

CAMFT: Conflict-Aware Mergeable Fine-Tuning for Large Language Models

Jingang Zhou^{1,2} Haiyang Guo^{1,2} Yuan Ma² Han Zhu² Xu-Yao Zhang^{1,2,*}

¹School of Advanced Interdisciplinary Sciences, University of Chinese Academy of Sciences

²Institute of Automation, Chinese Academy of Sciences

zhoujingang2025@ia.ac.cn xyz@nlpr.ia.ac.cn

Abstract

Model merging has emerged as a promising paradigm for integrating multiple task-specific capabilities into a single large language model. However, existing methods predominantly focus on post-hoc processing of independently fine-tuned models, overlooking how the training phase itself impacts cross-task compatibility. Resolving parameter conflicts after fine-tuning is inherently sub-optimal. To address this, we propose **CAMFT**, a **Conflict-Aware Mergeable Fine-Tuning** method that makes task adaptation both efficient and merge-aware. CAMFT treats mergeability as a property shaped during fine-tuning, rather than only a problem to be solved after fine-tuning. By guiding each task to update sparse coordinates with lower cross-task conflict, CAMFT produces task updates that are efficient to train and more compatible for downstream model merging. Extensive experiments demonstrate that CAMFT outperforms standard fine-tuning baselines in multi-task merging scenarios. Codes are available at <https://github.com/gyanchow/CAMFT-LLM>.

1 Introduction

Large language models (LLMs) have become general-purpose backbones for a wide range of downstream tasks (Brown et al., 2020; Chowdhery et al., 2023). As these models are adapted to increasingly diverse domains, a central challenge is how to integrate multiple task-specific capabilities into a single model without repeatedly training and deploying separate models. A straightforward solution is multi-task fine-tuning, which jointly trains one model on the union of all task data (Sanh et al., 2022; Wei et al., 2022). However, this paradigm is often impractical in practice: task data may be distributed across different users, organizations, or time periods; new tasks may arrive after initial training; and retraining large models on accumulated

*Corresponding author.

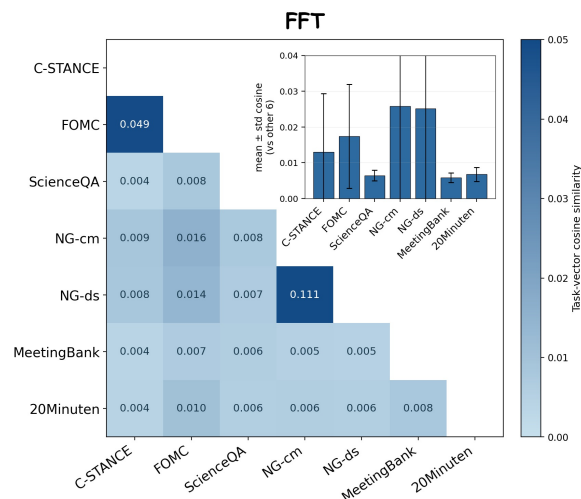


Figure 1: Pairwise cosine similarities among task vectors obtained by independently full-parameter fine-tuning on seven tasks.

data can be prohibitively expensive. These limitations motivate modular adaptation, where task-specific knowledge is learned independently and later integrated into a unified model.

One prominent solution addresses this problem through task vectors (Ilharco et al., 2023) and model merging (Yang et al., 2026a), which arithmetically combine parameter updates to enable data-free, modular reuse of task-specific expertise. Recent merging methods improve fusion quality through post-hoc techniques such as importance weighting, sign alignment, and sparse resampling (Yadav et al., 2023; Yu et al., 2024; Matena and Raffel, 2022; Yang et al., 2024; Zeng et al., 2025). These underlying task updates are typically obtained via full-parameter fine-tuning or parameter-efficient fine-tuning (PEFT) methods like LoRA (Houlsby et al., 2019; Hu et al., 2022) to reduce training costs. However, this entire paradigm operates strictly at the *post-hoc fusion* stage. Regardless of whether tasks update all parameters or an efficient subspace, indepen-

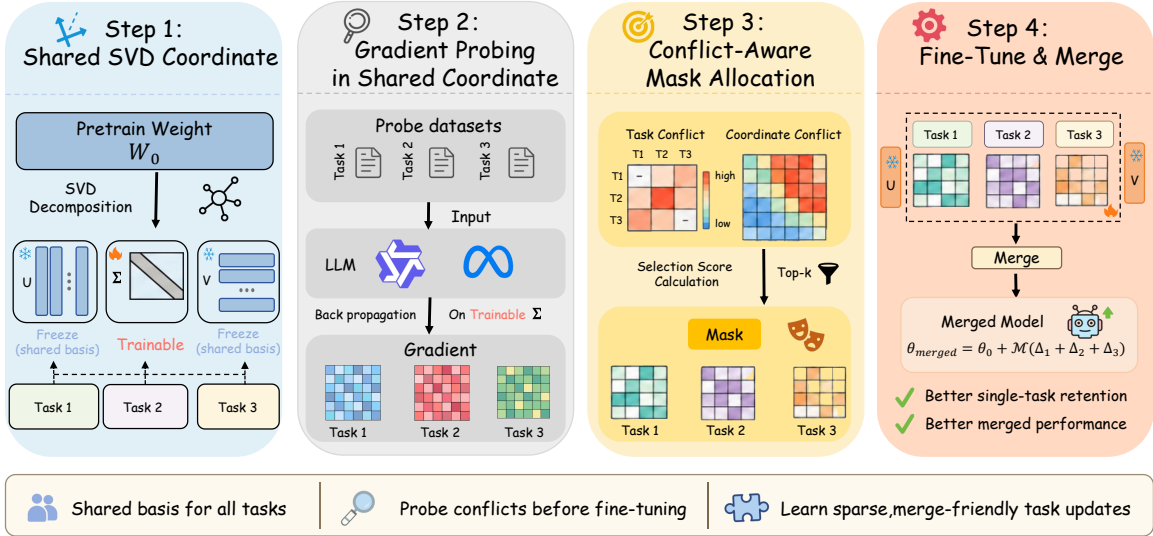


Figure 2: Overview of CAMFT. CAMFT constructs a shared SVD coordinate system and uses gradient probing to quantify cross-task conflicts. It then allocates sparse, conflict-aware masks to each task. By isolating parameter updates, CAMFT ensures the independently fine-tuned models are highly compatible for merging.

dent fine-tuning inevitably writes incompatible information into overlapping parameter directions. Figure 1 shows that task vectors obtained by independent full-parameter fine-tuning are not mutually orthogonal, indicating substantial parameter-space entanglement before merging. By attempting to resolve these interferences strictly after the fact, both post-hoc merging and parameter-efficient methods overlook the opportunity to optimize cross-task compatibility during training.

Recent work has begun to explore merge-aware fine-tuning by explicitly considering mergeability during training (Lee et al., 2025; Yang et al., 2026b; Zhang and Zhou, 2025). While these methods improve fusion outcomes through strategies such as sharpness minimization or subspace orthogonalization, they typically operate at a coarse granularity. By enforcing global properties or subspace-level separation, they overlook a fundamental question: *which specific parameter coordinates should each task update to minimize merge conflict?* Answering this question is crucial, because even if tasks are separated in broad subspaces, updating overlapping parameter coordinates will still cause severe weight-level interference during merging.

To answer this question, we argue that effective modular adaptation must deliberate not only *how many* parameters a task updates, but specifically *where* it is allowed to place them. If multiple tasks update the same parameter directions with inconsistent requirements, post-hoc merging inevitably compresses these incompatible changes,

leading to performance collapse. Instead of treating this conflict as noise to be resolved after training, we leverage it as a constructive signal to shape the updates beforehand. Motivated by this insight, we propose **CAMFT** (Conflict-Aware Mergeable Fine-Tuning). As illustrated in Figure 2, CAMFT constructs a shared SVD coordinate system from the pretrained weights to explicitly probe cross-task conflicts prior to fine-tuning. By quantifying these conflicts, CAMFT allocates sparse trainable coordinates for each task, guiding their adaptations toward mutually compatible regions. Consequently, CAMFT bridges PEFT and model merging by producing task updates that are intrinsically merge-friendly. Extensive evaluations on the diverse TRACE benchmark (Wang et al., 2023) demonstrate that this explicit coordinate allocation significantly improves the fused model’s performance over standard fine-tuning while fully preserving competitive single-task capabilities.

Our contributions are summarized as follows:

- We introduce a novel perspective for merge-aware fine-tuning by highlighting the need for weight-level interference control, emphasizing *where* a task should update its parameters.
- We formulate multi-task adaptation as a conflict-aware coordinate allocation problem and propose CAMFT. By leveraging gradient probing within a shared spectral space, CAMFT explicitly guides task updates toward less conflicting coordinates.

- Experiments on the TRACE benchmark demonstrate that CAMFT consistently improves merged performance across various downstream merging algorithms while strictly preserving single-task capabilities.

2 Related Work

Spectral PEFT. PEFT methods adapt pretrained models with a small number of trainable parameters, often through structured low-rank updates such as LoRA (Hu et al., 2022). Recent spectral PEFT methods use the SVD of pretrained weights to define better adaptation subspaces: PiSSA (Meng et al., 2024) updates principal singular components, MiLoRA (Wang et al., 2025) adapts minor singular components, and SVFT (Lingam et al., 2024) learns sparse coefficients over singular-vector directions. Unlike these methods, which mainly improve single-task adaptation, CAMFT uses the SVD basis to compare task updates and prevent cross-task conflicts during fine-tuning.

Model merging. Post-hoc merging methods combine task-specific models after fine-tuning. Weight averaging and Task Arithmetic directly merge model parameters or task vectors (Wortsman et al., 2022; Ilharco et al., 2023), but can suffer from conflicts among task updates. Fisher merging and AdaMerging estimate merging weights from parameter importance or unlabeled data (Matena and Raffel, 2022; Yang et al., 2024), while TIES and DARE reduce conflicts by pruning, sign alignment, or rescaling (Yadav et al., 2023; Yu et al., 2024). In contrast, CAMFT aims to prevent conflicts during fine-tuning and remains compatible with post-hoc merging algorithms.

Merge-aware fine-tuning. Recent work shows that fine-tuning strategies can substantially affect model mergeability. SAFT (Lee et al., 2025) uses sharpness-aware optimization to find flatter task-specific minima and reduce parameter interference. MergOPT (Yang et al., 2026b) improves merging robustness by simulating merge-induced perturbations during training. OSRM (Zhang and Zhou, 2025) constrains LoRA updates into orthogonal subspaces to mitigate cross-task interference before merging. These methods mainly enhance global robustness or enforce subspace-level separation, whereas CAMFT explicitly identifies coordinate-level conflicts in a shared spectral basis and allo-

cates trainable regions accordingly.

3 Method

CAMFT addresses model merging at the fine-tuning stage by constraining *where* each task is allowed to update its parameters in a shared spectral basis, rather than resolving conflicts post hoc. We formulate merge-aware fine-tuning as a sparse coordinate allocation problem and present the method in five parts. Section 3.1 formalizes the multi-task fine-tuning and merging setup. Section 3.2 introduces a shared SVD coordinate system from the pretrained weights, in which task updates from different tasks can be compared coordinate by coordinate. Section 3.3 estimates per-coordinate importance and cross-task conflict from a small probing set. Section 3.4 allocates a sparse trainable mask for each task that balances task importance against shared-coordinate conflict. Section 3.5 describes masked fine-tuning of each task and the post-hoc merging step.

3.1 Problem Setup

Let θ_0 denote the pretrained parameters and $\mathcal{T} = \{1, \dots, T\}$ the set of tasks. Fine-tuning task t produces an update Δ_t , yielding $\theta_t = \theta_0 + \Delta_t$. A merger $\mathcal{M}$ combines these updates as

$$\begin{aligned}\Delta_{\text{merge}} &= \mathcal{M}(\Delta_1, \dots, \Delta_T), \\ \theta_{\text{merge}} &= \theta_0 + \Delta_{\text{merge}}\end{aligned}\quad (1)$$

CAMFT does not alter $\mathcal{M}$; it changes how each Δ_t is produced so that the resulting updates are more compatible before merging.

3.2 Shared SVD Coordinate System

To compare and control task updates across different tasks, CAMFT first establishes a shared coordinate system for each target linear layer. Inspired by SVFT (Lingam et al., 2024), we choose the singular value decomposition (SVD) of pretrained weights as this shared basis because it provides a natural, task-agnostic decomposition of the parameter space.

For a pretrained weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we compute its compact full-rank SVD:

$$W_0 = U\Sigma V^\top, \quad r = \min(d_{\text{out}}, d_{\text{in}}). \quad (2)$$

where $U \in \mathbb{R}^{d_{\text{out}} \times r}$ and $V \in \mathbb{R}^{d_{\text{in}} \times r}$ are orthonormal matrices containing the left and right singular vectors, and $\Sigma \in \mathbb{R}^{r \times r}$ is a diagonal matrix of

singular values. The pretrained weight W_0 and its SVD bases U and V are frozen throughout training.

For each task t , CAMFT learns a residual update in this SVD coordinate system. We introduce a trainable coordinate matrix $M_t \in \mathbb{R}^{r \times r}$ and a binary mask $B_t \in \{0, 1\}^{r \times r}$. The effective update is:

$$\Delta W_t = \alpha \cdot U(M_t \odot B_t)V^\top, \quad (3)$$

where $\odot$ denotes element-wise multiplication and α is a scaling factor. We denote each entry of M_t by its index $p = (i, j)$, which represents the interaction between the i -th left singular direction and the j -th right singular direction. The set of all coordinates is denoted by Ω_ℓ , with $|\Omega_\ell| = r^2$.

This SVD-based parameterization places all task updates in a common coordinate system, so their locations can be compared by coordinate. The singular directions of the pretrained weights define a model-specific basis for allocating update capacity, and the mask B_t determines which coordinates task t is allowed to modify. CAMFT therefore formulates merge-aware fine-tuning as a sparse coordinate allocation problem, restricting each task update to coordinates that are important for the task and less likely to conflict with other tasks.

3.3 Probing Coordinate Importance and Conflict

Before fine-tuning, CAMFT probes task gradients to estimate two coordinate-level quantities: how strongly each task depends on a coordinate and whether different tasks prefer inconsistent update directions on that coordinate.

Gradient probing. For each task t , we sample a small probing set $\mathcal{D}_t^{\text{probe}}$ from the task’s training data. Starting from the pretrained model, we compute gradients with respect to the SVD-coordinate matrix M_t at each coordinate $p \in \Omega_\ell$. For each coordinate, CAMFT records two statistics from the gradient distribution:

$$\begin{aligned} I_{t,p} &= \mathbb{E}_{(x,y) \sim \mathcal{D}_t^{\text{probe}}} [|\nabla_{M_p} \ell_t(x, y)|], \\ R_{t,p} &= \mathbb{E}_{(x,y) \sim \mathcal{D}_t^{\text{probe}}} [\nabla_{M_p} \ell_t(x, y)]. \end{aligned} \quad (4)$$

where $\ell_t(x, y)$ is the task-specific loss function. The *importance score* $I_{t,p} \geq 0$ measures how strongly coordinate p influences task t ’s loss. The *signed gradient* $R_{t,p} \in \mathbb{R}$ captures the preferred update direction for coordinate p . Both statistics are essential: two tasks may have high importance on the same coordinate, but whether this causes

merge interference depends on whether they prefer similar or opposite directions.

Task specificity. Raw importance scores are not directly comparable across layers due to differing gradient scales. We therefore apply layer-wise rank normalization to obtain $\tilde{I}_{t,p} \in [0, 1]$. Using these normalized scores, we define the *task specificity* of coordinate p for task t :

$$D_{t,p} = \frac{\tilde{I}_{t,p}}{\sum_{j=1}^T \tilde{I}_{j,p} + \epsilon}. \quad (5)$$

The specificity $D_{t,p}$ represents the fraction of total demand for coordinate p that comes from task t . When $D_{t,p} \approx 1$, the coordinate is *task-specific*: it is crucial for task t but not other tasks. When $D_{t,p}$ is more uniformly distributed across tasks, the coordinate is *shared*: multiple tasks rely on it.

Coordinate conflict score. A shared coordinate is not necessarily problematic for merging. If all tasks prefer similar update directions, they can share the coordinate without interference. The coordinate becomes risky only when it is both shared and subject to heterogeneous update directions. To capture this intuition, we define a *conflict score* that combines task overlap with directional heterogeneity.

First, we measure how broadly a coordinate is shared using the normalized entropy of the specificity distribution:

$$O_p = -\frac{1}{\log T} \sum_{t=1}^T D_{t,p} \log(D_{t,p} + \epsilon), \quad (6)$$

where $O_p \approx 0$ indicates a task-specific coordinate and $O_p \approx 1$ indicates a broadly shared one.

Second, we measure directional heterogeneity by normalizing signed gradients within each task and layer, then computing the variance across tasks:

$$\begin{aligned} \hat{R}_{t,p} &= \frac{R_{t,p}}{s_{t,\ell}}, \\ s_{t,\ell} &= \sqrt{\frac{1}{|\Omega_\ell|} \sum_{q \in \Omega_\ell} R_{t,q}^2 + \epsilon}. \end{aligned} \quad (7)$$

$$C_p = O_p \cdot \text{Var}_{t \in \mathcal{T}} (\hat{R}_{t,p}). \quad (8)$$

The conflict score C_p is high only when both conditions hold: the coordinate is shared by multiple tasks (O_p large), and the tasks prefer different update directions (variance large). This design ensures that we penalize coordinates that genuinely cause interference, not simply those shared across tasks.

3.4 Conflict-Aware Mask Allocation

Based on the probing statistics, CAMFT allocates a sparse trainable mask B_t for each task t . The mask determines which SVD coordinates the task is permitted to update during fine-tuning. Our goal is to assign each task to coordinates that are both important for its performance and unlikely to conflict with other tasks.

Selection score. For each task-coordinate pair (t, p) , we compute a *selection score* that balances three considerations:

$$S_{t,p} = \frac{\tilde{I}_{t,p}(1 + \gamma D_{t,p})}{1 + \lambda C_p(1 - D_{t,p})}. \quad (9)$$

- **Importance** ($\tilde{I}_{t,p}$): Prioritizes coordinates that the task relies on.
- **Specificity** ($D_{t,p}$): Rewards coordinates specific to the current task, reducing competition. The hyperparameter $\gamma \geq 0$ controls this reward.
- **Conflict** (C_p): Penalizes shared high-conflict coordinates. The penalty is attenuated when the coordinate is specific to the current task ($D_{t,p} \approx 1$), ensuring that task-critical coordinates remain available even if they occasionally conflict. The hyperparameter $\lambda \geq 0$ controls the conflict penalty strength.

Top- k mask construction. For each layer ℓ , we allocate a fixed budget of trainable coordinates. Given a sparsity ratio $\rho \in (0, 1]$, each task selects the top- k coordinates according to $S_{t,p}$, where $k_\ell = \lfloor \rho \cdot |\Omega_\ell| \rfloor$. We first define the selected coordinate set:

$$\mathcal{K}_{t,\ell} = \text{TopK}_{q \in \Omega_\ell}(S_{t,q}, k_\ell), \quad (10)$$

and construct the binary mask as

$$B_{t,p} = \mathbb{I}[p \in \mathcal{K}_{t,\ell}], \quad (11)$$

where $\mathbb{I}[\cdot]$ is the indicator function. Coordinates with $B_{t,p} = 1$ are trainable; all others remain fixed at zero during fine-tuning.

High-conflict exclusivity. The selection score in Eq. (9) discourages high-conflict coordinates through a soft penalty, but highly important coordinates may still be selected by multiple tasks. To further reduce severe interference, CAMFT applies an *exclusivity rule* to the most conflict-prone coordinates.

Let $\mathcal{H}_\ell$ denote the set of top- $h\%$ coordinates with the highest conflict scores C_p . For each coordinate $p \in \mathcal{H}_\ell$, only the task with the highest specificity

Algorithm 1 Conflict-Aware Fine-Tuning

Require: Pretrained model θ_0 ; tasks $\mathcal{T}$; budget ρ .
Ensure: Merged model θ_{merge} .

- 1: **► Step 1: SVD coordinate construction**
- 2: **for** each target layer W_0 **do**
- 3: Decompose $W_0 = U\Sigma V^\top$.
- 4: Freeze U, V .
- 5: **end for**
- 6: **► Step 2: Coordinate-level conflict estimation**
- 7: **for** each task $t \in \mathcal{T}$ **do**
- 8: Probe gradients on task data to estimate coordinate importance $I_{t,p}$ and signed update direction $R_{t,p}$.
- 9: **end for**
- 10: Normalize importance scores $\{I_{t,p}\}$ to obtain $\{\tilde{I}_{t,p}\}$.
- 11: Compute task specificity $D_{t,p}$ and coordinate conflict C_p .
- 12: **► Step 3: Conflict-aware mask allocation**
- 13: **for** each task $t \in \mathcal{T}$ **do**
- 14: Compute selection score $S_{t,p}$ using $\tilde{I}_{t,p}$, $D_{t,p}$, and C_p .
- 15: Select top- ρ coordinates in each layer to construct task mask B_t .
- 16: **end for**
- 17: Apply high-conflict exclusivity and refill masks to keep the same budget.
- 18: **► Step 4: Masked fine-tuning and merging**
- 19: **for** each task $t \in \mathcal{T}$ **do**
- 20: Fine-tune masked SVD update:
 $\Delta W_t = \alpha U(M_t \odot B_t)V^\top$.
- 21: Reconstruct and save task update Δ_t .
- 22: **end for**
- 23: Merge task updates:
 $\Delta_{\text{merge}} = \mathcal{M}(\Delta_1, \dots, \Delta_T)$.
- 24: **return** $\theta_0 + \Delta_{\text{merge}}$.

is allowed to retain it:

$$t^*(p) = \arg \max_{t \in \mathcal{T}} D_{t,p}. \quad (12)$$

For all other tasks $t \neq t^*(p)$, coordinate p is removed from their masks: $B_{t,p} \leftarrow 0$. If removal causes a task's mask to have fewer than k_ℓ trainable coordinates, we refill with the next highest-scoring available coordinates (excluding those in $\mathcal{H}_\ell$ assigned to other tasks), preserving the per-task budget.

Together, the selection score and exclusivity rule assign coordinates according to both task specificity and cross-task compatibility. This allows CAMFT to preserve task-critical updates while restricting shared high-conflict coordinates that are likely to cause destructive interference during merging.

3.5 Task Fine-Tuning and Merging

Once the masks $\{B_t\}_{t=1}^T$ are constructed, each task is fine-tuned independently from the same pre-trained model θ_0 . For task t , we optimize only the entries of M_t where the mask is active:

$$\min_{M_t} \mathbb{E}_{(x,y) \sim \mathcal{D}_t} [\ell_t(f_{\theta_0 + \Delta_t}(x), y)], \quad (13)$$

Single-task Performance										
Method	Params	C-ST	FOMC	SciQA	NG-cm	NG-ds	MtgBk	20Min	Avg ($n = 3$)	Merge.
Full FT	Full	0.500	0.649	0.791	0.407	0.588	0.649	0.400	0.569±0.002	–
LoRA	45.1M	0.451	0.679	0.848	0.247	0.619	0.664	0.399	0.558±0.004	–
PiSSA	45.1M	0.431	0.653	0.835	0.264	0.603	0.673	0.382	0.549±0.002	–
SVFT	34.4M	0.423	0.626	0.617	0.257	0.533	0.526	0.365	0.478±0.008	–
OSRM [†]	45.1M	0.436	0.673	0.845	0.231	0.566	0.643	0.384	0.540±0.005	–
MergOPT [†]	Full	0.495	0.673	0.678	0.413	0.582	0.651	0.392	0.555±0.008	–
CAMFT [†]	34.4M	0.502	0.675	0.661	0.494	0.640	0.530	0.400	0.557±0.003	–

Merged Performance										
Method	Params	C-ST	FOMC	SciQA	NG-cm	NG-ds	MtgBk	20Min	Avg ($n = 3$)	Merge.
Full FT	Full	0.449	0.587	0.522	0.296	0.443	0.241	0.389	0.418±0.006	73.5%
LoRA	45.1M	0.446	0.613	0.173	0.099	0.557	0.020	0.394	0.329±0.007	59.0%
PiSSA	45.1M	0.444	0.609	0.265	0.187	0.532	0.165	0.382	0.369±0.005	67.2%
SVFT	34.4M	0.398	0.573	0.357	0.203	0.326	0.253	0.355	0.352±0.008	73.6%
OSRM [†]	45.1M	0.425	0.659	0.282	0.309	0.511	0.184	0.391	0.394±0.004	72.9%
MergOPT [†]	Full	0.468	0.588	0.424	0.235	0.582	0.406	0.390	0.442±0.002	79.7%
CAMFT [†]	34.4M	0.470	0.599	0.485	0.259	0.572	0.474	0.391	0.464±0.002	83.3%

Table 1: Performance comparison on TRACE 7-task benchmark with Llama-3.2-1B. **Single**: single-task performance before merging. **Merged**: performance after merging all 7 tasks with TIES. Task columns show the original fixed-seed breakdown; **Avg** is the mean $\pm$ standard deviation of the macro-average over three random seeds; **Merge.** is the ratio of the corresponding three-seed means. **Params**: trainable parameters per task (M = million; Full = full-model fine-tuning). All task metrics are normalized to [0,1]; best results are **bolded**. [†]: methods designed for mergeability.

where $\mathcal{D}_t$ is the task’s training data, ℓ_t is the task-specific loss, and Δ_t is reconstructed via Eq. (3). All pretrained weights and SVD bases (W_0 , U , Σ , V) remain frozen throughout; only the masked entries of M_t are updated. This ensures that each task learns within its allocated coordinate region.

After training, we reconstruct the task updates $\{\Delta_t\}_{t=1}^T$ into the original weight space. These reconstructed updates can then be merged using any standard model merging algorithm:

$$\begin{aligned}\Delta_{\text{merge}} &= \mathcal{M}(\Delta_1, \dots, \Delta_T), \\ \theta_{\text{merge}} &= \theta_0 + \Delta_{\text{merge}}.\end{aligned}\quad (14)$$

CAMFT is agnostic to the choice of $\mathcal{M}$, in our experiments, we use TIES (Yadav et al., 2023) with default hyperparameters to demonstrate that our method improves update quality independent of the merging algorithm.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on the TRACE benchmark (Wang et al., 2023), consisting of 7 diverse tasks: C-STANCE (Zhao et al., 2023), FOMC (Shah et al., 2023), MeetingBank (Hu et al.,

2023), ScienceQA (Lu et al., 2022), NumGLUE-cm and NumGLUE-ds (Mishra et al., 2022), and 20Minuten (Rios et al., 2021). The statistics and evaluation metrics of these datasets are summarized in Appendix A.

Models. We conduct experiments on Llama-3.2-1B (Meta AI, 2024) as the main backbone. We use Qwen3-4B (Yang et al., 2025) as an additional backbone and report its results in Appendix D.1.

Fine-tuning methods compared. We compare CAMFT with two groups of baselines. The first group consists of representative parameter-efficient fine-tuning methods, including **LoRA** (Hu et al., 2022), **PiSSA** (Meng et al., 2024), and **SVFT** (Lingam et al., 2024). LoRA and PiSSA provide standard low-rank and spectral PEFT baselines. SVFT is a sparse spectral baseline that updates a fixed set of SVD coordinates shared across tasks, without task-specific importance filtering or conflict-aware coordinate allocation. The second group includes merge-friendly fine-tuning methods, including **OSRM** (Zhang and Zhou, 2025), and **MergOPT** (Yang et al., 2026b). These methods aim to improve the compatibility of independently fine-tuned task updates through regularization or

merge-aware optimization.

Implementation details. All reported results are selected by validation performance from predefined hyperparameter grids, with search ranges and task-specific training schedules provided in Appendix B. All methods use the same optimizer family and training protocol unless otherwise specified. LoRA, PiSSA, and OSRM use rank 64 and have 45.1M trainable parameters per task on Llama-3.2-1B. SVFT and CAMFT use the same target modules with sparsity $\rho = 0.10$, giving 34.4M trainable parameters per task. CAMFT searches over λ , γ , h , and α on a predefined grid (full search ranges and selected values are reported in Appendix C.3), and uses 500 probing examples per task. All experiments are run on 4 NVIDIA A100 80GB GPUs. Unless otherwise specified, task updates are merged with TIES (Yadav et al., 2023), whose hyperparameters are also selected on validation data.

4.2 Main Results

Table 1 reports a fixed-seed task-level breakdown together with three-seed aggregate results for merging seven independently trained task experts. For the aggregate evaluation, we repeat every method with three random seeds while keeping the data splits and selected training and merging configurations fixed across runs. CAMFT achieves the best merged macro-average score of 0.4643, outperforming MergOPT (0.4420) by 0.0223. It also obtains the highest mergeability ratio, 83.3% versus 79.7% for MergOPT. This gain does not come from simply improving single-task performance: CAMFT’s single-task macro-average is 0.5573, comparable to LoRA (0.5582) and MergOPT (0.5547). Together with the lower results of SVFT and the component ablations below, these findings support conflict-aware coordinate allocation rather than sparsity or spectral parameterization alone as the source of CAMFT’s mergeability gains.

4.3 Ablation Study

Sensitivity to trainable coordinate ratio. We analyze the effect of the trainable coordinate ratio ρ , which controls the fraction of SVD coordinates allocated to each task. As shown in Figure 3, increasing ρ from 0.00 to 0.10 improves the merged average from 0.3125 to 0.4644. Further increasing ρ hurts merged performance, dropping to 0.3883 at

Variant	Merged Avg	Δ
CAMFT (Full)	0.464	—
w/o Conflict Estimation	0.395	−0.069
w/o Gradient Probing	0.376	−0.088
w/o SVD-space Importance	0.427	−0.037

Table 2: Ablation study of CAMFT’s core components on Llama-3.2-1B. Δ denotes the absolute drop from full CAMFT.

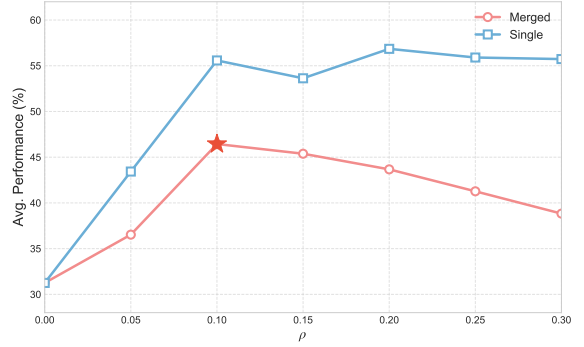


Figure 3: Sensitivity of CAMFT to the trainable coordinate ratio ρ . The star marks the **best** merged performance.

$\rho = 0.30$, although the single-task average remains high. This divergence suggests that larger update capacity can improve task adaptation but also reintroduce merge conflicts, supporting a compact and conflict-aware update space.

Core component ablation. We ablate three core designs of CAMFT: conflict estimation, gradient probing, and SVD-space importance estimation. *w/o Conflict Estimation* selects masks only by probing importance $\tilde{I}_{t,p}$, without the conflict score C_p or high-conflict exclusivity. *w/o Gradient Probing* replaces probing-based masks with random masks under the same sparsity budget. *w/o SVD-space Importance* estimates importance outside the SVD coordinate space. All variants use the same training and TIES merging configuration as CAMFT.

As shown in Table 2, removing any component degrades merged performance. Random masking performs worst, confirming that sparsity alone is insufficient without task-aware coordinate selection. Importance-only masking also suffers a large drop, showing that high-gradient coordinates can still induce cross-task interference if conflict is ignored. The degradation without SVD-space importance further suggests that the pretrained spectral basis provides a useful coordinate system for estimating and allocating task updates. Overall, these results

# Samples	10	100	200	500	1000
Merged Avg	0.4057	0.4276	0.4592	0.4644	0.4497

Table 3: Effect of probing sample size on merged performance. Best result is **bolded**.

Merging Method	Full FT	MergOPT	CAMFT
Average	0.368	0.378	0.389
Task Arithmetic	0.388	0.395	0.402
DARE	0.378	0.385	0.391
TIES	0.418	0.442	0.464

Table 4: CAMFT improves mergeability across different merging algorithms. We report the best merged average after hyperparameter search for each method.

show that CAMFT’s gains come from combining task relevance and conflict awareness in a shared SVD coordinate space.

Effect of probing sample size. We further study the effect of probing set size on CAMFT. As shown in Table 3, very small probing sets lead to weaker performance, while a moderate number of examples is sufficient to estimate task conflicts reliably. Using more probing samples does not further improve performance, suggesting that CAMFT does not require extensive probing.

4.4 Robustness to merging methods.

CAMFT aims to improve the intrinsic mergeability of task updates rather than relying on a particular post-hoc merging algorithm. We therefore evaluate Full FT, MergOPT, and CAMFT under four representative merging rules: Average, Task Arithmetic, DARE, and TIES, using the same predefined hyperparameter search space for each rule. As shown in Table 4, CAMFT consistently outperforms both Full FT and MergOPT across all merging methods. Its advantage under simple Average suggests that CAMFT already reduces update interference during fine-tuning, before any explicit post-hoc conflict resolution is applied. The consistent gains under Task Arithmetic, DARE, and TIES further show that CAMFT is complementary to existing merging algorithms and improves mergeability in a method-agnostic manner. Compared with MergOPT, CAMFT’s stronger performance indicates that coordinate-level conflict-aware allocation provides benefits beyond generic merge-aware optimization. The hyperparameter search ranges for each merging method are provided in

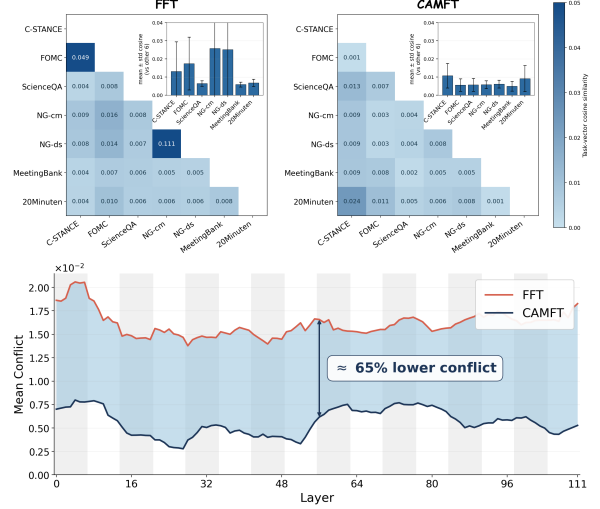


Figure 4: Cross-task interference before and after applying CAMFT. **Top:** pairwise cosine similarity of task vectors on the seven TRACE tasks under FFT (left) and CAMFT (right); **Bottom:** mean off-diagonal cosine at each of the 112 linear layers of Llama-3.2-1B.

Appendix C.2.

4.5 Analysis

Task-pair conflict. Figure 4 shows that task conflicts under FFT are highly uneven and concentrated in a few task pairs. CAMFT markedly alleviates these concentrated conflicts, resulting in lower and more balanced per-task overlap across all seven tasks.

Layer-level conflict. The bottom panel of Figure 4 shows that FFT maintains substantial cross-task interference across the network depth, whereas CAMFT consistently suppresses conflicts across layers. This indicates that CAMFT reduces interference broadly rather than only in a few outlier layers. A finer breakdown by projection type is provided in Appendix D.2.

5 Conclusion

We present CAMFT, a conflict-aware fine-tuning framework that bridges parameter-efficient fine-tuning and model merging. By constructing a shared SVD coordinate system and estimating coordinate-level task importance and conflicts before fine-tuning, CAMFT allocates sparse trainable coordinates in a merge-aware manner, preventing destructive interference before it must be repaired post hoc. Empirical results on seven datasets show that this approach substantially improves upon existing merging strategies across multiple LLMs.

Limitations

Although CAMFT improves model merging, it has several limitations. First, it relies on a small amount of probing data to estimate task importance and inter-task conflict, so the quality of mask allocation may depend on how representative the probing samples are. Second, CAMFT currently uses a fixed coordinate budget across layers and tasks, while different layers and tasks may require different update capacities. Future work could therefore explore adaptive budget allocation based on measured importance and conflict. Finally, CAMFT assumes that all tasks are available before fine-tuning. For incremental settings, one possible extension is to maintain a historical conflict map and allocate coordinates for each new task according to its conflict with previously learned tasks, without recomputing masks for all tasks.

As with other model merging methods, the merged model may inherit or combine undesirable behaviors from the base model or individual task experts, including biases, hallucinations, or unsafe capabilities. Since merging multiple experts can lead to behavior that is not fully predictable from each individual model, practical deployment should include task-specific safety evaluation and standard risk assessment. Our experiments are limited to academic benchmarks and do not involve high-stakes deployment.

Acknowledgments

This work was supported by the CAS Project for Young Scientists in Basic Research (Grant No. YSBR-083) and Lingang Laboratory (Grant No. LGL-2616-04).

References

- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2023. [Palm: Scaling language modeling with pathways](#). *Journal of Machine Learning Research*, 24(240):1–113.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. [Parameter-efficient transfer learning for NLP](#). In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Yebowen Hu, Timothy Ganter, Hanieh Deilamsalehy, Franck Dernoncourt, Hassan Foroosh, and Fei Liu. 2023. [MeetingBank: A benchmark dataset for meeting summarization](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16409–16423, Toronto, Canada. Association for Computational Linguistics.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hananeh Hajishirzi, and Ali Farhadi. 2023. [Editing models with task arithmetic](#). In *The Eleventh International Conference on Learning Representations*.
- Yeoreum Lee, Jinwook Jung, and Sungyong Baik. 2025. [Mitigating parameter interference in model merging via sharpness-aware fine-tuning](#). In *The Thirteenth International Conference on Learning Representations*.
- Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. 2024. [Svft: Parameter-efficient fine-tuning with singular vectors](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 41425–41446. Curran Associates, Inc.
- Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521.
- Michael S. Matena and Colin A. Raffel. 2022. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems*, volume 35, pages 17703–17716.
- Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. [PiSSA: Principal singular values and singular vectors adaptation of large language models](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

- Meta AI. 2024. Llama 3.2: 1b and 3b model card. <https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct>. Accessed: 2026-05-14.
- Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3505–3523.
- Annette Rios, Nicolas Spring, Tannon Kew, Marek Kostrzewa, Andreas Säuberli, Mathias Müller, and Sarah Ebling. 2021. A new dataset and efficient baselines for document-level text simplification in german. In *Proceedings of the Third Workshop on New Frontiers in Summarization*, pages 152–161.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, and 21 others. 2022. [Multitask prompted training enables zero-shot task generalization](#). In *International Conference on Learning Representations*.
- Agam Shah, Suvan Paturi, and Sudheer Chava. 2023. [Trillion dollar words: A new financial dataset, task & market analysis](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6664–6679, Toronto, Canada. Association for Computational Linguistics.
- Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. 2025. [MiLoRA: Harnessing minor singular components for parameter-efficient LLM finetuning](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4823–4836, Albuquerque, New Mexico. Association for Computational Linguistics.
- Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. [Trace: A comprehensive benchmark for continual learning in large language models](#). Preprint, arXiv:2310.06762.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. [Finetuned language models are zero-shot learners](#). In *International Conference on Learning Representations*.
- Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. [Model soups: averaging weights of multiple finetuned models improves accuracy without increasing inference time](#). In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.
- Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. [TIES-merging: Resolving interference when merging models](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2026a. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):1–41.
- Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2024. [Adamerging: Adaptive model merging for multi-task learning](#). In *The Twelfth International Conference on Learning Representations*.
- Enneng Yang, Qun Yang, Peng Wang, Anke Tang, Guibing Guo, Xiaochun Cao, and Li Shen. 2026b. [MergeOPT: A merge-aware optimizer for robust model merging](#). In *The Fourteenth International Conference on Learning Representations*.
- Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. [Language models are super mario: Absorbing abilities from homologous models as a free lunch](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 57755–57775. PMLR.
- Fanhu Zeng, Haiyang Guo, Fei Zhu, Li Shen, and Hao Tang. 2025. Robustmerge: Parameter-efficient model merging for mllms with direction robustness. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, volume 38, pages 71071–71095.
- Haobo Zhang and Jiayu Zhou. 2025. [Unraveling LoRA interference: Orthogonal subspaces for robust model merging](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26459–26472, Vienna, Austria. Association for Computational Linguistics.
- Chenye Zhao, Yingjie Li, and Cornelia Caragea. 2023. [C-STANCE: A large dataset for Chinese zero-shot stance detection](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13369–13385, Toronto, Canada. Association for Computational Linguistics.

A Datasets

Table 5 summarizes the statistics of the seven TRACE tasks used in our experiments, including the source domain, language, average context length, and the evaluation metric used for each task. For all metrics in this table, larger values indicate better performance. All task scores are normalized to $[0, 1]$ before averaging in the main experiments.

B Implementation Details

B.1 Training Schedules

Table 6 reports the task-specific training schedules. We select the number of epochs separately for each task because the tasks differ in dataset size and convergence behavior. For each task, we select the final configuration with the best validation performance before merging and then apply the same TIES merging protocol as in the main experiments.

B.2 Trainable Parameter Counting

For LoRA, PiSSA, and OSRM, we use rank 64 on all attention and MLP linear layers. For a linear layer with input dimension d_{in} and output dimension d_{out} , the number of trainable LoRA-style parameters is

$$r(d_{\text{in}} + d_{\text{out}}). \quad (15)$$

For Llama-3.2-1B, the adapted modules in each transformer block are $q_{\text{proj}}, k_{\text{proj}}, v_{\text{proj}}, o_{\text{proj}}, gate_{\text{proj}}, up_{\text{proj}}$, and $down_{\text{proj}}$. Using hidden size 2048, key/value projection size 512, intermediate size 8192, 16 layers, and rank $r = 64$, the trainable parameter count is

$$\begin{aligned} N_{\text{LoRA}} &= 64 \times 16 \times [(2048 + 2048) \\ &\quad + 2(2048 + 512) + (2048 + 2048) \\ &\quad + 3(2048 + 8192)] \\ &= 45,088,768. \end{aligned} \quad (16)$$

We therefore report LoRA, PiSSA, and OSRM as 45.1M trainable parameters per task on Llama-3.2-1B. Full FT and MergOPT update the full model and are marked as Full. CAMFT trains only the selected SVD-coordinate entries. For a target linear layer, CAMFT has $\lfloor \rho \min(d_{\text{in}}, d_{\text{out}})^2 \rfloor$ trainable SVD-coordinate entries. Using the same target modules as LoRA/PiSSA/OSRM and $\rho = 0.10$,

the CAMFT parameter count is

$$\begin{aligned} N_{\text{CAMFT}} &= 0.10 \times 16 \times (2 \cdot 512^2 + 5 \cdot 2048^2) \\ &= 34,393,293. \end{aligned} \quad (17)$$

We therefore report CAMFT as 34.4M trainable parameters per task.

C Hyperparameter Search Details

We tune hyperparameters for fine-tuning, merging, and CAMFT-specific components on predefined grids and select configurations based on validation performance. The same merging-side search space is shared across all fine-tuning methods to ensure a fair comparison. Sections C.1, C.2, and C.3 report the search ranges for the three groups respectively.

C.1 Fine-Tuning Hyperparameters

Table 7 reports the hyperparameter search ranges used for Full FT, LoRA, PiSSA, OSRM, and SVFT. For each task, we select the configuration with the best validation performance before merging and then apply the same TIES merging protocol as in the main experiments. LoRA, PiSSA, and OSRM use rank $r = 64$ on all attention and MLP linear layers (45.1M trainable parameters per task on Llama-3.2-1B).

C.2 Merging Hyperparameters

For a fair comparison, we tune the key hyperparameters of each merging algorithm on the same predefined grid for all fine-tuning methods. For Average and Task Arithmetic, we tune the scaling coefficient. For DARE, we tune the density of retained task-vector entries and the scaling coefficient. For TIES, we tune the density used for trimming and the scaling coefficient. The detailed search grids are shown in Table 8. For each method, we report the best merged average over validation tasks.

C.3 CAMFT-specific Hyperparameters

CAMFT introduces four method-specific hyperparameters: the specificity reward γ and the conflict penalty λ in the selection score (Eq. 9), the high-conflict exclusivity ratio h (Section 3.4), and the update scaling factor α (Eq. 3). Because γ , λ , and h govern *cross-task* coordinate allocation, they are shared across all tasks within a single merging run; only α , which rescales each task’s reconstructed update, is tuned per task. The total joint search budget is bounded by $|\gamma| \cdot |\lambda| \cdot |h| \cdot |\alpha| = 4 \cdot 4 \cdot 3 \cdot 4 = 192$

Dataset	Source	Language	Avg. Len.	Metric
<i>Domain-specific</i>				
ScienceQA	Science	English	210	Accuracy
FOMC	Finance	English	51	Accuracy
MeetingBank	Meeting	English	2,853	ROUGE-L
<i>Multi-lingual</i>				
C-STANCE	Social media	Chinese	127	Accuracy
20Minuten	News	German	382	SARI
<i>Mathematical reasoning</i>				
NumGLUE-cm	Math	English	32	Accuracy
NumGLUE-ds	Math	English	21	Accuracy

Table 5: An overview of dataset statistics in experiments. **Source** indicates the origin of the context. **Avg. Len.** denotes the average length in words for English and German datasets, and in characters for Chinese. **SARI** is a metric specific to simplification. For all metrics, the larger the corresponding value, the better.

Task	Epochs
C-STANCE	5
FOMC	3
ScienceQA	7
NumGLUE-cm	5
NumGLUE-ds	5
MeetingBank	7
20Minuten	7

Table 6: Task-specific training schedules.

Hyperparameter	Search Grid
Learning rate (Full FT)	{1e-5, 5e-5, 1e-4}
Learning rate (PEFT)	{5e-5, 1e-4, 5e-4}
Weight decay	{0, 0.01}
LoRA α (LoRA/PiSSA/OSRM)	{16, 32, 64}
Batch size	16
Optimizer	AdamW

Table 7: Hyperparameter search grids for fine-tuning baselines. The same grid is used across all tasks; per-task best configurations are selected on validation.

configurations per backbone, comparable in scale to the joint (learning rate, weight decay, α) grid used for LoRA-style baselines in Appendix C.1.

Table 9 lists the search grid, and Table 10 reports the final values used in the main experiments on both backbones. The same global (γ, λ, h) configuration was selected on Llama-3.2-1B and Qwen3-4B, suggesting that the conflict-aware allocation is not sensitive to the specific backbone scale.

Merging Method	Hyperparameters
Average	—
Task Arithmetic	$s \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$
DARE	$d \in \{0.1, 0.2, 0.25, 0.3, 0.5\};$ $s \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$
TIES	$d \in \{0.1, 0.2, 0.25, 0.3, 0.5\};$ $s \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$

Table 8: Hyperparameter search grids for different merging algorithms. The same search space is used for all fine-tuning methods.

Hyperparameter	Search Grid
Specificity reward γ	{0.5, 1.0, 2.0, 4.0}
Conflict penalty λ	{0.5, 1.0, 2.0, 4.0}
Exclusivity ratio h (%)	{5, 10, 20}
Scaling factor α	{0.5, 1.0, 2.0, 4.0}

Table 9: Search grids for CAMFT-specific hyperparameters. γ, λ, h are shared across tasks; α is tuned per task.

Hyperparameter	Llama-3.2-1B	Qwen3-4B
<i>Global (shared across tasks)</i>		
γ	1.0	1.0
λ	2.0	2.0
h (%)	10	10
<i>Per-task scaling factor α</i>		
C-STANCE	1.0	1.0
FOMC	1.0	1.0
ScienceQA	2.0	2.0
NumGLUE-cm	2.0	2.0
NumGLUE-ds	2.0	1.0
MeetingBank	1.0	1.0
20Minuten	1.0	1.0

Table 10: Final CAMFT hyperparameter values used in the main experiments. Global values for γ, λ, h are shared across all tasks within a merging run; the per-task α values are reported in the bottom block.

D Additional Experimental Results

D.1 Results on Qwen3-4B

Table 11 reports results on Qwen3-4B under the same seven-task TRACE merging protocol.

Single-task Performance									
Method	C-ST	FOMC	SciQA	NG-cm	NG-ds	MtgBk	20Min	Avg	Merge.
Full FT	0.536	0.712	0.842	0.491	0.651	0.684	0.439	0.622	–
LoRA	0.512	0.724	0.881	0.362	0.663	0.702	0.431	0.611	–
PiSSA	0.501	0.704	0.868	0.381	0.647	0.711	0.417	0.604	–
SVFT	0.477	0.681	0.691	0.351	0.593	0.583	0.399	0.539	–
OSRM [†]	0.505	0.718	0.873	0.345	0.626	0.681	0.422	0.596	–
MergOPT [†]	0.532	0.719	0.751	0.497	0.646	0.693	0.429	0.610	–
CAMFT [†]	0.542	0.721	0.738	0.548	0.682	0.596	0.438	0.609	–
Merged Performance									
Method	C-ST	FOMC	SciQA	NG-cm	NG-ds	MtgBk	20Min	Avg	Merge.
Full FT	0.492	0.647	0.583	0.371	0.517	0.377	0.421	0.487	78.3%
LoRA	0.487	0.665	0.244	0.163	0.612	0.092	0.424	0.384	62.9%
PiSSA	0.486	0.657	0.331	0.252	0.588	0.246	0.414	0.425	70.3%
SVFT	0.443	0.630	0.414	0.271	0.405	0.317	0.392	0.410	76.1%
OSRM [†]	0.476	0.699	0.356	0.384	0.579	0.332	0.424	0.464	77.9%
MergOPT [†]	0.514	0.648	0.493	0.317	0.642	0.516	0.423	0.508	83.3%
CAMFT [†]	0.523	0.662	0.557	0.365	0.637	0.610	0.429	0.540	88.7%

Table 11: Performance comparison on TRACE 7-task benchmark with Qwen3-4B. **Single**: single-task performance before merging. **Merged**: performance after merging all 7 tasks with TIES. **Merge.**: mergeability (Merged Avg / Single Avg $\times 100\%$). All task metrics are normalized to $[0,1]$. Best results are **bolded**. [†]: methods designed for mergeability.

D.2 Per-layer Conflict by Projection Type

Figure 5 decomposes the per-layer conflict in the bottom panel of Figure 4 along the seven projection types (q, k, v, o, gate, up, down). The reduction from FFT to CAMFT is consistent across all projection types and all 16 transformer blocks, with the largest absolute gap observed on q_proj and down_proj. The bottom-right panel summarises the per-projection mean over blocks.

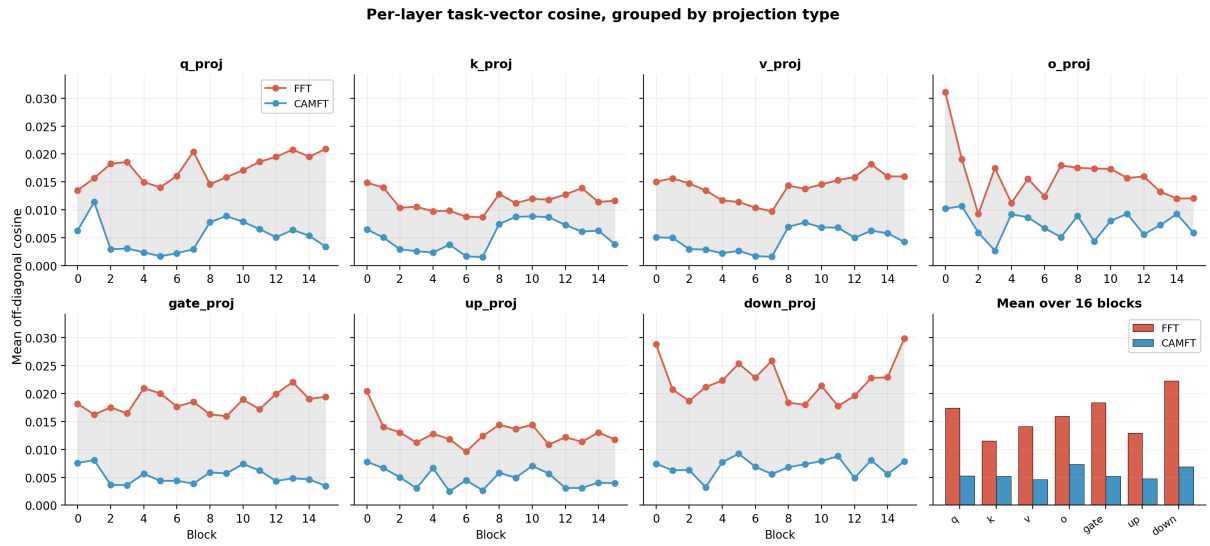


Figure 5: Per-layer task-vector cosine, decomposed by projection type. Each of the seven panels plots the 16 transformer blocks for one projection; the bottom-right panel reports the per-projection mean over all blocks.