Dynamic Model Merging Made Slim

Guodong Du **Wanyu Lin***
Department of Computing, The Hong Kong Polytechnic University
guodong.du@connect.polyu.hk wan-yu.lin@polyu.edu.hk

Abstract

Model merging consolidates independently fine-tuned checkpoints without joint training or access to task data. Dynamic methods retain task-specific capacity, but must divide a limited storage budget between shared and expert parameters. We propose DiDi-Merging, which jointly allocates low-rank capacity to these components through differentiable rank selection, followed by data-free parameter-space refinement. On the eight-task ViT-B/32 benchmark, the small configuration reaches 89.3% average accuracy (98.7% normalized retention) at $1.24\times$ the storage of one backbone, including that backbone; the medium configuration reaches 89.8% at $1.4\times$. Removing refinement reduces the medium configuration by 4.3 percentage points, while static-merge plus uniform-recovery controls are 4.7–8.9 points below the small configuration. Additional evaluations cover language and multimodal models at their separately reported budgets. The main results assume known task identity; practical routing and accuracy under larger task counts at fixed capacity remain distinct evaluation questions. Code is available at https://github.com/WanyuGroup/NeurIPS2026-DiDi-Merging.

1 Introduction

Fine-tuning many task-specific models from a shared backbone [82, 91] is now standard, but deploying them at scale is expensive, and joint multi-task training across the original datasets is often infeasible due to compute or privacy constraints [104]. Model merging combines fine-tuned models into a single unified model at the parameter level without access to the original training data [49, 23].

Most existing work focuses on static merging [47, 115], which fuses fine-tuned models into a single monolithic checkpoint and suffers from task interference. Recent dynamic merging methods [52] route inputs to task-relevant expert components, avoiding interference and supporting modular expansion (Fig. 1). The remaining bottleneck is storage: dynamic methods pay extra parameter overhead per task, motivating slim dynamic merging schemes.

Existing dynamic merging methods exhibit two failure modes that lie at opposite ends of the same axis: *how parameters are split between shared and expert components*. The first (*heavy shared*, Fig. 2a) keeps a full-size shared model and stores only tiny per-task expert vectors: TALL-Mask [94], EMR-Merging [45], Twin-Merging [66], FREE-Merging [128], and MMER [52] all fall into this regime, with parameters concentrated in a shared component no smaller than the original model. The second (*heavy expert*, Fig. 2b) does the opposite: TSV-C [30] and SMILE [90] drop the shared component and push all task knowledge into per-task expert vectors, so the total expert storage grows with the number of tasks even when each vector is low-rank. Neither extreme balances the two (Fig. 2c). To address these limitations, we propose **DiDi**-Merging (**D**yna**mi**c Merging with **Di**fferentiable Rank), which balances parameters between shared and task-specific experts, yielding a slim and efficient dynamic merging framework.

*Corresponding author.

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

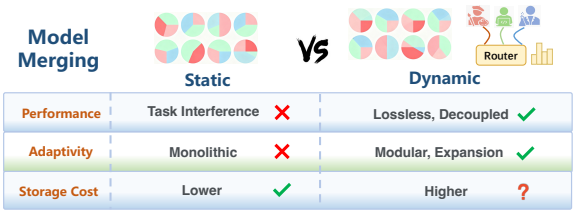
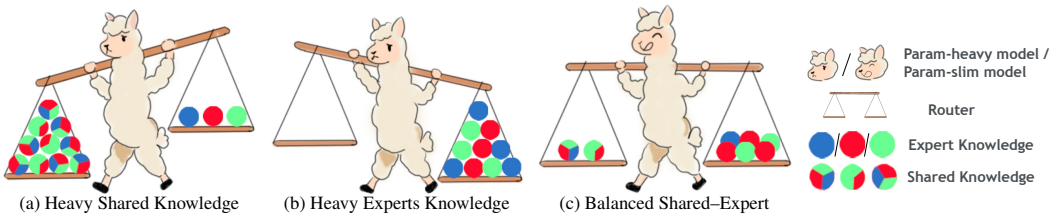


Figure 1: Static and dynamic model merging.



(a) Heavy Shared Knowledge (b) Heavy Experts Knowledge (c) Balanced Shared–Expert

Figure 2: Parameter allocation in dynamic model merging. The balance beam denotes the router, and baskets hold shared (multi-colour circles) and expert (single-colour circles) knowledge; legend at right. (a) *Heavy shared*: full-size shared model, tiny per-task experts. (b) *Heavy expert*: no shared component; per-task expert storage grows with task count. (c) *Balanced shared–expert*: slim shared module + small per-task experts, the regime `DiDi-Merging` targets via differentiable rank optimisation, with capacity adapting to task similarity.

DiDi-Merging proceeds in three steps. (1) Task vectors are statically merged into a shared knowledge vector, with per-task residuals as expert vectors (exact but parameter-heavy). (2) Low-rank decomposition recasts parameter budgeting as a differentiable rank allocation problem, learned end-to-end against a task-vector reconstruction objective. (3) The truncated low-rank parameters are refined in a data-free pass to recover task-specific performance. Unlike prior differentiable-SVD methods for compressing a *single* model [95], our *cross-task* rank allocation jointly optimizes shared and expert components against task-vector reconstruction targets. The full pipeline runs in two stages (rank optimisation followed by data-free LoRA refinement), illustrated in Fig. 6.

On the eight-task ViT-B/32 benchmark, `DiDi-Merging`-S reaches 89.3% average accuracy (98.7% normalized retention) at $1.24\times$ the storage of one backbone; `DiDi-Merging`-M reaches 89.8% at $1.4\times$. The language and multimodal evaluations use their separately reported budgets and metrics, rather than inheriting the vision operating point. The method combines a shared–expert decomposition, coupled rank allocation, and parameter-space refinement. Its contribution is this budgeted consolidation formulation and the resulting evaluated trade-off, not the invention of SVD, LoRA, or differentiable rank selection.

This paper makes three **contributions**:

- We identify two failure modes in existing dynamic merging methods (*heavy shared* and *heavy expert*), both stemming from one-sided shared/expert allocation.
- We propose **DiDi**-Merging, a **d**ynam**i**c merging framework with **di**fferentiable rank allocation: ranks are jointly optimized for shared and per-task expert components against task-vector reconstruction targets, without access to original training data.
- We evaluate accuracy and storage across vision, language, and multimodal settings, with controlled comparisons that separate the value of adaptive rank allocation from that of uniform-rank recovery.

2 Related Work

Dynamic Model Merging. Recent surveys organize model fusion across parameter, representation, and behavior levels [3, 130, 131]; behavior-level alternatives include preference-based fusion and graph-on-logits distillation [37, 99]. Static merging [47, 115, 22, 24, 28] fuses fine-tuned models

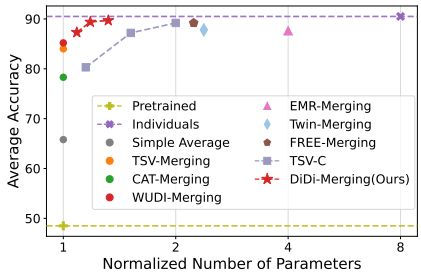


Figure 3: **DiDi**-Merging achieves a superior Pareto trade-off between parameter efficiency and accuracy across all dynamic merging baselines.

Table 1: Model merging approaches and requirements.

Method	Parameters	Data Free	Slim Shared	Slim Experts
Multiple Models [105]	8 X	-	-	-
Traditional MTL [8]	1 X	×	-	-
Static Merging [118]	1 X	✓	-	-
Dynamic Merging Methods				
WEMoE [89]	6.27 X	×	✓	×
TALL-Mask [94]	2.25 X	✓	×	✓
EMR-Merging [45]	2.25 X	✓	×	✓
Twin-Merging [66]	2.25 X	✓	×	✓
Free-Merging [128]	2.16 X	✓	×	✓
TSV-C [30]	2.08 X	✓	✓	×
SMILE [90]	3.07 X	×	✓	×
DiDi-Merging (Ours)	**1.24** X	✓	✓	✓

into a single checkpoint and suffers from task interference. Dynamic methods route inputs to task-specific experts to avoid interference [89, 118, 25, 58, 59], but their parameter allocation is fixed: *heavy shared* variants (TALL-Mask [94], EMR-Merging [45], Twin-Merging [66], FREE-Merging [128], MMER [52]) keep a full-size shared model with masks or rescalers to identify task-relevant subspaces, while *heavy expert* variants (TSV-C [30], SMILE [90]) drop the shared component and compress per-task vectors via SVD or subspace modelling. Recent methods also construct reusable component-level expert libraries for input-aware recombination [80] or evolve experts through manifold-aware, training-free routing for continual merging [81]. Complementary work spans VLA policy generation, resource-aware routing, and post-training calibration [19, 16, 54], merge-aware expert training [39], post-merge calibration or quantization [38, 96], and merging scaling laws or expert-read budgets [101, 100]. `DiDi-Merging` instead *optimizes* shared/expert capacity under a fixed storage budget via a differentiable rank objective (App. A).

Task Vector Compression and Differentiable Rank Selection. Most merging methods build on task-vector arithmetic [47, 124, 115, 122, 22, 116] or further compress task vectors via masking [77, 41, 137], SVD [97, 28, 120, 87, 66, 36], or quantization [29, 61, 64, 78, 53], yet they allocate storage uniformly across tasks and overlook the shared/expert distinction. Differentiable rank selection has been studied outside merging: AdaLoRA [125] prunes singular values during fine-tuning, DyLoRA [92] trains a single LoRA across a range of ranks for inference-time selection, and Dobi-SVD [95] introduces differentiable truncation for single-model LLM compression. `DiDi-Merging` differs in three ways: *(i) Setting:* we operate on multiple pre-existing task vectors for post-hoc merging rather than fine-tuning or single-model compression; *(ii) Data:* we are fully data-free, supervised by the stored task vectors themselves rather than a calibration set; *(iii) Allocation scope:* ranks are jointly optimized across task vectors *and* the shared/expert decomposition, rather than within a single matrix or model. Related multimodal studies also organize shared and specialized information: Meng et al. [74] propose DBR, which rebalances shared and private branches to reduce shared redundancy while preserving modality-specific evidence.

3 Background and Motivation

3.1 Preliminaries

Consider a pre-trained model f with parameters $\boldsymbol{\theta}_0$ and T downstream tasks fine-tuned to $\boldsymbol{\theta}_t$, with *task vector* $\boldsymbol{\tau}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$. **Static merging** fuses these into a single input-agnostic model $\boldsymbol{\theta}^* = \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \lambda_t \boldsymbol{\tau}_t$, while **dynamic merging** activates shared and task-specific components conditional on the input:

$$\boldsymbol{\theta}^*(\boldsymbol{x}) = \boldsymbol{\theta}_0 + \boldsymbol{M}_s + \sum_{t=1}^{T} w_t(\boldsymbol{x}) \boldsymbol{M}_t, \tag{1}$$

where $\boldsymbol{M}_s$ and $\boldsymbol{M}_t$ encode shared and expert knowledge, and $w_t(\boldsymbol{x})$ are lightweight, input-dependent routing weights from a small router. Following prior dynamic merging work [66, 52], we use an MoE-style *ideal router* that takes task identity at inference; routing design is orthogonal to our

contribution, which targets parameter organisation across shared and expert components. All reported total-storage ratios include one pretrained backbone. The ideal task-ID selector has no learned router parameters; a learned router would incur an additional cost that must be counted separately. The main results assume known task identity and do not evaluate routing for unknown tasks.

3.2 Rethinking Dynamic Model Merging

Q1: How should shared and expert knowledge be represented? As Fig. 4 shows, prior dynamic methods take one of two extremes: *heavy shared* (full-size $\boldsymbol{M}_s$, e.g., [45, 66, 128]) or *heavy expert* (no $\boldsymbol{M}_s$, e.g., [30]). We instead represent *both* as low-rank modules with flexible rank allocation: similar tasks rely on shared ranks, diverse tasks on expert ranks (Fig. 7).

Q2: How to allocate ranks across shared and expert modules? At a fixed parameter budget, the choice of ranks substantially affects accuracy (Fig. 5). We introduce a differentiable rank mechanism: each weight matrix is SVD-decomposed and its singular values are reweighted by a smooth truncation function controlled by a learnable continuous rank r. Jointly minimising a task reconstruction loss and a compression penalty yields a slim allocation for each module.

Q3: How to recover task fidelity after rank truncation? Direct truncation degrades accuracy, so we apply a data-free optimization that refines the low-rank module parameters using the stored task vectors themselves as reconstruction targets.

4 Method

We present **DiDi**-Merging, a slim dynamic merging framework (Fig. 6). Our approach consists of three key steps: establishing shared and expert matrices (§4.1), differentiable optimization of ranks to determine optimal allocations under parameter constraints (§4.2), and a data-free refinement that restores performance lost from rank truncation (§4.3).

4.1 Low-Rank Decomposition

Building upon the task vector representation introduced in the preliminary section, we decompose both shared and expert knowledge into low-rank modules to enable efficient dynamic merging. The shared matrix $\boldsymbol{M}_s$ is obtained via static merging of the task vectors $\{\boldsymbol{\tau}_t\}_{t=1}^{T}$, e.g., by averaging:

$$\boldsymbol{M}_s = \mathrm{avg}(\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_T), \tag{2}$$

and each expert matrix is defined as the residual relative to the shared component:

$$\boldsymbol{M}_t = \boldsymbol{\tau}_t - \boldsymbol{M}_s. \tag{3}$$

To achieve controllable and efficient parameterization, we apply singular value decomposition (SVD):

$$\boldsymbol{M}_s = \boldsymbol{U}_s \boldsymbol{\Sigma}_s \boldsymbol{V}_s^\top, \quad \boldsymbol{M}_t = \boldsymbol{U}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^\top. \tag{4}$$

Retaining only the top-r singular components yields the low-rank modules:

$$\tilde{\boldsymbol{M}}_s^{(r)} = \boldsymbol{U}_s^{(r)} \boldsymbol{\Sigma}_s^{(r)} {\boldsymbol{V}_s^{(r)}}^\top, \quad \tilde{\boldsymbol{M}}_t^{(r)} = \boldsymbol{U}_t^{(r)} \boldsymbol{\Sigma}_t^{(r)} {\boldsymbol{V}_t^{(r)}}^\top. \tag{5}$$

Finally, the dynamically merged parameters are

$$\boldsymbol{\theta}^*(\boldsymbol{x}) = \boldsymbol{\theta}_0 + \tilde{\boldsymbol{M}}_s^{(r)} + \sum_{t=1}^{T} w_t(\boldsymbol{x}) \cdot \tilde{\boldsymbol{M}}_t^{(r)}. \tag{6}$$

4.2 Differentiable Optimization of Ranks

To efficiently determine the optimal rank of each shared and expert module in a layer, we leverage a differentiable approach based on the smooth truncation of the singular values from the SVD of the matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$, $m > n$:

$$\boldsymbol{M} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^\top, \quad \boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n), \tag{7}$$

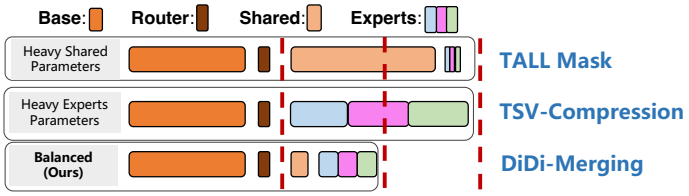


Figure 4: Parameter accounting under three dynamic merging regimes. *Heavy shared* inflates the shared component; *heavy expert* inflates per-task experts; `DiDi-Merging` keeps both small. Decomposition $|\boldsymbol{\theta}_0| + |\text{router}| + r_s(m{+}n) + T\, r_t(m{+}n)$ and the $1.24\times$ / $1.4\times$ / $2.0\times$ budgets in App. B.7.

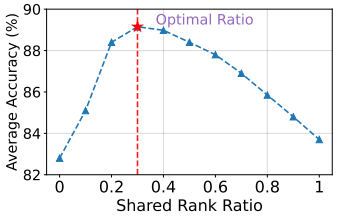


Figure 5: Effect of rank allocation on dynamic merging accuracy at a fixed parameter budget on the 8-task vision benchmark.

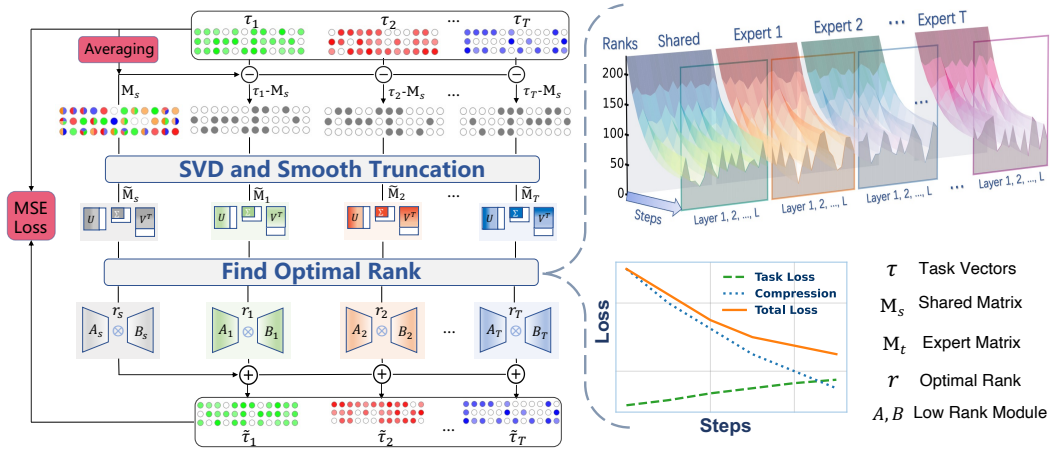


Figure 6: The **DiDi**-Merging pipeline. *Left:* task vectors $\{\boldsymbol{\tau}_t\}$ are split into a shared component $\boldsymbol{M}_s$ via averaging and per-task expert residuals $\boldsymbol{\tau}_t - \boldsymbol{M}_s$; SVD with smooth truncation produces low-rank $\tilde{\boldsymbol{M}}_s, \tilde{\boldsymbol{M}}_t$, the Find-Optimal-Rank stage learns per-module ranks $r_s, r_1, \ldots, r_T$, and LoRA factors (A, B) are refined data-free against an MSE reconstruction loss with respect to the original task vectors. *Right:* during optimisation, per-layer ranks evolve separately for shared and expert modules (top), while the joint objective trades task reconstruction against compression (bottom).

where $\boldsymbol{U} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times n}$. We define a smooth truncation function to retain a differentiable subset of singular values:

$$\tilde{\sigma}_i = \sigma_i \left[0.5 \cdot \tanh\left(\beta(r-i)\right) + 0.5\right], \tag{8}$$

where r is a learnable continuous rank parameter, and β controls the smoothness of the truncation. Comparison with AdaLoRA [125] and Dobi-SVD [95] is in §2 and App. D. The truncated low-rank matrix is reconstructed as

$$\tilde{\boldsymbol{M}} = \boldsymbol{U}\,\mathrm{diag}(\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_n)\,\boldsymbol{V}^\top. \tag{9}$$

During optimization, only the rank variable r is updated via gradient descent, while other low-rank module parameters remain frozen.

Task Reconstruction Loss. The reconstructed per-layer task vector is given by

$$\tilde{\boldsymbol{\tau}}_{t,l} = \tilde{\boldsymbol{M}}_{s,l} + \tilde{\boldsymbol{M}}_{t,l}, \tag{10}$$

where $\tilde{\boldsymbol{M}}_{s,l}$ and $\tilde{\boldsymbol{M}}_{t,l}$ denote the truncated shared and expert matrices at layer l. The task reconstruction loss is the mean squared error over all tasks and layers:

$$L_{\text{task}} = \frac{1}{T}\sum_{t=1}^{T}\sum_{l=1}^{L}\left\|\tilde{\boldsymbol{\tau}}_{t,l} - \boldsymbol{\tau}_{t,l}\right\|_F^2, \tag{11}$$

where T is the number of tasks and L the number of layers.

Coupled parameter budget. For a layer of shape $m_l \times n_l$, storing both shared factors and all expert factors costs

$$B_{\text{factors}} = \sum_l (m_l + n_l) \left(r_{s,l} + \sum_{t=1}^{T} r_{t,l} \right). \tag{12}$$

The rank penalty controls this total allocation, rather than granting an independent full budget to each expert. R_{now} denotes the factor count normalized by the corresponding dense matrix count; R_{tar} sets its target. The reported deployment ratio additionally includes the pretrained backbone and any router, as detailed in Appendix B.7. Shared and expert factors compete for the same storage budget, and integer ranks must be counted after truncation.

Total Loss. The objective balances task reconstruction performance and compression ratio:

$$L = L_{\text{task}} + \gamma \cdot \left| R_{\text{now}} - R_{\text{tar}} \right|, \tag{13}$$

where L_{task} is the task reconstruction loss, R_{tar} the target compression ratio, and γ a weighting factor.

4.3 Low-rank Module Update

After obtaining the optimal rank allocation $\{r_{t,l}\}$ for each task t and layer l, we instantiate low-rank LoRA-style modules to recover performance. For each layer l and task t, we introduce factor matrices

$$\boldsymbol{A}_{t,l} \in \mathbb{R}^{m_l \times r_{t,l}}, \qquad \boldsymbol{B}_{t,l} \in \mathbb{R}^{r_{t,l} \times n_l},$$

and express the low-rank module parameter update as

$$\boldsymbol{M}_{t,l} = \boldsymbol{A}_{t,l} \boldsymbol{B}_{t,l}. \tag{14}$$

Initialization. To provide a good starting point and accelerate convergence, we initialize each pair $(\boldsymbol{A}_{*,l}, \boldsymbol{B}_{*,l})$ for $* \in \{s, 1, \ldots, T\}$ (one shared module and T per-task experts) using the truncated SVD of the precomputed matrix $\boldsymbol{M}^{(0)}_{*,l}$:

$$\boldsymbol{M}^{(0)}_{*,l} \approx \boldsymbol{U}^{(r)}_{*,l} \boldsymbol{\Sigma}^{(r)}_{*,l} {\boldsymbol{V}^{(r)}_{*,l}}^{\top},$$

We initialize the factors as

$$\boldsymbol{A}_{*,l} \leftarrow \boldsymbol{U}^{(r)}_{*,l} (\boldsymbol{\Sigma}^{(r)}_{*,l})^{1/2}, \qquad \boldsymbol{B}_{*,l} \leftarrow (\boldsymbol{\Sigma}^{(r)}_{*,l})^{1/2} {\boldsymbol{V}^{(r)}_{*,l}}^{\top},$$

so that $\boldsymbol{A}_{*,l} \boldsymbol{B}_{*,l} \approx \boldsymbol{M}^{(0)}_{*,l}$.

Data-free Task Reconstruction. We supervise using the existing task vectors $\{\boldsymbol{\tau}_t\}$. For task t, the reconstructed vector is computed by summing the contributions of the low-rank shared and task-specific LoRA modules across layers:

$$\tilde{\boldsymbol{\tau}}_{t,l} = \boldsymbol{A}_{s,l} \boldsymbol{B}_{s,l} + \boldsymbol{A}_{t,l} \boldsymbol{B}_{t,l}, \tag{15}$$

where $\boldsymbol{A}_{s,l} \boldsymbol{B}_{s,l}$ and $\boldsymbol{A}_{t,l} \boldsymbol{B}_{t,l}$ denote shared and task-specific LoRA modules, respectively. The task reconstruction loss is then defined as the mean squared error over all tasks and layers:

$$L_{\text{task}} = \frac{1}{T} \sum_{t=1}^{T} \sum_{l=1}^{L} \left\| \tilde{\boldsymbol{\tau}}_{t,l} - \boldsymbol{\tau}_{t,l} \right\|_F^2. \tag{16}$$

Optimization Protocol. In this stage, the shared and task-specific factor matrices are refined with the ranks fixed, while the pretrained backbone remains frozen. We perform a lightweight, data-free optimization using Adam to minimize L_{task} (more optimization details in App. D).

5 Experiments

We evaluate vision (§5.1), LLM (§5.2), and MLLM (§5.3) merging, with zero-shot and separate fine-tuned models as references. Total-storage ratios include one backbone and all shared and expert parameters; the task-ID selector has no learned router. We report accuracy or the specified benchmark score. Baseline, dataset, and training details appear in Appendices H–J.2.

The evaluated budgets are $1.24\times$ (S), $1.4\times$ (M), and $2\times$ (L), set by scaling the optimized per-layer ranks. Vision uses S/M, while LLM and MLLM use L. These are the evaluated operating points; relative storage alone does not isolate the effects of rank allocation and refinement. PEFT comparisons and additional modality results are in Appendices B.5 and C.

5.1 Merging Vision Models

Setup. Following Ilharco et al. [47, 22], we evaluate on CLIP-ViT-B/32 and ViT-L/14 [82] across eight classification benchmarks (SUN397, Cars, RESISC45, EuroSAT, SVHN, GTSRB, MNIST, DTD; details in App. I), and compare against five static and seven dynamic baselines listed in Tab. 2.

Vision Results. Tab. 2 reports accuracy and parameter overhead on the 8-task suite. Static methods add no storage but trail dynamic baselines by 4–6% in accuracy, with the strongest static method (WUDI-Merging at 85.2%) still 4.3% below the best dynamic baseline. At 1.24×, `DiDi-Merging`-S matches the strongest dynamic baseline (89.3% vs. TSV-C at 89.5%) using under 60% of their storage; at 1.4×, `DiDi-Merging`-M attains the highest score overall (89.8%), surpassing all baselines. Scaling to 14 / 20 tasks and ViT-L/14 (Tab. 3), `DiDi-Merging`-M leads at 14 / 20 tasks (99.5%/99.4% retention) and ties TSV-C within 0.06% on ViT-L/14 at less than half the overhead. These evaluations cover different backbone sizes and task counts, with the corresponding settings and operating points reported in the tables.

5.2 Merging Large Language Models

Setup. We merge four fine-tuned LLMs derived from Llama-3.1-8B-Instruct [26] and Qwen-2.5-7B-Instruct [2], each obtained by sequential SFT and DPO on a different task domain (general, mathematics, code, and instruction following), following FuseLLM [93] and FuseChat [121]. Training configurations are in App. J.1. We compare against state-of-the-art dynamic baselines Twin-Merging and FREE-Merging.

Datasets. Evaluation spans ten benchmarks across four categories: general (MMLU-Pro, MMLU-Redux, GPQA-Diamond), mathematics (GSM8K, MATH, AMC23), code (HumanEval, MBPP), and instruction following (AlpacaEval-2, MT-Bench). Details in App. I.

LLM Results. LLM merging is harder than vision (more diverse capabilities, larger task heterogeneity); we report `DiDi-Merging`-L (2×) for parity with prior dynamic baselines that all use $\geq 2\times$ storage (Tab. 4). `DiDi-Merging` consistently outperforms prior methods, particularly on challenging benchmarks like AlpacaEval-2, where the gains over FREE-Merging are 5.4 and 5.1 points for Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct, respectively. The table-wide averages increase by 1.8 and 1.4 points.

5.3 Merging Multimodal Large Language Models

Setup. We merge four MLLMs (Vision, Audio, Video, Point Cloud), each fine-tuned from Vicuna-7B-v1.5 [126] under identical environments following [4, 76]; configurations are in App. J.2. We compare against representative data-free dynamic baselines NaiveMC [4] and MMER [52], reporting accuracy/scores and retention relative to each fine-tuned counterpart.

Datasets. We use fourteen dual-modal tasks, each pairing text with another modality: vision (VQAv2 [35], GQA [46], TextVQA [85], VizWiz [40], ScienceQA [65], POPE [57], OK-VQA [70]), audio (TUT [75], VocalSound [33], Clotho [18]), video (MSRVTT [113], MSVD [5]), and point cloud (ModelNet40 [106], Objaverse [14]).

MLLM Results. Tab. 5 shows `DiDi-Merging` consistently outperforms prior dynamic methods at a slim parameter footprint; NaiveMC's efficient merging suffers substantial performance degradation. Notably, all dynamic methods (including ours) exceed the fine-tuned baseline on several audio and point-cloud tasks: this is consistent with useful transfer across the evaluated models. The benchmark scores do not by themselves identify a specific transfer mechanism; additional analysis is in App. B.4.

6 Analysis

We evaluate component contributions and solving cost. Additional rank, PEFT, and task-relationship analyses are reported in the appendix.

Table 2: Test set performance when merging ViT-B/32 models on 8 vision tasks, with normalized averages shown in parentheses.

Task(→) Method(↓)	**#Params(↓)**	**Average(%)(↑)**	**Test Set Performance**							
			SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD
Individuals	x8	90.5 (100)	75.3	77.7	96.1	99.7	97.5	98.7	99.7	79.4
Multitask[84]	x1	88.9 (99.2)	74.4	77.9	98.2	98.9	99.5	93.9	72.9	95.8
Static Merging Methods										
Weight Averaging[ICLR23] [47]	x1	65.8 (72.7)	65.3	63.4	71.4	71.7	64.2	52.8	87.5	50.1
TSV-Merging[CVPR25] [30]	x1	84.0 (92.8)	69.1	70.7	85.5	94.3	92.0	91.9	99.3	69.2
CAT-Merging[ICML25] [88]	x1	78.3 (86.5)	68.1	65.4	80.5	89.5	85.5	78.5	98.6	60.7
ISO-CTS[ICML25] [69]	x1	81.4 (89.9)	74.4	74.4	87.2	90.4	76.8	83.3	97.4	67.0
WUDI-Merging[ICML25] [11]	x1	85.2 (94.1)	71.1	71.0	85.7	95.6	94.2	94.7	99.5	69.7
Dynamic Merging Methods										
WEMoE[ICML24] [89]	x6.27	89.2 (98.6)	73.7	76.8	93.4	98.2	96.8	98.2	**99.6**	76.6
SMILE[TPAMI25] [90]	x3.07	89.3 (98.7)	73.6	77.8	92.0	98.3	**96.9**	98.1	**99.6**	**78.1**
TALL-Mask+TIES[ICML24] [94]	x2.25	88.7 (98.0)	75.8	72.5	93.2	99.3	96.8	**98.5**	99.1	74.4
Twin-Merging[NeurIPS24] [66]	x2.25	87.8 (97.0)	73.6	71.7	92.1	99.3	95.3	97.2	99.1	74.0
EMR-Merging[NeurIPS24] [45]	x2.25	87.7 (96.9)	74.1	72.7	91.9	99.4	95.8	96.9	99.1	72.1
FREE-Merging[ICCV25] [128]	x2.16	89.2 (98.6)	76.4	77.6	93.4	**99.5**	96.3	98.2	99.5	75.4
TSV-C[CVPR25] [30]	x2.08	89.5 (98.9)	77.0	77.3	93.4	99.2	96.3	98.3	99.3	75.3
`DiDi-Merging`**-S (ours)**	**x1.24**	89.3 (98.7)	76.9	77.2	93.2	99.3	96.5	98.1	99.2	74.1
`DiDi-Merging`**-M (ours)**	x1.4	**89.8** (99.2)	**77.3**	**77.9**	**93.5**	**99.5**	96.4	98.2	99.5	76.2

Table 3: Average accuracy and parameter overhead across dynamic merging benchmarks for various models and task counts.

Task(→) Method(↓)	**#Params**	**ViT-B/32**		**ViT-L/14**
		14 tasks(%)	20 tasks(%)	8 tasks(%)
Individuals	x14, 20, 8	90.88 (100)	91.37 (100)	95.81 (100)
TALL-Mask+TIES	x2.25	90.13 (99.2)	90.91 (99.5)	93.92 (98.0)
EMR-Merging	x2.25	89.93 (98.9)	89.48 (97.9)	92.73 (96.8)
Twin-Merging	x2.25	89.54 (98.5)	89.97 (98.5)	92.81 (96.9)
FREE-Merging	x2.16	90.32 (99.4)	90.23 (98.7)	93.78 (97.9)
TSV-C	x2.08	90.29 (99.4)	90.64 (99.2)	**95.68** (99.9)
`DiDi-Merging`**-M (ours)**	**x1.4**	**90.42** (99.5)	**90.84** (99.4)	95.62 (99.8)

Table 4: Overall results of dynamic merging 4 different LLMs across 10 benchmark tasks. Twin and FREE denote Twin-Merging and Free-Merging.

LLM	**Benchmark**	**Llama-3.1-8B-Instruct**					**Qwen-2.5-7B-Instruct**				
		Zero-shot	Fine-tuned	Twin	FREE	**DiDi-Merg.-L**	Zero-shot	Fine-tuned	Twin	FREE	**DiDi-Merg.-L**
General	MMLU-Pro	49.7	51.3	50.2	50.4	**51.1**	54.0	55.6	54.5	54.8	**55.2**
	MMLU-redux	70.5	73.0	72.6	72.4	**72.8**	75.1	76.6	74.8	75.2	**76.2**
	GPQA-Diamond	33.6	37.7	36.1	35.9	**37.2**	34.7	38.1	36.8	37.2	**37.4**
Mathematics	GSM8K (0 shot, CoT)	85.9	88.8	86.4	**88.2**	88.0	91.7	92.6	91.3	91.3	**91.6**
	MATH (0 shot, CoT)	50.7	56.2	55.2	55.0	**55.3**	75.0	75.3	72.1	**74.2**	74.0
	AMC 23 (0 shot, CoT)	25.0	37.5	27.5	27.5	**35.0**	52.5	57.5	50.0	47.5	**52.5**
Coding	HumanEval (0 shot)	68.3	72.0	68.8	69.1	**70.4**	85.4	85.7	81.9	84.1	**84.4**
	MBPP (0 shot)	66.9	73.0	70.3	71.2	**71.8**	80.2	84.8	81.6	82.5	**83.3**
Instruction Following	AlpacaEval-2 (LC %)	28.3	65.4	52.4	54.1	**59.5**	34.2	63.6	53.3	53.6	**58.7**
	MT-Bench	8.4	9.0	8.2	8.3	**8.7**	8.4	9.0	7.8	8.2	**8.7**
Average		48.7	56.4	52.8	53.2	**55.0**	59.0	63.8	60.4	60.8	**62.2**
Parameters		x1	x4	x2.25	x2.08	**x2.0**	x1	x4	x2.25	x2.08	**x2.0**

Table 5: Results of multi-modality dynamic merging experiments. The performance retention is shown in parentheses.

Task (→) Method (↓)	**#Params(↓)**	**2 Point Tasks** Score (%) / Acc. (%)	**3 Audio Tasks** Score (%) / Acc. (%)	**2 Video Tasks** Acc. (%)	**7 Image Tasks** Acc. (%)
Individuals	[x2, x7]	23.15 / 21.27	25.30 / 24.71	39.79	62.23
NaiveMC[ACL2024] [4]	[x1.24, x1.84]	22.65 (97.8) / 20.49 (96.3)	24.59 (97.2) / 30.65 (124.8)	36.92 (93.0)	52.56 (83.6)
MMER[ACL2025] [52]	[x2.13, x2.44]	23.14 (99.9) / 22.49 (105.7)	25.20 (99.6) / 38.51 (155.6)	39.28 (98.5)	62.40 (100.3)
FREE-Merging[ICCV2025] [128]	[x2.04, x2.14]	23.02 (99.5) / 21.14 (99.4)	25.23 (99.7) / 38.58 (156.1)	38.52 (96.8)	62.54 (100.5)
`DiDi-Merging` **(ours)**	[x1.4, x2.0]	**23.08** (99.7) / **22.58** (106.2)	**25.27** (99.9) / **38.74** (156.8)	**39.51** (99.3)	**62.93** (101.1)

Table 6: Component ablation on DiDi-Merging at a matched parameter budget. When an optimal-rank component is ablated, its rank is replaced with a budget-preserving uniform value.

Method	ViT-B/32	ViT-L/14
DiDi-Merging **(full)**	**89.8**	**95.6**
– w/o Shared Matrix	87.3	93.7
– w/o Shared Optimal Rank	88.9	94.7
– w/o Expert Optimal Rank	87.4	93.6
– w/o Layer-wise Optimal Rank	88.2	94.1
– w/o Low-rank Module Update	85.5	91.2
– w/o Low-rank Module Init.	89.6	95.5

Table 7: Solving time and GPU memory on ViT-B/32, against optimisation-based merging baselines. Solving time and memory describe different resource costs.

Method	Acc.	Time	GPU Mem.
TIES Merging[NeurIPS23] [115]	72.4	4 s	0 GB
Adamerging[ICLR24] [119]	81.1	127 min	17.1 GB
TSV-C[CVPR25] [30]	89.1	3 min 27 s	0 GB
WUDI-Merging[ICML25] [11]	85.2	1 min 54 s	4.0 GB
DiDi-Merging **(Ours)**	**89.8**	14 min 37 s	1.1 GB
–Find Optimal Rank	–	10 min 22 s	1.1 GB
–Low-rank Module Update	–	4 min 15 s	0.6 GB

6.1 Ablation Study

Tab. 6 ablates the design choices of DiDi-Merging at a matched parameter budget. Removing the shared low-rank matrix or the per-expert optimal rank degrades ViT-B/32 accuracy by 2.5% and 2.4%, respectively, while disabling layer-wise rank optimization causes a smaller drop. Removing low-rank refinement reduces ViT-B/32 accuracy from 89.8 to 85.5% (4.3 points) and ViT-L/14 from 95.6 to 91.2% (4.4 points), showing that rank selection alone does not account for the complete method. SVD-based initialization of the LoRA factors has a small effect on the final score (-0.2%) but substantially accelerates convergence.

Is the rank optimization necessary? A static merge followed by uniform-rank recovery is a simpler alternative to learned allocation. App. B.2 tests this with four static mergers (Simple Averaging, TIES-Merging, Iso-CTS, WUDI-Merging) at a matched $1.24\times$ budget. The four alternatives reach 80.4, 81.5, 82.2, and 84.6%, compared with 89.3% for DiDi-Merging-S: gaps of 4.7–8.9 percentage points. This comparison supports the complete adaptive allocation design; a fully crossed allocation-by-refinement experiment would provide a more specific interaction analysis.

6.2 Computation Resource

Tab. 7 compares solving time and GPU memory on ViT-B/32 against optimisation-based merging baselines. DiDi-Merging uses 14 min 37 s in total, compared with 3 min 27 s for TSV-C and 1 min 54 s for WUDI in this timing table. Its memory use is 1.1 GB, below WUDI's 4.0 GB; the lower GPU memory requirement therefore does not imply lower wall-clock cost. The Find-Optimal-Rank stage runs sequentially per task and is the dominant wall-clock bottleneck of the pipeline.

Storage accounting. All total-storage ratios count one backbone and the shared and expert factors. The task-ID selector introduces no learned router parameters. Appendix B.7 gives the bit-level decomposition and learned rank statistics, including the global treatment of masks and scalars.

7 Conclusions

DiDi-Merging consolidates fine-tuned models through a shared–expert low-rank decomposition, coupled rank allocation, and data-free refinement. On the evaluated eight-task vision benchmark, it reaches 89.3% at $1.24\times$ storage and 89.8% at $1.4\times$, including the backbone. Language and multimodal results extend the evaluation at their own budgets. The present evidence supports this accuracy–storage trade-off under known task identity; practical routing and task-growth behavior remain separate empirical questions.

Acknowledgments and Disclosure of Funding

Funding statement. This work was supported by the Hong Kong Polytechnic University Internal Research Fund (the Projects of RIAIoT, Project ID P0065264), UDomain Web Hosting Company Limited (Collaboration Scheme, Project ID P0064507), and the Innovation and Technology Fund–Guangdong-Hong Kong Technology Cooperation Funding Scheme (ITF-TCFS) (Project ID P0049179).

Competing interests. The authors declare that they have no competing interests.

References

[1] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[2] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[3] S. Cai, Y. Gu, Z. Wang, Y. Wang, Y. Yan, W. Wang, Y. Liu, G. Zhu, S. Huang, M. Li, and H. Yang. From parameters to behaviors: A survey of model fusion for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, 2026. URL https://arxiv.org/abs/2609.19553.

[4] C. Chen, Y. Du, Z. Fang, Z. Wang, F. Luo, P. Li, M. Yan, J. Zhang, F. Huang, M. Sun, and Y. Liu. Model composition for multimodal large language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[5] D. L. Chen and W. B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 190–200, 2011.

[6] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[7] S. Chen, Y. Wu, C. Wang, S. Liu, D. Tompkins, Z. Chen, W. Che, X. Yu, and F. Wei. Beats: Audio pre-training with acoustic tokenizers. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 202, pages 5178–5193, 2023.

[8] S. Chen, Y. Zhang, and Q. Yang. Multi-task learning in natural language processing: An overview. *ACM Computing Surveys*, 56(12):1–32, 2024.

[9] Y. Chen, J. Li, W. Yao, X. Ma, G. Du, W. Wang, and J. Li. Vocabulary hijacking in lvlms: Unveiling critical attention heads by excluding inert tokens to mitigate hallucination. *arXiv preprint arXiv:2605.10622*, 2026.

[10] G. Cheng, J. Han, and X. Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[11] R. Cheng, F. Xiong, Y. Wei, W. Zhu, and C. Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[12] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, pages 3606–3613, 2014.

[13] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[14] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2023.

[15] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

[16] J. Dong, J. Li, T. Zheng, and W. Lin. HybridFlow: Resource-adaptive subtask routing for efficient edge-cloud LLM inference. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 25922–25943. PMLR, 2026. URL https://proceedings.mlr.press/v306/dong26g.html.

[17] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[18] K. Drossos, S. Lipping, and T. Virtanen. Clotho: an audio captioning dataset. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 736–740, 2020.

[19] G. Du and W. Lin. NestedVLA: Learning to consolidate and generate skills for vision-language-action models. In *Advances in Neural Information Processing Systems*, 2026. URL https://arxiv.org/abs/2610.23271.

[20] G. Du, H. Deng, J. Su, and Y. Huang. End-to-end rain streak removal with raw images. *arXiv preprint arXiv:2312.13304*, 2023.

[21] G. Du, R. Jiang, S. Yang, H. Li, W. Chen, K. Li, S. K. Goh, and H.-K. Tang. Impacts of darwinian evolution on pre-trained deep neural networks. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1907–1912. IEEE, 2024.

[22] G. Du, J. Lee, J. Li, R. Jiang, Y. Guo, S. Yu, H. Liu, S. K. Goh, H.-K. Tang, D. He, and M. Zhang. Parameter competition balancing for model merging. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[23] G. Du, J. Li, H. Liu, R. Jiang, S. Yu, Y. Guo, S. K. Goh, and H.-K. Tang. Knowledge fusion by evolving weights of language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 11727–11742, 2024.

[24] G. Du, Z. Fang, J. Li, J. Li, R. Jiang, S. Yu, Y. Guo, Y. Chen, S. K. Goh, H.-K. Tang, D. He, H. Liu, and M. Zhang. Neural parameter search for slimmer fine-tuned models and better transfer. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025. doi: 10.18653/v1/2025.acl-long.1570. URL https://aclanthology.org/2025.acl-long.1570/.

[25] G. Du, Z. Li, X. Zhou, J. Li, Z. Shi, W. Lin, H.-K. Tang, X. Li, F. Liu, W. Wang, M. Zhang, and J. Li. Knowledge fusion of large language models via modular skillpacks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

[26] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[27] Y. Dubois, B. Galambosi, P. Liang, and T. B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

[28] Z. Fang, G. Du, S. Yu, Y. Guo, Y. Zhang, Y. Cao, J. Li, H.-K. Tang, and S. K. Goh. To see a world in a spark of neuron: Disentangling multi-task interference for training-free model merging. *arXiv preprint arXiv:2503.05320*, 2025.

[29] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[30] A. A. Gargiulo, D. Crisostomi, M. S. Bucarelli, S. Scardapane, F. Silvestri, and E. Rodola. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18695–18705, 2025.

[31] A. P. Gema, J. O. J. Leang, G. Hong, A. Devoto, A. C. M. Mancino, R. Saxena, X. He, Y. Zhao, X. Du, M. R. G. Madani, et al. Are we done with mmlu? *arXiv preprint arXiv:2406.04127*, 2024.

[32] A. P. Gema, J. O. J. Leang, G. Hong, A. Devoto, A. C. M. Mancino, R. Saxena, X. He, Y. Zhao, X. Du, M. R. G. Madani, et al. Are we done with mmlu? *arXiv preprint arXiv:2406.04127*, 2024.

[33] Y. Gong, J. Yu, and J. R. Glass. Vocalsound: A dataset for improving human vocal sounds recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 151–155, 2022.

[34] Y. Gong, H. Luo, A. H. Liu, L. Karlinsky, and J. R. Glass. Listen, think, and understand. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[35] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6325–6334, 2017.

[36] H. Gu, W. Li, L. Li, Q. Zhu, M. Lee, S. Sun, W. Xue, and Y. Guo. Delta decompression for moe-based llms compression. *arXiv preprint arXiv:2502.17298*, 2025.

[37] Y. Gu, Y. Wang, Z. Yan, Y. Zhang, Q. Zhou, F. Wu, and H. Yang. InfiFPO: Implicit model fusion via preference optimization in large language models. In *Advances in Neural Information Processing Systems*, volume 38, pages 15645–15672, 2025. doi: 10.52202/085713-0529. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1730c56d7d415ec0fb1f9fa70b271c8f-Abstract-Conference.html.

[38] Y. Gu, S. Cai, Z. Wang, W. Wang, Y. Wang, P. Wang, S. Huang, S. Lu, J. Wu, and H. Yang. FeatCal: Feature calibration for post-merging models. In *Advances in Neural Information Processing Systems*, 2026. URL https://arxiv.org/abs/2605.13030.

[39] Y. Gu, Y. Wang, Z. Li, S. Cai, Y. Liu, J. Li, Z. Wang, and H. Yang. SMAT: Simple and efficient merge-aware training, 2026. URL https://arxiv.org/abs/2609.33437.

[40] D. Gurari, Q. Li, A. J. Stangl, A. Guo, C. Lin, K. Grauman, J. Luo, and J. P. Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3608–3617, 2018.

[41] Y. He, Y. Hu, Y. Lin, T. Zhang, and H. Zhao. Localize-and-stitch: Efficient model merging via sparse task arithmetic. *Transactions on Machine Learning Research (TMLR)*, 2024.

[42] P. Helber, B. Bischke, A. Dengel, and D. Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[43] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[44] E. J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[45] C. Huang, P. Ye, T. Chen, T. He, X. Yue, and W. Ouyang. Emr-merging: Tuning-free high-performance model merging. *arXiv preprint arXiv:2405.17461*, 2024.

[46] D. A. Hudson and C. D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6700–6709, 2019.

[47] G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[48] J. Jin, H. Liu, Y. Bai, Y. Lou, Z. Wang, T. Yuan, J. Chen, Y. Zhu, F. Zeng, X. Zhu, T. Feng, and Y. Xu. Unveiling fine-grained visual traces: Evaluating multimodal interleaved reasoning chains in multimodal STEM tasks, 2026. URL https://arxiv.org/abs/2604.19697.

[49] X. Jin, X. Ren, D. Preotiuc-Pietro, and P. Cheng. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

[50] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV workshops*, pages 554–561, 2013.

[51] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[52] J. Li, G. Du, J. Li, S. K. Goh, W. Wang, Y. Wang, F. Liu, H.-K. Tang, S. Alharbi, D. He, et al. Multi-modality expansion and retention for llms through parameter merging and decoupling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025. doi: 10.18653/v1/2025.acl-long.1491. URL https://aclanthology.org/2025.acl-long.1491/.

[53] J. Li, S. Song, G. Du, N. Wong, X. Liu, Y. Li, M. Zhang, J. Li, and X. Li. D-qrelo: Training-and data-free delta compression for large language models via quantization and residual low-rank approximation. *arXiv preprint arXiv:2604.16940*, 2026.

[54] S. Li, G. Du, A. Zhao, W. Lin, T. Yuan, and X. Shen. Post-training shifts confidence: A three-stage analysis of how SFT, RL, and OPD shape CoT calibration. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 34126–34141, Budapest, Hungary, 2026. Association for Computational Linguistics. URL https://aclanthology.org/2026.findings-emnlp.1602/.

[55] T. Li, W.-L. Chiang, E. Frick, L. Dunlap, T. Wu, B. Zhu, J. E. Gonzalez, and I. Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.

[56] X. Li, T. Zhang, Y. Dubois, R. Taori, I. Gulrajani, C. Guestrin, P. Liang, and T. B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models, 2023.

[57] Y. Li, Y. Du, K. Zhou, J. Wang, W. X. Zhao, and J. Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 292–305, 2023.

[58] Z. Li, G. Du, W. Guo, Y. Zhou, X. Li, W. Wang, F. Liu, Y. Wang, D. Ye, M. Zhang, et al. Multi-objective large language model alignment with hierarchical experts. *arXiv preprint arXiv:2505.20925*, 2025.

[59] Z. Li, G. Du, Z. Shi, W. Guo, W. Yao, Y. Zhou, J. Zhang, and J. Li. Skill weaving: Efficient LLM improvement via modular skillpacks. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 40000–40023. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1989. URL https://aclanthology.org/2026.findings-acl.1989/.

[60] B. Lin, B. Zhu, Y. Ye, M. Ning, P. Jin, and L. Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

[61] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

[62] J. Lin, C. Zhu, P. J. Kneuertz, Y. Bai, and Y. Xue. Medcausalx: Adaptive causal reasoning with self-reflection for trustworthy medical vision-language models. *arXiv preprint arXiv:2603.23085*, 2026.

[63] H. Liu, C. Li, Y. Li, and Y. J. Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26296–26306, 2024.

[64] J. Liu, G. Xiao, K. Li, J. D. Lee, S. Han, T. Dao, and T. Cai. Bitdelta: Your fine-tune may only be worth one bit. *Advances in Neural Information Processing Systems (NeurIPS)*, 37: 13579–13600, 2024.

[65] P. Lu, S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, and A. Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 2507–2521, 2022.

[66] Z. Lu, C. Fan, W. Wei, X. Qu, D. Chen, and Y. Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. *arXiv preprint arXiv:2406.15479*, 2024.

[67] R. Luo, Z. Zhao, M. Yang, J. Dong, D. Li, P. Lu, T. Wang, L. Hu, M. Qiu, and Z. Wei. Valley: Video assistant with large language model enhanced ability. *arXiv preprint arXiv:2306.07207*, 2023.

[68] M. Maaz, H. A. Rasheed, S. Khan, and F. Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

[69] D. Marczak, S. Magistri, S. Cygert, B. Twardowski, A. D. Bagdanov, and J. van de Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[70] K. Marino, M. Rastegari, A. Farhadi, and R. Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3195–3204, 2019.

[71] X. Mei, C. Meng, H. Liu, Q. Kong, T. Ko, C. Zhao, M. D. Plumbley, Y. Zou, and W. Wang. Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research. *IEEE ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 32:3339–3354, 2024.

[72] C. Meng, P. Feng, R. Fu, H. L. Lee, X. Du, Z. Kang, Z. Zhang, W. Zhou, C. Ouyang, and Z. Gan. Group cognition learning: Making everything better through governed two-stage agents collaboration. *arXiv preprint arXiv:2605.00370*, 2026.

[73] C. Meng, P. Feng, J. J. Pang, C.-T. Liao, R. Fu, Z. Kang, Z. Gan, and C. Ouyang. Adaptive hierarchical representation alliance for multimodal learning. *arXiv preprint arXiv:2608.22863*, 2026.

[74] C. Meng, J. Luo, P. Feng, Z. Yan, C. Hu, Z. Gan, and C. Ouyang. Mitigating shared-private branch imbalance via dual-branch rebalancing for multimodal sentiment analysis. *arXiv preprint arXiv:2604.25179*, 2026.

[75] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen. DCASE2017 challenge setup: Tasks, datasets and baseline system. In *Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, pages 85–92, 2017.

[76] A. Panagopoulou, L. Xue, N. Yu, J. Li, D. Li, S. Joty, R. Xu, S. Savarese, C. Xiong, and J. C. Niebles. X-instructblip: A framework for aligning x-modal instruction-aware representations to llms and emergent cross-modal reasoning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[77] A. Panigrahi, N. Saunshi, H. Zhao, and S. Arora. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*, 2023.

[78] B. Ping, S. Wang, H. Wang, X. Han, Y. Xu, Y. Yan, Y. Chen, B. Chang, Z. Liu, and M. Sun. Delta-come: Training-free delta-compression with mixed-precision for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[79] M. Qi, R. Fu, P. Feng, X. Li, Y. Cai, Y. Guo, S. Zhang, S. J. Fong, L. Ma, and B. Li. CACR: Reinforcing temporal answer grounding in instructional video via candidate-aware causal reasoning. *arXiv preprint arXiv:2606.08436*, 2026.

[80] H. Qiu, X. Wu, L. Feng, and K. C. Tan. Fine-grained model merging via modular expert recombination. *arXiv preprint arXiv:2602.06552*, 2026. URL https://arxiv.org/abs/2602.06552.

[81] H. Qiu, X. Wu, and K. C. Tan. Towards adaptive continual model merging via manifold-aware expert evolution. *arXiv preprint arXiv:2604.22464*, 2026. URL https://arxiv.org/abs/2604.22464.

[82] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[83] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[84] V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

[85] A. Singh, V. Natarajan, M. Shah, Y. Jiang, X. Chen, D. Batra, D. Parikh, and M. Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8317–8326, 2019.

[86] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *IJCNN*, pages 1453–1460. IEEE, 2011.

[87] G. Stoica, P. Ramesh, B. Ecsedi, L. Choshen, and J. Hoffman. Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*, 2024.

[88] W. Sun, Q. Li, Y.-a. Geng, and B. Li. Cat merging: A training-free approach for resolving conflicts in model merging. *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[89] A. Tang, L. Shen, Y. Luo, N. Yin, L. Zhang, and D. Tao. Merging multi-task models via weight-ensembling mixture of experts. In *ICML*, 2024.

[90] A. Tang, L. Shen, Y. Luo, S. Xie, H. Hu, L. Zhang, B. Du, and D. Tao. Zero-shot sparse mixture of low-rank experts construction from pre-trained foundation models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–12, 2025. doi: 10.1109/TPAMI.2025.3612480.

[91] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[92] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023.

[93] F. Wan, X. Huang, D. Cai, X. Quan, W. Bi, and S. Shi. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*, 2024.

[94] K. Wang, N. Dimitriadis, G. Ortiz-Jimenez, F. Fleuret, and P. Frossard. Localizing task information for improved model merging and compression. *arXiv preprint arXiv:2405.07813*, 2024.

[95] Q. Wang, J. Ke, M. Tomizuka, K. Keutzer, and C. Xu. Dobi-SVD: Differentiable SVD for LLM compression and some new perspectives. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[96] W. Wang, Y. Gu, S. Cai, Y. Wang, P. Wang, J. Wu, and H. Yang. E-PMQ: Expert-guided post-merge quantization with merged-weight anchoring, 2026. URL https://arxiv.org/abs/2605.16882.

[97] X. Wang, Y. Zheng, Z. Wan, and M. Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*, 2024.

[98] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, T. Li, M. Ku, K. Wang, A. Zhuang, R. Fan, X. Yue, and W. Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[99] Y. Wang, Z. Yan, Y. Zhang, Q. Zhou, Y. Gu, F. Wu, and H. Yang. InfiGFusion: Graph-on-logits distillation via efficient gromov–wasserstein for model fusion. In *Advances in Neural Information Processing Systems*, volume 38, pages 119677–119713, 2025. doi: 10.52202/085713-3996. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/ad7c21456da003836ef6be2261e940df-Abstract-Conference.html.

[100] Y. Wang, Y. Gu, S. Lu, Y. Yang, Z. Yan, C. Xie, J. Wu, and H. Yang. Access sets matter: Budgeting expert reads for scalable weight-space model merging, 2026. URL https://arxiv.org/abs/2605.29489. ICML 2026 Workshop on Weight-Space Symmetries.

[101] Y. Wang, Y. Gu, Y. Zhang, Q. Zhou, Z. Yan, C. Xie, X. Wang, J. Yuan, and H. Yang. Model merging scaling laws in large language models. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 126402–126437. PMLR, 2026. URL https://proceedings.mlr.press/v306/wang26bl.html.

[102] Y. Wang, Y. Li, J. Liu, F.-L. Zhang, X. Zheng, and N. A. Dodgson. RL-ScanIQA: Reinforcement-learned scanpaths for blind 360° image quality assessment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 37401–37412, 2026.

[103] G. Wei, X. Yuan, Y. Zhou, H. Jing, Y. Liu, X. Qi, C. Zhao, H. Zhang, and R. Xiao. Open-text aerial detection: A unified framework for aerial visual grounding and detection. *arXiv preprint arXiv:2602.07827*, 2026.

[104] Y. Wei, Z. Hu, L. Shen, Z. Wang, Y. Li, C. Yuan, and D. Tao. Task groupings regularization: Data-free meta-learning with heterogeneous pre-trained models. In *ICML*, 2024.

[105] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. 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.

[106] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[107] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva. Sun database: Exploring a large collection of scene categories. *IJCV*, 119:3–22, 2016.

[108] X. Xiao, Y. Zhang, X. Li, T. Wang, X. Wang, Y. Wei, J. Hamm, and M. Xu. Visual instance-aware prompt tuning. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 2880–2889, 2025. doi: 10.1145/3746027.3754858.

[109] X. Xiao, X. Li, C. Han, T. Wang, L. Zhao, Y. Zhang, G. Hu, R. Jiang, X. Li, X. Wang, and M. Xu. Adapting vision foundation models with cascaded semantics. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=SSsobNZJPO.

[110] X. Xiao, X. Li, Y. Zhang, C. Han, T. Liu, T. Wang, R. Jiang, J. Hamm, X. Wang, and M. Xu. Layer-specific prompt fusion discovery via differentiable search in vision foundation models. In *Computer Vision – ECCV 2026*, volume 17077 of *Lecture Notes in Computer Science*, pages 391–409. Springer, 2026. doi: 10.1007/978-3-032-37041-9_21.

[111] X. Xiao, C. Liu, C.-T. Liao, Y. Zhang, Q. Lan, Y. Wei, L. Zhao, J. Wang, J. Gu, M. Ye, T. Wang, and H. Xu. Staying VIGILant: Mitigating visual laziness via counterfactual visual alignment in MLLMs. In *Computer Vision – ECCV 2026*, volume 17014 of *Lecture Notes in Computer Science*, pages 287–305. Springer, 2026. doi: 10.1007/978-3-032-37232-1_16.

[112] X. Xiao, C. Ma, Y. Zhang, C. Liu, Z. Wang, Y. Li, L. Zhao, G. Hu, T. Wang, and H. Xu. Not all directions matter: Towards structured and task-aware low-rank model adaptation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2132–2154. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.97.

[113] J. Xu, T. Mei, T. Yao, and Y. Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5288–5296, 2016.

[114] R. Xu, X. Wang, T. Wang, Y. Chen, J. Pang, and D. Lin. Pointllm: Empowering large language models to understand point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[115] P. Yadav, D. Tam, L. Choshen, C. A. Raffel, and M. Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2024.

[116] Z. Yan, Y. Zhang, B. He, Y. Fu, Q. Zhou, Z. Sang, C. Ji, S. Zhang, F. Wu, and H. Yang. Infifusion: A unified framework for enhanced cross-model reasoning via llm fusion. *arXiv preprint arXiv:2501.02795*, 2025.

[117] A. Yang, B. Zhang, B. Hui, B. Gao, B. Yu, C. Li, D. Liu, J. Tu, J. Zhou, J. Lin, et al. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[118] E. Yang, L. Shen, G. Guo, X. Wang, X. Cao, J. Zhang, and D. Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024.

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

[120] Y. Yang, Y. Li, H. Wang, X. Wei, J. Yu, Y. Chen, and G. Chen. Impart: Importance-aware delta-sparsification for improved model compression and merging in llms. *arXiv preprint arXiv:2504.13237*, 2025.

[121] Z. Yang, F. Wan, L. Zhong, C. Huang, G. Liang, and X. Quan. Fusechat-3.0: Preference optimization meets heterogeneous model fusion. *arXiv preprint arXiv:2503.04222*, 2025.

[122] L. Yu, B. Yu, H. Yu, F. Huang, and Y. Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

[123] N. Yuval. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[124] J. Zhang, J. Liu, J. He, et al. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:12589–12610, 2023.

[125] Q. Zhang, M. Chen, A. Bukharin, N. Karampatziakis, P. He, Y. Cheng, W. Chen, and T. Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023.

[126] L. Zheng, W. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[127] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *NeurIPS Datasets and Benchmarks Track*, 2023.

[128] S. Zheng and H. Wang. Free-merging: Fourier transform for efficient model merging. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3863–3873, 2025.

[129] Y. Zheng, R. Zhang, J. Zhang, Y. Ye, and Z. Luo. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410, 2024.

[130] Q. Zhou, Y. Zhang, Y. Gu, Y. Wang, Z. Sang, Z. Yan, Z. Li, S. Zhang, F. Wu, and H. Yang. Democratizing AI through model fusion: A comprehensive review and future directions. *Nexus*, 2025.

[131] Q. Zhou, Y. Zhang, Y. Gu, Y. Wang, Z. Yan, Z. Li, C. Y. Chung, and H. Yang. Model fusion for scalable and sustainable artificial intelligence: A review and outlook. *Journal of Modern Power Systems and Clean Energy*, 14(1):37–49, 2026.

[132] B. Zhu, E. Frick, T. Wu, H. Zhu, and J. Jiao. Starling-7b: Improving llm helpfulness & harmlessness with rlaif, November 2023.

[133] B. Zhu, B. Lin, M. Ning, Y. Yan, J. Cui, H. Wang, Y. Pang, W. Jiang, J. Zhang, Z. Li, C. Zhang, Z. Li, W. Liu, and L. Yuan. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[134] C. Zhu, Y. Lin, J. Shao, J. Lin, and Y. Wang. Pathology-aware prototype evolution via llm-driven semantic disambiguation for multicenter diabetic retinopathy diagnosis. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 9196–9205, 2025.

[135] C. Zhu, Y. Lin, S. Chen, Y. Wang, and J. Lin. Medeyes: Learning dynamic visual focus for medical progressive diagnosis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 13916–13924, 2026.

[136] C. Zhu, J. Zeng, J. Jiang, J. Lin, and Y. Wang. Medsynapse-v: Bridging visual perception and clinical intuition via latent memory evolution, 2026. URL https://arxiv.org/abs/2604.26283.

[137] D. Zhu, Z. Sun, Z. Li, T. Shen, K. Yan, S. Ding, K. Kuang, and C. Wu. Model tailor: Mitigating catastrophic forgetting in multi-modal large language models. *arXiv preprint arXiv:2402.12048*, 2024.

A Novelty and Contribution

DiDi-Merging combines a shared–expert decomposition, differentiable rank allocation under a coupled budget, and parameter-space refinement. Its contribution is the consolidation formulation and evaluated accuracy–storage trade-off, rather than a new SVD or low-rank primitive.

Static methods [115, 22] form one input-agnostic policy. In contrast, DiDi retains task-specific capacity selected by known task identity in the main evaluation. TALL-Mask and EMR [94, 45] represent task variation using masks and scalars around shared dense state; Twin and FREE [66, 128] use their corresponding shared and expert representations. DiDi compresses both shared and expert matrices through joint rank allocation. Storage comparisons must use a common convention for the backbone, auxiliary state, and any learned router.

TSV-C and SMILE [30, 90] provide other expert-based operating points. Comparisons with these methods in the result tables concern the specified checkpoints and budgets. LoRA-MoE [124] addresses mixtures of low-rank modules and routing; DiDi's fitting signal instead comes from already fine-tuned model parameters. Retaining separate task components makes conditional reconstruction possible, but does not establish forget-free incremental learning or constant accuracy as tasks are added under a fixed total budget.

For broader multimodal context, related studies examine hierarchical shared–private representation learning, governed agent collaboration, reinforcement-learned panoramic scanpaths for quality assessment, and temporal answer grounding in instructional video and multimodal STEM reasoning evaluation [73, 72, 102, 79, 48]. Complementary visual adaptation work studies instance-aware prompting, cascaded semantics, layer-specific prompt fusion, structured low-rank updates, and counterfactual visual alignment [108–110, 112, 111]. Earlier visual studies also investigate evolutionary refinement of pretrained networks and RAW-image restoration [21, 20].

B Additional Analysis

B.1 Effectiveness of Differentiable Rank Allocation

To further demonstrate the effectiveness of our differentiable rank allocation strategy, we compared it with several non-optimization-based rank allocation methods. In the ViT-B/32 vision experiments, we constrained the total rank budget to $0.4\times$ the number of parameters and evaluated uniform allocation, random allocation, and energy-based allocation. The energy-based method dynamically assigns ranks by computing the cumulative singular value energy across different tasks and layers. For random allocation, we report the average results over five runs. As shown in Tab. 8, random allocation can degrade model performance, while the energy-based strategy offers minimal improvement, underscoring the advantage of our approach.

B.2 Controlled Comparison Against Static Merge + Uniform-rank LoRA

To address the concern that the gains of `DiDi-Merging` might be attributed primarily to the LoRA recovery stage rather than to differentiable rank allocation, we constructed a controlled comparison at a strictly matched parameter budget. We pair four static merging strategies (Simple Averaging, TIES-Merging, Iso-CTS, and WUDI-Merging) with a LoRA recovery stage whose rank is fixed to a single uniform value r across all tasks and layers. The rank r is selected so that the total parameter count (backbone + shared + per-task LoRAs) falls within $\pm 2\%$ of $1.24\times$ the single fine-tuned model, matching the budget of `DiDi-Merging`-S.

Table 9 reports the results. The WUDI plus uniform-recovery configuration is a relevant alternative, but the table compares complete configurations rather than a fully crossed factorial. The four controls reach 80.4–84.6%, compared with 89.3% for DiDi-S. This supports the evaluated adaptive design at similar storage; a separate allocation-by-refinement factorial is needed to quantify their interaction or assign the complete gap to one component.

B.3 Hyperparameter Analysis

Tab. 10 summarizes the hyperparameters used for the different models. While the optimal settings vary slightly depending on the model type, `DiDi-Merging` remains stable when these values fluctuate

Table 8: Rank allocation strategies on ViT-B/32 at a total storage budget of $1.4\times$ the backbone.

Strategies	Uniformed	Random	Energy-based	**Ours**
ViT-B/32	81.5	80.2($\pm$0.4)	81.9	**85.5**
ViT-L/16	87.9	87.1($\pm$0.3)	88.1	**91.2**

Table 9: Controlled comparison against simpler data-free baselines that pair a static merge with a uniform-rank LoRA recovery. The LoRA rank is fixed across tasks and layers and chosen so that the total parameter count falls within $\pm 2\%$ of $1.24\times$ the single fine-tuned model. The table compares complete static-merge plus uniform-recovery alternatives with the adaptive-allocation configuration.

Method (@ $1.24\times$ budget)	**ViT-B/32**	**ViT-L/14**
Simple Averaging + uniform-rank LoRA	80.4	82.3
TIES-Merging + uniform-rank LoRA	81.5	83.4
Iso-CTS + uniform-rank LoRA	82.2	83.7
WUDI-Merging + uniform-rank LoRA	84.6	86.3
`DiDi-Merging`**-S (Ours)**	**89.3**	**93.9**

within a reasonable range. Sensitivity analyses on key hyperparameters (β, γ, and learning rates) yield consistent results, suggesting that our method is largely insensitive to specific configurations. Moreover, varying the static merging strategy used to initialize the shared expert consistently yields comparable results, further demonstrating the robustness of our approach.

We further analyzed the accuracy of dynamic model merging across different models, including vision models (ViT-B/32), large language models (LLaMA3-8B), and PEFT models (QWEN-14B finetuned with LoRA), as shown in Tab. 11. Among all the hyperparameters, β and the learning rates mainly affect optimization speed, with very little impact on accuracy. Different choices of loss functions and γ can have some effect. Specifically, γ is introduced to normalize the task reconstruction loss across different models, since task vectors vary in magnitude across tasks, different normalization values are required to keep the loss comparable to the compression ratio.

B.4 Performance Gain Beyond Individual Modalities in MLLM Merging

In dynamic model fusion for MLLMs, it has been observed that the merged models can achieve performance exceeding that of any single individual modality, sometimes reaching more than 100% relative to the original single-modality performance. Our MLLM dynamic merging experiments indicate that this improvement is not due to simply removing redundant parameters, but rather results from the shared knowledge across different modalities. This shared knowledge provides additional information, such as prior knowledge or instruction-following capabilities, which enhances model performance. When the shared knowledge is removed, the performance gain disappears, confirming its crucial role. These analyses agree with prior findings [52].

B.5 Comparison of PEFT Merging Strategies

We compare `DiDi-Merging` against representative PEFT merging strategies along several practical dimensions (Tab. 12). KnOTS [87] and Core Space are static methods that produce a full matrix or a shared basis without routing, leading to performance degradation when many tasks must coexist. Twin-Merging [66] and FREE-Merging [128] introduce a router and recover near-lossless performance, but at the cost of heavy per-task parameter overhead. `DiDi-Merging` preserves the dynamic-routing benefits of the latter group while keeping the per-task expert footprint slim through differentiable rank allocation.

Implementation on LoRA modules. `DiDi-Merging` extends seamlessly to PEFT settings, where each fine-tuned model is parameterised as $\boldsymbol{\theta}_t = \boldsymbol{\theta}_0 + \mathrm{LoRA}_t$ with LoRA_t a low-rank module specific to task t. The task vector reduces to the LoRA module itself, $\boldsymbol{\tau}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0 = \mathrm{LoRA}_t$, so the static merge of task vectors is $\boldsymbol{M}_s = \mathrm{avg}(\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_T) = \mathrm{LoRA}_s$, and each expert residual is

Table 10: Hyperparameters used in different models. lr_1 and lr_2 denote learning rates for the *Find-Optimal-Rank* and *Low-rank Module Update* stages, respectively.

Model	*loss*	β	γ	lr_1	lr_2
Vision	MSE	20	300	0.2	0.001
LLM	L1	30	500	0.2	0.001
PEFT	MSE	20	100	0.2	0.001
MLLM	L1	50	500	0.2	0.001

Table 11: Impact of hyperparameters on accuracy across different models.

Hyper-Params		**Vision**	**LLM**	**PEFT**
loss	MSE	**89.8**	54.8	**73.5**
	L1	89.7	**55.0**	73.2
γ	100	89.7	54.7	**73.5**
	300	**89.8**	54.8	73.2
	500	89.4	**55.0**	73.1

Table 12: Comparison of PEFT merging strategies. `DiDi-Merging` combines a dynamic router with lightweight per-task LoRA outputs while preserving near-lossless performance.

Method (→)	**KnOTS & Core Space**	**Twin-Merging & FREE-Merging**	`DiDi-Merging` **(Ours)**
Input	LoRA	LoRA	LoRA
Output	Full matrix	LoRA	LoRA
Router	None	Yes	Yes
Additional parameters	None	Heavy	Lightweight
Type	Static	Dynamic	Dynamic
Performance	Degradation	Near-lossless	Near-lossless

$\boldsymbol{M}_t = \boldsymbol{\tau}_t - \boldsymbol{M}_s = \mathrm{LoRA}_t - \mathrm{LoRA}_s$. The dynamic merging form (Eq. 1) then specialises to:

$$\begin{aligned}\boldsymbol{\theta}^*(\boldsymbol{x}) &= \boldsymbol{\theta}_0 + \tilde{\boldsymbol{M}}_s + \sum_{t=1}^{T} w_t(\boldsymbol{x})\, \tilde{\boldsymbol{M}}_t \\ &= \boldsymbol{\theta}_0 + \widetilde{\mathrm{LoRA}}_s + \sum_{t=1}^{T} w_t(\boldsymbol{x})\, \widetilde{\mathrm{LoRA}_t - \mathrm{LoRA}_s} \\ &= \boldsymbol{\theta}_0 + \mathrm{LoRA}^*,\end{aligned} \tag{17}$$

where $\widetilde{\cdot}$ denotes the differentiable rank-truncated version of a matrix and $\mathrm{LoRA}^* = \widetilde{\mathrm{LoRA}}_s + \sum_{t=1}^{T} w_t(\boldsymbol{x})\, \widetilde{\mathrm{LoRA}_t - \mathrm{LoRA}_s}$ is the merged LoRA module. Hence `DiDi-Merging` on the full model $\boldsymbol{\theta}$ is equivalent to `DiDi-Merging` applied directly to the LoRA factors: the backbone $\boldsymbol{\theta}_0$ never enters the optimisation, and only the LoRA factors are stored, decomposed, and refined. This makes the framework directly compatible with existing PEFT pipelines. Empirical results validating this equivalence on Qwen-14B LoRA adapters are reported in App. C.1.

B.6 Task Relationships and Rank Allocation

Fig. 7 compares pairwise task-vector similarity with the optimal shared-rank ratio across four vision tasks, and the two heatmaps align tightly. MNIST and SVHN, both digit-classification tasks, form the most similar pair (similarity 0.18) and receive the largest shared-rank ratio (0.42), while less related pairs settle at a lower 0.28. This mapping emerges purely from optimisation, without hand-tuned similarity priors, and is consistent with the separately reported extension from 8 to 20 vision tasks (Tab. 3) retains $\sim 99\%$ of fine-tuned accuracy without re-tuning the rank budget.

B.7 Storage Accounting

Table 13 gives the per-module bit cost across methods. Heavy-shared baselines duplicate the backbone in their shared component ($32mn$); heavy-expert baselines pay $32Tr_t(m+n)$ on uncompressed experts. Differentiable allocation keeps both small ($\bar{r}_s, \bar{r}_t \sim 15$ on ViT-B/32; App. B.7), giving the $\leq 1.4\times$ total budget. The optimised ranks themselves reflect architectural structure: attention-output projections receive the smallest rank, MLP modules the largest, and the ordering is preserved across the $1.24\times$ and $1.4\times$ budgets, matching the relative information density of each module type.

This section makes the parameter budgets reported in the main paper ($1.24\times$ / $1.4\times$ / $2.0\times$) explicit and clarifies how they are computed. Let P_{model} denote all parameters in one pretrained backbone, R

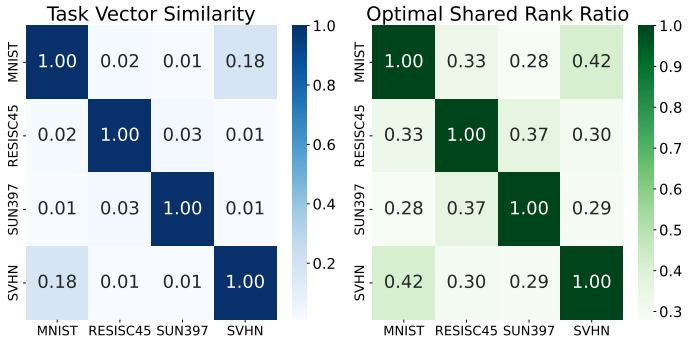


Figure 7: Task-vector similarity (left) vs. optimal shared-rank ratio (right) for pairs of four tasks.

Table 13: Per-module storage cost (in bits) for a module with dimensions $d_{\text{in}} \times d_{\text{out}}$ shared across T tasks. $P{=}d_{\text{in}}d_{\text{out}}$, R is the router size, charged once globally; $R = 0$ for the task-ID selector. Floating-point parameters are stored at 32 bits, binary masks at 1 bit per parameter. EMR additionally stores one global 32-bit rescaler per task, charged once per complete model rather than per module. r_{tsv} is a fixed rank in TSV-C; r_s, r_t are adaptively determined by differentiable rank optimisation in `DiDi-Merging`. Rows colour-coded by regime: static (gray), heavy shared (green), heavy expert (blue), balanced (pink).

Method	Backbone	Router	Shared	Experts
TIES-Merging [115] (*etc.*)	$32P$	0	0	0
TALL-Mask [94]	$32P$	$32R$	$32P$	TP
EMR-Merging [45]	$32P$	$32R$	$32P$	TP
TSV-C [30]	$32P$	$32R$	0	$32\, r_{\text{tsv}}(d_{\text{in}} + d_{\text{out}})$
`DiDi-Merging` **(Ours)**	$32P$	$32R$	$32\, r_s(d_{\text{in}} + d_{\text{out}})$	$32 \sum_{t=1}^{T} r_t(d_{\text{in}} + d_{\text{out}})$

the number of parameters in a learned router, and B_{factors} the shared and task-specific factor count in Eq. 12. Total fp32 storage is

$$32\big(P_{\text{model}} + B_{\text{factors}} + R\big) \quad \text{bits}, \tag{18}$$

and the normalized ratio is $1 + B_{\text{factors}}/P_{\text{model}} + R/P_{\text{model}}$. The backbone is included once. The main experiments use an ideal task-ID selector, for which $R = 0$; a learned router would be counted once at model level. Per-module tables decompose the weight terms, and their router column must not be charged again for every layer.

The bit-level decomposition across baselines is reported in Tab. 13 (this appendix).

Average optimal ranks across modules. The ranks found by the differentiable optimisation vary across module types (because $d_{\text{in}}, d_{\text{out}}$ differ) and across tasks (because the spectra of task vectors differ). Tab. 14 reports the average optimal rank for each module type in ViT-B/32, aggregated over the 12 transformer blocks and the $T{=}8$ vision tasks at the $1.24\times$ (`DiDi-Merging`-S) and $1.4\times$ (`DiDi-Merging`-M) budgets. Two patterns emerge: (i) ranks scale with module size: attention output ($768{\times}768$) requires the smallest rank, attention input ($2304{\times}768$) is intermediate, and MLP modules ($3072{\times}768$ and $768{\times}3072$) require the largest; and (ii) the per-module ratio between the two budgets is roughly constant ($\sim 1.7\times$), consistent with the linear scaling of the budget delta from $0.24\times$ to $0.4\times$, indicating that the optimisation rescales ranks proportionally rather than redistributing them across modules.

A finer-grained breakdown that separates the shared and per-task expert ranks is reported in Tab. 15. Within each module type, $\bar{r}_s$ and $\bar{r}_t$ are tightly correlated, indicating the shared module captures cross-task structure that each task vector also relies on. Plugging the finer-grained averages back into Eq. 12 recovers the $1.24\times$ budget within rounding noise.

Table 14: Average optimal ranks of different modules in ViT-B/32, averaged over the 12 transformer blocks and the T=8 vision tasks. `DiDi-Merging`-S and `DiDi-Merging`-M denote the $1.24\times$ and $1.4\times$ parameter budgets, respectively.

Module	attn-in-proj	attn-out-proj	mlp-c-fc	mlp-c-proj
`DiDi-Merging`-S ($1.24\times$)	15	10	16	16
`DiDi-Merging`-M ($1.4\times$)	25	17	27	27

Table 15: Average optimal ranks $\bar{r}_s$ (shared) and $\bar{r}_t$ (per-task expert) of different modules in CLIP-ViT-B/32, averaged over the 12 transformer blocks and the T=8 vision tasks at the two reported budgets. Values plug back into Eq. (14) and reproduce the $1.24\times$ / $1.4\times$ ratios.

Module	$1.24\times$ budget		$1.4\times$ budget	
	$\bar{r}_s$	$\bar{r}_t$	$\bar{r}_s$	$\bar{r}_t$
Self-attn QKV (in_proj, 2304×768)	18.3	17.1	27.4	26.1
Self-attn Out (out_proj, 768×768)	16.1	14.9	27.7	26.4
MLP fc1 (c_fc, 3072×768)	13.0	11.7	21.3	19.7
MLP fc2 (c_proj, 768×3072)	8.9	8.5	17.3	16.7
Average	**14.1**	**13.0**	**23.4**	**22.2**

C Additional Results

C.1 Detailed PEFT Merging Results

PEFT methods [15] efficiently adapt large language models to new tasks, but storing separate adapters per task can be costly [124]. We merge LoRA [44] adapters of Qwen-14B [2] on three generative tasks [32], following [66, 128]. As Tab. 16 shows, DiDi-Merging-M matches prior baselines with fewer parameters, and DiDi-Merging-L matches their parameter overhead while attaining 0.7% higher performance.

C.2 Detailed Multimodal Results

We report the detailed results of each method on the corresponding datasets, including vision-language (Tab. 17), point-cloud (Tab. 18), audio, and video (Tab. 19) tasks, illustrating that DiDi-Merging consistently achieves superior performance while maintaining a slim parameter footprint. As an experimental detail, we evaluate DiDi-Merging on multiple multimodal tasks, ensuring that all models are fine-tuned under identical environments. Performance is measured in terms of accuracy or overall scores, as well as the percentage of performance retained relative to each model's fine-tuned counterpart. We compare DiDi-Merging with representative data-free dynamic mergers.

D Method Details

D.1 Differentiable Optimization of Ranks

To efficiently allocate ranks for shared and expert modules, we adopt a differentiable approach inspired by smooth SVD truncation. Each parameter matrix is decomposed via SVD, and a smooth truncation function with a learnable continuous rank r produces a low-rank approximation while keeping the rest of the module parameters frozen. The reconstructed task vectors, obtained by combining truncated shared and expert matrices, are optimized with a task reconstruction loss. A compression-aware term penalizes deviations from a target rank budget, balancing task fidelity and parameter efficiency. By jointly optimizing r across layers, the method adaptively allocates ranks between shared and expert matrices to maintain compact yet informative low-rank modules.

D.2 Optimization for Low-rank Module Update

During refinement, the shared and task-specific low-rank factors are updated with fixed ranks, while all pretrained backbone parameters remain frozen. The goal of this stage is to efficiently recover

Table 16: LoRA merging results on three generative tasks with QWEN-14B as the pretrained model.

Method	**#Params**	MMLU	Tru.QA	BBQ	**Average**
Zero-shot	x1	69.30	51.27	80.69	67.09
Fine-tuned	x3	68.35	54.34	93.53	72.07
EMR-Merging [45]	x2.09	67.82	55.01	90.13	70.98
Twin-Merging [66]	x2.13	68.32	55.76	90.98	71.68
FREE-Merging [128]	x2.04	68.83	57.39	92.14	72.78
DiDi-Merging-M(ours)	**x1.4**	68.85	57.32	92.44	72.87
DiDi-Merging-L(ours)	x2.0	**68.97**	**58.24**	**93.32**	**73.51**

Table 17: Performance of each method on seven multimodal vision–language QA tasks.

Task (→)	**7 Image Tasks**						
	VQAv2	GQA	TextVQA	VizWiz	ScienceQA	POPE	OK-VQA
Method (↓)	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.
Individuals	78.11	61.52	55.89	51.51	71.12	86.17	31.33
NaiveMC [ACL2024] [4]	59.73	45.83	42.29	47.87	68.52	79.41	24.28
MMER [ACL2025] [52]	77.95	61.85	55.74	52.26	71.16	86.58	31.27
FREE-Merging [ICCV2025] [128]	**78.05**	61.92	55.84	52.86	71.06	86.73	**31.31**
`DiDi-Merging` **(Ours)**	78.03	**62.08**	**55.88**	**53.12**	**72.32**	**87.83**	31.29

Table 18: Performance of each method on two point-cloud tasks, including ModelNet40 (classification) and Objaverse (captioning).

Task (→)	**2 Point Tasks**					
	ModelNet40	Objaverse-captioning				
Method (↓)	Acc.	BLEU-1	ROUGE-L	METEOR	Sentence-BERT	SimCSE
Individuals	21.27	4.73	8.51	12.02	44.18	46.31
NaiveMC [ACL2024] [4]	20.49	4.43	8.24	11.37	43.22	45.97
MMER [ACL2025] [52]	22.49	**5.06**	8.53	11.90	43.72	**46.51**
FREE-Merging [ICCV2025] [128]	21.14	4.95	8.49	11.80	43.69	46.18
`DiDi-Merging` **(Ours)**	**22.58**	4.92	**8.58**	**11.94**	**44.08**	46.37

Table 19: Performance of each method on three audio tasks and two video tasks, including TUT, VocalSound, MSVD, and MSRVTT for classification, and Clotho for captioning.

Task (→)	**3 Audio Tasks**					**2 Video Tasks**	
	TUT	VocalSound	Clotho			MSVD	MSRVTT
Method (↓)	Acc.	Acc.	CIDEr	SPICE	SPIDEr	Acc.	Acc.
Individuals	22.23	27.19	38.63	11.98	25.29	48.40	31.18
NaiveMC [ACL2024] [4]	29.50	31.80	37.56	11.61	24.61	44.53	29.31
MMER [ACL2025] [52]	34.14	42.88	38.49	11.93	25.18	48.12	30.43
FREE-Merging [ICCV2025] [128]	34.27	42.92	38.53	11.95	25.22	47.42	29.83
`DiDi-Merging` **(Ours)**	**34.49**	**43.02**	**38.56**	**11.98**	**25.25**	**48.37**	**30.89**

task-specific performance that may have been lost during low-rank truncation. The optimization is performed as follows:

- **Optimizer:** Adam with a small learning rate, typically in the range $1e{-4}$ to $1e{-3}$.
- **Batch-free:** Optimization is fully data-free; the supervision signal comes solely from the stored task vectors $\boldsymbol{\tau}_t$, so no original training data are required.
- **Regularization:** Optionally, a term $\lambda \|\boldsymbol{A}_{t,l}\boldsymbol{B}_{t,l} - \boldsymbol{M}_{t,l}^{(0)}\|_F^2$ or standard weight decay can be applied to stabilize updates.
- **Multi-task Optimization:** When optimizing multiple tasks simultaneously, different task losses are balanced using dynamic weighting. A temperature-based scaling factor is applied

to adjust the relative contribution of each loss over time, ensuring that no single task dominates the optimization and that all tasks maintain stable improvement.

This lightweight procedure produces compact per-task low-rank modules that respect the allocated rank budget, restoring task-specific fidelity while maintaining high efficiency and performance.

E Limitations

The main evaluation supplies task identity to select experts. A practical router may make errors, and its supervision, latency, and parameters must be included in an end-to-end deployment assessment. Data-free merging still requires the source fine-tuned checkpoints and performs optimization; it is not optimization-free. Parameter reconstruction is a surrogate for downstream behavior, and its relation to accuracy depends on the input distribution. Rank allocation and refinement add wall-clock cost compared with one-pass or CPU-based alternatives. A fixed total storage target can be maintained as tasks are added only by reallocating finite capacity; this is not a guarantee of constant accuracy or cost-free incremental addition. The current evidence leaves practical routing, fully crossed component attribution, and larger-scale task growth as further evaluations.

F Notation

Tab. 20 summarises the key symbols used throughout the paper.

G Broader Impact

`DiDi-Merging` reduces the storage and serving cost of multi-task systems, lowering the deployment barrier for collections of specialised models. Like all efficient deployment methods, lower cost can amplify both beneficial uses (low-resource personalisation, on-device multi-task inference) and harmful ones (easier serving of biased or unsafe fine-tunes). `DiDi-Merging` inherits the biases of its source checkpoints; we introduce no additional training data or labels. We see particular promise in high-stakes multimodal deployments such as medical diagnosis, where heterogeneous expert models for imaging, language, and clinical reasoning must coexist under tight storage and latency budgets [134, 135, 62, 136, 103, 9]; slim dynamic merging offers a path to deploy such expert ensembles without retraining or exposing the underlying patient data.

H Baselines details

This section provides a detailed description of the baselines, as outlined below.

- **Task Arithmetic** [47] first defines the concept of "task vectors" and merges these vectors into a pre-trained model to execute multi-task learning. The model is produced by scaling and adding the task vectors to the initial model as $\theta_m = \theta_{\text{init}} + \lambda \cdot \sum_{t=1}^{n} \tau_t$.
- **Ties-Merging** [115] further solves the task conflict problem in Task Arithmetic [47]. It eliminates redundant parameters and resolves symbol conflicts through three steps: Trim, Elect Sign, and Disjoint Merge.
- **DARE** [122] sets the majority of delta parameters to zero and rescales the rest by $\theta' = \theta \cdot (1/(1-p))$ where p is the proportion of delta parameters dropped, therefore efficiently reducing parameter redundancy.
- **PCB-Merging** [22] effectively adjusts parameter coefficients through balancing parameter competition within model population.
- **CAT-Merging** [88] introduces a training-free framework that trims conflict-prone task vector components via projection for linear weights and masking for normalization parameters.
- **TALL-Mask** [94] localizes the task-specific information in a multi-task vector, which deactivates irrelevant parts for each task in the merged vector using binary task masks.
- **EMR-Merging** [45] first selects a unified model from all weights, then generates lightweight task-specific modulators (masks and rescalers) to align direction and magnitude with each source model.

Table 20: Notation summary.

Symbol	Meaning
Problem setting	
T	Number of tasks
L	Number of layers
$\boldsymbol{\theta}_0$	Pre-trained backbone parameters
$\boldsymbol{\theta}_t$	Fine-tuned parameters for task t
$\boldsymbol{\tau}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$	Task vector for task t
λ_t	Coefficient of $\boldsymbol{\tau}_t$ in static merging
Dynamic merging form (Eq. 1)	
$\boldsymbol{M}_s$	Shared module
$\boldsymbol{M}_t$	Per-task expert module
$w_t(\boldsymbol{x})$	Input-dependent routing weight for task t
Low-rank decomposition (§4.2)	
$d_{\text{in}}, d_{\text{out}}$ (or m_l, n_l)	Module weight dimensions
r_s, r_t	Shared, per-task expert ranks
$r_{s,l}, r_{t,l}$	Per-layer shared / per-task ranks at layer l
$\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$	SVD factors
$\sigma_i, \tilde{\sigma}_i$	Singular values; smooth-truncated values
β	Smoothness factor of the truncation function
Low-rank module update (§4.3)	
$\boldsymbol{A}_{t,l}, \boldsymbol{B}_{t,l}$	LoRA factor matrices for task t at layer l
$\boldsymbol{A}_{s,l}, \boldsymbol{B}_{s,l}$	Shared LoRA factor matrices at layer l
L_{task}	Task reconstruction loss
$R_{\text{now}}, R_{\text{tar}}$	Current / target compression ratio
γ	Weight for compression-ratio penalty
Storage accounting (App. B.7)	
$P = d_{\text{in}} d_{\text{out}}$	Per-module parameter count
R	Router parameter count

- **Twin-Merging** [66] proposes a method that modularizes knowledge into shared and exclusive components, while compression reduces redundancy and improves parameter efficiency.
- **FREE-Merging** [128] reveals that task interference is evident in the frequency domain of model parameters.
- **TSV-Compression** [30] leverages low-rank space to define a new measure of task interference based on the interaction of singular vectors from different tasks.
- **NaiveMC** [4] composes existing MLLMs into a new model that retains each source model's modality-specific understanding.
- **MMER** [52] introduces a training-free approach for seamless multimodal expansion of LLMs through parameter merging and decoupling.

I Datasets details

SUN397 [107] is a scene classification dataset, which contains images in 397 classes, with a total of 108,754 images, and each class has at least 100 images.

Stanford Cars (Cars) [50] is a car classification dataset, which contains 196 classes of cars and a total of 16,185 images. Each class is split equally between the training and test sets.

RESISC45 [10] is a remote sensing image scene classification dataset. It contains 45 classes of scenes and a total of 31,500 images, with approximately 700 per class.

EuroSAT [42] is a satellite image classification dataset containing 27,000 labeled and geo-referenced images in 10 classes.

SVHN [123] is a real-world digital classification dataset extracted from house numbers in Google Street View images. There are 10 classes in total. The training set contains 73,257 samples, the test set contains 26,032 samples, and 531,131 additional simple samples can be used as additional training data.

GTSRB [86] is a traffic sign classification dataset, which contains 43 classes of traffic signs with a total sample size of more than 50,000.

MNIST [51] is a benchmark dataset for image classification. It contains grayscale images of handwritten digits in 10 classes. The number of images in the training and test sets is 60,000 and 10,000 respectively. The number of images in each class is balanced.

DTD [12] is a texture classification dataset, which contains 47 classes, a total of 5,640 images, with each class having approximately 120 images.

AlpacaEval-2 [56] comprises 805 instructions from five different datasets and assesses models using two metrics: length-controlled (LC) win rate and raw win rate (WR) [27]. GPT-4-Preview-1106 serves as both the baseline model and the evaluator for the other models.

MT-Bench [127] contains 80 multi-turn dialogues across eight categories, including writing, roleplay, reasoning, math, coding, extraction, STEM, and humanities. Each response is evaluated by GPT-4 on a scale from 1 to 10, with the average score reported for each dialogue turn across the 80 dialogues. We use GPT-4-0613 as the judge model following the official setting.

MMLU-Pro [98] is an enhanced version of the MMLU dataset, designed to address issues such as noisy data and reduced difficulty due to advances in model capabilities and increased data contamination. MMLU-Pro increases challenge levels by expanding multiple-choice options from 4 to 10, requiring reasoning across more questions, and incorporating expert-reviewed annotations for improved quality and reduced noise.

MMLU-redux [31] is a re-annotated subset of the MMLU dataset created through manual assessment from 14 human experts.

GPQA-Diamond [83] is a challenging knowledge benchmark crafted by PhD-level domain experts in biology, physics, and chemistry. The dataset contains questions that are straightforward for experts but difficult for laypersons. We evaluate the highest quality diamond set comprising 198 questions.

Arena-Hard [55] is a challenging instruction-following benchmark that closely aligns with the human preference ranking from Chatbot Arena, a crowd-sourced platform for evaluating LLMs. It spans 250 high-quality topic clusters including 500 well-defined technical problem-solving queries. We report the win rate against GPT-4-0314 using GPT-4-Preview-1106 as the judge model.

GSM8K [13] contains grade-school math word problems for evaluating mathematical reasoning.

MATH [43] is a dataset of math problems ranging in difficulty from middle school to high school competition level. It tests a wide range of mathematical skills, including algebra, calculus, number theory, and probability.

AMC23 [117] refers to the 2023 American Mathematics Competition, featuring 25 multiple-choice questions that test advanced high school mathematics, including trigonometry, advanced algebra, and elements of calculus.

HumanEval [6] evaluates code generation capabilities by presenting models with function signatures and docstrings and requiring them to implement the function body in Python.

MBPP [1] is a dataset of simple programming problems designed to assess the ability of models to generate short Python code snippets from natural language descriptions.

J Implementation Details

In this section, we provide detailed descriptions of the experimental setup used for both LLM (§J.1) and multimodal LLM (MLLM) (§J.2) fine-tuning. We first outline the procedures for obtaining task-specific fine-tuned LLMs and constructing the associated datasets, followed by the fine-tuning

protocol for MLLMs across multiple modalities. All experiments are designed to ensure consistency and comparability, with hyperparameters and training configurations aligned with prior studies. All our experiments are conducted on an NVIDIA 8×A800-SXM4-80GB machine.

J.1 LLM Fine-tuning

In our LLM experiments, we follow a structured pipeline to fine-tune one model for each expert task.

SFT Experiments. We use the `Llama-Factory` library[2] [129] for implementation. All target models are fine-tuned for 3 epochs with a batch size of 128 and a maximum sequence length of 2048 tokens. A cosine learning rate schedule with a warmup ratio of 0.1 is employed.

DPO Experiments. We use the `alignment-handbook`[3] as the training framework. All post-SFT models are trained for one epoch with the same batch size and sequence length. Cosine learning rate scheduling with 0.1 warmup is applied. Checkpoints are saved every 100 steps, and the best checkpoint from the last two is selected. Detailed hyperparameter configurations for different models are listed in Tab. 21.

Dataset Construction. The dataset plays a critical role in enabling the LLM model merging approach, as demonstrated in the previous study FuseChat-3.0 [121]. The dataset is carefully assembled following these procedures:

- **Prompt Selection**: To enhance the target LLMs' abilities across instruction-following, math, coding, and Chinese, we select high-quality samples from open-source datasets. Filtering and preprocessing ensure relevance and quality.
- **Response Sampling**: For each prompt, responses are generated from four leading source LLMs using `vLLM`[4] [132] as the inference backend. Multiple sampling runs with different seeds ensure diversity. Sampling parameters are: for `Gemma-2-27B-it`, `Mistral-Large-Instruct-2407`, and `Llama-3.1-70B-Instruct`, temperature 0.8, top-p 0.95; for `Qwen-2.5-(Math)-72B-Instruct`, we use temperature 0.7, top-p 0.8, and repetition penalty 1.05.
- **Preference Pairs**: To build preference pairs, the best and worst responses from the same source model are selected. This intra-model pairing reduces reward bias due to heterogeneous response styles, prevents reward hacking, and ensures a reliable preference signal. For instruction-following and conversational data, responses are evaluated with an external reward model; for math and coding, rule-based verification is applied.

The final dataset $\mathcal{D}$ contains 158,667 entries: 94,539 for the SFT phase ($\mathcal{D}_{\text{SFT}}$) and 64,128 preference pairs for the DPO phase ($\mathcal{D}_{\text{DPO}}$). Dataset composition is summarized in Tab. 22.

J.2 MLLM Fine-tuning

For fine-tuning the original MLLMs, we employed consistent training data and model components across the four modalities, following the setup in NaiveMC [4]. Further details are provided in Tab. 23. Hyperparameters were chosen in line with prior studies [4, 63, 76, 60, 114]. During the alignment stage, only the connector parameters were updated, whereas in the full fine-tuning stage, both the connector and base LLM parameters were optimized. To improve training efficiency, we use DeepSpeed ZeRO Stage 3; Tab. 24 lists all hyperparameters.

[2] https://github.com/hiyouga/LLaMA-Factory
[3] https://github.com/huggingface/alignment-handbook
[4] https://github.com/vllm-project/vllm

Table 21: Hyperparameters for different target models during the SFT and DPO stages.

Target Model	SFT Learning Rate	DPO Learning Rate	DPO λ	DPO Loss Type
Llama-3.1-8B-Instruct	5×10^{-6}	8×10^{-7}	10	$\mathcal{L}_{\text{LN-DPO}}$
Qwen-2.5-7B-Instruct	2×10^{-6}	3×10^{-7}	0.01	$\mathcal{L}_{\text{DPO}}$

Table 22: The composition of the Implicit Knowledge Fusion dataset in SFT phase and DPO phase. As no suitable reward models were available for Chinese, we used all samples for SFT and omitted the DPO phase.

Category	Dataset	Count	$\#\mathcal{D}_{\text{SFT}}$	$\#\mathcal{D}_{\text{DPO}}$
Instruction Following	UltraFeedback	51,098	20,439	30,659
	Magpie-Pro-DPO	20,374	8,149	12,225
	HelpSteer2	9,435	3,774	5,661
Mathematics	OpenMathInstruct-2	51,803	40,188	11,615
Coding	LeetCode	3,113	1,877	1,236
	Self-Oss-Instruct-SC2	12,892	10,160	2,732
Chinese Language	Alpaca-GPT4-Zh	2,471	2,471	0
	Magpie-Qwen2-Pro-Zh	7,481	7,481	0
Total		158,667	94,539	64,128

Table 23: Training data and components of MLLMs for different modalities.

Modality	Modality Encoder	Connector	Alignment Data	Fine-tuning Data	Referenced Work
Image	CLIP-ViT-L-336px [17]	MLP	LCS 558K [114]	LLaVA-mixed 665K [114]	LLaVA-1.5 [63]
Audio	BEATs-Iter3+ [7]	Q-Former	WaveCaps 400K [71]	OpenAQA filtered 350K [34]	X-InstructBLIP [76]
Video	LanguageBind [133]	MLP	LCS 558K, Valley 702K [67]	Video-ChatGPT 100K [68], LLaVA-mixed sampled 140K	Video-LLaVA [60]
Point Cloud	Point Encoder [114]	MLP	PointLLM brief description 660K [114]	Point complex instruction 70K [114]	PointLLM [114]

Table 24: Hyperparameters of different MLLMs.

Stage	Hyperparameter	Image	Audio	Video	Point Cloud
Alignment-Stage	Batch size	256	256	256	128
	LR	1e-3	1e-3	1e-3	2e-3
	LR Schedule	cosine decay			
	Warmup Ratio	0.03			
	Epoch	1	1	1	3
Fine-tuning-Stage	Batch size	128	64	128	64
	LR	2e-5	1e-5	2e-5	2e-5
	LR Schedule	cosine decay			
	Warmup Ratio	0.03			
	Epoch	1	3	1	3

K Additional Formulation and Evaluation Details

K.1 Storage units and task-count scaling

The denominator in total-storage ratios is one dense 32-bit backbone with P_{model} parameters. For EMR, storing the pretrained backbone, a unified dense vector, T one-bit masks, and one global fp32 scalar per task costs

$$S_{\text{EMR}} = 2 + \frac{T}{32} + \frac{T}{P_{\text{model}}}. \tag{19}$$

The scalar term is negligible at model scale, giving $2.25\times$ for eight tasks and approximately $2.09\times$ for three. Counting masks as byte tensors would describe an unpacked implementation rather than the bit-packed storage convention. Under the same convention, TALL's backbone, unified vector, and masks contribute $2 + T/32$. Router storage, if present, is added separately.

For DiDi, the factor contribution is given by Eq. 12. At a fixed target budget, increasing T requires reducing or reallocating ranks. Holding ranks fixed instead gives a task-count-dependent storage cost. These two regimes must not be conflated: a fixed numerical budget is a design constraint, whereas retained accuracy as more tasks are consolidated is an empirical question.

K.2 What the refinement objective measures

For a linear layer with reconstruction error $E = \widetilde{W} - W$ and input second moment $\Sigma_x = \mathbb{E}[xx^\top]$,

$$\mathbb{E}\|Ex\|_2^2 = \mathrm{tr}(E\Sigma_x E^\top). \tag{20}$$

A Frobenius reconstruction objective corresponds to isotropic weighting of input directions. It is attractive when task data are unavailable, but smaller parameter error does not generally guarantee higher downstream accuracy in a nonlinear model. A correlation study would need to specify its ranks, tasks, refinement steps, and individual observations; the component-ablation gains in the existing tables answer a different question.

K.3 Interpretation of component comparisons

The existing vision table directly isolates removal of the refinement stage: the reductions are 4.3 points on ViT-B/32 and 4.4 on ViT-L/14. The uniform-rank recovery comparison is performed at approximately $1.24\times$ storage and supports the full adaptive design. These results do not, by themselves, determine a two-factor interaction effect, nor do they establish the same refinement gains for LLM or MLLM settings without corresponding stage-one measurements.

K.4 Routing and shared initialization

The formulation permits input-dependent routing, but the main numerical results use known task identity. Evaluating a learned router requires both route accuracy and the final merged model's task performance; unknown-task rejection and mixed-task batches are separate settings. The static aggregator initializes shared and residual matrices. Subsequent factor refinement changes this initialization, but does not make the final solution provably invariant to the aggregator or rank allocation. A conflicting-task aggregator-swap study is therefore a separate experiment, not a consequence of the reconstruction objective.

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 abstract and §1 state the three contributions (heavy-shared/heavy-expert characterization, differentiable rank allocation, and near-lossless performance under tight parameter budgets), each of which is supported by the experiments in §5 and the analysis in §6. Quantitative claims use the softened phrasing "match at $1.24\times$, surpass at $1.4\times$" consistent with the reported numbers.

   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: A dedicated Limitations section (§E) discusses the ideal-router assumption, the data-free-but-checkpoint-dependent setting, and the wall-clock cost of sequential rank optimization relative to closed-form SVD baselines.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper is empirical; it does not introduce new theorems. The few derivations (e.g., the equivalence between applying `DiDi-Merging` to full parameters and to LoRA factors in the PEFT-merging appendix) are stated with all intermediate steps in-line, and any assumptions are made explicit at the point of use.

   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: §4 fully describes the training-free smooth-SVD truncation, the differentiable rank-allocation objective, and the two-stage optimization. The appendix lists per-modality datasets, baselines, optimizer settings, learning rates, and hyperparameters (β, γ, loss type) used to obtain every reported number.

   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: The core DiDi-Merging algorithm is publicly available at https://github.com/WanyuGroup/NeurIPS2026-DiDi-Merging, with installation instructions and a minimal example. The current release does not include the benchmark-specific training and evaluation pipelines or configurations needed to reproduce every reported result. All datasets used in the paper are public benchmarks (image classification, GLUE, AudioCaps, ScienceQA, ModelNet40, etc.); the appendix lists their sources. The method, hyperparameters, and baselines are fully described in the main paper and appendix.

   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: §5 states the modalities, backbones, task suites, and parameter budgets (S/M/L). The appendix specifies optimizer (Adam), learning rates (lr_1 for rank search and lr_2 for low-rank update), per-modality hyperparameter values, dataset splits, and how each hyperparameter was selected (sensitivity tables included).

   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: Following the convention in the dynamic-merging literature (e.g., Twin-Merging, FREE-Merging), we report single-seed numbers. Reproducing the full benchmark suite (vision, LLM, MLLM, PEFT) across multiple seeds is computationally expensive; we leave multi-seed std reporting to future work. Trends are consistent across the four modalities and the three budget regimes (S/M/L), which we view as evidence that conclusions are not seed-sensitive.

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: §6 (Computation Resource) reports end-to-end wall-clock time of `DiDi-Merging` against the main baselines. The appendix specifies the GPU type used for each modality and memory footprint of the merged models.

Guidelines:

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: The work uses only publicly available models and benchmark datasets, does not collect new data from human subjects, preserves anonymity in the submission, and does not involve any deployment that could pose foreseeable harm. We have reviewed the NeurIPS Code of Ethics and confirm conformance.

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: A Broader Impact section in the appendix discusses the positive impact of reducing storage and serving cost for multi-task deployment, as well as potential negative impacts inherited from the underlying foundation models that are merged (e.g., bias, mis-use of generative capabilities).

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 proposes a model-merging algorithm; it does not release new pre-trained foundation models, generative image models, or scraped datasets that pose elevated misuse risk.

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 datasets, pre-trained backbones (CLIP-ViT, LLaMA, etc.), and baseline implementations used in this work are public and are cited in the appropriate sections. We use each asset under its respective license and within its terms of use, and do not redistribute any restricted content. We disclaim that we have not independently re-verified each upstream license text; if an asset's terms have changed, we will defer to the most recent terms specified by its creators.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The released algorithm code is documented at `https://github.com/WanyuGroup/NeurIPS2026-DiDi-Merging`, which provides installation instructions, a minimal example, the scope of the release, and its license. The paper and appendix document the method and experimental protocol.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, and therefore no IRB review was required.

    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: LLMs appear in the paper only as one of the model classes being merged in the experimental study; they are not an original or non-standard component of the core method, and were not used to develop the methodology.

    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.