathbf{Z}'^{(h)} &= \mathbf{Z}^{(h)} + \mathbf{Z}^{(h)}\,\Delta\mathbf{S}^{(h)}, \\ \widetilde{\mathbf{O}}^{(h)} &= \mathbf{Z}'^{(h)}\,\mathbf{W}_o^{(h)\top} \in \mathbb{R}^{n\times d}. \end{aligned} \tag{15}$$

Initialize each model embedding to zero: $\Delta\mathbf{S}^{(h)} = \mathbf{0}_{d_h}\ \forall h$, and it causes *zero* deviation from the pre-trained model at initialization. This structure is illustrated in Figure 2 (c).

3 Discussion

In this section, we discuss the comparison between tuning and merging in the model embedding space and in the weight space, *i.e.*, LoRA.

3.1 The Effect of Aligned Subspace

Let $\boldsymbol{\tau}_i = \mathrm{vec}(\Delta\mathbf{W}_i) \in \mathbb{R}^D$ denote the update of model i and let $\mathbf{T} = [\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_M] \in \mathbb{R}^{D\times M}$. Choose a rank-$r$ aligned subspace basis $U \in \mathbb{R}^{D\times r}$, which is obtained by the top-r SVD of $\mathbf{T}$, and its projector $\mathbf{P} = UU^\top$. The *arithmetic averaging* gives $\bar{\boldsymbol{\tau}} = \frac{1}{M}\sum_i \boldsymbol{\tau}_i$, while aligned merging gives $\hat{\boldsymbol{\tau}} = \mathbf{P}\,\bar{\boldsymbol{\tau}}$.

Proposition 3.1. *$\hat{\boldsymbol{\tau}}$ is the* unique minimizer *of projection of $\bar{\boldsymbol{\tau}}$ onto the subspace spanned by U:*

$$\hat{\boldsymbol{\tau}} = \arg\min_{\boldsymbol{\tau}' \in \{Uz : z\in\mathbb{R}^r\}} \|\bar{\boldsymbol{\tau}} - \boldsymbol{\tau}'\|_2.$$

Proposition 3.2. *The alignment error between $\bar{\boldsymbol{\tau}}$ and $\hat{\boldsymbol{\tau}}$ is*

$$\|\bar{\boldsymbol{\tau}} - \hat{\boldsymbol{\tau}}\|_2 \;\le\; \frac{1}{M}\sum_{i=1}^{M} \|\boldsymbol{\tau}_i\|_2 \,\sin\theta_i,$$

where θ_i is the largest principal angle between $\mathrm{span}(U)$ *and* $\mathrm{span}(\boldsymbol{\tau}_i)$.

Table 1: **SVHN reconstruction MSE.**

	6	7	8	9	Avg. ↓
Pre-trained	18.58	18.56	18.86	19.28	18.82
Fine-tuned LoRA	13.50	12.43	14.06	13.85	13.46
Fine-tuned Ours	12.54	11.04	12.94	12.76	12.32
LoRA Avg.	17.61	17.44	17.83	18.28	17.79
LoRA Aligned	14.19	13.69	14.23	14.58	14.17
Ours Avg.	**13.72**	**13.16**	**13.52**	**13.86**	**13.56**

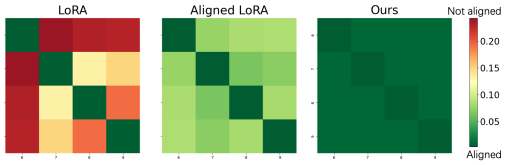


Figure 3: **Aligned subspace enables more effective model merging.** Across three methods, we compare the principal angles between subspaces over four SVHN subtasks, where smaller angles indicate stronger subspace alignment.

Proposition 3.3. *Let the downstream effect of an update be $\mathcal{F}(\boldsymbol{\tau})$, e.g., output or reconstruction map. If $\mathcal{F}$ is linear with operator norm $\|\mathcal{A}\|_2$, i.e., $\mathcal{F}(\boldsymbol{\tau}) = \mathcal{A}\boldsymbol{\tau}$, the excess MSE from unaligned vs. aligned merging satisfies*

$$\|\mathcal{F}(\bar{\boldsymbol{\tau}}) - \mathcal{F}(\hat{\boldsymbol{\tau}})\|_2 \;\le\; \frac{\|\mathcal{A}\|_2}{M} \sum_{i=1}^{M} \|\boldsymbol{\tau}_i\|_2 \, \sin\theta_i.$$

Remark 3.4. The error of the arithmetic averaging is bounded by the alignment of subspace. If the subspace is more aligned, *i.e.*, θ_i is smaller, the arithmetic average leads to better performance. The model merging is more effective. The details of the analysis are in Appendix B.

3.2 Demonstration

We evaluate the effectiveness of our approach, particularly the validity of the common space hypothesis, in a controlled setting based on the SVHN dataset [56].

Experimental setting. Our base model is a fully connected autoencoder trained on digits 0–5 from the SVHN dataset. We then fine-tune its model embedding separately on digits 6–9 to obtain adaptation variants. We compare our adaptation method with LoRA [28]. For each method, we fuse the resulting experts using arithmetic average. In addition, for LoRA, we also evaluate an aligned variant using (11). The details are provided in Appendix E.2.

Metrics. Image reconstruction is evaluated using mean-squared error (MSE) on the SVHN test sets for digits 6–9. To further analyze subspace alignment, we compute the principal angles between the update subspaces of different digits. Specifically, given two subspaces $\mathcal{U}, \mathcal{V} \subset \mathbb{R}^n$ with orthonormal bases $U \in \mathbb{R}^{n\times p}$ and $V \in \mathbb{R}^{n\times q}$, the *principal angles* $\{\theta_i\}_{i=1}^{k}$ $(k = \min(p,q))$ are defined recursively as

$$\cos\theta_i = \max_{\mathbf{u}_i\in\mathcal{U},\, \mathbf{v}_i\in\mathcal{V}} \mathbf{u}_i^\top \mathbf{v}_i,$$
$$\text{s.t. } \|\mathbf{u}_i\| = \|\mathbf{v}_i\| = 1,\; \mathbf{u}_i^\top \mathbf{u}_j = \mathbf{v}_i^\top \mathbf{v}_j = 0,\; j < i.$$

Smaller principal angles indicate greater overlap and thus a more shared representation space.

Capability of model embedding adaptation. Table 1 reports the MSE of the reconstructed images. Compared with LoRA, fine-tuning only the model embeddings $\Delta\mathbf{S}$ achieves consistently lower MSE across all four digits, despite using substantially fewer parameters (16k vs. 84k).

Aligned subspace helps model merging. Figure 3 shows the principal angles between updates on different digits. Compared with LoRA, our method yields the smallest principal angles as well as the lowest MSE. It demonstrates that our task subspaces are well aligned within the shared model embedding space, making it naturally suited for simple model merging. When subspace alignment is applied to LoRA, the principal angles decrease substantially, and the reconstruction loss is also reduced, confirming that subspace alignment improves both merging effectiveness.

4 Experiments

In this section, we present empirical results on both vision and multimodal settings, including vision transformers (ViT) and multimodal large language models (MLLM).

Table 2: Performance of ViT-B/32 on eight vision tasks. We compare model adaptation and merging in both the weight space and the proposed model embedding space $\mathcal{S}$. (i) For model fine-tuning, adapting model embeddings $\Delta\mathbf{S}$ achieves higher accuracy than tuning LoRA. (ii) For model merging, our aligned subspace merging methods in $\mathcal{S}$ achieve the best result among all approaches. The aligned subspace merging is also effective in weight space using LoRA.

Space	Method	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC	SUN397	SVHN	**Average**
-	Pre-trained model	59.74	44.15	46.63	32.8	48.3	61.57	63.35	31.8	48.42
-	Fine-tuned model (LoRA)	74.0	58.33	99.0	92.7	99.3	88.4	64.5	96.2	84.1
-	Fine-tuned model (Ours, $\Delta\mathbf{S}$)	75.89	55.63	99.0	93.69	98.5	89.9	68.1	96.24	84.61
Weight space (LoRA)	Arithmetic average	60.61	43.0	48.4	39.06	52.25	63.1	62.09	39.22	50.97
	Task Arithmetic [31]	60.66	43.0	48.48	39.16	52.75	63.21	62.86	39.68	51.23
	Ties-merging [80]	60.95	42.42	49.41	33.95	56.76	61.33	62.51	42.87	51.28
	DARE [82]	60.78	43.0	48.86	35.02	56.23	62.0	62.79	40.52	51.15
	TSV [20]	61.75	44.07	50.08	41.74	58.89	64.82	62.78	47.36	53.94
	ISO-C [51]	59.32	48.43	50.93	69.3	70.22	70.62	63.64	48.29	60.09
	KnOTS [69]	59.8	48.43	57.67	68.91	67.19	70.45	63.56	47.97	60.5
	Core [57]	61.41	**49.65**	50.44	**78.1**	70.64	**74.6**	**63.89**	51.5	62.53
	Ours (Aligned subspace)	59.83	44.68	41.93	47.55	54.44	65.3	63.31	42.95	52.5
$\mathcal{S}$	**Ours (Arithmetic average)**	63.0	39.41	55.22	38.44	55.36	64.44	60.51	41.36	53.48
	Ours (Aligned subspace)	**73.19**	42.55	**60.82**	61.16	**79.4**	69.7	63.73	**67.99**	**64.82**

4.1 Merging in Vision Tasks

We first evaluate our method on vision model merging using ViT-B/32 [14] across eight downstream image classification tasks. We report the absolute accuracy, which is different from the normalized accuracy used in some of the prior papers [57, 69]. Implementation details are provided in Appendix E.3. Additional experimental results for ViT-B/32, ViT-B/16, and ViT-L/14 on vision benchmarks comprising 14 and 20 datasets are presented in Appendix F.4.

Model embedding enables effective task adaptation. In Table 2, "Fine-tuned model" represents the test accuracy of models fine-tuned separately on each downstream task. It shows that fine-tuning in the model embedding space consistently achieves higher classification accuracy than fine-tuning in the weight space using LoRA, while requiring substantially fewer trainable parameters. Specifically, LoRA fine-tuning updates 1.18M parameters, whereas our method updates only 0.58M parameters, reducing the parameter count by nearly 50%.

Model embedding space for effective merging. Table 3 demonstrates the effectiveness of the proposed model embedding space for model merging. In particular, we adapt representative weight-space merging methods to the model embedding space by merging task-specific model embeddings instead of LoRA weight updates. Applying Task Arithmetic, TSV, and ISO-C in the model embedding space yields approximately a 2% absolute accuracy improvement compared to their weight-space counterparts.

Table 3: The accuracy of different merging methods improves when applied in the model embedding space $\mathcal{S}$ rather than in the weight space.

	Task Arithmetic	TSV	ISO-C
Weight space (LoRA)	51.23	53.94	60.09
Model embedding space	**53.48**	**56.10**	**62.37**

Aligned subspace enhances model merging. In Table 2, applying aligned subspace in the weight space when merging LoRA checkpoints also leads to improved performance over these methods. When aligned subspace merging is applied directly in the model embedding space, our approach achieves the highest overall accuracy among all merged models, highlighting the effectiveness of structured alignment in the proposed embedding space.

4.2 Merging in MLLMs

We further evaluate our method on MLLM merging using MM-MergeBench [83] and the LLaVA [42] codebase. Implementation details are provided in Appendix E.4.

Seen tasks evaluation of MLLM merging. Table 4 shows that our model embedding space is more merge-friendly than the standard LoRA space. Even arithmetic averaging in this space yields a strong merged model, suggesting better-aligned task updates with less cancellation. Aligned subspace merging further improves both spaces, and achieves the best result when applied in our embedding

Table 4: Performance of MM-Merge-Bench on eight seen and four unseen tasks.

		SEEN TASKS									UNSEEN TASKS				
Space	Method	SciQA	Image	VQA	REC	OCR	VizWiz	Flickr	IconQA	**Average**	AVQA	Image-R	S2W	TabMWP	**Average**
-	Pre-trained model	61.73	40.87	62.88	36.10	41.16	41.03	49.07	14.09	43.37	51.62	28.27	5.98	15.01	25.22
-	Fine-tuned models (LoRA)	83.74	96.02	67.58	43.40	65.50	64.80	57.29	75.54	69.23	-	-	-	-	-
-	Fine-tuned models (Ours, $\Delta\mathbf{S}$)	85.50	95.05	68.08	48.17	66.09	59.72	56.67	77.09	69.55	-	-	-	-	-
Weight space (LoRA)	Arithmetic average	70.71	56.30	**67.87**	37.88	65.40	42.37	44.20	24.97	51.21	71.79	42.64	8.06	17.59	35.02
	Task Arithmetic [31]	71.94	57.49	67.06	38.90	62.87	44.80	49.20	39.21	53.93	74.78	37.37	7.52	13.57	33.31
	DARE [82]	71.59	57.25	66.26	39.38	62.56	44.93	49.13	39.59	53.84	73.75	37.67	7.56	13.62	33.15
	Ties-merging [80]	71.49	55.88	66.73	39.67	65.12	44.35	47.06	34.46	53.09	73.43	38.44	7.47	13.23	33.14
	PCB-merging [22]	71.10	57.82	67.59	38.22	64.35	44.58	48.90	37.01	53.70	74.57	36.28	7.84	15.44	33.53
	WUDI-Merging [12]	75.17	62.73	63.31	40.65	61.20	45.66	**53.96**	40.86	55.44	78.95	43.56	9.64	17.59	37.44
	RobustMerge [83]	73.43	65.54	67.20	44.80	62.97	**46.61**	52.80	**45.90**	57.33	**79.30**	45.79	9.23	17.62	37.99
	Ours (Aligned subspace)	72.74	61.39	67.49	43.27	63.81	45.38	49.73	37.66	55.18	77.82	44.81	8.46	18.06	37.29
$\mathcal{S}$	**Ours (Arithmetic average)**	73.50	60.97	67.71	40.65	**65.61**	46.16	48.74	23.76	53.39	76.07	45.21	8.41	17.63	36.83
	Ours (Aligned subspace)	**82.76**	**74.85**	66.56	**46.67**	62.47	45.43	52.13	39.16	**58.75**	78.52	**50.63**	**10.55**	**19.20**	**39.73**

space, increasing the average score from 57.33 [83] to **58.75**. The gains are most pronounced on SciQA and Image, with about **10%** improvement compared to the previous methods.

Unseen tasks evaluation of MLLM merging. A similar pattern holds on the four unseen benchmarks, where the advantage of the proposed model embedding space is more evident. Although publicly available LoRA checkpoints can be directly merged, weight-space LoRA merging shows limited generalization to unseen tasks. In contrast, even arithmetic averaging in our model embedding space provides a stronger unseen-task baseline, suggesting better preservation of transferable knowledge and reduced cross-task interference under distribution shifts. With aligned subspace merging, the LoRA-space result becomes comparable to previous best methods, while our embedding-space alignment further achieves the best average score of **39.73**. Overall, model embedding space merging offers stronger unseen-task generalization than directly merging LoRA checkpoints.

4.3 Ablation Study

We empirically choose the aligned-subspace rank r under the constraint $r < r_i$, ensuring that the merged embedding lies in a projected subspace. Table 5 reports the ViT merging accuracy as a function of the ratio r/r_i. More details are attached in Appendix F.2.

The ViT results show that performance improves as r/r_i increases from small values and becomes stable once the aligned subspace is sufficiently expressive. This indicates that the method does not require a narrowly tuned rank. In our ViT experiments, we use a high-rank setting with $r/r_i = 98\%$.

4.4 Advantages of Model Embedding Space over LoRA in Model Merging

We summarize the advantages of the proposed model embedding space over standard LoRA. As shown in Figure 1(a), LoRA updates the original weight space, where task-specific updates may remain misaligned. Our method instead constrains adaptation to the pre-trained-derived model embedding space $\mathcal{S}$, producing more structured and merge-friendly updates.

- **Knowledge preservation.** Figure 1(b) shows that LoRA may distort the pre-trained distribution, while our method better preserves the original structure after adaptation and merging. Table 6 further demonstrates adaptation and knowledge-preservation capabilities of our method. The model embedding space achieves *better adaptation* than LoRA, yielding higher target-task accuracy. It also exhibits *minimal forgetting*, as indicated by its near-preservation of the original ImageNet performance. The experimental details for Table 6 are provided in Appendix E.5.
- **Parameter-efficient adaptation.** Table 2 shows that model embedding fine-tuning improves average ViT accuracy over LoRA while using only 0.58M trainable parameters, compared to 1.18M for LoRA, reducing the parameter count by nearly 50%.
- **Merge-friendly and generalizable updates.** By constraining task updates within the same model embedding space, our method makes them more comparable and easier to align. Figure 3 shows

Table 5: **Sensitivity to r/r_i on ViT-B/32.** We vary the ratio r/r_i and report the merged accuracy.

r/r_i	12.5%	25%	37.5%	50%	62.5%	75%	87.5%	98%
Accuracy	52.45	57.43	60.35	62.99	64.74	65.43	65.24	64.82

Table 6: Knowledge preservation and target-task adaptation across different merging settings.

	ImageNet top1	ImageNet top5	Target Task Acc
Pre-trained model	60.77	85.56	48.42
Weight Space (Full parameter)	53.47 (-7.3)	79.89 (-5.67)	86.3
Weight Space (LoRA)	58.82 (-1.95)	84.59 (-0.97)	60.09
Model Embedding Space	**60.62 (-0.15)**	**85.6 (+0.04)**	64.82

smaller principal angles than LoRA, and Table 3 shows improved merging performance in $\mathcal{S}$. Table 4 further shows stronger unseen MLLM task generalization than directly merging LoRA checkpoints, indicating better transferability under distribution shifts.

Overall, the model embedding space provides a more structured and preservation-friendly alternative to LoRA, enabling compact adaptation, better knowledge retention, and stronger merging generalization.

5 Related Work

5.1 PEFT and Representation Tuning

Training all weights of large foundation models is often impractical, so PEFT freezes the backbone and learns small task-specific components, such as adapters, prefixes, or low-rank updates [27, 28, 40]. Recent variants improve decomposition, allocation, and robustness [6, 7, 9, 43, 44, 52, 53, 86], and extend PEFT to vision–language models through adapterized bridges and visual prompting [32, 70]. Representation tuning instead directly edits hidden activations while keeping the backbone fixed, offering parameter-efficient and controllable adaptation [1, 45, 79, 78].

5.2 Model Merging

Model merging combines multiple specialized checkpoints into a single model without additional multi-task training. Existing methods typically merge checkpoints offline through weight arithmetic, such as checkpoint averaging and task vectors [31, 77], and improve robustness using tie-breaking, rescaling, training-free recipes, or spectral projections to reduce interference [20, 22, 29, 68, 80, 82, 83]. Other approaches make adapters more merge-friendly by learning layer-wise, subspace, or task-wise coefficients [15, 51, 62, 71, 81]. In contrast, our method constructs a merge-friendly model embedding space and further introduces an aligned subspace for data-free model merging. Additional related work on MLLM adaptation and merging is provided in Appendix D.

6 Conclusion

In summary, our work introduces a unified framework for model merging through a shared low-rank embedding space, *i.e.*, *model embedding space*, derived from the pre-trained model's intrinsic structure. By decomposing the model into fixed encoder and decoder bases, we constrain all downstream adaptations to evolve within the same low-dimensional subspace, ensuring consistency across tasks. Within this space, arithmetic average is more effective, as it preserves the intrinsic structure of the pre-trained representations. Moreover, applying an aligned subspace to among updates further enhances the performance of the merged model. Experiments demonstrate that our approach not only improves fusion stability and performance but also provides a coherent interpretation of adaptation and combination among pre-trained models. The limitations are discussed in Appendix G.

Border Impact

Our work enables effective merging of task-specific adaptations, offering a promising direction for efficient knowledge integration and reuse in pretrained models. By representing adaptations within a shared low-dimensional space, our approach has the potential to reduce the computational and storage overhead associated with maintaining multiple specialized models, contributing to more resource-efficient AI systems. Beyond the image classification, natural language processing, and multimodal tasks evaluated in this work, the proposed framework could be extended to broader applications, including concept composition in image generation and reasoning. Furthermore, the ability to integrate task-specific knowledge through model merging may facilitate continual learning and federated learning, where knowledge accumulation across tasks or distributed clients is essential.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *PAMI*, 35(8):1798–1828, 2013.

[2] Chi Chen, Yiyang Du, Zheng Fang, Ziyue Wang, Fuwen Luo, Peng Li, Ming Yan, Ji Zhang, Fei Huang, Maosong Sun, et al. Model composition for multimodal large language models. In *ACL*, 2024.

[3] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *NeurIPS*, 2024.

[4] Wei Chen, Kartikeya Bhardwaj, and Radu Marculescu. Fedmax: Mitigating activation divergence for accurate and communication-efficient federated learning. In *ECML-PKDD*, 2020.

[5] Wei Chen, Huidong Liu, Yang Liu, Chien-Chih Wang, Moyan Li, Hongdong Li, and Bryan Wang. Zero-shot customized video editing with diffusion feature transfer. In *ICCVW*, 2025.

[6] Wei Chen, Zichen Miao, and Qiang Qiu. Parameter-efficient tuning of large convolutional models. *arXiv preprint arXiv:2403.00269*, 2024.

[7] Wei Chen, Zichen Miao, and Qiang Qiu. Large convolutional model tuning via filter subspace. In *ICLR*, 2025.

[8] Wei Chen and Qiang Qiu. Extra clients at no extra cost: Overcome data heterogeneity in federated learning with filter decomposition. *arXiv preprint arXiv:2503.08652*, 2025.

[9] Wei Chen, Jingxi Yu, Zichen Miao, and Qiang Qiu. Sparse fine-tuning of transformers for generative tasks. In *ICCV*, 2025.

[10] Wei Chen, Jingxi Yu, Zichen Miao, and Qiang Qiu. In-context compositional learning via sparse coding transformer. *NeurIPS*, 2026.

[11] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

[12] 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.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[15] Yiyang Du, Xiaochen Wang, Chi Chen, Jiabo Ye, Yiru Wang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, Zhifang Sui, et al. AdaMMS: Model merging for heterogeneous multimodal large language models with unsupervised coefficient optimization. In *CVPR*, 2025.

[16] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. VLMEvalKit: An open-source toolkit for evaluating large multi-modality models. In *MM*, 2024.

[17] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1936.

[18] An Evgeniou and Massimiliano Pontil. Multi-task feature learning. *NeurIPS*, 2007.

[19] Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael Jordan, and Zheng-Jun Zha. Rank diminishing in deep neural networks. In *NeurIPS*, 2022.

[20] Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. In *CVPR*, 2025.

[21] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.

[22] DU Guodong, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim Kuan Goh, Ho-Kin Tang, Daojing He, et al. Parameter competition balancing for model merging. In *NeurIPS*, 2024.

[23] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *CVPR*, 2018.

[24] Ligong Han, Yinxiao Li, Han Zhang, Peyman Milanfar, Dimitris Metaxas, and Feng Yang. Svdiff: Compact parameter space for diffusion fine-tuning. In *ICCV*, 2023.

[25] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021.

[26] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[27] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019.

[28] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[29] Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. EMR-Merging: Tuning-free high-performance model merging. In *NeurIPS*, 2024.

[30] Yerlan Idelbayev and Miguel A Carreira-Perpinán. Low-rank compression of neural nets: Learning the rank of each layer. In *CVPR*, 2020.

[31] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023.

[32] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022.

[33] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014.

[34] Taewook Kim, Wei Chen, and Qiang Qiu. Learning to customize text-to-image diffusion in diverse context. *arXiv preprint arXiv:2410.10058*, 2024.

[35] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023.

[36] Jose Lezama, Wei Chen, and Qiang Qiu. Run-sort-rerun: Escaping batch size limitations in sliced wasserstein generative models. In *ICML*, 2021.

[37] Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench: Benchmarking multimodal large language models. In *CVPR*, 2024.

[38] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *ICLR*, 2018.

[39] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[40] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

[41] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *CVPR*, 2024.

[42] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[43] Jun Liu, Zhenglun Kong, Peiyan Dong, Xuan Shen, Pu Zhao, Hao Tang, Geng Yuan, Wei Niu, Wenbin Zhang, Xue Lin, et al. Rora: Efficient fine-tuning of llm with reliability optimization for rank adaptation. In *ICASSP*, 2025.

[44] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *ICML*, 2024.

[45] Yiyang Liu, James Chenhao Liang, Ruixiang Tang, Yugyung Lee, Majid Rabbani, Sohail Dianat, Raghuveer Rao, Lifu Huang, Dongfang Liu, Qifan Wang, et al. Re-imagining multimodal instruction tuning: A representation view. *arXiv preprint arXiv:2503.00723*, 2025.

[46] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. MMBench: Is your multi-modal model an all-around player? In *ECCV*, 2024.

[47] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*, 2022.

[48] Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *ICLR*, 2023.

[49] Pan Lu, Liang Qiu, Jiaqi Chen, Tony Xia, Yizhou Zhao, Wei Zhang, Zhou Yu, Xiaodan Liang, and Song-Chun Zhu. Iconqa: A new benchmark for abstract diagram understanding and visual language reasoning. In *NeurIPS Track on Datasets and Benchmarks*, 2021.

[50] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016.

[51] Daniel Marczak, Simone Magistri, Sebastian Cygert, Bartłomiej Twardowski, Andrew D Bagdanov, and Joost van de Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. In *ICML*, 2025.

[52] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *NeurIPS*, 2024.

[53] Zichen Miao, Wei Chen, and Qiang Qiu. Coeff-tuning: A graph filter subspace view for tuning attention-based large models. In *CVPR*, 2025.

[54] Zichen Miao, Ze Wang, Wei Chen, and Qiang Qiu. Continual learning with filter atom swapping. In *ICLR*, 2021.

[55] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *ICDAR*, 2019.

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

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

[58] Bryan A. Plummer, Liwei Wang, Christopher M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *IJCV*, 123(1):74–93, 2017.

[59] Vishal Purohit, Wei Chen, and Qiang Qiu. Blockwise divide-and-aggregate for image restoration using diffusion priors. In *CVPR*, 2026.

[60] Qiang Qiu, Xiuyuan Cheng, Guillermo Sapiro, et al. Dcfnet: Deep neural network with decomposed convolutional filters. In *ICML*, 2018.

[61] Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. In *NeurIPS*, 2023.

[62] Huaizhi Qu, Xinyu Zhao, Jie Peng, Kwonjoon Lee, Behzad Dariush, and Tianlong Chen. UQ-Merge: Uncertainty guided multimodal large language model merging. In *ACL*, 2025.

[63] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[64] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge. In *ECCV*, 2022.

[65] Atul Sharma, Wei Chen, Joshua Zhao, Qiang Qiu, Saurabh Bagchi, and Somali Chaterji. Flair: Defense against model poisoning attack in federated learning. In *ACM Asia*, 2023.

[66] Atul Sharma, Wei Chen, Joshua Zhao, Qiang Qiu, Somali Chaterji, and Saurabh Bagchi. Tesseract: Gradient flip score to secure federated learning against model poisoning attacks. *arXiv preprint arXiv:2110.10108*, 2021.

[67] Atul Sharma, Joshua C Zhao, Wei Chen, Qiang Qiu, Saurabh Bagchi, and Somali Chaterji. How to learn collaboratively-federated learning to peer-to-peer learning and what's at stake. In *DSN-S*, 2023.

[68] George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *ICLR*, 2024.

[69] George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. In *ICLR*, 2025.

[70] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *CVPR*, 2022.

[71] Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. Parameter-efficient multi-task model fusion with partial linearization. In *ICLR*, 2024.

[72] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[73] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, 2021.

[74] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[75] Ze Wang, Zichen Miao, Jun Hu, and Qiang Qiu. Adaptive convolutions with per-pixel dynamic filter atom. In *ICCV*, 2021.

[76] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022.

[77] 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 *ICML*, 2022.

[78] Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Jianhao Zhu, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. Advancing parameter efficiency in fine-tuning via representation editing. In *ACL*, 2024.

[79] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. In *NeurIPS*, 2024.

[80] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In *NeurIPS*, 2023.

[81] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *ICLR*, 2024.

[82] 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 *ICML*, 2024.

[83] Fanhu Zeng, Haiyang Guo, Fei Zhu, Li Shen, and Hao Tang. Robustmerge: Parameter-efficient model merging for mllms with direction robustness. In *NeurIPS*, 2025.

[84] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

[85] Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, et al. Lmms-eval: Reality check on the evaluation of large multimodal models. *arXiv preprint arXiv:2407.12772*, 2024.

[86] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*, 2023.

A Examples of the Model Embedding Space Through Layer Decomposition

A.1 Motivation

Modern pretrained models operate on data and features with inherently low intrinsic dimensionality. To study model behavior directly within this intrinsic space while preserving the knowledge in the pretrained model, we construct an encoder/decoder derived from the pretrained weights. This factorization provides a structure in which input features are projected into a shared low-dimensional latent space and decoded back. Within this low-rank space, the mapping from input latent features to output latent features defines a *model embedding*, providing a pretrained-aligned subspace in which downstream updates and fusions can be meaningfully performed.

A.2 MLP / Linear Layers

Let the pretrained linear map be $\mathbf{W} \in \mathbb{R}^{d_o \times d_i}$ with compact SVD

$$\mathbf{W} \;=\; U\,\Sigma\,V^\top,$$

where $U \in \mathbb{R}^{d_o \times r},\ \Sigma = \mathrm{diag}(\sigma_1 \geq \cdots \geq \sigma_r) > 0,\ V \in \mathbb{R}^{d_i \times r}$, $r = \mathrm{rank}(W)$. Fix an integer $H \geq 1$. Define a partition of $\{1, \ldots, r\}$ into H contiguous blocks with approximately equal cardinality. Let $p \triangleq \lfloor r/H \rfloor$ and $q \triangleq r - Hp$, which is the remainder. For $h = 1, \ldots, H$, define block sizes

$$r_h \;=\; \begin{cases} p+1, & h \leq q, \\ p, & h > q. \end{cases}$$

The h-th block corresponds to indices $\mathcal{I}_h = \{\sum_{j=1}^{h-1} r_j + 1, \ldots, \sum_{j=1}^{h} r_j\}$. Extract per-block SVD factors

$$\begin{aligned} U^{(h)} \;&\triangleq\; U(:, \mathcal{I}_h) \in \mathbb{R}^{d_o \times r_h}, \\ V^{(h)} \;&\triangleq\; V(:, \mathcal{I}_h) \in \mathbb{R}^{d_i \times r_h}, \\ \Sigma^{(h)} \;&\triangleq\; \mathrm{diag}\big(\{\sigma_j\}_{j \in \mathcal{I}_h}\big) \in \mathbb{R}^{r_h \times r_h}. \end{aligned} \tag{16}$$

Then

$$\mathbf{W} \;=\; \sum_{h=1}^{H} U^{(h)}\,\Sigma^{(h)}\,V^{(h)\top}, \tag{17}$$

where encoder $\Phi^{(h)} = V^{(h)}$, decoder $\Psi^{(h)} = U^{(h)}$. When adapting to downstream tasks, we update the model embedding $\Delta\mathbf{S}^{(h)} \in \mathbb{R}^{r_h \times r_h}$. The weight updates become,

$$\Delta\mathbf{W} \;=\; \sum_{h=1}^{H} U^{(h)}\,\Delta\mathbf{S}^{(h)}\,V^{(h)\top}. \tag{18}$$

The structure is illustrated in Figure 4.

A.3 Convolutional Layers

Setting. Consider a convolutional layer with input $\mathbf{X} \in \mathbb{R}^{c_i \times h_i \times w_i}$ and output $\mathbf{O} \in \mathbb{R}^{c_o \times h_o \times w_o}$, parameterized by a kernel $\mathcal{F} \in \mathbb{R}^{c_i \times c_o \times k_h \times k_w}$.

Encoder and decoder over channels. We introduce an encoder and decoder over channels, shared across spatial locations. For a latent dimension r and H heads, the encoder and decoder for head h are

$$\Phi^h \in \mathbb{R}^{c_i \times r}, \qquad \Psi^h \in \mathbb{R}^{r \times c_o}, \qquad h = 1, \ldots, H.$$

Given an input feature map $\mathbf{X}$, the encoder performs a 1×1 convolution over channels:

$$\mathbf{Z}^h \;=\; \sum_{i=1}^{c_i} \Phi^h_i\,\mathbf{X}_i, \qquad \mathbf{Z}^h \in \mathbb{R}^{r \times h_i \times w_i}, \tag{19}$$

mapping c_i input channels to r latent channels for each head h.

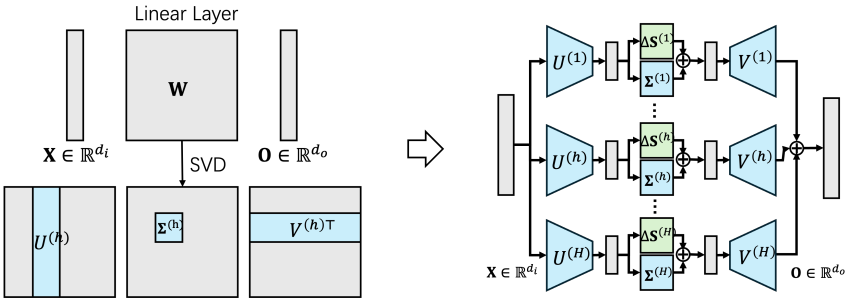


Figure 4: An illustration of extracting the encoder and decoder from a linear layer.

Model embedding over spatial filters. The model embedding at this layer is given by a bank of spatial atoms for each head,

$$\mathbf{S}^h \in \mathbb{R}^{r \times k_h \times k_w}.$$

These atoms act on the encoded latent channels via depthwise convolution:

$$\mathbf{Z}'^h \;=\; \mathbf{S}^h * \mathbf{Z}^h, \qquad \mathbf{Z}'^h \in \mathbb{R}^{r \times h_o \times w_o}, \tag{20}$$

where $*$ denotes 2D convolution in the spatial dimensions.

Decoding process. Finally, each head decodes latent channels back to output channels via another 1×1 convolution:

$$\mathbf{O}^h \;=\; \sum_{i=1}^{r} \Psi_i^h \, \mathbf{Z}_i'^h, \qquad \mathbf{O}^h \in \mathbb{R}^{c_o \times h_o \times w_o}, \tag{21}$$

and the layer output is obtained by summing over heads:

$$\mathbf{O} \;=\; \sum_{h=1}^{H} \mathbf{O}^h. \tag{22}$$

Model embedding space. If the encoder and decoder $\{\Psi^h, \Phi^h\}_{h=1}^H$ are fixed from the pretrained model, then all task-specific adaptation at this layer is captured by the collection of spatial atoms $\{\mathbf{S}^h\}_{h=1}^H$, which defines a *model embedding space* for this convolutional layer. Each adapted model corresponds to a point in this space, and model fusion reduces to linear operations on the $\{\mathbf{S}^h\}$.

Reconstruction of the convolutional kernel. The above encoder/decoder structure induces an explicit approximation of the original kernel $\mathcal{F}$. Plugging (19)–(21) into the standard convolution expression shows that

$$\mathcal{F} \;\approx\; \sum_{h=1}^{H} \sum_{i=1}^{r} \Psi_i^h \, \mathbf{S}_i^h \, \Phi_i^h. \tag{23}$$

The approximation in (23) becomes exact, *i.e.*, zero reconstruction error, whenever the true kernel $\mathcal{F}$ admits a factorization of this form with rank at most Hr. In particular, if we choose H and r such that Hr is at least the tensor rank of $\mathcal{F}$ and optimize $\{\Psi^h, \mathbf{S}^h, \Phi^h\}$ without additional constraints, we can recover $\mathcal{F}$ exactly.

B Analysis

B.1 Structural Relation Between Model Embedding Space and Feature Space

Consider a linear layer parameterized by an encoder/decoder decomposition

$$\mathbf{W} \;=\; \Psi \, \mathbf{S} \, \Phi^\top,$$

where $\Phi \in \mathbb{R}^{d_i \times r}$, $\Psi \in \mathbb{R}^{d_o \times r}$ are fixed encoder and decoder matrices, and $\mathbf{S} \in \mathbb{R}^{r \times r}$ is the model embedding. Given fixed input $\mathbf{x} \in \mathbb{R}^{d_i}$, the output is

$$\mathbf{o} = \mathbf{W}\mathbf{x} = \Psi \, \mathbf{S} \, (\Phi^\top \mathbf{x}) \;=\; \Psi \, \mathbf{S} \, \mathbf{z},$$

where $\mathbf{z} = \Phi^\top \mathbf{x}$ denotes the encoded latent feature.

Inner product structure. For two model embeddings $\mathbf{S}_1, \mathbf{S}_2$, the corresponding outputs are $\mathbf{o}_1 = \Psi\,\mathbf{S}_1\mathbf{z}$ and $\mathbf{o}_2 = \Psi\,\mathbf{S}_2\mathbf{z}$. Their output-space inner product satisfies

$$\langle \mathbf{o}_1, \mathbf{o}_2 \rangle \;=\; \mathbf{z}^\top \mathbf{S}_1^\top (\Psi^\top \Psi) \mathbf{S}_2 \mathbf{z}.$$

Thus, the output-space geometry is induced by the bilinear form

$$\langle \mathbf{S}_1, \mathbf{S}_2 \rangle_{\Psi,\mathbf{z}} \;=\; \mathbf{z}^\top \mathbf{S}_1^\top (\Psi^\top \Psi) \mathbf{S}_2 \mathbf{z},$$

showing that the model embedding space inherits a weighted inner product defined by the decoder and latent feature direction.

Bound. If Ψ has orthonormal columns, i.e. $\Psi^\top \Psi = \mathbf{I}_r$, then

$$\|\mathbf{o}_1 - \mathbf{o}_2\|_2 \;=\; \|\mathbf{S}_1\mathbf{z} - \mathbf{S}_2\mathbf{z}\|_2 \;\le\; \|\mathbf{S}_1 - \mathbf{S}_2\|_2\, \|\mathbf{z}\|_2,$$

which means the change in output is linearly bounded by the spectral norm distance between model embeddings.

For the general case, $\Psi^\top \Psi$ induces a condition number $\kappa(\Psi)$:

$$\sigma_{\min}(\Psi)\, \|\mathbf{S}_1 - \mathbf{S}_2\|_2\, \|\mathbf{z}\|_2 \;\le\; \|\mathbf{o}_1 - \mathbf{o}_2\|_2,$$
$$\|\mathbf{o}_1 - \mathbf{o}_2\|_2 \;\le\; \sigma_{\max}(\Psi)\, \|\mathbf{S}_1 - \mathbf{S}_2\|_2\, \|\mathbf{z}\|_2.$$

B.2 The Effect of Common Basis

Let $\boldsymbol{\tau}_i = \mathrm{vec}(\Delta \mathbf{W}_i) \in \mathbb{R}^D$ denote the update of model i and let $\mathbf{T} = [\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_M] \in \mathbb{R}^{D \times M}$. Choose a rank-$r$ common basis $U \in \mathbb{R}^{D \times r}$, which is obtained by the top-r SVD of $\mathbf{T}$, and its projector $\mathbf{P} = UU^\top$. Unaligned averaging gives $\bar{\boldsymbol{\tau}} = \frac{1}{M}\sum_i \boldsymbol{\tau}_i$, while aligned fusion gives $\hat{\boldsymbol{\tau}} = \mathbf{P}\,\bar{\boldsymbol{\tau}}$.

Proof of proposition 3.1. We first show that $\hat{\boldsymbol{\tau}}$ is the *unique minimizer* of projection of $\bar{\boldsymbol{\tau}}$ onto the r-dimensional subspace spanned by U:

$$\hat{\boldsymbol{\tau}} = \arg \min_{\boldsymbol{\tau}' \in \{Uz : z \in \mathbb{R}^r\}} \|\bar{\boldsymbol{\tau}} - \boldsymbol{\tau}'\|_2.$$

Proof. Since $\{Uz : z \in \mathbb{R}^r\}$ is a linear subspace, the minimizer of the distance from $\bar{\boldsymbol{\tau}}$ is given by the orthogonal projection:

$$P\bar{\boldsymbol{\tau}} = UU^\top \bar{\boldsymbol{\tau}}.$$

To see uniqueness, suppose $\boldsymbol{\tau}' \in \{Uz : z \in \mathbb{R}^r\}$ also attains the minimum:

$$\|\bar{\boldsymbol{\tau}} - \boldsymbol{\tau}'\|_2 = \|\bar{\boldsymbol{\tau}} - P\bar{\boldsymbol{\tau}}\|_2.$$

Because P is an orthogonal projector, $\bar{\boldsymbol{\tau}} - P\bar{\boldsymbol{\tau}}$ is orthogonal to $\{Uz : z \in \mathbb{R}^r\}$, hence

$$\langle \bar{\boldsymbol{\tau}} - P\bar{\boldsymbol{\tau}}, \boldsymbol{\tau}' - P\bar{\boldsymbol{\tau}} \rangle = 0.$$

The Pythagorean theorem then yields

$$\|\bar{\boldsymbol{\tau}} - \boldsymbol{\tau}'\|_2^2 = \|\bar{\boldsymbol{\tau}} - P\bar{\boldsymbol{\tau}}\|_2^2 + \|\boldsymbol{\tau}' - P\bar{\boldsymbol{\tau}}\|_2^2.$$

Equality with the minimum requires $\|\boldsymbol{\tau}' - P\bar{\boldsymbol{\tau}}\|_2^2 = 0$, which implies $\boldsymbol{\tau}' = P\bar{\boldsymbol{\tau}}$. Thus $\hat{\boldsymbol{\tau}} = P\bar{\boldsymbol{\tau}}$ is the *unique* minimizer of (3.1). □

Proof of proposition 3.2.

Proof. The alignment error follows

$$\begin{aligned}
\|\bar{\boldsymbol{\tau}} - \hat{\boldsymbol{\tau}}\|_2 &= \|(I - P)\bar{\boldsymbol{\tau}}\|_2 \\
&\le\; \frac{1}{M}\sum_{i=1}^{M} \|(I - P)\boldsymbol{\tau}_i\|_2 \\
&\le\; \frac{1}{M}\sum_{i=1}^{M} \|\boldsymbol{\tau}_i\|_2\, \sin\theta_i,
\end{aligned}$$

where θ_i is the largest principal angle between $\mathrm{span}(U)$ and $\mathrm{span}(\boldsymbol{\tau}_i)$. □

Remark B.1. Because U is chosen by SVD, it minimizes the total residual $\sum_i \|(I-P)\boldsymbol{\tau}_i\|_2^2 = \|(I-P)\mathbf{T}\|_F^2 = \sum_{j>r}\sigma_j(\mathbf{T})^2$, so the bound follows:

$$\|\bar{\boldsymbol{\tau}} - \hat{\boldsymbol{\tau}}\|_2 \;\le\; \frac{\|(I-P)\mathbf{T}\|_F}{\sqrt{M}} \;=\; \frac{\left(\sum_{j>r}\sigma_j(\mathbf{T})^2\right)^{1/2}}{\sqrt{M}}.$$

B.3 Computational Overhead

We analyze the computational overhead of the proposed aligned-subspace merging. Let M be the number of tasks and let r_m denote the latent feature dimension. For simplicity, we assume $r_m = r_i = r_o$, and r denotes the rank of the shared aligned basis. The time complexity of our merging procedure is

$$\mathcal{O}(Mr_m^3), \tag{24}$$

which is comparable to existing merging methods because r_m is much smaller than the original feature dimension d.

The cost can be decomposed as follows:

$$\text{Per-task decomposition:} \quad \mathcal{O}(Mr_m^3), \tag{25}$$

$$\text{Aligned subspace basis:} \quad \mathcal{O}(Mr_m^2 r), \tag{26}$$

$$\text{Coefficient projection:} \quad \mathcal{O}(Mr_m^2 r), \tag{27}$$

$$\text{Coefficient averaging:} \quad \mathcal{O}(Mr_m r), \tag{28}$$

$$\text{Merged embedding reconstruction:} \quad \mathcal{O}(r_m^2 r). \tag{29}$$

The dominant term is $\mathcal{O}(Mr_m^3)$, and the total cost scales linearly with the number of tasks M.

Table 7: **Wall-clock merging time and complexity comparison.** Here $d = d_i = d_o$ is the feature dimension and r_l is the LoRA rank.

Method	Merging Time (s)	Complexity
Task Arithmetic	4.17	$\mathcal{O}(Md^2)$
TSV	12.02	$\mathcal{O}(Md^3)$
KnOTS	48.45	$\mathcal{O}(M^2d^3)$
Core	10.14	$\mathcal{O}(Md^2r_l)$
Ours	5.88	$\mathcal{O}(Mr_m^3)$

As shown in Table 7, our method has a low wall-clock merging time. Since r_m is usually much smaller than d, the proposed method remains efficient while performing alignment in the model embedding space.

C Property of Model Embedding Space

The model embedding space provides a unified low-dimensional representation of parameter updates derived from a pretrained model. Because Φ and Ψ are derived from the pretrained model, they remain *frozen*, and all task-specific knowledge is encoded solely by modifying $\mathbf{S}$. This structure induces several key properties.

Low-dimensionality. The intrinsic ranks $r_i, r_o \ll d_i, d_o$ reflect the assumption that the pretrained model operates on a low-rank feature manifold. Thus, updates to $\mathbf{S}$ correspond to adjusting the model only within this intrinsic subspace, avoiding unnecessary changes in orthogonal directions.

Shared coordinate system. By fixing Φ and Ψ, all adaptations are expressed in the same latent coordinates. Two updates $\Delta\mathbf{S}_1$ and $\Delta\mathbf{S}_2$ are therefore represented in a common basis, making addition, averaging, alignment, or similarity comparisons well-posed. This resolves a fundamental difficulty in model fusion: conventional fine-tuning does not guarantee that updates across tasks lie in comparable subspaces.

Table 8: **Sensitivity to r_i/d and r/r_i on ViT-B/32.** We report fine-tuned accuracy, trainable parameters, and merged accuracy under different intrinsic rank ratios r_i/d and aligned-subspace ratios r/r_i.

	Fine-tuned Acc.	Params (M)	$r/r_i = 25\%$	$r/r_i = 50\%$	$r/r_i = 75\%$	$r/r_i = 87.5\%$	$r/r_i = 98\%$
$r_i/d = 8.33\%$	84.03	0.58	52.63	55.79	61.96	61.72	60.85
$r_i/d = 12.5\%$	84.68	0.87	54.42	58.99	62.58	62.96	63.67
$r_i/d = 16.7\%$	84.99	1.16	55.32	60.90	64.23	66.02	65.89
$r_i/d = 25\%$	85.34	1.75	58.83	64.16	67.32	**68.28**	68.10

Table 9: **Sensitivity to r/r_i on MLLMs.** We vary the aligned-subspace ratio r/r_i and report seen-task and unseen-task averages.

r/r_i	25%	50%	75%	99%
Seen task accuracy	54.95	57.54	58.48	**58.75**
Unseen task accuracy	37.31	38.39	39.47	**39.73**

Preservation of pretrained knowledge. Because Φ and Ψ reconstruct the pretrained weights exactly, the update $\Delta\mathbf{S}$ modifies only the intrinsic transformation, leaving the pretrained structure intact. This yields parameter-efficient adaptation while preventing catastrophic drift from the pretrained model.

We illustrate this behavior in Figure 1 (b). In this experiment, we train an MLP-based VAE on a synthetic 2D Swiss Roll dataset and then adapt the pretrained model to two downstream tasks, each derived from a small sub-distribution. We compare our method with LoRA by visualizing the generated samples. As shown in the figure, our approach remains well aligned with the pretrained distribution while still adapting to the fine-tuned data. In contrast, LoRA tends to overfit to the downstream distribution, thereby compromising the structure learned during pretraining. Consequently, the fused LoRA model also deviates from the original data distribution, whereas our method preserves the pretrained structure and successfully adapts the downstream data distributions.

D Related Work

D.1 MLLM Adaptation and Merging

Multimodal LLMs are typically modular, consisting of a frozen vision encoder, a language decoder, and a lightweight connector such as a projector or cross-attention block. This structure makes PEFT especially effective for multimodal adaptation. Recent instruction-tuning pipelines commonly introduce LoRA or small adapters into decoder and cross-modal layers to efficiently acquire VQA, OCR, REC, and captioning abilities [41, 42], while newer open models further improve perception quality and input resolution [74]. Built on these PEFT backbones, model-level merging methods either combine task-specialized heads or adapters into a single checkpoint [2, 83], or directly merge fully fine-tuned MLLMs. Large-scale studies further provide practical recipes, scaling strategies, and unified evaluation toolkits for assessing robustness and holistic multimodal capability [3, 11, 16, 37, 46, 85].

D.2 Comparison with Other Decomposition Methods

In this section, we provide comparison between our method and other related methods.

LoRA with SVD reparameterization. LoRA [28] constrains adaptation to a learned low-rank update $\Delta\mathbf{W} = \mathbf{BA}$, but the basis $(\mathbf{A}, \mathbf{B})$ is optimized independently for each task, so different fine-tuning runs generally produce updates that reside in *task-specific and mutually misaligned* subspaces. DoRA [44] improves stability by decoupling magnitude and direction, yet it still learns task-dependent directions and does not guarantee that updates from different tasks are expressed in a common coordinate system. In contrast, our method derives a *task-independent* encoder/decoder decomposition directly from the pretrained model, ensuring exact reconstruction of the base weights and constraining all downstream adaptations to occur within a *shared intrinsic subspace*. This yields updates that are naturally aligned and geometrically comparable, a property that neither LoRA nor DoRA provides without additional post-hoc alignment.

Table 10: We demonstrate the effectiveness of our aligned subspace merging on a masked-image reconstruction task on MNIST. We pre-train an autoencoder on digit 0, then fine-tune model embeddings separately on digits 2 and 8. The *aligned subspace merging* produces the merged model by aligning these two model embeddings in a shared subspace, which leads to better reconstructions than arithmetic average merging. The details are provided in Appendix E.1.

Task	Method (MSE ↓)	
	Arithmetic average	Aligned subspace
Digit 2	0.21	**0.13**
Digit 8	0.27	**0.14**

Post-hoc SVD merging of weight updates. Several model-merging approaches apply SVD or PCA directly to a set of vectorized task updates $\{\Delta\mathbf{W}_i\}$, *i.e.*, SVDiff [24]. These methods produce a shared basis in parameter space but do not account for the pretrained model's structure; the derived basis depends solely on the fine-tuning trajectories. This conflates signal and noise in the updates. Our model embedding space instead derives its basis *from the pretrained model itself*, reflecting the intrinsic geometry of its feature manifold. Task updates are thus expressed in a subspace that is both stable and architecturally aligned.

Weight-space SVD reparameterization. Reparameterizing each layer as $\mathbf{W} \approx \mathbf{USV}^\top$ is a direct low-rank approximation commonly used for model compression. While this reduces parameters, it discards residual components and modifies the pretrained model. In contrast, our encoder/decoder decomposition maintains *exact* reconstruction: Ψ and Φ span the full row/column spaces of the pretrained weight, and all information is preserved. Adaptation is therefore stable and does not degrade the base model.

Subspace-based decomposition. Prior studies on task subspace modeling represent tasks as compositions or linear combinations of latent basis tasks [18]. Extending this idea to convolutional networks, DCFNet [60] introduces a filter subspace that represents convolutional filters as linear combinations of a predefined set of filter atoms. Filter subspace modeling has since been explored in a wide range of applications, including continual learning [54], distributed learning [8], and adaptive convolution [75]. Building upon this line of research, our work extends subspace decomposition beyond convolutional filters to linear layers and multi-head attention (MHA), enabling structured model representations in Transformer architectures.

E Implementation Details

E.1 MNIST Experiment

Experimental setting. Our base model is a lightweight fully connected autoencoder for 32×32 MNIST images (normalized to $[-1, 1]$). The encoder is $1024 \to 512 \to 256 \to 64$, and the decoder mirrors it $64 \to 256 \to 512 \to 1024$. We pretrain the autoencoder on digit 0 using AdamW. Training uses an MAE-style masked reconstruction loss to compute.

Adaptation models. Starting from the pretrained autoencoder, we apply model embedding adapters into all linear layers in both encoder and decoder, freeze the pretrained weights, and fine-tune only the model embedding parameters. We fine-tune digit-2 and digit-8 adapters separately with AdamW. We then merge the digit-2 and digit-8 model embeddings layer-wise using arithmetic average and aligned subspace merging to explore the effectiveness of our method.

Metrics. While training uses MSE reconstruction loss, all reported fusion results are evaluated using mean squared error (MSE) on the MNIST test sets for digits 2 and 8.

E.2 SVHN Experiment

Experimental setting. Our base model is a fully connected autoencoder. The encoder consists of two linear layers with ReLU activations, and the decoder mirrors this with a final sigmoid to produce outputs in [0,1]. The latent dimension is 128. We pretrain this dense autoencoder on digits 0–5 from SVHN dataset for 20 epochs using Adam. The training and validation objective is binary cross-entropy (BCE) reconstruction loss between the output and the input image.

Metrics. While training uses BCE reconstruction loss, all reported fusion results are evaluated using mean squared error (MSE) on the SVHN test sets for digits 6–9.

E.3 Vision Tasks

We conduct experiments using ViT-B/32 [14] as the backbone model and evaluate performance on eight downstream image classification tasks, following the standard setup in prior work [69]. For the model embedding fine-tuning in the ViT experiment, we exactly follow the experiment setting as in [69] and [57], with the same random seed. We use AdamW optimizer with learning rate 3e-4, weight decay 0.1. For merging in the model embedding space, we mainly tune the r and select $r = r_i - 1$ to provide the best accuracy. Notably, when $r/r_i > 50\%$, our method always produces better results than baseline methods, demonstrating robustness to this hyperparameter. To evaluate other merging methods in the model embedding space, we directly reuse the hyperparameters reported in [57]. All experiments are done on NVIDIA RTX5000 GPUs.

We compare our approach against a range of strong model merging baselines, including Task Arithmetic [31], TIES-merging [80], DARE [82], TSV [20], ISO-C [51], KnOTS [69], and Core [57]. All baselines operate in the *weight space* and rely on task-specific fine-tuned checkpoints obtained via LoRA [28], where rank-16 adapters are applied to all attention projection matrices, including queries, keys, values, and outputs, across all transformer layers.

In contrast, our method constructs a *model embedding space* from the pre-trained attention weights and adapts each downstream task by updating only the corresponding model embedding parameters, following the same formulation described in Section 2.5. After obtaining task-specific embeddings, we evaluate both *arithmetic average* and *aligned subspace merging* strategies in the model embedding space. Additionally, to study the effect of the aligned subspace, we apply the same aligned subspace merging procedure directly in the weight space by *aligning LoRA* checkpoints from prior methods before merging.

E.4 Multimodal Large Language Models

We evaluate multimodal model merging on MM-MergeBench [83] using the LLaVA [42] codebase. We introduce trainable *model embeddings* into multi-head attention and linear layers (excluding the *K* and *O* projection parameters). And we freeze all original pre-trained weights. For each task, we fine-tune only these model embedding parameters for about 1 epoch using AdamW optimizer, with learning rate 2e-4 and warmup ratio 0.03. We use DeepSpeed ZeRO-3 in bf16, with per-device train/eval batch sizes of 16/4 on NVIDIA A100 GPUs. Depending on the dataset, the training time varies from 0.3 to 11 hours. For merging in the model embedding space, we adopt the same strategy and parameter setting in Section E.3.

MM-MergeBench contains eight *seen* datasets: ScienceQA [47], ImageNet [13], VQAv2 [21], REC-COCO [33, 50], OCRVQA [55], VizWiz-caption [23], Flickr30k [58], IconQA [49]; and four *unseen* datasets: AOKVQA [64], ImageNet-R [25], Screen2Words [73], TabMWP [48]. We follow the official splits and evaluation protocol. We compare our method against standard weight space, *i.e.*, LoRA, merging baselines from MM-MergeBench, including arithmetic average, Task Arithmetic [31], Ties-Merging [80], DARE [82], PCB-Merging [22], WUDI-Merging [12], and RobustMerge [83].

E.5 Knowledge Preservation Experiment

We use ImageNet accuracy to measure knowledge preservation and target-task accuracy to measure adaptation. As shown in Table 6, full-parameter merging achieves strong target-task accuracy but causes substantial forgetting, with a 7.3% ImageNet top-1 drop. In contrast, our model embedding

space achieves higher target-task accuracy than LoRA while almost preserving the original ImageNet performance, indicating better adaptation with minimal forgetting.

Experimental details. We follow [69, 57, 51] and use ViT-B/32 of the CLIP image encoder as the pre-trained model. We use ISO-C [51] for both full-parameter and LoRA merging. The full-parameter checkpoints are from [51], and the LoRA checkpoints are from [69].

F Additional Experimental Results

F.1 Accuracy Metric Clarification

Prior model-merging papers sometimes report *normalized accuracy*, which is defined as

$$\mathrm{Acc}_{\mathrm{norm}}(i) = \frac{\mathrm{Acc}_{\mathrm{merged}}(i)}{\mathrm{Acc}_{\mathrm{finetuned}}(i)} \times 100\%. \tag{30}$$

Table 11: **Comparison between normalized and absolute accuracy.** We report absolute accuracy in the main paper. Normalized KnOTS results are converted to absolute accuracy for reference.

Method	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC	SUN397	SVHN	Average
LoRA	74.00	58.30	99.00	92.70	99.30	88.40	64.50	96.20	84.10
KnOTS Normalized [69]	82.70	73.70	49.30	48.90	68.90	70.90	95.50	53.80	68.00
KnOTS Absolute	61.20	42.97	48.80	45.33	68.42	62.68	61.60	51.76	55.35
KnOTS Normalized [57]	80.81	83.03	58.25	74.34	67.66	79.69	98.54	49.86	74.02
KnOTS Absolute	59.80	48.43	57.67	68.91	67.19	70.45	63.56	47.97	60.50

In contrast, we report the original absolute accuracy. This is because our fine-tuned checkpoints are trained in the model embedding space, while many baselines use existing LoRA checkpoints. Since the fine-tuned checkpoints can have different single-task accuracies, normalizing by each fine-tuned model may introduce an unfair comparison. Therefore, the main paper follows the standard absolute-accuracy reporting protocol.

To clarify the relationship between normalized and absolute accuracy, Table 11 converts representative KnOTS results from normalized accuracy to absolute accuracy using the corresponding LoRA fine-tuned accuracy.

For example, a normalized accuracy of 82.70% on Cars corresponds to an absolute accuracy of 61.20%, because $61.20/74.00 = 82.70\%$. This explains why the main paper reports absolute accuracy: it directly reflects the final merged model performance without being affected by different fine-tuned checkpoint strengths.

F.2 Rank Sensitivity Study

ViT Rank Sensitivity We study two rank-related factors: the intrinsic rank ratio r_i/d and the aligned-subspace ratio r/r_i. The ratio r_i/d controls the capacity of the model embedding space, while r/r_i controls how much information is retained when constructing the aligned subspace for merging. In the main experiments, we tune only MHA blocks, where $r_i = d/H$ is fixed by the number of heads. To explicitly vary r_i/d, we conduct an additional ablation on MLP layers.

Table 8 shows that increasing r_i/d generally improves merging performance, while single-task fine-tuning is less sensitive to this ratio. This suggests that a richer model embedding space is especially beneficial for merging, because it preserves more task-relevant directions shared across models.

The aligned-subspace ratio r/r_i controls the trade-off between compression and information preservation during merging. Very small aligned ranks discard useful task information, while moderate-to-large ratios lead to stronger and more stable performance. Overall, the method benefits from a sufficiently expressive embedding and aligned subspace, but does not require a narrowly tuned rank.

MLLM Rank Sensitivity We also evaluate the effect of the aligned-subspace rank on MM-Merge-Bench. Table 9 reports both seen-task and unseen-task averages as a function of r/r_i.

The MLLM results follow the same trend as ViT. Increasing r/r_i improves the merged model, while performance remains stable once the rank is sufficiently large. Given the efficiency of the proposed merging method, we set $r = r_i - 1$ in our main experiments, corresponding to $r/r_i = 99\%$ for MLLMs.

F.3 Random Seed Robustness

Once the merged model is obtained, inference is deterministic, as no randomness is applied. In particular, for MLLM experiments, it is required to ensure deterministic text generation for fair comparison. Once the optimal merging hyperparameters are obtained, the model merging is also deterministic.

Randomness arises only during the training stage, due to factors such as SGD and stochastic data sampling. For a fair comparison with [57, 69, 83], we use the same random seed when fine-tuning in the model embedding space. Therefore, the main paper does not report variations across different random seeds.

To illustrate the impact of randomness, we apply different random seeds during training. For the ViT experiments, we report the fine-tuning results for both LoRA and our method. For the MLLM experiments, due to the high computational cost, we report fine-tuning results of only our method.

Table 12: **Random seed robustness.** We report mean and standard deviation over three random seeds.

(a) ViT fine-tuning robustness.

Method	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC	SUN397	SVHN	Average
LoRA	74.20 ± 0.17	57.86 ± 0.33	98.90 ± 0.03	92.68 ± 0.12	99.10 ± 0.22	89.10 ± 0.42	64.73 ± 0.68	96.19 ± 0.14	84.09 ± 0.23
Ours	75.62 ± 0.12	55.93 ± 0.80	99.10 ± 0.02	93.71 ± 0.17	98.50 ± 0.29	89.90 ± 0.30	68.25 ± 0.45	96.22 ± 0.31	84.65 ± 0.30

(b) MLLM fine-tuning robustness.

Method	SciQA	Image	VQA	REC	OCR	VizWiz	Flickr	IconQA	Average
Ours	85.69 ± 0.27	96.07 ± 1.44	68.36 ± 0.39	47.81 ± 0.51	65.98 ± 0.16	59.90 ± 0.25	56.56 ± 0.16	76.75 ± 0.49	69.64 ± 0.13

(c) Aligned-subspace merging robustness.

ViT	MLLM Seen Tasks	MLLM Unseen Tasks
65.20 ± 0.38	59.06 ± 0.45	39.64 ± 0.13

Using model embeddings fine-tuned with different random seeds, we report the aligned subspace merging results in terms of mean and standard deviation in Table 12. Overall, our method demonstrates robustness to random seeds. For individual tuning, adaptation in the model embedding space achieves higher accuracy than LoRA, and the merged models obtained via aligned subspace merging consistently outperform the baselines.

F.4 Vision Experiments with Different Architectures

Table 13 contains the results for ViT-B/32, ViT-B/16, and ViT-L/14 on 8, 14, and 20 tasks. Our Model Embedding Space with Aligned Subspace Merging consistently achieves the best performance across different backbone architectures, model scales, and numbers of tasks, demonstrating stable and significant advantages over LoRA-merging baselines. The LoRA baselines were reproduced following the guidance and official implementations of KnOTS, ISO-C, and Core Space. The reported fine-tuning baseline uses LoRA with rank (r=16), rather than full-parameter fine-tuning.

F.5 Experiments on NLP Tasks

To show the performance on pure NLP (LLMs) tasks, we conduct experiment on NLI setting. It includes 6 datasets: SNLI, MNLI, SICK, QNLI, RTE, and SCITAIL.

Table 14 shows that our Model Embedding Space and aligned-subspace merging framework generalizes effectively to pure NLP tasks. Across six NLI benchmarks, it achieves the highest average merged accuracy of 88.11%, outperforming the strongest baseline. This demonstrates that our approach extends beyond vision models and MLLMs, effectively mitigating cross-task interference in large language model merging.

Table 13: Comparison of fine-tuning and model merging performance across different ViT architectures and numbers of tasks. All results are reported as absolute accuracy (%).

Type	Method	ViT-B/32			ViT-B/16			ViT-L/14		
		8	14	20	8	14	20	8	14	20
Zero-shot	–	48.42	57.20	56.10	55.30	61.30	59.70	64.70	68.20	65.20
Fine-tuning	LoRA	84.10	85.16	87.01	89.83	88.98	89.99	92.31	92.03	92.96
	Model Embedding	84.61	86.43	88.02	90.51	89.84	90.92	92.93	92.31	93.25
LoRA Merging	Task Arithmetic	51.23	61.28	62.41	68.96	71.10	68.19	68.57	66.47	75.22
	TIES	51.28	59.75	60.07	69.25	69.14	65.42	69.30	71.80	69.75
	DARE-TIES	51.15	61.16	61.61	72.89	70.91	67.02	69.32	72.86	71.36
	TSV	53.94	63.44	64.36	77.85	69.79	63.72	71.65	78.06	78.24
	ISO-C	60.09	58.06	57.63	63.14	70.04	69.30	77.00	69.38	66.97
	KnOTS	60.50	63.66	62.97	65.03	71.31	69.27	77.43	75.51	74.08
	Core	62.53	62.16	61.87	78.61	72.49	67.82	79.26	75.07	72.36
	Ours	52.50	61.16	60.73	77.40	73.91	68.47	70.85	73.88	71.91
Model Embedding Merging	Ours	**64.82**	**74.01**	**69.31**	**81.91**	**77.59**	**72.56**	**89.67**	**85.70**	**82.91**

Table 14: Comparison of fine-tuning and model merging performance across six natural language inference tasks. All results are reported as absolute accuracy (%).

Type	Method	SNLI	MNLI	SICK	QNLI	RTE	SCITAIL	Avg.
Fine-tuning	LoRA	92.50	90.31	91.58	94.49	89.86	96.52	92.54
	Model Embedding	93.04	91.07	91.72	94.67	89.13	96.90	92.76
LoRA Merging	TA	86.55	86.05	80.55	64.92	89.86	93.36	83.55
	TIES+KnOTS	84.93	85.06	85.14	74.24	89.86	94.21	85.57
	TIES+Core	85.16	84.45	85.75	79.11	89.13	94.26	86.31
	DARE-TIES+KnOTS	84.75	87.35	68.59	**80.08**	89.39	**95.68**	84.31
	DARE-TIES+Core	85.19	84.51	85.81	79.07	89.13	94.26	86.33
	TSV+KnOTS	85.59	86.54	75.80	72.77	89.86	93.70	84.04
	TSV+Core	88.67	86.43	81.74	79.27	**92.03**	94.45	87.10
	Iso-C+KnOTS	78.88	47.74	81.90	51.88	67.40	75.03	67.14
	Iso-C+Core	84.67	81.37	80.47	71.67	89.13	94.03	83.56
Model Embedding Merging	Ours	**90.75**	**88.95**	**89.89**	76.06	91.30	91.72	**88.11**

G Future Work

Empirical results demonstrate that model merging via arithmetic average is significantly more effective within the model embedding space, as evidenced by the enhanced accuracy of the merged models. However, a rigorous theoretical characterization of the mapping between the model embedding space and the task-specific functional space remains to be established. Future research will focus on elucidating this connection to determine which geometric operations in the embedding space most directly translate to gains in downstream task performance.

Our work provides extensive experimental evaluations across image classification, natural language processing (NLP), and multimodal large language model (MLLM) tasks. Beyond these settings, our framework has the potential to benefit a broader range of applications, including concept merging in image and video generation [5, 26, 34, 36, 59] and reasoning tasks [10, 76]. Furthermore, the proposed model embedding space offers a promising direction for knowledge accumulation and integration, with potential applications in continual learning [84] and federated learning [4, 39, 65, 66, 67].

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We have clarified our contributions in both the abstract and introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations at the end of the appendix.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: We provide assumptions and complete proofs for each theoretical result.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide the experimental settings, datasets, baselines, hyperparameters, and evaluation protocols needed to reproduce the main results. We also provide experimental results of random seed robustness in the appendix.

   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: [No]

Justification: We provide detailed experimental settings, datasets, hyperparameters, and evaluation protocols in the paper and supplementary material. However, the code is not publicly released at submission time.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: We specify the datasets, data splits, training settings, hyperparameters, optimizer choices, and evaluation protocols in the paper.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: In the appendix, we report mean and standard deviation over multiple runs for the experiments where statistical variability is evaluated.

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 compute resources used for our experiments in the implementation detail section of the appendix.

   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 in the paper conforms to the NeurIPS Code of Ethics.

   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: [N/A]

    Justification: This work is a foundational method for model merging. It does not target a specific societal application, and therefore has no direct societal impact specific to the proposed method.

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: There's no such risk in the paper.

    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 properly cite the original papers and sources for all existing assets used in this work. We follow their corresponding licenses and terms of use.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [N/A]

    Justification: This paper does not introduce or release new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing experiments or 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: This paper does not involve crowdsourcing experiments, user studies, or 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 method development does not involve LLMs.

    Guidelines:

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