

CASS: Mitigating Parameter Interference for Model Merging via Contribution-Aware Structured Sparsity

Abstract

Model merging has emerged as a pivotal technique for integrating task-specific experts into a single multi-task model without expensive retraining. However, parameter interference remains a critical bottleneck. Existing methods typically mitigate conflicts by pruning task vectors based on weight magnitude or random heuristics, treating Transformers as unstructured “bags of parameters” and overlooking their inherent modularity. In this paper, we propose **Contribution-Aware Structured Sparsity (CASS)**, a unified framework that disentangles task capabilities by identifying and preserving functional subspaces. Drawing on mechanistic interpretability, we reformulate Transformer layers into a unified summation form, establishing that masking heads and neurons is physically equivalent to removing specific additive updates to the residual stream. Guided by an activation-weighted metric, CASS generates structural masks that isolate task-essential components. This framework operates in two paradigms: as a *post-training filter* to denoise task vectors for existing methods (CASS-Merging), and as a *training-time constraint* to proactively induce orthogonality (CASS-Tuning). Extensive experiments on Qwen (Language) and ViT (Vision) architectures demonstrate that CASS consistently enhances state-of-the-art baselines.

1. Introduction

The proliferation of Foundation Models has catalyzed a paradigm shift in the deep learning community. Adapting a single pre-trained model to diverse downstream tasks, ranging from natural language processing to computer vision, has become standard practice. However, deploying independent fine-tuned checkpoints for every specific task incurs prohibitive storage and memory costs. To address

Correspondence to: Anonymous Author
<anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

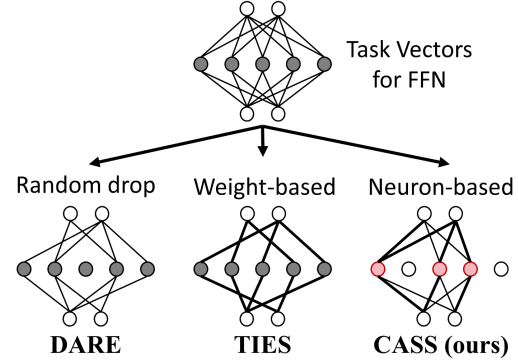


Figure 1. Sparsification-based methods for model merging.

this, Model Merging has emerged as a pivotal technique, enabling the integration of multiple task-specific expert models into a single multi-task model without the need for expensive retraining or access to original training data (Tang et al., 2024; He et al., 2025b).

Despite its promise, current merging techniques face a fundamental challenge: parameter interference, where conflicting updates across tasks degrade multi-task performance. Early approaches attempt to combine capabilities through simple linear operations on weights or task vectors (Wortsman et al., 2022; Ilharco et al., 2022). To resolve conflicts more effectively, distinct lines of research have emerged. Optimization-based methods (Matena & Raffel, 2022; Jin et al., 2022) and recent subspace-oriented approaches (Gargiulo et al., 2025; Cheng et al., 2025) utilize second-order information, inner product matrices, or singular value decomposition to reweight parameters or guide merging trajectories. While theoretically grounded, these methods often incur significant computational overhead or rely on implicit optimization in the full parameter space. Conversely, sparsification-based heuristics aim for efficiency by pruning conflicting updates, which rely on weight magnitude (Yadav et al., 2023; Wang et al., 2024) or random pruning (Yu et al., 2024) to mask parameters. While effective to a degree, these methods treat the model as an unstructured “bag of parameters.” As shown in Fig. 1, they indiscriminately preserve large-magnitude updates or randomly discard updates, failing to account for the model’s internal modular structure and the actual functional contribution of its components.

To overcome these limitations, we draw inspiration from Mechanistic Interpretability (Gao et al., 2025), specifically the observations regarding functional specialization in Transformers. Research indicates that Feed-Forward Networks (FFN) operate as Key-Value memories with highly sparse activation patterns (Geva et al., 2021; Xiao et al., 2024), and Attention Heads perform distinct functional roles that vary across tasks (Clark et al., 2019; Hanna et al., 2023; Michel et al., 2019). This implies that task-specific knowledge is not uniformly distributed but resides in distinct functional subspaces. Consequently, merging conflicts largely arise from the forced integration of disjoint functional modules. We argue that the metric of a component’s importance is not its static weight magnitude, but its dynamic contribution to the residual stream, which can be effectively captured by the interaction between activation norms and weight matrices.

Building on this insight, we propose **Contribution-Aware Structured Sparsity (CASS)**, a unified framework designed to mitigate merging interference by explicitly identifying and preserving task-specific functional subspaces. We first formulate both Multi-Head Attention (MHA) and FFN layers into a unified summation form, establishing that masking operations physically correspond to removing specific terms from the residual stream. Guided by an activation-weighted importance metric, CASS applies structured masks to task vectors, ensuring that only heads and neurons critical to a specific task are preserved. Crucially, CASS operates as a versatile framework: it can serve as a post-training denoising filter to clean up task vectors during merging, or as a proactive constraint during the fine-tuning stage to enforce orthogonality from the source.

Our main contributions are summarized as follows:

- We formulate a unified mathematical framework for Transformer components, treating MHA and FFN layers as sums of independent functional vectors. This provides a rigorous physical interpretation for structured masking: zeroing out a head or neuron is mathematically equivalent to removing its additive update to the residual stream.
- We propose CASS, a contribution-aware method that identifies functional subspaces via activation-weighted metrics. CASS unifies two paradigms: it operates as a post-training denoising filter to resolve conflicts in existing models (CASS-Merging), and as a mask-guided training constraint (CASS-Tuning) to proactively induce task orthogonality during fine-tuning.
- We conduct extensive experiments across different architectures (Qwen, ViT) and modalities (Language, Vision) to demonstrate the effectiveness of the proposed CASS.

2. Related Work

2.1. Model Merging

Model merging aims to integrate the capabilities of multiple task-specific models into a single entity. Early approaches, such as Model Soup (Wortsman et al., 2022) and Task Arithmetic (Ilharco et al., 2022), demonstrated that simple linear interpolation or vector addition of fine-tuned weights can often retain multi-task capabilities. However, these methods suffer significantly from parameter interference when task vectors point in conflicting directions. To mitigate this, recent research has diverged into two main streams.

Optimization-based approaches aim to resolve conflicts by utilizing second-order information or projecting weights into low-interference subspaces. Fisher Merging (Matena & Raffel, 2022) and RegMean (Jin et al., 2022) rely on the Fisher Matrix or inner product matrices to weigh parameters. More recently, subspace-oriented methods like TSV-Merging (Gargiulo et al., 2025) and WUDI-Merging (Cheng et al., 2025) have utilized SVD or linear subspace assumptions to guide merging. While theoretically grounded, these methods typically require heavy computation or rely on implicit optimization in the full parameter space.

Sparsification-based methods mitigate interference by pruning conflicting updates. TIES-Merging (Yadav et al., 2023) posits that parameter importance correlates with weight magnitude, resolving conflicts by keeping only the largest updates. DARE (Yu et al., 2024) employs random pruning followed by rescaling. More recently, TALL-masks proposed constructing task-specific binary masks based on weight magnitude differences to localize information within multi-task vectors. While TALL-masks (Wang et al., 2024) improves upon global heuristics by tailoring masks to each task, it remains an *unstructured* method. These approaches operate at the individual parameter level, effectively treating the model as a “bag of parameters” and ignoring the inherent modular structure (e.g., Attention Heads and FFN neurons) of Transformer architectures.

2.2. Mechanistic Interpretability

Our work is grounded in the growing field of mechanistic interpretability (Gao et al., 2025), which views Transformers not as monolithic blocks but as collections of specialized functional components. Extensive research suggests that Feed-Forward Networks (FFNs) act as Key-Value memories (Geva et al., 2021), where specific neurons store discrete knowledge patterns. These neurons often exhibit high activation sparsity (Xiao et al., 2024). Similarly, Attention Heads have been shown to perform distinct roles (Michel et al., 2019), such as induction heads for in-context learning (Olsson et al., 2022) or retrieval heads for factual recall (Wu et al., 2024).

CASS leverages these functional specialization properties directly. Unlike previous merging methods that rely on static weights, we identify functional subspaces using contribution-based statistics, aligning the merging process with the model’s intrinsic topological structure.

3. Preliminaries and Motivation

3.1. Unified Summation Form of Transformer

Standard merging methods often treat model parameters as unstructured vectors, ignoring the modular architecture of Transformers. To provide physical grounding for our method, we reformulate both MHA and FFN into a unified summation form. Consider a vanilla Transformer layer (Vaswani et al., 2017) with input $\mathbf{x} \in \mathbb{R}^d$. The layer updates the residual stream through two sub-layers. We demonstrate that the output of both can be decomposed into a sum of independent update vectors.

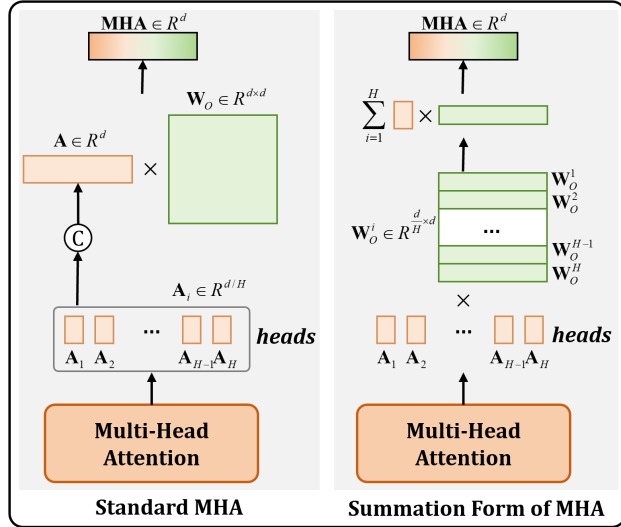


Figure 2. The summation form of MHA.

MHA as a Sum of Heads. Let an MHA layer have H heads. The standard formulation concatenates the output of individual heads $\mathbf{A}_i(\mathbf{x}) \in \mathbb{R}^{d/H}$ and projects them with $\mathbf{W}_O \in \mathbb{R}^{d \times d}$. By viewing this through the lens of block matrix multiplication, we can decompose \mathbf{W}_O into H submatrices $\mathbf{W}_O^{(i)} \in \mathbb{R}^{(d/H) \times d}$ along the row dimension. As shown in Fig. 2, the output becomes an additive sum:

$$\begin{aligned} \text{MHA}(\mathbf{x}) &= \text{Concat}(\mathbf{A}_1(\mathbf{x}), \dots, \mathbf{A}_H(\mathbf{x})) \mathbf{W}_O \\ &= \sum_{i=1}^H \mathbf{A}_i(\mathbf{x}) \mathbf{W}_O^{(i)} \end{aligned} \quad (1)$$

Physically, each term $\mathbf{A}_i(\mathbf{x}) \mathbf{W}_O^{(i)}$ represents an independent update vector from head i to the residual stream. Masking

a head is thus mathematically equivalent to removing its specific additive contribution.

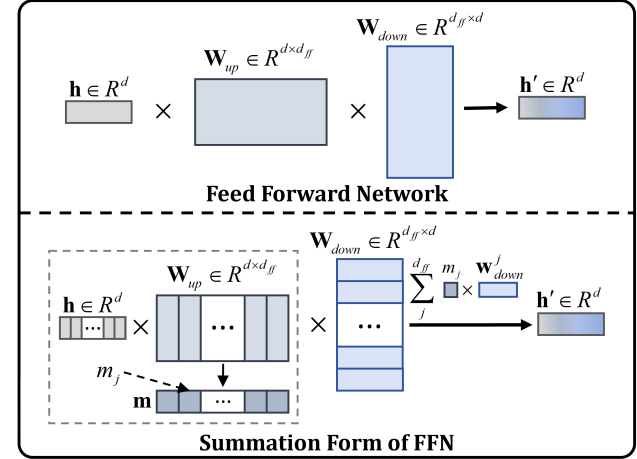


Figure 3. The summation form of FFN.

FFN as a Sum of Hidden Neurons. Similarly, the FFN layer (typically an MLP) projects the input to an intermediate dimension d_{ff} via \mathbf{W}_{up} , applies a non-linearity $\sigma(\cdot)$, and projects back via \mathbf{W}_{down} . As shown in Fig. 3, let $\mathbf{m} = \sigma(\mathbf{x} \mathbf{W}_{up}) \in \mathbb{R}^{d_{ff}}$ be the intermediate activation vector, where m_j is the scalar activation of the j -th neuron. By decomposing \mathbf{W}_{down} into row vectors $\mathbf{w}_{down}^{(j)} \in \mathbb{R}^d$, the output is:

$$\text{FFN}(\mathbf{x}) = \sum_{j=1}^{d_{ff}} m_j \cdot \mathbf{w}_{down}^{(j)} \quad (2)$$

Here, each neuron j contributes a vector scaled by its activation intensity m_j . This unified summation form reveals that the functional capability of a Transformer is distributed across these additive terms.

3.2. Importance Metric via Contribution-Awareness

Having established the summation form, the challenge lies in identifying which components (terms) are critical for a specific task. While traditional pruning relies on weight magnitude as a proxy for importance, recent work in model compression highlights that the input activation norm is equally critical (Sun et al., 2023). A parameter with a large weight magnitude contributes little to the residual stream if its input activation is consistently near zero.

Building on this insight, we propose a *Contribution Score* \mathcal{I} that measures the expected magnitude of the output vector generated by a component over a small probe set \mathcal{D}_{probe} . Taking an **FFN Neuron** j as an example, its importance is determined by the interaction between its activation magnitude and its output weight norm:

$$\mathcal{I}_{neuron}^{(j)} = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{probe}} \left[|m_j| \cdot \|\mathbf{w}_{down}^{(j)}\|_2 \right] \quad (3)$$

This metric directly reflects the dynamic contribution of a component to the model’s inference process, offering a more faithful representation of functional specialization than static weight analysis.

3.3. Empirical Analysis

To validate our motivation, we conduct an empirical analysis on the **Qwen2.5-1.5B-Instruct** model (Qwen et al., 2025). We use the contribution metric defined above to analyze the distribution of functional components across diverse tasks, including Coding (Wei et al., 2023), Instruction Following (Lambert et al., 2024), Mathematics (Tong et al., 2024), and Safety (Han et al., 2024). More visualizations across different layers and tasks are provided in the Appendix.

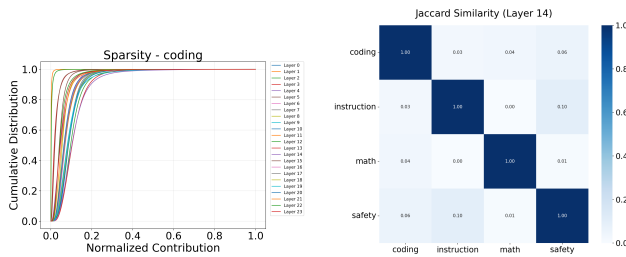


Figure 4. **Visualization on Qwen2.5-1.5B-Instruct.** (Left) The Cumulative Distribution Function of neuron contribution scores in the FFN layers for the *Coding* task. (Right) Jaccard Similarity Heatmap of the most important neurons across different tasks.

We first investigate whether task-specific knowledge requires the entire network. The left of Fig. 4 plots the Cumulative Distribution Function of normalized neuron importance scores for the *Coding* task. The curve exhibits a sharp rise near zero, indicating that the vast majority ($> 80\%$) of neurons have negligible contributions to the coding capability. This confirms that task knowledge is stored sparsely in a small subset of “loud” components.

A critical question for model merging is whether these sparse subspaces overlap between tasks. We select the top-20% most important neurons for each task in a representative middle layer (Layer 14 for Qwen2.5-1.5B-Instruct) and compute the Jaccard Similarity Coefficient of their indices. As shown in the right of Fig. 4, the heatmap reveals a striking pattern: the overlap between distinct tasks (e.g., Coding vs. Safety) is near zero.

These findings suggest that tasks reside in *orthogonal functional subspaces*. Parameter interference in standard merging arises because simple averaging (or unstructured pruning) forces the “loud” components of one task to mix with the “silent” noise of another. By explicitly identifying and masking these subspaces, we can theoretically disentangle task capabilities, merging them without destructive interference.

4. Methodology: CASS

Based on the theoretical and empirical insights in Sec. 3, we propose **CASS**, a unified framework to mitigate model merging interference. As illustrated in Fig. 5, CASS operates in two phases: (1) **Functional Subspace Identification**, where we generate task-specific structural masks using the pre-trained base model; and (2) **Subspace-Aware Integration**, where these masks are applied either to clean up task vectors post-training or to constrain gradient updates during fine-tuning.

4.1. Phase 1: Functional Subspace Identification

The core of CASS is to convert the continuous contribution scores derived in Eq. 3 into discrete binary masks that define the functional subspace of a task. Here, we derive these masks using the **pre-trained base model** θ_{base} , rather than individual fine-tuned checkpoints.

For a given task t with probe data D_t , we perform a forward pass using θ_{base} to compute the contribution scores $\mathcal{I}^{(c)}$ for each component c (head or neuron). We then employ a sparsity-constrained thresholding strategy. Let Θ be the set of all functional components. We select the top- $k\%$ components with the highest scores to form the active set, setting the binary mask $\mathbf{M}_t^{(c)} = 1$ for selected components and 0 otherwise. To accommodate the varying roles of different layers, we apply this ranking **globally** for Attention Heads and **layer-wise** for FFN Neurons.

4.2. Phase 2 (Scenario A): CASS-Merging

This scenario addresses the standard setting where we aim to merge independently fine-tuned models $\{\theta_t\}_{t=1}^K$. Standard methods typically merge task vectors $\tau_t = \theta_t - \theta_{base}$, which accumulate noise from components irrelevant to task t .

In CASS-Merging, we utilize the structural masks derived from Phase 1 as a **functional denoising filter**. Before merging, we project the raw task vector onto its identified subspace:

$$\tilde{\tau}_t = \mathbf{M}_t \odot \tau_t \quad (4)$$

where \odot denotes structured broadcasting multiplication. Physically, this operation reverts the parameters of non-essential heads and neurons back to their pre-trained state (since τ_t becomes 0 at masked positions). The sparsified vectors $\tilde{\tau}_t$ are then aggregated using standard operators (e.g., TIES, DARE, or simple summation):

$$\theta_{merged} = \theta_{base} + \lambda \sum_{t=1}^K \text{MergeOp}(\tilde{\tau}_t) \quad (5)$$

By filtering out “loud but irrelevant” updates, CASS-Merging explicitly reduces the probability of collision between tasks.

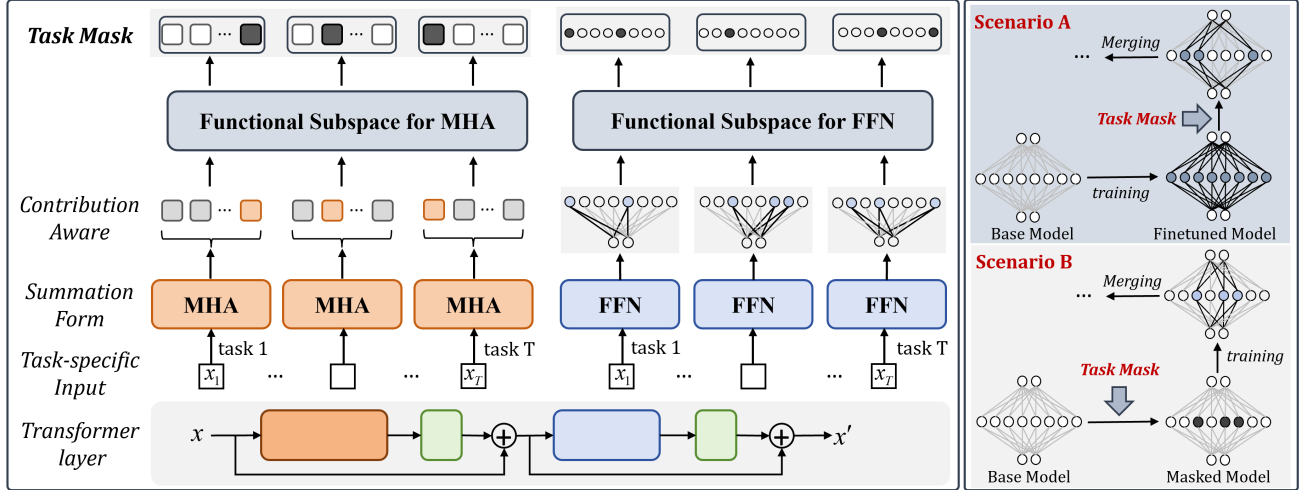


Figure 5. The proposed Contribution-Aware Structured Sparsity (CASS) framework.

4.3. Phase 2 (Scenario B): CASS-Tuning

While post-training merging is effective, it is inherently *reactive*—it attempts to fix interference after it has occurred. We propose CASS-Tuning as a **proactive** alternative that prevents interference at the source.

In this scenario, we use the mask \mathbf{M}_t (derived from θ_{base}) to constrain the optimization trajectory during the fine-tuning of task t . Specifically, we freeze the parameters belonging to the inactive subspace ($\mathbf{M}_t^{(c)} = 0$) by masking their gradients:

$$\mathbf{g}'_t = \mathbf{M}_t \odot \nabla_{\theta} \mathcal{L}_t \quad (6)$$

This ensures that the model updates are strictly confined to the pre-identified functional subspace.

Models trained via CASS-Tuning are “born orthogonal.” Since the masks \mathbf{M}_t for dissimilar tasks (e.g., Coding vs. Safety) are naturally disjoint (as shown in Sec. 3.3), the resulting task vectors occupy non-overlapping regions of the parameter space by design. This renders the subsequent merging process trivial and highly robust, effectively eliminating the need for complex conflict-resolution heuristics.

5. Experiments

5.1. Experimental Setup

Datasets and Models. To ensure a rigorous and standardized evaluation, we adopt the experimental protocols established in widely recognized benchmarks for both textual and visual modalities. For the **Text Modality**, we follow the settings of **MergeBench** (He et al., 2025b). We employ the Qwen2.5-0.5B-Instruct and Qwen2.5-1.5B-Instruct models as our base models. The evaluation encompasses four distinct downstream capabilities: Instruction Following, Mathematics, Coding, and Safety. For the **Vision Modality**, we adopt

the **FusionBench** (Tang et al., 2024) toolkit. We utilize ViT-B/32 and ViT-L/14 architectures (Dosovitskiy, 2020) pre-trained on CLIP (Radford et al., 2021). The evaluation covers a diverse set of eight downstream image classification tasks, ranging from object recognition to texture classification. Detailed specifications regarding the fine-tuning datasets, hyperparameters, and evaluation metrics are provided in the Appendix.

Baselines. We evaluate CASS as a versatile, plug-and-play framework designed to **enhance** existing merging algorithms. Instead of treating prior works as competitors, we integrate CASS with them to demonstrate its ability to mitigate interference across diverse merging paradigms. We select the following methods as our base aggregators: Task Arithmetic (Ilharco et al., 2022), TIES (Yadav et al., 2023), DARE (Yu et al., 2024), Localize-and-Stitch (He et al., 2025a), and Task Singular Vectors (Gargiulo et al., 2025). To ensure fair comparison, we report **normalized performance**, aligning with the standard metrics defined in the respective benchmarks.

5.2. Results on Text Modality

Table 1 presents the comparative results on Qwen2.5 models. We apply CASS-Merging as a denoising filter prior to aggregation. The results reveal several key insights:

Consistent Enhancement via Functional Denoising. CASS consistently boosts the performance of all base aggregators across both model scales. Whether applied to simple arithmetic (TA) or heuristic sparsification (TIES, DARE), our structural masks effectively filter out interference. Notably, on the 0.5B model, standard TIES and Localize-and-Stitch (LS) struggle significantly (averaging 0.695 and 0.745, respectively). This is likely because magnitude-based heuristics become unreliable in smaller models with less parameter redundancy. CASS effectively revitalizes these

Table 1. Results on Text Modality. Normalized performance comparison on Qwen2.5-0.5B-Instruct and Qwen2.5-1.5B-Instruct.

Method	Qwen2.5-0.5B-Instruct					Qwen2.5-1.5B-Instruct				
	Code	Safe	Math	Instr.	Avg.	Code	Safe	Math	Instr.	Avg.
Base Model	0.675	0.708	0.765	0.731	0.720	0.687	0.775	0.855	0.754	0.768
Fine-tuned Model	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Post-Mask Model	0.964	0.836	0.961	0.878	0.910	0.953	1.015	1.051	0.882	0.975
Task Arithmetic	0.800	0.687	0.765	0.718	0.742	0.735	0.778	0.942	0.703	0.789
w/ CASS-Merging	0.820	0.684	0.814	0.788	0.776	0.808	0.777	1.065	0.698	0.837
TIES	0.671	0.659	0.873	0.577	0.695	0.829	0.752	0.978	0.651	0.803
w/ CASS-Merging	0.753	0.731	0.863	0.743	0.773	0.846	0.767	1.094	0.610	0.829
DARE	0.857	0.697	0.892	0.743	0.797	0.933	0.722	1.072	0.641	0.842
w/ CASS-Merging	0.903	0.684	0.853	0.788	0.807	0.924	0.782	1.087	0.692	0.871
Localize-and-Stitch	0.613	0.713	0.941	0.711	0.745	0.781	0.745	0.935	0.749	0.802
w/ CASS-Merging	0.813	0.723	0.961	0.743	0.810	0.846	0.766	1.000	0.718	0.832

methods (boosting TIES to 0.773 and LS to 0.810), confirming that contribution-aware subspace identification is a more robust proxy for importance than weight magnitude alone.

Scaling and State-of-the-Art Performance. On the larger Qwen2.5-1.5B model, the benefits of CASS are further amplified. DARE w/ CASS-M achieves the highest average score of 0.871, establishing a new state-of-the-art for this setting. The scaling trend suggests that as model size increases, functional subspaces become more clearly defined (as hypothesized in Sec. 3), allowing CASS to separate tasks more precisely.

The “Post-Mask” Upper Bound. A striking observation is the performance of the “Post-Mask” models (Row 3). Even before merging, simply applying our masks to the fine-tuned models retains 91.0% (0.5B) and 97.5% (1.5B) of the original Fine-Tuned (FT) performance. In some cases (e.g., Math and Safety on 1.5B), the sparse expert even outperforms the dense FT model (scores > 1.0). This phenomenon validates our core premise: the vast majority of parameter updates in fine-tuning are noise or redundancy. By stripping them away, CASS not only reduces the collision probability for merging but also acts as a regularizer that can enhance individual task expertise.

5.3. Results on Vision Modality

To verify the cross-modal generalization of our framework, we evaluate CASS on eight diverse vision tasks using ViT-B/32 and ViT-L/14, and the results are shown in Table 2.

A key observation is that standard DARE performs relatively poorly on ViT-B/32 (Avg 0.741), falling behind even simple Task Arithmetic (0.756). This stands in contrast to its success in LLMs. However, DARE w/ CASS-M yields a dramatic recovery, boosting performance to 0.819 (+7.8%).

This confirms that structural awareness is critical for vision encoders: by preserving complete heads and neurons, CASS maintains the spatial integrity of visual features while still enabling the benefits of sparsity.

We explicitly compare CASS with TSV-Merging, a strong baseline that models task subspaces using Singular Value Decomposition (SVD) on weight matrices. While TSV is highly effective (0.901 on ViT-B/32), applying CASS on top of it further enhances performance to 0.907, and achieves the highest overall score of 0.964 on ViT-L/14. TSV defines subspaces based on static weight variance, whereas CASS defines them based on dynamic activation patterns. The fact that CASS adds value suggests that *activation-aware metrics capture functional nuances that weight decomposition alone misses*.

Moreover, CASS demonstrates exceptional robustness in tasks that typically cause high interference. For instance, on the SVHN task with ViT-B/32, standard TIES achieves 0.834, while TIES w/ CASS-M jumps to 0.939. SVHN (digit recognition) represents a distinct domain from natural images (e.g., Cars, EuroSAT). The significant gain indicates that CASS effectively disentangles the “digit recognition” subspace from the “natural object” subspace, preventing feature overwriting during merging.

5.4. Results of Mask-Guided Fine-Tuning (Scenario B)

We further evaluate the efficacy of Mask-Guided Fine-Tuning (Scenario B), where we explicitly constrain the optimization trajectory to task-specific subspaces identified by the base model. As shown in Table 3, this strategy fundamentally alters the mergeability of models, particularly for methods sensitive to parameter interference.

The performance of “Pre-Mask” models provides validation

Table 2. Results on Vision Modality. Normalized performance on 8 vision tasks using ViT-B/32 and ViT-L/14.

Method	DTD	EuroSAT	GTSRB	MNIST	RESISC	Cars	SUN397	SVHN	Average
<i>ViT-B/32 Architecture</i>									
Base Model	0.559	0.450	0.330	0.484	0.626	0.767	0.839	0.324	0.548
Fine-tuned Model	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Task Arithmetic	0.631	0.775	0.706	0.976	0.694	0.707	0.734	0.823	0.756
w/ CASS-Merging	0.739	0.770	0.723	0.963	0.789	0.827	0.869	0.889	0.821
TIES	0.684	0.769	0.703	0.968	0.777	0.828	0.863	0.834	0.803
w/ CASS-Merging	0.765	0.823	0.761	0.967	0.824	0.840	0.884	0.939	0.850
DARE	0.595	0.733	0.694	0.963	0.674	0.716	0.758	0.799	0.741
w/ CASS-Merging	0.733	0.765	0.722	0.964	0.789	0.824	0.868	0.887	0.819
TSV-Merging	0.778	0.925	0.947	0.994	0.864	0.896	0.849	0.954	0.901
w/ CASS-Merging	0.872	0.901	0.864	0.980	0.884	0.889	0.904	0.962	0.907
<i>ViT-L/14 Architecture</i>									
Base Model	0.659	0.614	0.510	0.766	0.732	0.843	0.829	0.595	0.693
Fine-tuned Model	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Task Arithmetic	0.780	0.927	0.875	0.992	0.890	0.889	0.900	0.896	0.894
w/ CASS-Merging	0.813	0.913	0.833	0.977	0.906	0.930	0.923	0.959	0.907
TIES	0.805	0.944	0.828	0.990	0.903	0.915	0.911	0.874	0.896
w/ CASS-Merging	0.843	0.935	0.876	0.977	0.927	0.948	0.945	0.986	0.930
DARE	0.796	0.848	0.812	0.976	0.873	0.915	0.906	0.963	0.886
w/ CASS-Merging	0.813	0.912	0.834	0.976	0.906	0.929	0.923	0.959	0.907
TSV-Merging	0.898	0.974	0.973	0.994	0.959	0.972	0.950	0.975	0.962
w/ CASS-Merging	0.938	0.975	0.945	0.982	0.955	0.968	0.960	0.991	0.964

for our subspace hypothesis. Even with updates restricted to a fixed sparse subset of parameters, the models recover the vast majority of the fully fine-tuned performance (e.g., 0.954 vs. 1.000 on Qwen2.5-0.5B). This confirms that the functional subspaces identified by CASS are sufficient to encapsulate task-specific capabilities, effectively serving as “trainable lottery tickets” (Frankle & Carbin, 2018) within the pre-trained weights.

The most notable impact of CASS-Tuning is observed in TIES-Merging. Standard TIES often struggles on smaller models (0.695 on 0.5B) because it relies on weight magnitude as a proxy for importance, which can be noisy in unconstrained fine-tuning. However, under CASS-Tuning, TIES performance surges to 0.816 (+12.1%). This is because our gradient masking ensures that any parameter with a non-zero magnitude is genuinely task-relevant by design. Consequently, CASS-Tuning converts TIES’s heuristic assumption (Magnitude \approx Importance) into a rigorous property, eliminating the interference noise that typically plagues magnitude-based merging.

Comparing CASS-Tuning (Table 3) with CASS-Merging

(Table 1) reveals an interesting trade-off between *optimization freedom* and *structural compatibility*. While CASS-Tuning yields superior results for TIES, it performs comparably or slightly lower for DARE and Task Arithmetic on the 1.5B model. We attribute this to the fact that CASS-Merging benefits from unconstrained exploration during fine-tuning, potentially capturing richer features before pruning. In contrast, CASS-Tuning trades a small fraction of single-task performance for *inherent orthogonality*. This makes CASS-Tuning particularly valuable for scenarios requiring storage efficiency (sparse checkpoints) or when using merging algorithms that demand strict conflict avoidance like TIES.

5.5. OOD Generalization and Robustness

Beyond task-specific performance, a critical desideratum for model merging is preserving the general capabilities and truthfulness of the pre-trained base model, preventing catastrophic forgetting during the integration of specialized experts. To evaluate this, we test the merged Qwen2.5-1.5B-Instruct model (using Task Arithmetic) on two out-of-distribution (OOD) benchmarks: MMLU (Hendrycks et al., 2020) (general knowledge) and TruthfulQA (Lin et al.,

Table 3. Results of Mask-Guided Fine-Tuning. Normalized performance comparison on Qwen2.5.

Method	Qwen2.5-0.5B-Instruct					Qwen2.5-1.5B-Instruct				
	Code	Safe	Math	Instr.	Avg.	Code	Safe	Math	Instr.	Avg.
Base Model	0.676	0.708	0.765	0.731	0.720	0.687	0.775	0.855	0.754	0.768
Fine-tuned Model	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Pre-Mask Model	0.966	0.988	0.902	0.962	0.954	0.885	1.008	0.978	0.826	0.924
Task Arithmetic	0.801	0.687	0.765	0.718	0.743	0.735	0.778	0.942	0.703	0.789
w/ CASS-Tuning	0.889	0.705	0.863	0.788	0.811	0.798	0.772	1.007	0.703	0.820
TIES	0.671	0.659	0.873	0.577	0.695	0.829	0.752	0.978	0.651	0.803
w/ CASS-Tuning	0.788	0.715	1.000	0.763	0.816	0.969	0.748	1.073	0.677	0.867
DARE	0.857	0.697	0.892	0.744	0.797	0.933	0.722	1.073	0.641	0.842
w/ CASS-Tuning	0.853	0.690	0.922	0.776	0.810	0.933	0.743	1.022	0.703	0.850
Localize-and-Stitch	0.613	0.713	0.941	0.711	0.745	0.781	0.745	0.935	0.749	0.802
w/ CASS-Tuning	0.757	0.692	0.843	0.731	0.756	0.828	0.748	1.036	0.692	0.826

2022) (truthfulness generation).

As presented in Table 4, standard Task Arithmetic exhibits the lowest performance, suggesting that unconstrained parameter updates during fine-tuning induce noise that interferes with the model’s pre-existing knowledge. In contrast, applying CASS consistently enhances OOD generalization. CASS-Merging recovers a portion of the lost capability (e.g., boosting MMLU from 0.4531 to 0.4563), acting as a retrospective filter that removes task-irrelevant updates likely responsible for overwriting general knowledge.

Table 4. OOD Generalization Analysis.

Method	MMLU	TruthfulQA
Task Arithmetic	0.4531	0.2717
w/ CASS-Merging	0.4563	0.2729
w/ CASS-Tuning	0.4603	0.2778

Crucially, CASS-Tuning achieves the highest scores across both benchmarks (0.4603 on MMLU). This reinforces the advantage of our mask-guided training strategy: by freezing parameters outside the task-specific functional subspace, CASS-Tuning effectively acts as a *knowledge safeguard*. It ensures that the vast majority of the base model’s general-purpose circuits remain untouched during fine-tuning. Consequently, the resulting merged model retains the “best of both worlds”, the specialized skills of the experts and the broad, robust foundation of the pre-trained model.

5.6. Ablation Study

To dissect the source of our performance gains, we conduct a component-wise ablation study on the Qwen2.5-0.5B-Instruct model using Task Arithmetic. We apply CASS-Merging structured masks exclusively to either the Multi-

Head Attention layers (Head Only) or the Feed-Forward Networks (Neuron Only) and compare them with the full framework.

As shown in Fig. 6, both components contribute positively to the overall performance, but with distinct characteristics. Applying masks solely to FFN neurons yields a higher average improvement (0.769) compared to masking attention heads (0.758). This aligns well with the mechanistic interpretation that FFNs serve as the primary “Key-Value memories” for storing task-specific knowledge; thus, denoising FFNs effectively removes conflicting knowledge patterns.

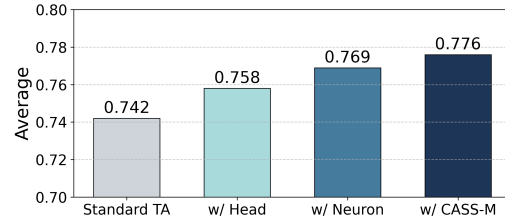


Figure 6. **Component-wise Ablation Study.** Performance comparison on Qwen2.5-0.5B-Instruct using Task Arithmetic (TA) as the base aggregator. We investigate the impact of applying CASS-Merging to Multi-Head Attention (w/ Head) or Feed-Forward Networks (w/ Neuron).

6. Conclusion

This paper presents **CASS**, a framework that mitigates merging interference by isolating functional subspaces via activation-aware metrics. Grounded in the summation form of Transformer layers, CASS provides a rigorous physical basis for structured sparsity. Experimental results across textual (Qwen) and visual (ViT) modalities demonstrate that CASS consistently enhances baseline methods.

7. Impact Statements

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

- Cheng, R., Xiong, F., Wei, Y., Zhu, W., and Yuan, C. Who-ever started the interference should end it: Guiding data-free model merging via task vectors. *arXiv preprint arXiv:2503.08099*, 2025.
- Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. What does bert look at? an analysis of bert’s attention. *arXiv preprint arXiv:1906.04341*, 2019.
- Dosovitskiy, A. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.
- Gao, L., Rajaram, A., Coxon, J., Govande, S. V., Baker, B., and Mossing, D. Weight-sparse transformers have interpretable circuits. *arXiv preprint arXiv:2511.13653*, 2025.
- Gargiulo, A. A., Crisostomi, D., Bucarelli, M. S., Scardapane, S., Silvestri, F., and Rodola, E. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18695–18705, 2025.
- Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.
- Han, S., Rao, K., Ettinger, A., Jiang, L., Lin, B. Y., Lambert, N., Choi, Y., and Dziri, N. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in Neural Information Processing Systems*, 37:8093–8131, 2024.
- Hanna, M., Liu, O., and Variengien, A. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060, 2023.
- He, Y., Hu, Y., Lin, Y., Zhang, T., and Zhao, H. Localize-and-stitch: Efficient model merging via sparse task arithmetic. *Transactions on Machine Learning Research*, 2025a.
- He, Y., Zeng, S., Hu, Y., Yang, R., Zhang, T., and Zhao, H. Mergebench: A benchmark for merging domain-specialized llms. *arXiv preprint arXiv:2505.10833*, 2025b.
- Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.
- Jin, X., Ren, X., Preotiuc-Pietro, D., and Cheng, P. Data-less knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.
- Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.
- Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.
- Matena, M. S. and Raffel, C. A. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.
- Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.
- Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.
- Qwen, :, Yang, A., et al. Qwen2.5 technical report, 2025. URL <https://arxiv.org/abs/2412.15115>.
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.
- Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

- Tang, A., Shen, L., Luo, Y., Hu, H., Du, B., and Tao, D. Fusionbench: A comprehensive benchmark of deep model fusion. *arXiv preprint arXiv:2406.03280*, 2024.
- Tong, Y., Zhang, X., Wang, R., Wu, R., and He, J. Dartmath: Difficulty-aware rejection tuning for mathematical problem-solving. *Advances in Neural Information Processing Systems*, 37:7821–7846, 2024.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Wang, K., Dimitriadis, N., Ortiz-Jimenez, G., Fleuret, F., and Frossard, P. Localizing task information for improved model merging and compression. *arXiv preprint arXiv:2405.07813*, 2024.
- Wei, Y., Wang, Z., Liu, J., Ding, Y., and Zhang, L. Magi-coder: Empowering code generation with oss-instruct. *arXiv preprint arXiv:2312.02120*, 2023.
- Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.
- Wu, W., Wang, Y., Xiao, G., Peng, H., and Fu, Y. Retrieval head mechanistically explains long-context factuality. *arXiv preprint arXiv:2404.15574*, 2024.
- Xiao, C., Zhang, Z., Song, C., Jiang, D., Yao, F., Han, X., Wang, X., Wang, S., Huang, Y., Lin, G., et al. Configurable foundation models: Building llms from a modular perspective. *arXiv preprint arXiv:2409.02877*, 2024.
- Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.
- Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.