PGSB: Pretrained-Guided Shared Basis for LoRA Model Merging

Muqing Liu[1,†], **Chongjie Si**[2,†], **Zhuoya Liu**[1], **Yuheng Jia**[3,4*]
[1]Chien-Shiung Wu College, Southeast University [2]MoE Key Lab of Artificial Intelligence, AI Institute, School of Computer Science, Shanghai Jiao Tong University [3]School of Computer Science and Engineering, Southeast University [4]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
liumq23@seu.edu.cn chongjiesi@sjtu.edu.cn zhuoyaliu@seu.edu.cn yhjia@seu.edu.cn

Abstract

Large pretrained models have achieved remarkable success across diverse domains, yet adapting them to multiple downstream tasks remains challenging: full fine-tuning is costly, while maintaining separate task-specific models is storage-inefficient and fails to produce a unified multi-task model. Low-Rank Adaptation (LoRA) offers an efficient alternative for Model Merging by representing each task adaptation as a low-rank update over frozen pretrained weights. However, existing LoRA merging methods often treat task-specific updates as directly comparable objects and merge them in their original low-rank parameter spaces. This overlooks two key factors: which update directions provide reliable shared support across tasks, and how these directions should be re-parameterized with respect to the pretrained weights before merging. To address this limitation, we propose Pretrained-Guided Shared Basis (PGSB), a two-stage framework for LoRA model merging. PGSB first identifies shared directions among task-specific LoRA updates, and then re-parameterizes these directions using their interaction with the pretrained weights, producing a more suitable basis for merging. Extensive experiments demonstrate that PGSB consistently outperforms state-of-the-art baselines across multiple benchmarks. Codes are available at `https://github.com/Kitschnada/Pretrained-Basis`.

1 Introduction

Large pretrained models have demonstrated strong performance across diverse tasks [11, 25, 31], making them powerful foundations for downstream adaptation. As independently adapted models become increasingly available through model hubs such as Hugging Face [21], model merging has emerged as an attractive paradigm for constructing a unified multi-task model by combining task-specific models without requiring access to their original training data [22, 37, 51, 60, 62]. However, most existing merging methods are designed for fully fine-tuned models [22, 62, 63, 64], which presupposes that each task model is obtained by updating all pretrained parameters. This requirement becomes increasingly costly in computation, memory, and storage as model scale grows [32, 45, 47]. In contrast, parameter-efficient fine-tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA) [56], learns compact task-specific adaptations over frozen pretrained weights with extremely low costs while retaining competitive performance [17, 19, 20, 30, 52, 68]. There-

*Corresponding author.
†Equal contribution.

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

fore, merging PEFT-based adaptations has attracted growing research attention as a scalable route toward unified multi-task models.

Despite this promise, directly extending existing merging methods to PEFT-based adaptations remains non-trivial. In LoRA, each task adaptation is represented by a pair of low-rank matrices rather than a full weight update, making its merging geometry fundamentally different from that of fully fine-tuned models. Recent methods such as KnOTS and Core Space [55, 42] address this issue by projecting LoRA updates into a shared subspace before merging. Although these methods have achieved promising results, they do not fully resolve the core difficulty of LoRA merging. A more principled solution must answer two challenging questions: how to distill a unified set of directions from diverse task-specific LoRA updates so that it can carry information shared across multiple tasks, and how to represent these directions in a way that remains compatible with the scale and geometry of the original pretrained weights.

To answer these questions, we propose a novel framework named **Pretrained-Guided Shared Basis** (PGSB) for LoRA model merging. PGSB consists of two stages: *Task-shared Subspace Extraction* (TSE) and *Pretrained-coupled Coordinate Refinement* (PCR), which accordingly determines the shared task directions and their pretrained-compatible merge coordinates. In the first stage, PGSB concatenates task-specific LoRA updates and performs singular value decomposition to extract a shared subspace, yielding a unified multi-task basis that captures common task-relevant directions. In the second stage, PGSB refines this basis using the geometric structure of the pretrained weights, producing a pretrained-compatible multi-task basis that reduces conflicts with the original weight space. By merging LoRA updates in this refined basis, PGSB better preserves task-specific knowledge while maintaining compatibility with the pretrained model, leading to more reliable multi-task adaptation. Extensive experiments on vision and language benchmarks show that PGSB consistently outperforms strong baselines and achieves state-of-the-art results. Our contributions are as follows:

- We show that effective LoRA merging requires both constructing a shared multi-task basis and ensuring its compatibility with the pretrained weight geometry.
- We propose PGSB, a two-stage LoRA merging framework that first extracts a task-shared subspace from concatenated LoRA updates and then refines the resulting basis using the geometric structure of pretrained weights.
- Extensive experiments on vision and language benchmarks demonstrate that PGSB consistently outperforms strong model merging baselines, validating the effectiveness of pretrained-guided basis refinement for scalable multi-task adaptation.

2 Related Work

2.1 Model Merging

With the proliferation of task-specific models, integrating them into a single model capable of handling multiple tasks without additional training has become a central problem in model merging [36]. A representative approach is task arithmetic [22], which constructs task vectors by subtracting base-model parameters from fine-tuned models and combines them through linear aggregation. However, it often suffers from performance degradation due to interference among task updates. To mitigate this issue, subsequent methods introduce more structured merging strategies [2, 51, 23, 41, 35]. TIES [62] resolves sign conflicts and filters updates by magnitude, while DARE [64] reduces redundancy through random dropping and rescaling. More recent methods, such as TSV [16] and Iso-C [33], further exploit the matrix structure of task updates, highlighting the importance of geometric alignment in effective merging.

Despite these advances, methods designed for fully fine-tuned models do not transfer well to LoRA-based adaptations [56]. In LoRA, task adaptations are represented as low-rank factors, leading to a fundamentally different merging geometry. KnOTS [55] addresses this by merging in a shared subspace, but incurs high computational cost due to SVD on concatenated full-size matrices. Core Space [42] improves efficiency via a shared low-rank space, yet relies on a fixed linear representation and insufficiently models interactions between task updates and pretrained weights. Thus, achieving efficient and reliable LoRA merging while preserving task information remains an open challenge. Detailed formulations of KnOTS and Core Space are deferred to Appendix A.

2.2 Parameter-efficient Fine Tuning (PEFT)

Large pretrained models have been applied in various downstream tasks, enabling significant improvement in generalization. Due to the huge cost fully fine-tuning brings to computation and storage, Parameter-efficient fine-tuning (PEFT) methods have been proposed, which uses only a small subset of parameters without reducing the performances of fully fine-tuning. PEFT techniques include strategies such as Adapter [17, 44, 19], Prompt Tuning [29, 50, 58, 15], and Low-Rank Adaptation (LoRA) [20, 67, 52, 53, 30]. Among these, LoRA [20] is widely adopted for its simplicity and effectiveness. It freezes the pretrained weights and learns task-specific low-rank updates for linear layers, enabling efficient adaptation with few trainable parameters. Since the learned updates can be merged into the original weights during inference, LoRA preserves the model architecture without introducing additional latency.

3 Method

3.1 Preliminary

Low-Rank Adaptation. For a pretrained weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA freezes $\mathbf{W}$ and introduces a trainable low-rank update $\Delta\mathbf{W}$:

$$\mathbf{W}' = \mathbf{W} + \Delta\mathbf{W}, \quad \Delta\mathbf{W} = \mathbf{B}\mathbf{A}, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ are the only trainable parameters. The rank $r \ll \min(d_{\text{out}}, d_{\text{in}})$ is chosen to be small, so that each task adaptation can be learned with substantially fewer parameters than full fine-tuning.

Model Merging. Given a set of task-specific models $\{\mathbf{W}_t\}_{t=1}^{T}$ fine-tuned from the same pretrained model $\mathbf{W}_0$, model merging aims to construct a single model that integrates knowledge from multiple tasks without additional training. A common formulation is to first define the task vector of each model as the difference from the pretrained weights [22], i.e., $\tau_t = \mathbf{W}_t - \mathbf{W}_0$. Then, a merging operator $\mathcal{M}$ is applied to obtain a merged task vector:

$$\tau^\star = \mathcal{M}(\tau_1, \tau_2, \ldots, \tau_T). \tag{2}$$

Finally, the merged model is reconstructed as

$$\mathbf{W}_{\text{merge}} = \mathbf{W}_0 + \tau^\star. \tag{3}$$

3.2 Overview

PGSB separates LoRA merging into two stages: task-shared subspace extraction (TSE) and pretrained-coupled coordinate refinement (PCR). TSE extracting a task-shared subspace that provides a unified multi-task basis, and PCR refines this basis with the pretrained weight geometry to obtain pretrained-compatible merge coordinate. The following sections describe these two stages in detail. The framework of PGSB is shown in Fig. 1.

3.3 Task-shared Subspace Extraction

PGSB first constructs a shared pair of support matrices for representing multiple LoRA updates. Since each update $\Delta\mathbf{W}_t$ is a matrix, its action involves two coupled spaces: the column space and the row space. Therefore, a single shared basis is insufficient; we need to extract common structures from both sides of the updates. We refer to the resulting bases as the left and right support matrices.

Specifically, PGSB first concatenates all task-specific updates and applies singular value decomposition (SVD):

$$[\Delta\mathbf{W}_1, \Delta\mathbf{W}_2, \ldots, \Delta\mathbf{W}_T] = \mathbf{U}_L\mathbf{\Sigma}_L\mathbf{V}_L^\top, \tag{4}$$

where $\mathbf{U}_L$ captures the shared column-space structure of task updates and is used as the left support matrix. To extract the shared row-space structure, we equivalently apply SVD to the concatenated transposed updates:

$$[\Delta\mathbf{W}_1^\top, \Delta\mathbf{W}_2^\top, \ldots, \Delta\mathbf{W}_T^\top] = \mathbf{U}_R\mathbf{\Sigma}_R\mathbf{V}_R^\top. \tag{5}$$

where $\mathbf{U}_R$ is used as the right support matrix. The transposition is introduced because the row space of $\Delta\mathbf{W}_t$ becomes the column space of $\Delta\mathbf{W}_t^\top$, allowing both support matrices to be obtained through the same SVD-based construction.

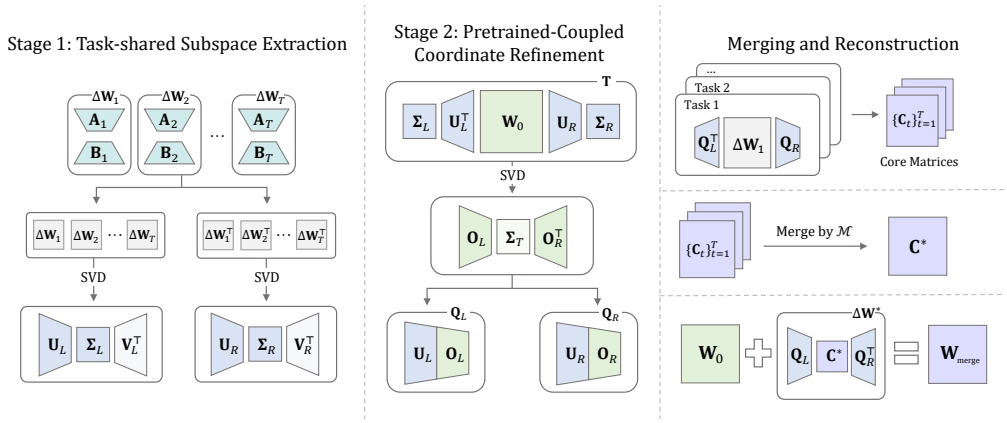

Figure 1: Overview of PGSB.Given task-specific LoRA updates, PGSB first performs Task-shared Subspace Extraction by decomposing the concatenated updates and their transposed counterparts, producing shared support matrices. PGSB then performs Pretrained-Coupled Coordinate Refinement, where the extracted supports are coupled with the pretrained weight geometry to obtain pretrained-compatible multi-task bases. Finally, task updates are projected into the refined bases, merged by a downstream merge operator $\mathcal{M}$, and reconstructed back to the original parameter space to obtain the final merged model.

3.3.1 Efficient Low-Rank Support Construction

Although the full-update formulation provides a direct definition of the task-shared support pair, explicitly factorizing concatenated full-size matrices is computationally expensive and undermines the efficiency advantage of LoRA. Since each LoRA update is intrinsically low-rank: $\Delta\mathbf{W}_t = \mathbf{B}_t\mathbf{A}_t$, we have $\mathrm{rank}(\Delta\mathbf{W}_t) \le r$. Therefore, the shared support subspaces can be recovered from compact low-rank factors rather than from the full matrices.

Table 1: Asymptotic preprocessing complexity for constructing the merging space of one layer before applying a downstream merge operator. We assume square weight matrices of size $n \times n$, T tasks, and LoRA rank r with $Tr \ll n$.

Method	Construction object	Complexity
Baseline	Directly merge $\Delta\mathbf{W}_t = \mathbf{B}_t\mathbf{A}_t$	$\mathcal{O}(n^2Tr)$
KnOTS	SVD on $[\Delta\mathbf{W}_1, \dots, \Delta\mathbf{W}_T]$	$\mathcal{O}(n^3T^2)$
Core Space	SVD on stacked low-rank factors	$\mathcal{O}(n^2Tr + T^3r^3)$
PGSB	Two full-space SVDs	$\mathcal{O}(n^3T^2 + n^2Tr + T^3r^3)$
PGSB (efficient)	Two low-rank SVDs	$\mathcal{O}(n^2Tr + T^3r^3)$

For each task t, let $\Delta\mathbf{W}_t = \mathbf{U}_t\mathbf{\Sigma}_t\mathbf{V}_t^\top$ be its compact SVD, where $\mathbf{U}_t$ and $\mathbf{V}_t$ contain only the nonzero singular directions. We define

$$\mathbf{L} = [\mathbf{U}_1\mathbf{\Sigma}_1, \dots, \mathbf{U}_T\mathbf{\Sigma}_T], \qquad \mathbf{R} = [\mathbf{V}_1\mathbf{\Sigma}_1, \dots, \mathbf{V}_T\mathbf{\Sigma}_T]. \tag{6}$$

Since each block has at most r columns, $\mathbf{L}$ and $\mathbf{R}$ have only Tr columns, making the decomposition substantially cheaper than SVD on concatenated full updates. We then perform SVD on these low-rank concatenations:

$$\mathbf{L} = \bar{\mathbf{U}}_L\bar{\mathbf{\Sigma}}_L\widetilde{\mathbf{V}}_L^\top, \qquad \mathbf{R} = \bar{\mathbf{U}}_R\bar{\mathbf{\Sigma}}_R\widetilde{\mathbf{V}}_R^\top, \tag{7}$$

and use $\bar{\mathbf{U}}_L$ and $\bar{\mathbf{U}}_R$ as the left and right support matrices. This yields the same support matrices and singular values (i.e., $\mathbf{U} = \bar{\mathbf{U}}_L$, $\mathbf{\Sigma}_L = \begin{bmatrix}\bar{\mathbf{\Sigma}}_L & \mathbf{0}\end{bmatrix}$, $\bar{\mathbf{U}}_R = \mathbf{U}_R$, and $\mathbf{\Sigma}_R = \begin{bmatrix}\bar{\mathbf{\Sigma}}_R & \mathbf{0}\end{bmatrix}$) as the full-update formulation, while avoiding SVD on large concatenated matrices. As shown in Table 1, this reduces the computational complexity while preserving the original task-shared support construction; the equivalence proof and detailed complexity derivation are provided in Appendices B and C, respectively.

3.4 Pretrained-Coupled Coordinate Refinement

TSE identifies a shared support pair $\mathbf{U}_L$ and $\mathbf{U}_R$ for LoRA updates. These support matrices specify a common subspace in which task updates can be represented, but they do not uniquely determine the coordinate system inside this subspace. Indeed, the same subspace can admit many equivalent bases. For any orthogonal matrices $\mathbf{R}_L$ and $\mathbf{R}_R$, $\mathbf{U}_L\mathbf{R}_L$ and $\mathbf{U}_R\mathbf{R}_R$ span exactly the same subspaces as $\mathbf{U}_L$ and $\mathbf{U}_R$, and thus are equally valid support matrices, yet they produce different coordinate representations for the same update. This distinction is crucial because practical merge rules, such as TIES [62], are coordinate-sensitive: magnitude trimming, sign selection, and conflict resolution are performed entry by entry and therefore depend on the chosen basis.

Therefore, after obtaining the task-shared supports, we further seek orthogonal transformations that preserve the shared subspaces while selecting a more suitable coordinate system for merging. Since every LoRA update is learned as an offset to the same pretrained weight $\mathbf{W}_0$, the merge coordinate should also respect how these shared directions are arranged in the pretrained model. To measure this relation, we project the pretrained weight onto the task-shared supports:

$$\mathbf{G} = \mathbf{U}_L^\top \mathbf{W}_0 \mathbf{U}_R. \tag{8}$$

The matrix $\mathbf{G}$ describes how the pretrained weight is expressed under the task-shared coordinates. In particular, its entries reflect the coupling between shared directions induced by $\mathbf{W}_0$. Meanwhile, the singular values $\mathbf{\Sigma}_L$ and $\mathbf{\Sigma}_R$ quantify the aggregate importance of the corresponding shared directions across task updates themselves. We therefore use them to rescale the pretrained coupling:

$$\mathbf{T} = \mathbf{\Sigma}_L \mathbf{G} \mathbf{\Sigma}_R. \tag{9}$$

Thus, $\mathbf{T}$ represents the pretrained-weight coupling after taking into account the relative importance of task-shared directions. However, $\mathbf{T}$ is generally non-diagonal, which means that the current task-shared coordinates do not separate the dominant interactions well. For example, if one row of $\mathbf{T}$ contains several large off-diagonal entries, then one shared direction is strongly coupled with multiple directions on the other side. In this case, the effect of this direction is distributed across multiple coordinate entries, rather than being concentrated in a single dominant coordinate. This is undesirable for coordinate-wise merge rules such as TIES, where magnitude trimming, sign selection, and conflict resolution are performed independently for each entry. Such independent entry-wise decisions may break correlated interactions that should ideally be treated together.

To reduce this coordinate mixing, we compute the SVD of $\mathbf{T}$:

$$\mathbf{T} = \mathbf{O}_L \mathbf{\Sigma}_T \mathbf{O}_R^\top. \tag{10}$$

After this rotation, the dominant interactions are concentrated on diagonal coordinates, making each coordinate more clearly associated with one major pretrained-compatible interaction.

We combine the task-shared supports from TSE with the pretrained-coupled rotations:

$$\mathbf{Q}_L = \mathbf{U}_L \mathbf{O}_L, \qquad \mathbf{Q}_R = \mathbf{U}_R \mathbf{O}_R. \tag{11}$$

The resulting bases $\mathbf{Q}_L$ and $\mathbf{Q}_R$ preserve the original task-shared subspaces, while providing a coordinate system refined by the pretrained weight geometry.

3.5 Merging and Reconstruction

After obtaining the refined bases $\mathbf{Q}_L$ and $\mathbf{Q}_R$, PGSB first maps all task-specific LoRA updates into the same pretrained-compatible coordinate space. This gives a set of task matrices $\{\mathbf{C}_t\}_{t=1}^T$:

$$\mathbf{C}_t = \mathbf{Q}_L^\top \Delta\mathbf{W}_t \mathbf{Q}_R, \tag{12}$$

where each $\mathbf{C}_t$ represents the corresponding update under the shared task subspace and the refined merge coordinate. Any merge rule $\mathcal{M}$ can then be applied in this coordinate space:

$$\mathbf{C}^\star = \mathcal{M}(\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_T). \tag{13}$$

Finally, the merged coordinate matrix is mapped back to the original parameter space:

$$\Delta\mathbf{W}^\star = \mathbf{Q}_L \mathbf{C}^\star \mathbf{Q}_R^\top. \tag{14}$$

Thus, PGSB performs merging in a basis that retains the task-shared subspace structure while adapting its coordinates to the pretrained weight geometry, and then reconstructs the merged LoRA update for deployment. The specific algorithm of PGSB is deferred to Appendix D.

4 Experiment

4.1 Experimental Setting

Models. We evaluate PGSB on both vision and language model merging benchmarks. For vision experiments, we use CLIP [46] with ViT-B/32 and ViT-L/14 backbones [12] and merge LoRA adapters fine-tuned on eight image classification tasks. For language experiments, we use Llama-3-8B [1] and merge LoRA adapters fine-tuned on six Natural Language Inference (NLI) tasks. We also use Llama-3-8B [1] and merge LoRA adapters fine-tuned on eight Commonsense Reasoning tasks. All LoRA checkpoints and evaluation protocols follow prior alignment-based LoRA merging studies [55, 42].

Datasets. The vision benchmark covers diverse recognition scenarios, including texture classification, digit recognition, scene understanding, and remote sensing, with representative datasets such as DTD [8], EuroSAT [18], MNIST [28], and SUN397 [61]. The language benchmark covers standard NLI, entailment-style tasks and reasoning tasks, including MNLI [59], SciTail [24], and BoolQ [9].

Baselines. We compare PGSB with representative merge spaces, including KnOTS [55] and Core Space [42]. We also report the **Baseline**, where each merge operator is directly applied to the reconstructed LoRA updates $\Delta \mathbf{W}_t = \mathbf{B}_t \mathbf{A}_t$ without any space transformation. For all merge spaces, we use the same merge operators, including TA [22], TIES [62], DARE-TIES [64], TSV [16], CART [7], and Iso-C [33]. This setup isolates the effect of merge space: PGSB only changes coordinate space where LoRA updates are represented before applying downstream merge operator.

Table 2: Normalized accuracies of merged models on the visual tasks with ViT-B/32.

Method	Space	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC	SUN397	SVHN	Avg
Abs. Accuracy		74.00	58.30	99.00	92.70	99.30	88.40	64.50	96.20	-
TA		83.62	76.00	50.77	44.12	55.39	72.36	97.05	41.31	65.08
TIES	Baseline	82.24	73.26	51.44	37.92	56.28	69.88	96.95	43.89	63.98
	KnOTS	82.58	73.45	48.26	41.83	57.40	70.66	96.58	45.60	64.54
	Core	84.29	79.74	54.10	49.67	64.50	72.72	97.44	44.29	68.34
	PGSB	83.37	76.37	58.40	44.29	82.18	72.08	95.41	66.17	**72.28**
DARE-TIES	Baseline	82.35	73.26	51.44	37.92	56.26	69.96	97.01	43.81	64.00
	KnOTS	82.22	75.36	51.18	40.71	56.19	71.48	97.93	39.92	64.37
	Core	85.13	79.56	57.69	46.59	65.91	72.96	97.55	46.82	69.03
	PGSB	83.45	75.27	58.03	43.01	83.43	71.54	95.13	67.97	**72.23**
TSV	Baseline	83.40	75.45	52.49	44.99	59.50	73.26	96.33	49.31	66.84
	KnOTS	81.97	75.09	51.96	41.62	53.93	71.68	97.90	40.29	64.30
	Core	83.96	75.36	53.05	45.35	58.27	73.01	97.64	45.27	66.49
	PGSB	84.04	78.19	54.96	47.32	61.23	74.03	97.59	45.39	**67.84**
CART	Baseline	83.96	77.64	52.04	48.98	56.04	73.76	97.05	43.45	66.62
	KnOTS	83.40	73.63	50.69	47.98	60.57	72.79	96.83	49.12	66.88
	Core	84.04	77.00	50.77	48.30	60.12	73.13	96.46	46.17	67.00
	PGSB	83.20	76.00	47.48	49.62	62.89	72.92	96.11	49.56	**67.22**
TIES$_{+\text{Iso-C}}$	Baseline	81.88	75.18	60.08	38.76	63.26	69.88	98.95	40.34	66.04
	KnOTS	80.77	80.56	54.88	63.95	66.38	77.26	98.81	48.95	71.45
	Core	84.46	84.03	57.69	75.46	68.93	81.83	99.47	50.42	75.29
	PGSB	85.13	84.94	59.63	76.66	73.52	81.90	98.57	51.76	**76.51**
DARE-TIES$_{+\text{Iso-C}}$	Baseline	82.14	76.18	55.85	39.71	61.00	70.05	98.71	40.47	65.51
	KnOTS	76.38	81.75	45.98	65.35	68.50	78.07	97.71	46.57	70.04
	Core	83.40	82.66	56.86	80.24	69.37	82.26	99.13	53.18	75.89
	PGSB	85.30	85.03	59.60	76.79	73.62	81.97	98.73	51.74	**76.60**
TSV$_{+\text{Iso-C}}$	Baseline	78.04	80.38	61.92	66.54	65.99	79.83	98.37	45.24	72.04
	KnOTS	80.69	83.66	58.36	74.76	67.89	79.80	98.55	50.10	74.23
	Core	82.81	85.03	52.45	84.25	71.01	84.32	99.01	53.55	76.55
	PGSB	84.54	85.40	53.61	84.09	73.36	85.02	98.49	53.75	**77.28**
CART$_{+\text{Iso-C}}$	Baseline	81.59	77.83	54.4	44.76	52.81	73.06	98.91	36.88	65.03
	KnOTS	82.05	79.10	55.93	55.24	56.56	76.10	99.70	41.45	68.27
	Core	84.54	84.21	75.13	57.13	69.29	83.03	99.43	51.24	75.82
	PGSB	84.38	85.49	52.75	82.17	74.22	83.42	98.45	54.27	**76.89**
Iso-C	Baseline	80.41	82.48	53.65	74.12	69.91	79.99	98.62	49.61	73.60
	KnOTS	80.62	83.30	53.61	75.67	70.41	82.09	98.66	50.82	74.40
	Core	82.95	84.94	38.76	83.15	77.39	82.09	98.18	57.47	75.62
	PGSB	84.38	85.40	50.88	82.47	74.02	83.67	98.53	54.67	**76.75**

Table 3: Normalized accuracies of merged models on the visual tasks with ViT-L/14.

Method	Space	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC	SUN397	SVHN	Avg
Abs. Accuracy		99.77	70.05	98.59	97.20	99.53	95.70	79.60	97.72	-
TA		80.07	79.73	62.10	59.96	82.17	79.53	86.71	64.78	74.38
TIES	Baseline	79.54	78.66	63.11	60.42	84.12	78.94	87.03	69.60	75.18
	KnOTS	80.35	79.57	62.06	62.95	85.43	79.81	87.57	70.99	76.09
	Core	81.91	79.88	63.07	68.95	87.48	81.77	88.50	73.76	78.17
	PGSB	83.15	80.18	65.48	69.37	87.85	82.29	87.99	77.52	**79.23**
DARE-TIES	Baseline	79.56	78.66	62.96	60.39	84.11	78.95	86.99	69.61	75.15
	KnOTS	80.43	79.65	62.02	63.09	85.29	79.81	87.56	71.04	76.11
	Core	82.27	79.95	62.21	68.15	88.19	82.14	88.82	74.33	78.26
	PGSB	82.34	80.11	65.44	69.04	89.82	82.46	87.87	77.21	**79.29**
TSV	Baseline	82.10	79.88	64.05	66.79	87.31	82.09	87.56	71.67	77.68
	KnOTS	80.21	79.95	64.43	60.21	83.75	79.32	86.84	65.57	75.03
	Core	82.13	79.88	63.97	66.85	87.26	82.10	87.56	71.60	77.67
	PGSB	82.50	80.95	64.93	66.88	87.65	83.32	87.63	72.87	**78.34**
CART	Baseline	91.20	89.60	76.37	81.66	74.10	93.28	86.87	66.01	82.39
	KnOTS	82.27	81.17	62.43	67.66	71.38	83.16	88.24	69.26	75.70
	Core	86.02	86.33	72.39	82.82	91.03	85.93	88.46	72.67	83.21
	PGSB	92.97	88.40	75.10	80.05	90.63	92.90	86.11	66.48	**84.08**
Iso-C	Baseline	86.96	85.42	79.94	78.40	92.19	88.21	87.22	69.09	83.43
	KnOTS	88.27	88.61	77.24	82.45	87.74	89.43	88.55	68.63	83.86
	Core	91.48	90.74	77.87	85.83	87.40	91.29	89.66	70.75	85.63
	PGSB	91.45	91.66	77.76	85.94	87.38	91.33	89.69	71.33	**85.82**

Table 4: Normalized accuracies of merged models on the NLI tasks for Llama 3 8B.

Method	Space	SNLI	MNLI	SICK	QNLI	RTE	SCITAIL	Avg
Abs. Accuracy	–	92.50	90.31	91.58	94.49	89.86	96.52	–
TA	–	93.58	95.28	88.07	68.76	100.00	96.73	90.40
TIES	Baseline	95.17	96.21	84.20	74.21	100.00	96.73	91.09
	KnOTS	92.67	94.82	91.03	78.93	101.61	97.22	92.72
	Core	91.94	91.12	96.68	79.27	100.81	97.37	92.86
	PGSB	97.19	97.02	87.35	87.54	95.28	97.97	**93.73**
DARE-TIES	Baseline	94.83	96.47	83.44	74.84	98.39	96.00	90.66
	KnOTS	92.95	95.53	90.70	76.58	100.00	96.98	92.12
	Core	92.38	93.02	93.59	84.86	100.00	97.39	93.54
	PGSB	96.97	96.93	88.03	86.20	99.19	97.47	**94.13**
TSV	Baseline	93.95	96.05	82.26	80.00	99.19	96.88	91.39
	KnOTS	94.56	91.61	92.99	69.58	102.42	97.56	91.45
	Core	95.86	95.71	89.27	83.94	102.42	97.86	94.18
	PGSB	96.12	96.21	89.43	84.11	102.01	98.02	**94.31**
Iso-C	Baseline	55.00	39.04	76.54	55.90	46.77	69.25	57.08
	KnOTS	85.28	52.86	89.43	54.90	75.00	77.73	72.53
	Core	91.54	90.10	87.87	75.85	99.19	97.42	90.33
	PGSB	90.80	90.88	86.69	78.72	101.68	98.52	**91.22**

All hyper-parameters are selected on validation sets, and test labels are never used for model selection. Implementation details, compute resources, baselines, datasets, and hyperparameter search protocols are provided in Appendix E.

4.2 Visual Tasks

Table 2 reports the normalized accuracies on the visual benchmarks using ViT-B/32 [12]. Across different merge operators, PGSB consistently provides a stronger merging space than Baseline, KnOTS, and Core Space. The improvements are especially pronounced for coordinate-sensitive operators such as TIES and DARE-TIES, whose trimming and conflict-resolution steps directly depend on the chosen coordinate system. We further evaluate PGSB with a stronger visual backbone, ViT-L/14, and the results are shown in Table 3. PGSB again achieves the best average performance across the evaluated settings, demonstrating that the proposed pretrained-compatible merging space remains effective across different visual backbones.

4.3 Natural Language Inference

Table 4 reports the normalized accuracies on the NLI benchmarks using Llama 3 8B [1]. PGSB achieves the best average performance across all evaluated merge operators when compared with

Baseline, KnOTS [55], and Core Space [42]. These results indicate that the proposed merging space is not specific to visual encoders, but also transfers effectively to large language models. They further support the generality of representing LoRA updates in pretrained-compatible coordinates before applying downstream merge operators.

4.4 Commensense Reasoning

Table 5 reports the normalized accuracies on the Commonsense Reasoning benchmarks using Llama 3 8B [1]. We use TIES as the merge operator and compare PGSB against Baseline, KnOTS [55], and Core Space [42]. PGSB achieves the best average performance of 97.58%, outperforming all competing merging spaces. These results further demonstrate that PGSB effectively preserves reasoning capabilities during model merging.

Table 5: Normalized accuracies of merged models on the Commonsense Reasoning tasks for Llama 3 8B with TIES merging.

Space	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	SIQA	WinoG.	Avg
Abs. Accuracy	80.38	92.89	89.51	96.28	86.2	90.32	80.6	88.08	–
Baseline	99.68	99.23	97.56	92.8	96.29	96.75	98.1	97.13	97.19
KnOTS	102.34	100.3	98.14	90.11	96.98	97.11	97.02	95.96	97.25
Core	100.42	99.18	95.66	92.47	96.52	95.3	96.26	95.16	96.37
PGSB	100.21	99.55	97.66	93.72	97.68	97.11	97.46	97.22	**97.58**

4.5 Ablation Study

We study the necessity of the two stages of PGSB through two ablation variants. Both variants start from the task-shared support pair $\mathbf{U}_L, \mathbf{U}_R$ extracted by TSE. *TSE-only* removes PCR and directly merges task updates in the original support coordinates:

$$\mathbf{D}_t = \mathbf{U}_L^\top \Delta\mathbf{W}_t \mathbf{U}_R, \qquad \mathbf{D}^\star = \mathcal{M}(\mathbf{D}_1, \ldots, \mathbf{D}_T). \tag{15}$$

The merged update is then reconstructed as $\Delta\mathbf{W}^\star = \mathbf{U}_L \mathbf{D}^\star \mathbf{U}_R^\top$. *No-weighted* keeps PCR but removes the singular-value weighting. Instead of decomposing the weighted pretrained coupling $\mathbf{T} = \mathbf{\Sigma}_L(\mathbf{U}_L^\top \mathbf{W}_0 \mathbf{U}_R)\mathbf{\Sigma}_R$, it directly decomposes the unweighted coupling $\mathbf{G} = \mathbf{U}_L^\top \mathbf{W}_0 \mathbf{U}_R = \widetilde{\mathbf{O}}_L \mathbf{\Sigma}_G \widetilde{\mathbf{O}}_R^\top$, and constructs $\mathbf{Q}_L = \mathbf{U}_L \widetilde{\mathbf{O}}_L$ and $\mathbf{Q}_R = \mathbf{U}_R \widetilde{\mathbf{O}}_R$.

Table 6 shows that *TSE-only* reaches an average score of 68.46, outperforming the Baseline under TIES merging. This indicates that TSE provides an effective task-shared support pair for representing LoRA updates. Full PGSB further improves the score to 72.28, showing that PCR provides additional gains beyond support construction by refining the coordinates inside the same support. In contrast, *No-weighted* drops to 62.86, suggesting that directly decomposing $\mathbf{G}$ is insufficient for obtaining merge-friendly coordinates. Weighting $\mathbf{G}$ by both $\mathbf{\Sigma}_L$ and $\mathbf{\Sigma}_R$ is therefore important for emphasizing task-shared directions, reducing harmful coordinate mixing, and improving coordinate-sensitive merging.

Table 6: Ablations on the visual tasks with TIES merging. Each entry reports normalized accuracy on the corresponding dataset. Baseline directly applies TIES to the LoRA updates without any other operations.

Variant	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC	SUN397	SVHN	Avg
PGSB	83.37	76.37	58.40	44.29	82.18	72.08	95.41	66.17	72.28
Baseline	82.24	73.26	51.44	37.92	56.28	69.88	96.95	43.89	63.98
TSE-only	84.88	77.37	54.55	46.81	65.46	72.31	96.69	49.64	68.46
No-weighted	81.76	74.63	51.37	36.91	53.58	69.54	98.04	37.06	62.86

4.6 Extensibility Study

We next examine whether PCR is specific to the support pair constructed by PGSB, or whether it can serve as a general coordinate refinement module for existing alignment-based merge spaces. To this end, we keep the shared spaces produced by KnOTS [55] and Core Space [42] unchanged, and apply

PCR on top of them to obtain *KnOTS + PCR* and *Core Space + PCR*. These hybrid variants preserve the original support spaces of prior methods, but replace their inherited coordinates with pretrained-compatible coordinates refined by PCR. Implementation details for applying PCR to prior merge spaces are provided in Appendix A.3.

Table 7: Extensibility experiments on the visual tasks with TIES merging. Each entry reports normalized accuracy on the corresponding dataset.

Variant	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC	SUN397	SVHN	Avg
PGSB	83.37	76.37	58.40	44.29	82.18	72.08	95.41	66.17	72.28
KnOTS	82.58	73.45	48.26	41.83	57.40	70.66	96.58	45.60	64.54
KnOTS + PCR	83.00	72.26	56.12	37.34	70.88	70.41	96.66	61.28	68.49
Core Space	84.29	79.74	54.10	49.67	64.50	72.72	97.44	44.29	68.34
Core Space + PCR	83.79	77.00	50.24	47.75	77.04	68.07	96.19	57.93	69.75

Table 7 shows that PCR consistently improves prior alignment spaces. Specifically, *KnOTS + PCR* increases the average score from 64.54 to 68.49, while *Core Space + PCR* improves it from 68.34 to 69.75. These results indicate that pretrained-compatible coordinate refinement is not tied to PGSB's own support construction, but can also enhance existing merge spaces. Nevertheless, neither hybrid reaches the performance of full PGSB, which achieves an average score of 72.28. This suggests that PCR is broadly extensible, while the strongest results are obtained when it is combined with the task-shared support constructed by PGSB.

4.7 Pretrained Transfer Preservation Study

We further evaluate whether the merged model preserves the transfer ability inherited from the pretrained backbone. The motivation is that LoRA updates are learned as task-specific offsets to the same pretrained weight $\mathbf{W}_0$. After merging, the final model is obtained by adding the merged update back to $\mathbf{W}_0$. Therefore, an effective merge space should not only combine knowledge from the source tasks, but also avoid unnecessarily distorting the pretrained representation, so that the model can still generalize to tasks outside the merge set.

Concretely, we use the same eight-task visual merging setting as in Sec. 4.2. For each merge space, including Baseline, KnOTS, Core Space, and PGSB, we merge the same eight task-specific LoRA updates using TIES [62]. The resulting merged update is then added to the CLIP ViT-B/32 pretrained weight [46, 12]. Without any additional training, fine-tuning, or adaptation, we directly evaluate the merged models on seven additional visual datasets that are not used for training the source LoRA adapters or for constructing the merged update. These datasets cover diverse visual domains, including generic object recognition, fine-grained animal classification, food recognition, flower recognition, and natural image classification, with representative datasets such as CIFAR100 [27], Caltech101 [14], and Food101 [4].

Table 8 reports normalized accuracies, using the original pretrained model as the reference. PGSB achieves the highest average normalized accuracy of 98.38, remaining closest to the pretrained model among all merged models. Compared with Baseline, KnOTS, and Core Space, PGSB better preserves performance on tasks outside the merge set. These results show that PGSB not only improves multi-task merging on the source tasks, but also better maintains the general representation ability of the pretrained model after merging.

Table 8: Normalized accuracies of merged models on additional visual tasks after TIES merging. *Abs. Accuracy* reports the absolute accuracy of the pretrained model, which is used as the normalization reference for each visual task.

Space	CIFAR10	CIFAR100	Pets	Flowers102	Food101	Caltech101	Cats vs Dogs	Avg
Abs. Accuracy	88.86	64.35	86.24	65.98	84.43	79.90	99.62	81.34
Baseline	97.60	78.57	95.63	91.77	97.95	96.72	99.99	94.03
KnOTS	81.87	30.27	85.74	61.90	89.92	85.47	99.94	76.44
Core Space	99.07	90.58	97.84	97.01	98.65	99.50	99.96	97.52
PGSB	99.32	94.65	98.92	97.71	99.04	98.97	100.00	**98.38**

4.8 Rank Robustness Study

We further investigate the robustness of PGSB under diverse LoRA rank configurations on the eight visual tasks using ViT-B/32. Specifically, we consider three settings: (1) homogeneous LoRA ranks of $r \in \{4, 64, 256\}$, covering a wide range of adaptation capacities; (2) heterogeneous ranks, where Cars, DTD, EuroSAT, and GTSRB use $r = 16$, while MNIST, RESISC45, SUN397, and SVHN use $r = 64$; and (3) adaptive ranks, where all eight task adapters are trained using AdaLoRA [68] with a target rank of $r = 16$. For all settings, we apply TIES [62] as the merge operator and compare PGSB against Baseline, KnOTS [55], and Core Space [42].

Table 9 reports the normalized accuracies under these configurations. PGSB achieves the highest average performance in all five settings, demonstrating its effectiveness across homogeneous, heterogeneous, and adaptively allocated ranks. These results demonstrate that PGSB does not rely on a fixed or uniform LoRA rank and remains effective across diverse rank configurations.

Table 9: Normalized accuracies of merged models under different LoRA rank configurations on the eight visual tasks using ViT-B/32 with TIES merging.

Setting	Space	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Avg
$r = 4$	*Abs. Accuracy*	68.57	68.83	98.19	96.65	99.49	91.60	71.34	96.05	–
	Baseline	89.59	72.26	60.62	63.59	82.75	80.64	93.68	79.71	77.85
	KnOTS	92.74	75.50	61.60	68.37	82.13	78.06	93.56	76.39	78.54
	Core	92.38	76.66	66.09	66.59	79.32	77.54	92.98	72.31	77.98
	PGSB	92.81	75.81	68.58	65.45	86.83	81.13	93.33	81.66	**80.70**
$r = 64$	*Abs. Accuracy*	74.63	75.32	98.56	97.87	99.50	93.38	73.48	96.64	–
	Baseline	82.94	69.28	69.00	73.73	90.85	81.54	90.30	89.01	80.83
	KnOTS	86.36	70.62	68.96	76.24	90.35	79.69	89.56	88.17	81.24
	Core	87.58	78.67	80.72	79.04	86.63	82.95	91.05	79.46	83.26
	PGSB	86.33	82.49	67.91	83.08	89.20	83.78	91.26	83.36	**83.43**
$r = 256$	*Abs. Accuracy*	77.81	76.70	99.04	98.39	99.65	94.54	73.97	96.87	–
	Baseline	79.19	64.36	74.20	71.38	90.69	77.25	89.03	85.62	78.96
	KnOTS	81.95	63.31	66.57	65.05	90.25	75.65	89.07	86.51	77.30
	Core	83.44	68.86	74.91	70.81	88.96	79.13	90.51	80.09	79.59
	PGSB	82.32	72.19	72.70	71.94	88.56	80.42	90.44	80.51	**79.89**
Heterogeneous	*Abs. Accuracy*	74.00	58.30	99.00	92.70	99.50	93.38	73.48	96.64	–
	Baseline	86.31	74.72	69.47	40.66	89.00	80.16	89.41	78.69	76.05
	KnOTS	86.97	76.00	64.05	41.92	87.15	78.63	89.23	79.85	75.47
	Core	85.71	77.73	65.73	43.43	88.96	83.05	90.81	76.46	76.49
	PGSB	87.49	76.18	69.32	44.97	91.86	85.30	89.92	82.23	**78.41**
$\text{AdaLoRA}_{r=16}$	*Abs. Accuracy*	73.84	58.56	97.70	95.34	99.27	89.79	69.29	95.49	–
	Baseline	94.56	88.83	58.95	70.87	81.28	82.46	95.05	80.10	81.51
	KnOTS	95.31	92.33	65.66	71.60	82.35	82.36	95.03	78.70	82.92
	Core	95.43	91.46	71.19	80.62	80.19	87.45	95.80	78.73	85.11
	PGSB	96.74	91.35	73.01	78.61	84.57	88.99	96.15	78.84	**86.03**

5 Conclusion

In this paper, we studied LoRA model merging as a scalable route toward building unified multi-task models from parameter-efficient adaptations. We identified two key challenges that distinguish LoRA merging from conventional merging of fully fine-tuned models: extracting shared task directions from diverse low-rank updates, and representing these directions in coordinates compatible with the pretrained weight geometry. To address these challenges, we proposed PGSB, a two-stage framework consisting of Task-shared Subspace Extraction and Pretrained-coupled Coordinate Refinement. PGSB first constructs a shared subspace from concatenated LoRA updates, and then refines the merge coordinates using the geometric structure of the pretrained weights. This yields a pretrained-compatible basis that better supports coordinate-sensitive merging and improves multi-task adaptation. Extensive experiments on vision and language benchmarks demonstrate that PGSB consistently outperforms strong baselines. Further analyses confirm the necessity of both shared support construction and pretrained-guided coordinate refinement, show that PGSB better preserves the transfer ability of the pretrained model, and demonstrate the robustness of PGSB across diverse rank configurations.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China under Grants U24A20322 and 62576094. This research work was also supported by the Big Data Computing Center of Southeast University.

References

[1] AI@Meta. Llama 3 model card, 2024.

[2] Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, 2025.

[3] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[5] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. *Conference on Empirical Methods in Natural Language Processing*, 2015.

[6] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 2017.

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

[8] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014.

[9] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[10] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[13] Jeremy Elson, John (JD) Douceur, Jon Howell, and Jared Saul. Asirra: A captcha that exploits interest-aligned manual image categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS)*. Association for Computing Machinery, Inc., October 2007.

[14] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178, 2004.

[15] Marc Fischer, Alexander Bartler, and Bin Yang. Prompt tuning for parameter-efficient medical image segmentation. *Medical Image Analysis*, 91:103024, 2024.

[16] Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[17] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

[18] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[19] 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 *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[20] J. E. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2021.

[21] Hugging Face. https://huggingface.co, 2024.

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

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

[24] Tushar Khot, Ashish Sabharwal, and Peter Clark. SciTail: A textual entailment dataset from science question answering. In *AAAI Conference on Artificial Intelligence*, 2018.

[25] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[26] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D Object representations for fine-grained categorization. In *IEEE/CVF International Conference on Computer Vision Workshops*, 2013.

[27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[30] Muqing Liu, Chongjie Si, and Yuheng Jia. Flexlora: Entropy-guided flexible low-rank adaptation. *arXiv:2601.22905*, 2026.

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

[32] Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 15(1):654, 2024.

[33] 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 *International Conference on Machine Learning*, 2025.

[34] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, 2014.

[35] Imad Eddine Marouf, Subhankar Roy, Enzo Tartaglione, and Stéphane Lathuilière. Weighted ensemble models are strong continual learners. *arXiv preprint arXiv: 2312.08977*, 2023.

[36] Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems*, 2021.

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

[38] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

[39] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Neural Information Processing Systems Workshops*, 2011.

[40] Maria-Elena Nilsback and Andrew Zisserman. Automated Flower Classification over a Large Number of Classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, December 2008.

[41] Aniello Panariello, Emanuele Frascaroli, Pietro Buzzega, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. Modular embedding recomposition for incremental learning. *British Machine Vision Conference*, 2025.

[42] Aniello Panariello, Daniel Marczak, Simone Magistri, Angelo Porrello, Bartłomiej Twardowski, Andrew D. Bagdanov, Simone Calderara, and Joost van de Weijer. Accurate and efficient low-rank model merging in core space. *arXiv preprint arXiv:2509.17786*, 2026.

[43] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3498–3505, 2012.

[44] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

[45] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

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

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

[48] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[49] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

[50] Zhengxiang Shi and Aldo Lipani. Dept: Decomposed prompt tuning for parameter-efficient fine-tuning. *arXiv preprint arXiv:2309.05173*, 2023.

[51] Chongjie Si, Kangtao Lv, Jingjing Jiang, Yadao Wang, Yongwei Wang, Xiaokang Yang, Wenbo Su, Bo Zheng, and Wei Shen. Nan: A training-free solution to coefficient estimation in model merging. *arXiv preprint arXiv:2505.16148*, 2025.

[52] Chongjie Si, Xuehui Wang, Xue Yang, Zhengqin Xu, Qingyun Li, Jifeng Dai, Yu Qiao, Xiaokang Yang, and Wei Shen. Flora: Low-rank core space for n-dimension. *arXiv preprint arXiv:2405.14739*, 2024.

[53] Chongjie Si, Xiaokang Yang, and Wei Shen. See further for parameter efficient fine-tuning by standing on the shoulders of decomposition. *arXiv preprint arXiv:2407.05417*, 2024.

[54] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *International Joint Conference on Neural Networks*, 2011.

[55] George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. *International Conference on Learning Representations*, 2025.

[56] A. 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 *International Conference on Learning Representations*, 2023.

[57] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[58] Yaqing Wang, Jialin Wu, Tanmaya Dabral, Jiageng Zhang, Geoff Brown, Chun-Ta Lu, Frederick Liu, Yi Liang, Bo Pang, Michael Bendersky, et al. Non-intrusive adaptation: Input-centric parameter-efficient fine-tuning for versatile multimodal modeling. *arXiv preprint arXiv:2310.12100*, 2023.

[59] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

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

[61] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 2016.

[62] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2024.

[63] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *International Conference on Learning Representations*, 2024.

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

[65] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[66] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023.

[67] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2022.

[68] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

A Prior Alignment-Based Methods

Let the LoRA update for task t be

$$\mathbf{\Delta W}_t = \mathbf{B}_t \mathbf{A}_t \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}, \qquad t = 1, \ldots, T, \tag{16}$$

where $\mathbf{B}_t \in \mathbb{R}^{d_{\text{out}} \times r}$ and $\mathbf{A}_t \in \mathbb{R}^{r \times d_{\text{in}}}$ are the low-rank adaptation factors. Existing alignment-based LoRA merging methods can be viewed as attempts to construct a common representation space for the task updates before applying a merge operator. Below, we summarize two representative methods and discuss their relation to the two questions considered in PGSB: how to extract shared task directions, and how to choose pretrained-compatible merge coordinates.

A.1 KnOTS.

KnOTS aligns LoRA updates by reconstructing the full update matrices and extracting a shared basis from their concatenation. For a given layer, it computes

$$[\mathbf{\Delta W}_1, \mathbf{\Delta W}_2, \ldots, \mathbf{\Delta W}_T] = \mathbf{U\Sigma} \left[\mathbf{V}^{(1)}, \mathbf{V}^{(2)}, \cdots, \mathbf{V}^{(T)}\right]^\top. \tag{17}$$

Here, $\mathbf{U}$ provides a common left basis shared by all task updates, Σ contains the corresponding singular values, and each $\mathbf{V}^{(t)}$ gives the task-specific coordinates under this shared basis. A merge operator is then applied to $\{\mathbf{V}^{(t)}\}_{t=1}^{T}$ to obtain $\mathbf{V}^{(\text{merge})}$, and the merged update is reconstructed as

$$\mathbf{\Delta W}^{(\text{merge})} = \mathbf{U\Sigma}\big(\mathbf{V}^{(\text{merge})}\big)^\top. \tag{18}$$

This formulation shows that KnOTS addresses part of the first question: it extracts a shared set of directions from concatenated task updates and performs merging in the resulting aligned space. However, the alignment is essentially one-sided. The shared structure is made explicit through the common left basis U, while the right-side structure remains absorbed in the task-specific coordinate matrices $V^{(t)}$. Thus, KnOTS does not explicitly construct a shared left-right support pair for LoRA updates. Moreover, its merge coordinates are inherited directly from the SVD of task updates and are not further refined using the pretrained weight geometry. Finally, because KnOTS performs SVD on concatenated full update matrices, its preprocessing cost grows with the ambient weight dimension, which weakens the efficiency advantage of LoRA.

A.2 Core Space.

Core Space avoids full-space SVD by constructing shared reference spaces directly from the low-rank LoRA factors. Specifically, it computes

$$[\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_T] = \mathbf{U}_B^{\text{ref}} \mathbf{\Sigma}_B \mathbf{V}_B^\top, \tag{19}$$

and

$$\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_T \end{bmatrix} = \mathbf{U}_A \mathbf{\Sigma}_A \big(\mathbf{V}_A^{\text{ref}}\big)^\top. \tag{20}$$

Here, $\mathbf{U}_B^{\text{ref}}$ spans a shared left reference space for the $\mathbf{B}_t$ factors, while $\mathbf{V}_A^{\text{ref}}$ spans a shared right reference space for the $\mathbf{A}_t$ factors. Each task update is represented by a compact core matrix

$$\mathbf{M}_t = \big(\mathbf{U}_B^{\text{ref}\top} \mathbf{B}_t\big)\big(\mathbf{A}_t \mathbf{V}_A^{\text{ref}}\big), \tag{21}$$

and merging is performed on $\{\mathbf{M}_t\}_{t=1}^{T}$. Denoting the merged core matrix by $\mathbf{M}^{(\text{merge})}$, the final merged update is reconstructed as

$$\mathbf{\Delta W}^{(\text{merge})} = \mathbf{U}_B^{\text{ref}} \mathbf{M}^{(\text{merge})} \mathbf{V}_A^{\text{ref}\top}. \tag{22}$$

Compared with KnOTS, Core Space preserves the low-rank regime and provides an efficient way to represent LoRA updates in a shared compact space. In this sense, it also addresses part of the first question by introducing shared left and right reference spaces. However, these two spaces are constructed independently from the B_t and A_t factors, rather than from the aggregate structure of the

reconstructed task updates. Consequently, Core Space does not explicitly characterize the relative strength of shared directions in the full update space. More importantly, as with KnOTS, the merge coordinates are directly induced by the shared-space construction itself and are not coupled with the common pretrained weight $\mathbf{W}_0$. Therefore, Core Space improves the efficiency of LoRA merging, but it does not explicitly address the second question of how to choose pretrained-compatible coordinates within the shared supports.

Overall, both KnOTS and Core Space demonstrate the importance of aligning LoRA updates before merging. KnOTS emphasizes shared directions through full-update concatenation, while Core Space emphasizes efficient low-rank representation. PGSB builds on this alignment perspective, but separates the problem into two stages: first extracting task-shared left and right support subspaces with singular-value-based directional strength, and then refining the resulting basis using the geometry of the pretrained weight.

A.3 Applying PCR to Prior Methods

We describe how PCR is applied to prior merge spaces in the extensibility study. The key idea is simple: we keep the support construction of each prior method unchanged, and only apply PCR to rotate the coordinates inside the obtained merge space before applying the downstream merge operator.

KnOTS + PCR. For KnOTS, we keep its original one-sided merge form. After truncation, KnOTS represents each task update as

$$\Delta\mathbf{W}_t \approx \mathbf{U}_k\mathbf{\Sigma}_k(\mathbf{V}_k^{(t)})^\top. \tag{23}$$

Thus, the shared KnOTS basis is $\mathbf{U}_k\mathbf{\Sigma}_k$, and the merge operator is applied to the task-specific coordinate blocks $(\mathbf{V}_k^{(t)})^\top$.

To apply PCR without changing this one-sided structure, we refine only the coordinates inside the KnOTS shared basis. We form the pretrained-coupled matrix

$$\mathbf{T}_{\mathrm{K}} = (\mathbf{U}_k\mathbf{\Sigma}_k)^\top\mathbf{W}_0 = \mathbf{\Sigma}_k\mathbf{U}_k^\top\mathbf{W}_0, \tag{24}$$

and compute its compact SVD

$$\mathbf{T}_{\mathrm{K}} = \mathbf{O}_{\mathrm{K}}\mathbf{\Gamma}_{\mathrm{K}}\mathbf{R}_{\mathrm{K}}^\top. \tag{25}$$

The refined KnOTS basis is

$$\widehat{\mathbf{U}}_{\mathrm{K}} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{O}_{\mathrm{K}}, \tag{26}$$

and each task coordinate is correspondingly rotated as

$$\widehat{\mathbf{V}}_{\mathrm{K}}^{(t)\top} = \mathbf{O}_{\mathrm{K}}^\top(\mathbf{V}_k^{(t)})^\top. \tag{27}$$

This transformation preserves each task update before merging since

$$\mathbf{U}_k\mathbf{\Sigma}_k(\mathbf{V}_k^{(t)})^\top = \widehat{\mathbf{U}}_{\mathrm{K}}\widehat{\mathbf{V}}_{\mathrm{K}}^{(t)\top}. \tag{28}$$

We then apply the downstream merge operator to the refined coordinate blocks:

$$\widehat{\mathbf{V}}_{\mathrm{K}}^{\mathrm{merge}\top} = \mathcal{M}\left(\widehat{\mathbf{V}}_{\mathrm{K}}^{(1)\top}, \ldots, \widehat{\mathbf{V}}_{\mathrm{K}}^{(T)\top}\right), \tag{29}$$

and reconstruct the merged update as

$$\Delta\mathbf{W}_{\mathrm{K}}^{\mathrm{merge}} = \widehat{\mathbf{U}}_{\mathrm{K}}\widehat{\mathbf{V}}_{\mathrm{K}}^{\mathrm{merge}\top}. \tag{30}$$

Core Space + PCR. Core Space constructs shared reference spaces from the stacked LoRA factors:

$$[\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_T] = \mathbf{U}_B^{\mathrm{ref}}\mathbf{\Sigma}_B\mathbf{V}_B^\top, \qquad \begin{bmatrix}\mathbf{A}_1\\ \mathbf{A}_2\\ \vdots\\ \mathbf{A}_T\end{bmatrix} = \mathbf{U}_A\mathbf{\Sigma}_A(\mathbf{V}_A^{\mathrm{ref}})^\top. \tag{31}$$

Therefore, its original merge space is given by the left-right support pair

$$\mathbf{U}_B^{\mathrm{ref}}, \qquad \mathbf{V}_A^{\mathrm{ref}}, \tag{32}$$

with strength spectra $\mathbf{\Sigma}_B$ and $\mathbf{\Sigma}_A$. PCR keeps these supports fixed and forms

$$\mathbf{T}_{\mathrm{C}} = \mathbf{\Sigma}_B \left((\mathbf{U}_B^{\mathrm{ref}})^\top \mathbf{W}_0 \mathbf{V}_A^{\mathrm{ref}} \right) \mathbf{\Sigma}_A. \tag{33}$$

We compute

$$\mathbf{T}_{\mathrm{C}} = \mathbf{O}_{\mathrm{C}}^L \mathbf{\Gamma}_{\mathrm{C}} (\mathbf{O}_{\mathrm{C}}^R)^\top, \tag{34}$$

and obtain the PCR-refined Core Space bases:

$$\widehat{\mathbf{U}}_{\mathrm{C}} = \mathbf{U}_B^{\mathrm{ref}} \mathbf{O}_{\mathrm{C}}^L, \qquad \widehat{\mathbf{V}}_{\mathrm{C}} = \mathbf{V}_A^{\mathrm{ref}} \mathbf{O}_{\mathrm{C}}^R. \tag{35}$$

Each task update is represented as

$$\mathbf{C}_t^{\mathrm{C}} = \widehat{\mathbf{U}}_{\mathrm{C}}^\top \Delta \mathbf{W}_t \widehat{\mathbf{V}}_{\mathrm{C}} = \widehat{\mathbf{U}}_{\mathrm{C}}^\top \mathbf{B}_t \mathbf{A}_t \widehat{\mathbf{V}}_{\mathrm{C}}. \tag{36}$$

The merged coordinate matrix is

$$\mathbf{C}_{\mathrm{C}}^{\mathrm{merge}} = \mathcal{M}(\mathbf{C}_1^{\mathrm{C}}, \ldots, \mathbf{C}_T^{\mathrm{C}}), \tag{37}$$

and the final merged update is reconstructed as

$$\Delta \mathbf{W}_{\mathrm{C}}^{\mathrm{merge}} = \widehat{\mathbf{U}}_{\mathrm{C}} \mathbf{C}_{\mathrm{C}}^{\mathrm{merge}} \widehat{\mathbf{V}}_{\mathrm{C}}^\top. \tag{38}$$

In both cases, PCR does not change how the prior method constructs its merge space. It only replaces the original coordinates inside that space with pretrained-coupled coordinates, after which the same merge operator and reconstruction rule are applied.

B Equivalent Low-Rank Construction of Shared Supports

In this section, we show that the task-shared support pair used in TSE can be obtained without explicitly performing SVD on the concatenated full update matrices. For one LoRA-adapted weight matrix, define the full-update concatenations as

$$\mathbf{X} := [\Delta \mathbf{W}_1, \Delta \mathbf{W}_2, \ldots, \Delta \mathbf{W}_T] = \mathbf{U}_L \mathbf{\Sigma}_L \mathbf{V}_L^\top, \tag{39}$$

and

$$\mathbf{Y} := [\Delta \mathbf{W}_1^\top, \Delta \mathbf{W}_2^\top, \ldots, \Delta \mathbf{W}_T^\top] = \mathbf{U}_R \mathbf{\Sigma}_R \mathbf{V}_R^\top. \tag{40}$$

The full-update formulation of TSE uses $\mathbf{U}_L$ and $\mathbf{U}_R$ as the shared left and right supports, while $\mathbf{\Sigma}_L$ and $\mathbf{\Sigma}_R$ characterize the aggregate strength of the corresponding shared directions.

For each task t, let

$$\Delta \mathbf{W}_t = \mathbf{U}_t \mathbf{\Sigma}_t \mathbf{V}_t^\top \tag{41}$$

be its compact SVD. We construct two low-rank concatenations:

$$\mathbf{L} := [\mathbf{U}_1 \mathbf{\Sigma}_1, \mathbf{U}_2 \mathbf{\Sigma}_2, \ldots, \mathbf{U}_T \mathbf{\Sigma}_T], \qquad \mathbf{R} := [\mathbf{V}_1 \mathbf{\Sigma}_1, \mathbf{V}_2 \mathbf{\Sigma}_2, \ldots, \mathbf{V}_T \mathbf{\Sigma}_T]. \tag{42}$$

We now show that the SVDs of $\mathbf{L}$ and $\mathbf{R}$ recover the same shared supports and singular values as the full-update construction.

For the left support, since

$$\Delta \mathbf{W}_t = \mathbf{U}_t \mathbf{\Sigma}_t \mathbf{V}_t^\top, \tag{43}$$

the full concatenation can be written as

$$\mathbf{X} = \mathbf{L} \ \mathrm{blkdiag}(\mathbf{V}_1^\top, \mathbf{V}_2^\top, \ldots, \mathbf{V}_T^\top). \tag{44}$$

Because each $\mathbf{V}_t$ has orthonormal columns, the block-diagonal matrix above has orthonormal rows. Therefore,

$$\mathbf{X}\mathbf{X}^\top = \mathbf{L}\mathbf{L}^\top. \tag{45}$$

Thus, $\mathbf{X}$ and $\mathbf{L}$ have the same left Gram matrix, which implies that they share the same nonzero singular values and the same left singular subspace. Consequently, the shared left support $\mathbf{U}_L$ and the strength spectrum $\mathbf{\Sigma}_L$ can be recovered equivalently from the SVD of $\mathbf{L}$.

The same argument applies to the right support. Since

$$\mathbf{Y} = \mathbf{R} \ \mathrm{blkdiag}(\mathbf{U}_1^\top, \mathbf{U}_2^\top, \ldots, \mathbf{U}_T^\top), \tag{46}$$

and each $\mathbf{U}_t$ has orthonormal columns, we have

$$\mathbf{Y}\mathbf{Y}^\top = \mathbf{R}\mathbf{R}^\top. \tag{47}$$

Therefore, $\mathbf{Y}$ and $\mathbf{R}$ also share the same nonzero singular values and the same left singular subspace. Hence, $\mathbf{U}_R$ and $\mathbf{\Sigma}_R$ can be recovered equivalently from the SVD of $\mathbf{R}$.

This proves that the low-rank construction based on $\mathbf{L}$ and $\mathbf{R}$ yields the same task-shared left and right support subspaces as the full-update TSE formulation, while avoiding explicit SVD on the concatenated full update matrices. The equivalence holds up to the standard sign and rotation ambiguities of SVD within degenerate singular subspaces.

C Computational Complexity Analysis

We analyze the preprocessing complexity of PGSB for constructing the merge space of one LoRA-adapted layer before applying a downstream merge operator. Following prior analyses, we assume a square weight matrix of size $n \times n$, T task-specific LoRA updates, and LoRA rank r with $Tr \ll n$. The complexity of model merging operators, KnOTS, and Core Space follows the analysis in the Core Space paper [42].

PGSB Complexity. The direct formulation of PGSB first reconstructs each task update

$$\Delta\mathbf{W}_t = \mathbf{B}_t\mathbf{A}_t, \qquad t = 1, \ldots, T, \tag{48}$$

which costs $\mathcal{O}(n^2Tr)$ in total. It then constructs the left and right task-shared supports by applying SVD to the concatenated full updates

$$[\Delta\mathbf{W}_1, \ldots, \Delta\mathbf{W}_T], \qquad [\Delta\mathbf{W}_1^\top, \ldots, \Delta\mathbf{W}_T^\top]. \tag{49}$$

Each concatenated matrix has size $n \times nT$, and the full-space SVD cost is $O(n^3T^2)$ under the same convention as KnOTS. Since PGSB performs this operation on both the update matrix and its transpose, the asymptotic order remains $\mathcal{O}(n^3T^2)$.

After obtaining the support matrices $\mathbf{U}_L, \mathbf{U}_R$ and strength spectra $\mathbf{\Sigma}_L, \mathbf{\Sigma}_R$, PGSB performs pretrained-coupled coordinate refinement. Let $k \leq Tr$ be the dimension of the shared support. Computing

$$\mathbf{G} = \mathbf{U}_L^\top\mathbf{W}_0\mathbf{U}_R \tag{50}$$

costs $\mathcal{O}(n^2k)$, which is upper-bounded by $\mathcal{O}(n^2Tr)$. The SVD of the resulting $k \times k$ matrix

$$\mathbf{T} = \mathbf{\Sigma}_L\mathbf{G}\mathbf{\Sigma}_R \tag{51}$$

costs $\mathcal{O}(k^3)$, which is upper-bounded by $\mathcal{O}(T^3r^3)$. Therefore, the total preprocessing complexity of the direct PGSB formulation is $\mathcal{O}(n^3T^2 + n^2Tr + T^3r^3)$.

PGSB (Efficient) Complexity. The efficient implementation avoids SVD on concatenated full-size update matrices. For each task update, let

$$\Delta\mathbf{W}_t = \mathbf{U}_t\mathbf{\Sigma}_t\mathbf{V}_t^\top \tag{52}$$

be its compact SVD. Instead of factorizing $[\Delta\mathbf{W}_1, \ldots, \Delta\mathbf{W}_T]$ and $[\Delta\mathbf{W}_1^\top, \ldots, \Delta\mathbf{W}_T^\top]$, PGSB constructs the compact matrices

$$\mathbf{L} = [\mathbf{U}_1\mathbf{\Sigma}_1, \ldots, \mathbf{U}_T\mathbf{\Sigma}_T], \qquad \mathbf{R} = [\mathbf{V}_1\mathbf{\Sigma}_1, \ldots, \mathbf{V}_T\mathbf{\Sigma}_T], \tag{53}$$

whose widths are at most Tr. SVD on these low-rank concatenations costs $\mathcal{O}(n(Tr)^2 + (Tr)^3)$, which is no larger than the subsequent pretrained-coupled refinement cost under $Tr \ll n$. The refinement step is the same as in PGSB: computing $\mathbf{G} = \mathbf{U}_L^\top\mathbf{W}_0\mathbf{U}_R$ costs $\mathcal{O}(n^2Tr)$, and decomposing the $Tr \times Tr$ matrix $\mathbf{T}$ costs $\mathcal{O}(T^3r^3)$. Thus, the efficient implementation has total preprocessing complexity $\mathcal{O}(n^2Tr + T^3r^3)$.

This matches the asymptotic order of Core Space while preserving the two-sided task-shared support construction and the pretrained-coupled coordinate refinement of PGSB.

Algorithm 1 PGSB for LoRA Model Merging

Require: Pretrained weights $\{\mathbf{W}_0^\ell\}_{\ell=1}^L$, task updates $\{\Delta\mathbf{W}_t^\ell\}_{t=1,\ell=1}^{T,L}$, merge rule $\mathcal{M}$
Ensure: Merged model $\mathbf{W}_{\text{merge}}$
for each layer ℓ **do**
  Compute compact SVDs $\Delta\mathbf{W}_t^\ell = \mathbf{U}_t^\ell\mathbf{\Sigma}_t^\ell\mathbf{V}_t^{\ell\top}$ for all t
  Compute $[\mathbf{U}_1^\ell\mathbf{\Sigma}_1^\ell, \ldots, \mathbf{U}_T^\ell\mathbf{\Sigma}_T^\ell] = \mathbf{U}_L^\ell\mathbf{\Sigma}_L^\ell\widetilde{\mathbf{V}}_L^{\ell\top}$ and $[\mathbf{V}_1^\ell\mathbf{\Sigma}_1^\ell, \ldots, \mathbf{V}_T^\ell\mathbf{\Sigma}_T^\ell] = \mathbf{U}_R^\ell\mathbf{\Sigma}_R^\ell\widetilde{\mathbf{V}}_R^{\ell\top}$
  Form $\mathbf{T}^\ell = \mathbf{\Sigma}_L^\ell(\mathbf{U}_L^{\ell\top}\mathbf{W}_0^\ell\mathbf{U}_R^\ell)\mathbf{\Sigma}_R^\ell$
  Decompose $\mathbf{T}^\ell = \mathbf{O}_L^\ell\mathbf{\Sigma}_T^\ell\mathbf{O}_R^{\ell\top}$
  Set $\mathbf{Q}_L^\ell = \mathbf{U}_L^\ell\mathbf{O}_L^\ell$ and $\mathbf{Q}_R^\ell = \mathbf{U}_R^\ell\mathbf{O}_R^\ell$
  Compute $\mathbf{C}_t^\ell = \mathbf{Q}_L^{\ell\top}\Delta\mathbf{W}_t^\ell\mathbf{Q}_R^\ell$ for all t, then merge $\mathbf{C}_*^\ell = \mathcal{M}(\{\mathbf{C}_t^\ell\}_{t=1}^T)$
  Reconstruct $\Delta\mathbf{W}_*^\ell = \mathbf{Q}_L^\ell\mathbf{C}_*^\ell\mathbf{Q}_R^{\ell\top}$
end for
return $\mathbf{W}_{\text{merge}} = \mathbf{W}_0 + \{\Delta\mathbf{W}_*^\ell\}_{\ell=1}^L$

D Algorithm table of PGSB

The full procedure of PGSB is summarized in Algorithm 1.

E Experiment Details

All the experiments were performed on NVIDIA RTX 3090. Our implementation builds on the KnOTS codebase and uses the released LoRA checkpoints from [55]. For the original training and adaptation procedures, we refer readers to [55].

E.1 Baselines

Merging spaces. We compare our proposed PGSB with three alternative merging spaces, which are all plugins.

- **Baseline** performs merging on the updates $\Delta\mathbf{W}^{(t)} = \mathbf{B}^{(t)}\mathbf{A}^{(t)}$ directly.
- **KnOTS** [55] performs merging in the alignment space constructed from the concatenated updates.
- **Core Space** [42] performs merging in the shared low-rank basis induced by the stacked LoRA factors.
- **PGSB** performs merging in pretrained-guided shared basis proposed in Eq.12.

Merging methods. We evaluate each merging space using representative merging methods.

- **Task Arithmetic (TA)** [22] performs scaled summation of task vectors.
- **TIES** [62] trims small-magnitude parameters and resolves sign conflicts before averaging.
- **DARE-TIES** [64] further applies random dropping and rescaling before TIES merging.
- **TSV** [16] concatenates low-rank approximations of task updates and orthogonalizes them across tasks.
- **CART** [7] first centers task vectors by subtracting their average and then performs merging on the resulting low-rank approximations.
- **Iso-C** [33] flattens the singular value spectrum of a merged model and can be combined with other merging methods. We denote such variants by "+Iso-C".

Since TA is linear, its results are equivalent across different merging spaces after exact reconstruction, whereas the choice of merging space mainly affects the performance of nonlinear methods.

E.2 Datasets

Vision tasks. We evaluate vision model merging on eight image classification datasets: Cars196 [26], DTD [8], EuroSAT [18], GTSRB [54], MNIST [28], NWPU-RESISC45 [6], SUN397 [61], and SVHN [39].

To evaluate generalization beyond the original merging tasks, we further use seven additional datasets: CIFAR10, CIFAR100 [27], Oxford-IIIT Pets [43], Oxford Flowers102 [40], Food101 [4], Caltech101 [14], and Cats vs Dogs [13].

Language tasks. We evaluate language model merging on six NLI datasets: SNLI [5], MNLI [59], SICK [34], QNLI [57], RTE [57], and SciTail [24].

We also evaluate language model merging on eight Commonsense Reasoning datasets: ARC-Challenge and ARC-Easy [10], BoolQ [9], HellaSwag [65], OpenBookQA [38], PIQA [3], Social IQa [49], and WinoGrande [48].

Dataset usage. All datasets are public benchmark datasets and are used only for research and benchmarking purposes, following their original licenses or terms of use.

E.3 Hyperparameter Search Strategy

To ensure fair comparison, we follow prior works [55, 42] and apply the same validation-based hyperparameter search protocol to all merging spaces and merge operators. PGSB only changes the coordinate space in which task updates are represented and does not introduce a new downstream merge operator; therefore, the hyper-parameters of each merge operator are selected in the same way across all compared spaces. No test-set labels are used during hyperparameter selection.

For methods with an overall scaling coefficient, we search α from 0.1 with step size 0.1 until validation performance stops improving. For TIES [62] and DARE-TIES [64], we additionally search the top-K trimming parameter from 10 with step size 10. For DARE-TIES, the pruning factor p is searched from 0.1 with step size 0.1. For CART [7], we follow the original setting and search the pruning rank over $\{0.04, 0.08, 0.16, 0.32\}$. CART also introduces an additional scaling coefficient λ:

$$\mathbf{W}_{\text{merge}} = \mathbf{W}_0 + \alpha \left(\Delta \mathbf{W}_{\text{avg}} + \lambda \sum_{t=1}^{T} \bar{\tau}_t \right), \tag{54}$$

where $\Delta \mathbf{W}_{\text{avg}}$ denotes the average update and $\bar{\tau}_t$ denotes the centered task vector for task t. We search λ using the same validation strategy as α. For variants combined with Iso-C [33], we first apply the corresponding merge operator with selected hyper-parameters and then apply Iso-C post-processing.

E.4 Computational Efficiency and Memory Analysis

We further evaluate the practical computational efficiency of PGSB in terms of merging time, peak GPU memory consumption, and hyperparameter search overhead. Although PGSB introduces an additional pretrained-coupled coordinate refinement (PCR) stage, its implementation processes the pretrained weights layer by layer and releases intermediate tensors immediately, avoiding the need to maintain an additional full copy of W_0 on the GPU. Moreover, the shared bases and SVD results are independent of the downstream merging hyperparameters. Therefore, these quantities can be computed once and cached for reuse across subsequent merges and hyperparameter search trials.

Merging Time and Memory Consumption. Table 10 compares the merging time and peak GPU memory consumption of different merging spaces on ViT-B/32, ViT-L/14, and Llama 3 8B. We report both uncached and cached results for PGSB, where the uncached setting includes the construction of the shared bases and PCR, while the cached setting reuses the precomputed quantities.

Without caching, PGSB incurs additional computational overhead compared with Core Space due to PCR. However, caching substantially reduces the merging time to 0.84s, 1.39s, and 12.43s for ViT-B/32, ViT-L/14, and Llama 3 8B, respectively. Consequently, cached PGSB is faster than Core Space on both vision backbones and achieves comparable efficiency on Llama 3 8B. Compared

with KnOTS, cached PGSB achieves a $6.27\times$–$30.66\times$ speedup while reducing peak GPU memory consumption by 62.9%–85.5%. These results indicate that the additional coordinate refinement does not introduce prohibitive computational or memory overhead when intermediate computations are reused.

Table 10: Merging time and peak GPU memory consumption of different merging spaces. PGSB (uncached) includes shared basis construction and PCR, while PGSB (cached) reuses precomputed intermediate results.

Model	Merge Space	Merge Time (s)	Peak GPU Memory (MiB)
ViT-B/32	Baseline	0.73	103.5
	KnOTS	5.27	90.6
	Core Space	1.55	20.5
	PGSB (uncached)	5.76	60.1
	PGSB (cached)	0.84	33.6
ViT-L/14	Baseline	2.50	184.0
	KnOTS	19.46	153.1
	Core Space	3.01	28.3
	PGSB (uncached)	13.48	87.6
	PGSB (cached)	1.39	44.1
Llama 3 8B	Baseline	26.51	2272.0
	KnOTS	381.11	1750.5
	Core Space	11.43	273.6
	PGSB (uncached)	39.94	502.6
	PGSB (cached)	12.43	254.7

Hyperparameter Search Efficiency. The dimensions of the shared bases in PGSB are determined by the numerical ranks of the aggregated task supports at each layer, rather than by the downstream hyperparameter search. Thus, the shared subspace construction and PCR only need to be performed once for each set of task adapters. Subsequent hyperparameter configurations can directly reuse the cached representations without repeating the SVD computations.

To assess the practical impact, we measure the total hyperparameter search time on ViT-B/32 using TIES under the same linear-search protocol. As shown in Table 11, PGSB requires 727s, compared with 806s for Baseline, 794s for KnOTS, and 1187s for Core Space. PGSB therefore achieves the lowest total search time among the evaluated merging spaces, demonstrating that its additional coordinate refinement does not impose an extra validation-search burden in practice.

Table 11: Total hyperparameter search time on ViT-B/32 using TIES under the same linear-search protocol.

Merge Space	Search Time (s)
Baseline	806
KnOTS	794
Core Space	1187
PGSB	**727**

Overall, layer-wise processing and caching substantially reduce the computational and memory overhead of PGSB. The resulting implementation achieves competitive merging efficiency and low validation-search cost while retaining the superior merging performance demonstrated in the main experiments.

F Newer Backbones Experiments

To further evaluate the generalization ability of PGSB beyond CLIP-based visual encoders, we conduct additional experiments on the more recent SigLIP backbone [66]. We merge LoRA adapters trained on the same eight visual classification tasks as in the main experiments, using TIES [62] as the merge operator.

Table 12 reports the normalized accuracies of the merged models. PGSB achieves the highest average performance of 91.27, outperforming Baseline (89.13), KnOTS [55] (88.86), and Core Space [42] (90.91). These results indicate that the effectiveness of PGSB is not limited to the CLIP backbones evaluated in the main experiments, but also extends to newer vision encoders such as SigLIP.

Table 12: Normalized accuracies of merged models on the eight visual tasks using SigLIP-B/16 with TIES merging.

Space	Cars	DTD	EuroSAT	GTSRB	MNIST	RESISC45	SUN397	SVHN	Avg
Abs. Accuracy	94.13	81.81	98.78	97.89	99.53	94.21	76.93	97.27	–
Baseline	98.00	89.73	58.42	90.88	97.83	84.99	94.82	98.38	89.13
KnOTS	98.61	89.79	56.81	92.85	98.34	82.70	94.99	96.80	88.86
Core	99.05	91.35	65.73	95.85	98.34	84.85	95.46	96.61	90.91
PGSB	98.57	95.19	63.33	92.78	98.51	85.93	97.84	97.99	**91.27**

G Heterogeneous Tasks Merging Experiments

We further investigate the ability of PGSB to merge LoRA adapters trained on heterogeneous language tasks using Llama 3 8B [1]. Unlike the homogeneous task configurations evaluated in the main experiments, we jointly merge three commonsense reasoning tasks (OBQA, PIQA, and WinoGrande) and three natural language inference tasks (QNLI, RTE, and SciTail). We adopt TIES [62] as the merge operator and compare PGSB with Baseline, KnOTS [55], and Core Space [42].

Table 13 reports the normalized accuracies on the six tasks. PGSB achieves the highest average score of 97.06, surpassing Baseline (96.64), KnOTS (96.12), and Core Space (96.73). These results demonstrate that PGSB remains effective when merging adapters from different task families, supporting its applicability to heterogeneous language task collections.

Table 13: Normalized accuracies of merged models on heterogeneous language tasks using Llama 3 8B with TIES merging.

Space	OBQA	PIQA	WinoG.	QNLI	RTE	SciTail	Avg
Abs. Accuracy	86.20	90.32	88.08	94.49	89.86	96.52	–
Baseline	93.27	93.55	99.55	96.97	98.38	98.10	96.64
KnOTS	94.43	94.58	100.63	90.81	97.98	98.29	96.12
Core	100.93	95.54	98.92	88.51	99.19	97.27	96.73
PGSB	96.98	96.08	98.65	95.08	97.98	97.56	**97.06**

H PCR Effectiveness Analysis

We further investigate how different factors affect the effectiveness of Pretrained-coupled Coordinate Refinement (PCR). To isolate its contribution, we compare the full PGSB framework against TSE-only, which performs merging directly in the task-shared support coordinates without PCR. We measure the improvement introduced by PCR as $\Delta_{\mathrm{PCR}} = \mathrm{PGSB} - \mathrm{TSE\text{-}only}$, where the difference is computed using the average normalized accuracy across the evaluated tasks.

Specifically, we consider four factors: task similarity, number of adapters, LoRA rank, and merge operator. For task similarity, we measure the average pairwise cosine similarity between task updates and select the three most similar and three least similar four-task groups. For the number of adapters, we randomly sample three task subsets for each of $T = 2$ and $T = 4$, while $T = 8$ includes all eight tasks. We additionally vary the LoRA rank over $\{4, 16, 64, 256\}$ and evaluate four representative merge operators: TIES, DARE-TIES, TSV, and CART.

Table 14 summarizes the performance improvements introduced by PCR under these settings. PCR yields substantially larger gains for highly similar tasks (+4.40) than for less similar tasks (+0.84), suggesting that pretrained-guided coordinate refinement is particularly effective when task updates exhibit coherent shared structures. PCR also consistently improves performance across different numbers of adapters and LoRA ranks, although the gains are non-monotonic, indicating that its effectiveness depends on task composition rather than adapter count or rank alone.

Regarding merge operators, PCR achieves the largest improvements with TIES (+3.82) and DARE-TIES (+5.15). Both methods rely on coordinate-wise pruning and sign conflict resolution, making their merging behavior sensitive to the choice of basis. In contrast, the gains are smaller for TSV (+0.07) and CART (+0.99), which incorporate their own subspace transformations. Overall, these results indicate that PCR is most beneficial when task updates share coherent geometric structures and the downstream merge operator is sensitive to the coordinate representation.

Table 14: Effect of different factors on the performance improvement introduced by PCR. Δ_{PCR} denotes the difference in average normalized accuracy between PGSB and TSE-only. Positive values indicate improvements from PCR.

Factor	**Setting**	$\mathbf{\Delta_{PCR}}$
Task similarity	High / Low	+4.40 / +0.84
Number of adapters	$T = 2/4/8$	+1.30 / +5.68 / +3.82
LoRA rank	$r = 4/16/64/256$	+2.12 / +3.82 / +3.11 / +1.63
Merge operator	TIES / DARE-TIES / TSV / CART	+3.82 / +5.15 / +0.07 / +0.99

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect our paper's contributions and scope.

   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 this in the conclusion section.

   - 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: All the assumptions proposed in this paper have been proven.

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 all implementation details.

   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 will provide codes in the near future.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide all the experimental details.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: We use official evaluation protocols provided by the corresponding benchmarks.

   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 provide all the implementation details.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have checked our work.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss this in the conclusion and introduction sections.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    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: All are properly referred to.

    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: The paper does not 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: No such experiments.

    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: No such experiments.

    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: Not use LLM at all.

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.