Superpose Task-specific Features for Model Merging

Haiquan Qiu[1], You Wu[1], Dong Li[2], Jianmin Guo[2], Quanming Yao[1,3]*,
[1]Tsinghua University, [2]Huawei,
[3]State Key laboratory of Space Network and Communications
Correspondence: qyaoaa@tsinghua.edu.cn

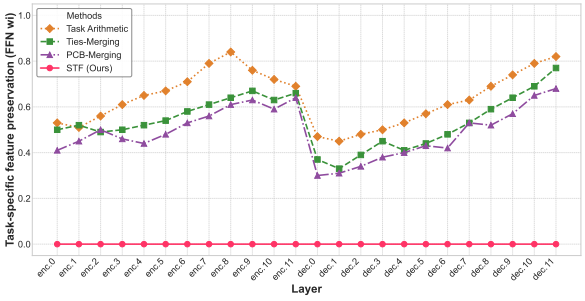
Abstract

Model merging enables powerful capabilities in neural networks without requiring additional training. In this paper, we introduce a novel perspective on model merging by leveraging the fundamental mechanisms of neural network representation. Our approach is motivated by the linear representation hypothesis, which states that neural networks encode information through linear combinations of feature vectors. We propose a method that superposes task-specific features from individual models into a merged model. Our approach specifically targets linear transformation matrices, which are crucial for feature activation and extraction in deep networks. By formulating the merging process as a linear system, we can preserve task-specific features from individual models and create merged models that effectively maintain multi-task capabilities compared to existing methods. Extensive experiments across diverse benchmarks and models demonstrate that our method outperforms existing techniques. Code is available at https://github.com/LARS-research/STF.

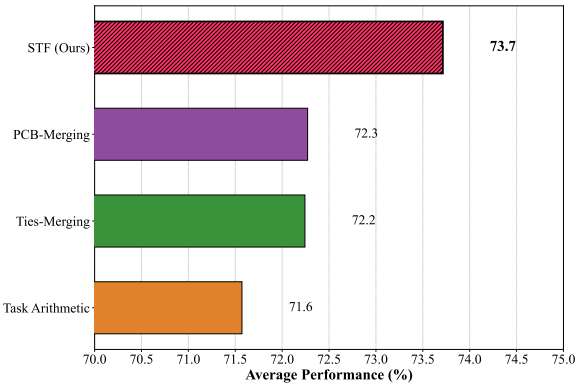
1 Introduction

Because of the increasing resource demands of training models, model merging has emerged as an efficient approach to consolidate capabilities from specialist models while reducing storage and deployment costs. Various model merging methods have been proposed, including parameter averaging (Choshen et al., 2022; Wortsman et al., 2022) and task vectors (Ilharco et al., 2022; Du et al., 2024). However, these methods primarily focus on parameter-level operations and do not explicitly incorporate the fundamental working mechanisms of neural networks in their design.

We argue that a principled approach to model merging should be conditioned on how deep neural networks represent and process information. Therefore, to design our merging method, we draw upon



(a) Task-specific feature preservation of merged methods



(b) Performance on T5 model merging

Figure 1: We measure the task-specific feature preservation of merged methods in (a) (refer to Appendix A), where smaller values indicate better preservation. The merging performance in (b) strongly correlates with task-specific feature preservation.

the linear representation hypothesis (Mikolov et al., 2013; Arora et al., 2016; Olah et al., 2020), which states that neural network representations can be decomposed into combinations of feature vectors. Recent works in mechanistic interpretability (Elhage et al., 2022; Bricken et al., 2023; Templeton et al., 2024; Lindsey et al., 2025) validate the hypothesis and also reveal that these representations often contain features both related and unrelated to the model input. This phenomenon motivates our approach: linearly superposing features from indi-

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 4200–4214
November 4-9, 2025

vidual models into the representation of the merged model can preserve task-specific capabilities.

Therefore, we propose a method that Superposes Task-specific Features(STF) from individual fine-tuned models. Our approach specifically targets linear transformation matrices, which comprise the majority of model parameters and form the foundation of the linear representation hypothesis. These matrices activate features through inner products with row vectors and extract new features through linear combinations via column vectors, making them crucial for detecting and processing information in deep neural networks. We design the merged linear transformation matrices to preserve the output features from individual models when processing the same input. By identifying task-specific features through singular value decomposition of task matrices, we formulate feature superposition as a linear system that derives the optimal merged task matrix. The final merged model is created by adding this merged matrix to the pre-trained model. Experiments verified the strong positive correlation between feature preservation and performance improvement(see Fig. 1). We validate our method on multiple benchmark datasets and models. The results demonstrate that our method consistently outperforms existing model merging techniques.

Notations Throughout this paper, we use bold uppercase letters (e.g., $\mathbf{X}$) to denote matrices, bold lowercase letters (e.g., $\mathbf{x}$) to denote vectors, and regular lowercase letters (e.g., x) to denote scalars. For a matrix $\mathbf{X}$, its transpose is denoted as $\mathbf{X}^\top$. We denote task i as T_i and its corresponding model parameters as θ_i. For linear transformations, a linear transformation matrix in the model trained on task i is represented as $\mathbf{P}_i$. The subscript $_{\text{pre}}$ indicates pre-trained parameters or matrices - for example, θ_{pre} and $\mathbf{P}_{\text{pre}}$ denote the parameters and linear transformation matrices of the pre-trained model, respectively. The notation $\mathbf{X} \circ \mathbf{Y}$ denotes the Hadamard product of matrices $\mathbf{X}$ and $\mathbf{Y}$.

2 Related Work

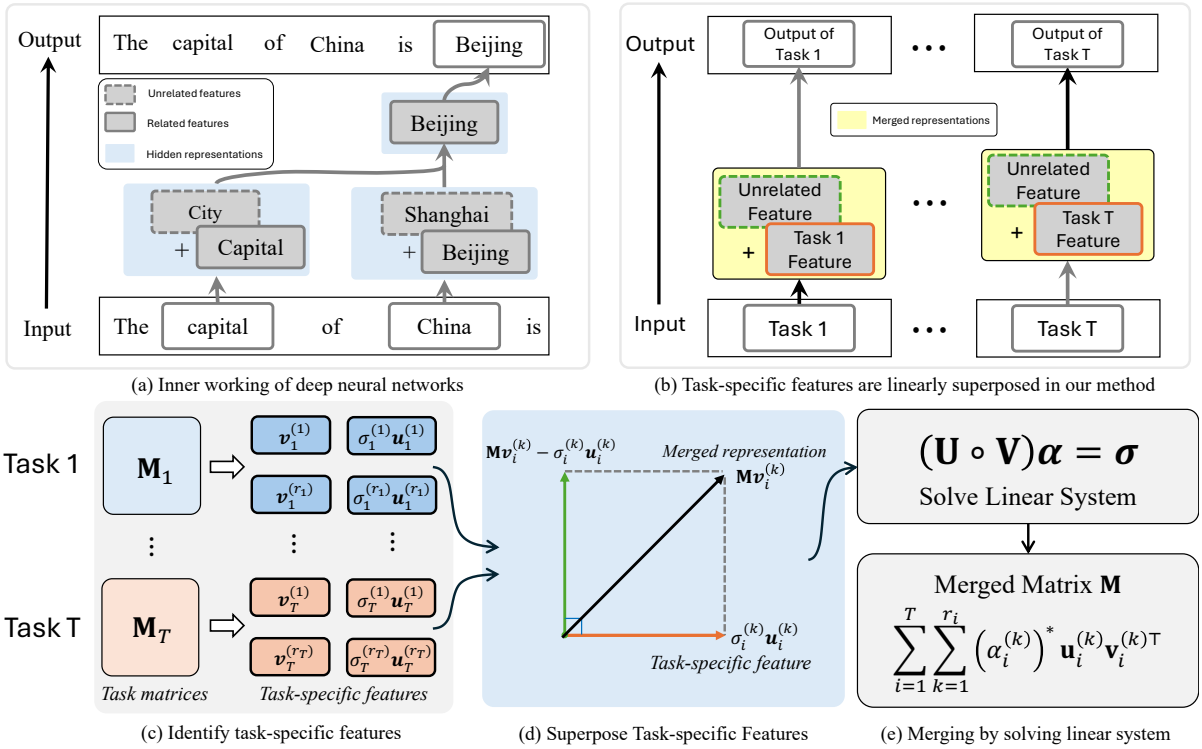
2.1 Model Merging of Fine-tuned Models

Model merging is a technique that combines multiple models into a single model to enhance performance or enable the model to perform multiple tasks. Previous studies have shown that averaging the weights of multiple models fine-tuned from the same pre-trained initialization is a promising approach for model merging. Fisher Merging (Matena and Raffel, 2022) advances beyond simple averaging by utilizing the Fisher information matrix to assess the importance of individual parameters, which are then weighted accordingly during the merging process. Similarly, RegMean (Jin et al., 2022) forms a linear regression problem with extra data for each layer and offers a closed-form solution for the merged model's parameters by solving the regression problem.

Beyond parameter averaging, Task Arithmetic (Ilharco et al., 2022) introduces task vectors and adding the task vectors of individual tasks to merge model, demonstrating their effectiveness and lightweight nature in facilitating cross-task generalization. Building on this concept, PEM Composition (Zhang et al., 2023) extends the task arithmetic framework to merge LoRA (Hu et al., 2021), while Ties-Merging (Yadav et al., 2023) addresses task conflicts by resetting redundant parameters and resolving sign conflicts. These methods, however, use a single merging coefficient across all task vectors, which limits their flexibility. In contrast, Lorahub (Huang et al., 2023) and AdaMerging (Yang et al., 2023) use different coefficients for enhanced adaptability. Lorahub's performance is limited as it only searches for coefficients at the task level, while AdaMerging requires complex training and unlabeled test datasets, making it applicable solely to classification problems. DARE (Yu et al., 2024) proposes drop and rescale as preprocessing steps when merging fine-tuned LLMs. PCB-Merging (Du et al., 2024) is a lightweight, training-free technique for model merging that balances parameter competition by intra-balancing parameter significance within tasks and inter-balancing parameter similarities across tasks, effectively enhancing performance across various scenarios.

2.2 Linear Representation Hypothesis

The linear representation hypothesis states that neural networks encode information by summing up feature vectors (Mikolov et al., 2013; Arora et al., 2016; Olah et al., 2020), i.e., a layer of a network represents a set of features as a weighted sum of task-associated vectors. This hypothesis has been observed in various models, including word embeddings (Mikolov et al., 2013; Conneau et al., 2017), sentence embeddings (Bowman et al., 2015), Transformer language models (Meng et al., 2022; Hendel et al., 2023), and vision-language models (Trager et al., 2023; Perera et al., 2023). The hypothesis has been exploited in various fields, especially in



(a) Inner working of deep neural networks

(b) Task-specific features are linearly superposed in our method

(c) Identify task-specific features

(d) Superpose Task-specific Features

(e) Merging by solving linear system

Figure 2: Deep neural networks (a) linearly add both relevant and irrelevant features in hidden representations. This insight motivates our approach to model merging through linear superposition of task-specific features. The second row illustrates how STF preserves and combines essential features from individual models to create an effective merged method.

probing (Alain and Bengio, 2018; Belinkov, 2022) and interpretability (nostalgebraist, 2020; Elhage et al., 2022; Bricken et al., 2023; Gao et al., 2024; Lindsey et al., 2025).

The linear representation hypothesis has been validated in recent research on mechanistic interpretability of language models (Elhage et al., 2022; Bricken et al., 2023; Templeton et al., 2024), which showed that neural networks learn to represent meaningful features through linear combinations of neurons.

3 Method

3.1 Problem Formulation

We start with a set of tasks $\{\mathsf{T}_1, \ldots, \mathsf{T}_T\}$ and various pre-trained models. The objective is to fine-tune these models either by updating all parameters or using parameter-efficient fine-tuning (PEFT) methods. The goal of model merging is to combine multiple fine-tuned models into a single model that can perform all tasks $\{\mathsf{T}_1, \ldots, \mathsf{T}_T\}$ effectively without requiring access to training data.

To superpose task-specific features, we focus on merging the linear transformation matrices of the models, as these matrices are core components contributed to the linear representation hypothesis (Elhage et al., 2022). In this paper, instead of merging linear transformation matrices directly, we merge the *task matrices* $\mathbf{M}_i = \mathbf{P}_i - \mathbf{P}_{\text{pre}} \in \mathbb{R}^{m \times n}$ where $\mathbf{P}_i$ is a linear transformation matrix for model of task T_i, $\mathbf{P}_{\text{pre}}$ is the linear transformation matrix for the pre-trained model, as the task matrices contain the task-specific information compared to model parameters that contains both task-specific and pre-trained information. Specifically, we merge task matrices $\{\mathbf{M}_i\}$, i.e., into a single merged matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$. For an input feature $\mathbf{x} \in \mathbb{R}^n$, we aim to ensure that the output of the merged model $\mathbf{M}\mathbf{x}$ maintains the features of the outputs $\mathbf{M}_i\mathbf{x}$ of the models for task T_i. The final merged model is then constructed by adding the merged matrix $\mathbf{M}$ to the pre-trained model with a scaling factor γ: $\mathbf{P} = \mathbf{P}_{\text{pre}} + \gamma\mathbf{M}$.

3.2 Task-specific Feature Superposition

Identify Task-specific Features To merge the linear transformation matrices to linearly superpose task-specific features, we need to identify these features first. In this paper, we choose the singular vectors of the task matrices as the

task-specific features, i.e., decomposing $\mathbf{M}_i$ as $\mathbf{M}_i = \sum_{k=1}^{r_i} \sigma_i^{(k)} \mathbf{u}_i^{(k)} \mathbf{v}_i^{(k)\top}$, where $\sigma_i^{(k)}$ is the k-th singular value, and $\mathbf{u}_i^{(k)} \in \mathbb{R}^m$ and $\mathbf{v}_i^{(k)} \in \mathbb{R}^n$ are the corresponding left and right singular vectors. This decomposition naturally provides task-specific features: the right singular vectors $\mathbf{v}_i^{(k)}$ form input features for the input $\mathbf{x}$, and $\sigma_i^{(k)} \mathbf{u}_i^{(k)}$ forms the task-specific feature for the output of task matrices.

Remark 1. *Singular vectors also reveal how the model processes information, i.e., the right singular vectors act as feature detectors through inner products with inputs, while the left singular vectors and singular values determine how detected features are combined into outputs. Refer to Appendix B for detailed analysis of singular vector.*

Objective of Superposing Features Given the input features $\mathbf{v}_i^{(k)}$ of task i, we want that after the features transformed by merged matrix $\mathbf{M}$, the output $\mathbf{M}\mathbf{v}_i^{(k)}$ contains the task-specific output feature $\sigma_i^{(k)} \mathbf{u}_i^{(k)}$. This leads to the following objective:

$$\langle \sigma_i^{(k)} \mathbf{u}_i^{(k)}, \underbrace{\mathbf{M}\mathbf{v}_i^{(k)} - \sigma_i^{(k)} \mathbf{u}_i^{(k)}}_{\text{Unrelated features}} \rangle = 0, \quad (1)$$

where $\langle \cdot, \cdot \rangle$ is the inner product. In (1), we first obtain the unrelated features by subtracting the task-specific output feature $\sigma_i^{(k)} \mathbf{u}_i^{(k)}$ from the transformed feature $\mathbf{M}\mathbf{v}_i^{(k)}$. The inner product (1) should be zero, which means that the unrelated features are orthogonal to the task-specific output feature. This ensures that the merged matrix $\mathbf{M}$ preserves the task-specific feature $\sigma_i^{(k)} \mathbf{u}_i^{(k)}$ while allowing other features to be superposed linearly.

Remark 2 (Comparison against RegMean). *Ideally, we would fully preserve features by requiring* $\mathbf{M}\mathbf{v}_i^{(k)} = \mathbf{M}_i \mathbf{v}_i^{(k)}$ *for all tasks (similar to RegMean). However, due to the vast number of features, merged parameters cannot simultaneously maintain all features, as this would lead to an overdetermined system of equations. Instead, our approach, STF, superposes task-specific features rather than requiring exact feature preservation, resulting in increased representation efficiency and superior performance.*

3.3 Merging by Solving Linear System

To merge the task matrices, we can see that the superposition objective (1) is a linear equation with the unknowns being the merging matrices $\mathbf{M}$. However, direct solving $\mathbf{M}$ is not feasible because of numerous unknowns and equations in the linear system consisting of (1). For efficient merging, we instead merge the task matrices $\mathbf{M}_i$ by decomposing them via SVD and merging their singular decomposition components:

$$\mathbf{M} = \sum_{i=1}^{T} \sum_{k=1}^{r_i} \alpha_i^{(k)} \mathbf{u}_i^{(k)} \mathbf{v}_i^{(k)\top}, \quad (2)$$

where $\alpha_i^{(k)}$ are the merging weights. This reduces the number of unknowns and equations to $r = \sum_{i=1}^{T} r_i$, where r_i is the rank of task matrix $\mathbf{M}_i$. By merging in the singular space, we can ensure that the resulting problem becomes a linear system of equations with the following theorem (proved in Appendix C).

Theorem 1. *Given the task matrices* $\mathbf{M}_i$ *with their SVD decompositions, the merging weights* $\alpha_i^{(k)}$ *can be obtained by solving the linear system:*

$$(\mathbf{U} \circ \mathbf{V})\boldsymbol{\alpha} = \boldsymbol{\sigma}, \quad (3)$$

where

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1^{(1)\top}\mathbf{u}_1^{(1)} & \cdots & \mathbf{u}_1^{(1)\top}\mathbf{u}_T^{(r_T)} \\ \vdots & \ddots & \vdots \\ \mathbf{u}_T^{(r_T)\top}\mathbf{u}_1^{(1)} & \cdots & \mathbf{u}_T^{(r_T)\top}\mathbf{u}_T^{(r_T)} \end{bmatrix},$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1^{(1)\top}\mathbf{v}_1^{(1)} & \cdots & \mathbf{v}_T^{(r_T)\top}\mathbf{v}_1^{(1)} \\ \vdots & \ddots & \vdots \\ \mathbf{v}_1^{(1)\top}\mathbf{v}_T^{(r_T)} & \cdots & \mathbf{v}_T^{(r_T)\top}\mathbf{v}_T^{(r_T)} \end{bmatrix},$$

$$\boldsymbol{\alpha} = [\alpha_1^{(1)}, \cdots, \alpha_T^{(r_T)}] \ , \ \boldsymbol{\sigma} = [\sigma_1^{(1)}, \cdots, \sigma_T^{(r_T)}].$$

By solving the linear system (3), we obtain the optimal weights $\boldsymbol{\alpha}^*$ for merging task matrices according to (2). The merged linear transformation matrix $\mathbf{P}$ is then obtained by adding the merged task matrix $\mathbf{M}$ to the pre-trained matrix $\mathbf{P}_{\text{pre}}$ with a scaling factor γ: $\mathbf{P} = \mathbf{P}_{\text{pre}} + \gamma\mathbf{M}$.

The algorithm for merging task matrices is summarized in Algorithm 1 and Fig. 2. Because there are T tasks matrices of size $m \times n$, SVD of these matrices takes the time complexity $O(Tmn^2)$. To merge T task matrices, the linear system has r equations and variables to solve, which takes the

Algorithm 1 Merging task matrices: $\mathsf{STF}(\{\mathbf{M}_i\})$

Input: task matrices $\mathbf{M}_i, i = 1, \ldots, T$;
Apply SVD to $\mathbf{M}_i$ to obtain $\sigma_i^{(k)}$, $\mathbf{u}_i^{(k)}$, and $\mathbf{v}_i^{(k)}$;
Prepare $\mathbf{U}$, and $\mathbf{V}$;
Solve $(\mathbf{U} \circ \mathbf{V})\boldsymbol{\alpha} = \boldsymbol{\sigma}$ to obtain $\boldsymbol{\alpha}^*$;
Obtain the merged task matrix $\mathbf{M}$ by (2);
Return: merged task matrices $\mathbf{M}$.

time complexity $O(r^3)$. Therefore, the overall time complexity of STF for merging T task matrices is $O(Tmn^2 + r^3)$. See Section 4.3 for the time consumption of model merging with STF for various models.

3.4 Complete Algorithm

Except the linear transformation matrices, there are parameters in the model that need to be merged, such as biases, normalization parameters, and embeddings. For biases and embeddings, we merge them with task arithmetic, i.e., adding their task vectors together. For normalization parameters, we merge them by averaging the normalization parameters of the individual tasks because the normalization can be seen as a linear transformation with diagonal matrix.

We also follow Ties-Merging (Yadav et al., 2023) and apply a trimming step. This involves keeping only the top $\eta\%$ parameters by magnitude while setting the remaining parameters to zero. This preprocessing step helps reduce noise and focus on the most significant parameters during merging. We present the complete algorithm for merging models in Algorithm 2.

Algorithm 2 Complete Merging Algorithm

Input: fine-tuned models $\{\theta_1, \ldots, \theta_T\}$, pre-trained model θ_{pre}
Parameters: sparsity ratio η, scaling factor γ
for each linear transformation layer **do**
  Extract task matrices $\mathbf{M}_i = \mathbf{P}_i - \mathbf{P}_{\text{pre}}$ for $i = 1, \ldots, T$
  Keep top η parameters by magnitude in each $\mathbf{M}_i$
  $\mathbf{M} \leftarrow \mathsf{STF}(\{\mathbf{M}_i\}_{i=1}^{T})$ {Algorithm 1}
  Merged matrix $\mathbf{P} = \mathbf{P}_{\text{pre}} + \gamma\mathbf{M}$
end for
for each bias, embedding, normalization layer **do**
  Extract task vectors $\tau_i = \theta_i - \theta_{\text{pre}}$ for $i = 1, \ldots, T$
  Keep top η parameters by magnitude in each τ_i
  if normalization layer **then**
    Merged parameters $\leftarrow \mathsf{mean}(\{\tau_1, \ldots, \tau_T\})$
  else
    Merged parameters $\leftarrow \gamma \sum_{i=1}^{T} \tau_i$
  end if
end for
Return: merged model parameters

3.5 Discussion

Merging Task Matrices or Fine-tuned Matrices? In this paper, our focus is on merging task matrices instead of fine-tuning matrices. We have discovered that fine-tuned linear transformation matrices tend to have more shared features across tasks. Therefore, merging the fine-tuned linear transformation matrix is not as efficient as merging task matrix because certain overlapping direction of singular vectors correspond to these common features. On the contrary, the singular vectors of the task matrix contain features that are more specific to individual tasks, making it a more effective approach for merging. See the results of merging fine-tuned matrices in Section 4.3.

Feature Interference and Scaling Factor In STF, we scale the merged matrix $\mathbf{M}$ to reduce interference between task-specific features. During merging, features from different tasks can interfere since we combine them through linear superposition. While using singular vectors as basis helps minimize interference within each task, some features may still be diminished when their directions conflict with features from other tasks. The scaling factor γ plays a crucial role in managing this tradeoff - a small γ reduces interference but may weaken task-specific features, while a large γ better preserves individual task capabilities but risks amplifying interference between tasks. Finding the optimal γ requires careful tuning based on the specific tasks and model architecture. In Section 5, we discuss future directions for reducing feature interference beyond scaling.

4 Experiments

4.1 Experimental Setup

Evaluation Setup. We evaluate STF across diverse fine-tuning methods, tasks, model architectures, and model size. For the fully fine-tuning, we evaluate NLP tasks on T5 following protocols of Yadav et al. (2023). For parameter-efficient fine-tuning (PEFT), we evaluate LoRA on GPT-2 for NLP tasks since it uses linear transformation matrices through low-rank adaptation, which aligns well with our approach, rather than vector parameters like IA3 (Liu et al., 2022) in previous research. To evaluate STF when model size scales up, we follow the setup of Du et al. (2024) to merge LLMs. Additionally, to evaluate the out-of-distribution robustness, we follow Yadav et al. (2023) to test it on

NLP tasks.

All experiments are conducted on NVIDIA A100 GPUs and AMD EPYC 7763 CPUs. STF is implemented in PyTorch and performed on NVIDIA A100 GPUs. The hyperparameter settings for STF are in Appendix D.2.

Baseline Methods. We compare STF against four established model merging approaches: (1) **Averaging** (Choshen et al., 2022; Wortsman et al., 2022), which computes the element-wise mean of individual models; (2) **Task Arithmetic** (Ilharco et al., 2022), which merges by scaling and adding task vectors to the initial model; (3) **Fisher Merging** (Matena and Raffel, 2022), which approximates Fisher Information Matrix to weight parameters based on their importance for each task and combines the weighted parameters into the final merged model; (4) **RegMean** (Jin et al., 2022), which computes a closed-form solution to a least-squares regression problem that aims to minimize the distance between the merged model's activations and the individual models' activations. (5) **Ties-Merging**, which enhances merging by eliminating redundant parameters and resolving sign conflicts; (6) **PCB-Merging**, which balances parameter competition through intra-task significance and inter-task similarity analysis. We also report results from individual **fine-tuned models** and a **pre-trained model** on all tasks; (7) **MetaGPT** (Zhou et al., 2024), which formalizes the objective of model merging into a multi-task learning framework, aiming to minimize the average loss difference between the merged model and each individual task model; (8) **Knots** (Stoica et al., 2024), which utilizes Singular Value Decomposition to transform and align task-updates from multiple LoRA models into a shared representational space; (9) **Consensus TA** (Wang et al., 2024) and (10) **Localize-and-Stitch** (He et al., 2024), both of which focus on constructing a sparse mask for each task to reduce inter-task conflict during merging.

4.2 Results

Fully Fine-tuned NLP Model: T5 Merging For NLP experiments, we evaluate on T5-base (Raffel et al., 2020), an encoder-decoder transformer (Vaswani et al., 2017) pre-trained with masked language modeling. We finetune T5-base on seven diverse tasks spanning question answering (QASC (Khot et al., 2020), WikiQA (Yang et al., 2015), QuaRTz (Tafjord et al., 2019)), paraphrase identification (PAWS (Zhang et al., 2019)), sentence completion (Story Cloze (Sharma et al., 2018)), and coreference resolution (Winogrande (Sakaguchi et al., 2020), WSC (Levesque et al., 2012)). We use the code from Yadav et al. (2023) to finetune the model on these tasks. To reduce variance and ensure reliable evaluation, we report the experimental results of the average performance over different templates (Bach et al., 2022) for each task.

As shown in Table 1, STF achieves state-of-the-art performance when merging T5-base models, outperforming existing methods by 1.4% on average across 7 tasks. STF shows particularly strong performance on PAWS and Story Cloze, with improvements of 5.9% and 12.5% respectively over PCB-Merging, the previous best method.

PEFT of NLP Model: LORA Adapters Merging For PEFT experiments, we evaluate GPT-2 Medium model on LoRA adapters (Hu et al., 2021), which are task-specific adapters that are fine-tuned on NLP task datasets. These tasks include converting tables (E2E(Novikova et al., 2017)), knowledge graph (WebNLG(Gardent et al., 2017)) and structured data (DART(Nan et al., 2020)) to natural language. We use the released checkpoints from Hu et al. (2021) and evaluate the performance of the merged model on these datasets. Specifically, we compare Knots, which is designed specifically for LoRA merging.

As shown in Table 2, STF outperforms existing baselines on all metrics. Specifically, comparing with previous best baselines, STF shows improvements of 2.2% for NIST of E2E, 4.8% over PCB-Merging for CIDEr of E2E.

Large Model: LLM Merging We evaluate model merging on three fine-tuned Llama-2-7B models (Touvron et al., 2023) focusing on different capabilities: Chinese language proficiency[1], mathematical reasoning (Yu et al., 2023)[2], and code generation (Rozière et al., 2023)[3]. Each model's performance was assessed using domain-specific benchmarks: CMMLU (Li et al., 2023) for Chinese language understanding, GSM8K (Cobbe et al., 2021) for mathematical reasoning, and HumanEval (Chen et al., 2021) for code generation abilities.

[1] https://huggingface.co/LinkSoul/Chinese-Llama-2-7b

[2] https://huggingface.co/meta-math/MetaMath-7B-V1.0

[3] https://huggingface.co/qualis2006/llama-2-7b-int4-python-code-18k

Method	Average	Test Set Performance						
		paws	qasc	quartz	story_cloze	wiki_qa	winogrande	wsc
Pre-trained	53.5	49.9	35.8	53.3	48.1	76.2	50.0	61.1
Fine-tune	81.0	91.4	95.5	79.1	79.6	95.2	62.8	63.6
Parameter Average	60.2	55.2	57.5	55.2	49.4	91.1	50.2	62.5
Fisher Merging	68.2	67.9	84.4	63.5	57.1	90.1	54.2	60.8
RegMean	71.5	76.2	92.8	62.6	63.6	89.4	57.4	58.3
Task Arithmetic	71.6	71.1	81.4	62.1	77.1	95.0	57.4	56.9
Ties-Merging	72.2	78.8	88.4	65.1	70.7	84.2	56.2	62.3
PCB-Merging	72.3	71.5	91.7	66.8	62.7	92.8	57.1	63.3
MetaGPT	58.5	56.7	67.3	58.1	45.4	80.2	55.5	46.2
Consensus TA	72.3	73.8	83.4	61.9	76.0	83.8	56.2	61.0
Localize-and-Stitch	72.9	75.7	88.6	62.3	73.6	93.9	55.9	60.6
STF (ours)	**73.7**	77.4	89.1	62.6	75.2	94.2	56.4	61.1

Table 1: Merge fully-fine-tuned T5-base model with different methods.

Method	Test Set Performance											Rank
	E2E					DART			WebNLG			
	BLEU↑	NIST↑	MET↑	ROUGE-L↑	CIDEr↑	BLEU↑	MET↑	TER↓	BLEU-A↑	MET-A↑	TER-A↓	
Pre-trained	0.2	0.58	1.5	5.2	0.002	0.2	2.0	152.4	0.15	2.0	179.1	–
LoRA	67.7	8.64	46.0	68.3	2.36	44.8	35.0	50.4	52.3	37.0	44.4	–
Parameter Average	63.4	8.00	40.8	66.6	2.01	40.0	32.0	53.7	43.9	32.0	49.2	7
Task Arithmetic	63.2	8.02	40.9	66.3	1.98	40.8	33.0	53.7	45.9	34.0	48.8	6
Ties-Merging	62.8	8.14	41.1	65.9	2.08	41.4	33.0	53.7	46.1	33.0	48.1	5
PCB-Merging	62.9	8.12	41.4	66.0	2.08	41.3	33.0	53.5	46.2	33.0	48.1	3
Knots	62.9	8.22	41.6	65.8	2.05	41.52	34	54.0	46.0	34	47.5	2
MetaGPT	63.4	7.88	40.7	66.1	2.01	42.23	33	52.1	45.9	33	47.4	4
STF (ours)	64.1	8.40	42.2	66.5	2.18	41.6	33.0	54.1	47.1	34.0	48.0	1

Table 2: Merge LoRA Adapters of GPT-2 M with Different Methods. ↑ indicates higher is better, ↓ indicates lower is better.

Method	Datasets			Average
	CMMLU	GSM8K	Humaneval	
Chinese	38.6	2.3	13.4	18.1
Math	31.2	65.6	0	32.3
Code	33.3	0	17.1	16.8
Parameter Average	35.6	48.5	6.7	30.3
Task Arithmetic	35.4	46.1	9.8	30.4
Ties-Merging	36.5	53.4	12.8	34.3
MetaGPT	36.2	50.6	16.9	34.6
PCB-Merging	36.4	52.3	16.5	35.1
STF (ours)	36.5	63.0	14.0	**37.8**

Table 3: Results on the CMMLU, GSM8K, and Humaneval datasets.

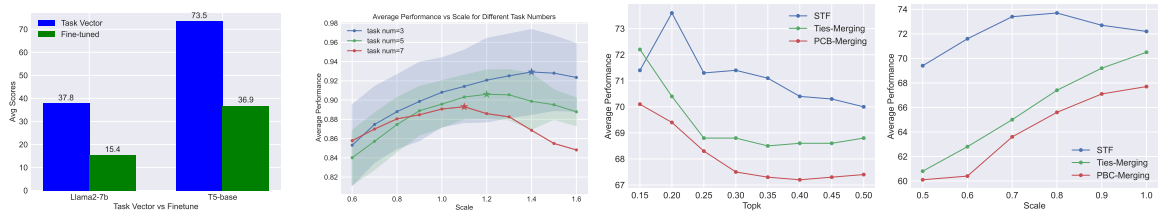
As shown in Table 3, STF achieves state-of-the-art performance across all three domains, improving overall performance by 2.7% compared to the best baseline. The most significant improvement is observed in mathematical reasoning, where STF outperforms other methods by approximately 10% on the GSM8K benchmark.

Out-of-Distribution Generalization To evaluate out-of-distribution generalization, we test the merged T5-base model (previously trained on seven tasks) on six held-out tasks from the T0 mixture (Sanh et al., 2021): three question answering tasks (Cosmos QA (Huang et al., 2019), Social IQA (Sap et al., 2019), QuAIL (Rogers et al., 2020)), word sense disambiguation (WiC (Pilehvar and Camacho-Collados, 2018)), and two sentence completion tasks (COPA (Roemmele et al., 2011), H-SWAG (Zellers et al., 2019)). As shown in Table 5, STF achieves 0.9% improvement over the best baseline on T5-base, demonstrating strong generalization capabilities to tasks outside the training distribution.

4.3 Additional Results and Analysis

Task Matrices versus Fine-tuned Matrices We compare merging task matrices versus fine-tuned matrices directly on T5-base models. As shown in Fig. 3(a), merging task matrices consistently outperforms merging fine-tuned matrices across all architectures by a large margin. This validates our discussion in Section 3.5 that task matrices better isolate task-specific features compared to fine-tuned matrices, making them more effective for model merging.

Feature Interference and Scaling Factor We examine how performance of T5 merging varies with different number of tasks and scaling factor in Fig. 3(b). We observe that as more tasks are



(a) Merge task matrices vs fine-tuned matrices (b) Performance vs number of tasks and scaling factor (c) Various η at $\gamma = 0.6$ (d) Various γ at $\eta = 0.2$

Figure 3: Experimental results on analyzing STF.

merged, the optimal scaling factor γ that achieves the best performance decreases. This trend indicates that larger γ values amplify interference between task-specific features when merging many tasks, requiring smaller scaling factors to maintain performance. The results validate our discussion in Section 3.5 about scaling merged matrix to balance feature preservation against interference.

Hyperparameter Sensitivity We analyze the sensitivity of STF to the hyperparameters η and γ on the T5-base. Three model merging methods are compared in the experiments: STF, Ties-Merging, and PCB-Merging. We follow the experiment setup of previous work (Du et al., 2024) to analyze the effect of two parameters. The first parameter is the scaling factor γ , which is used to scale the merged task vector. For STF and Ties-Merging, the second hyperparameter is the top-k percentage η, indicating that the $\eta\%$ parameters in task vector with the highest absolute value are retained. For PCB-Merging, the second hyperparameter is mask ratio η, indicating that the $\eta\%$ parameters in parameter competition balancing matrix with the biggest absolute value will be retained. We vary the hyperparameters γ with step size 0.1 from 0.5 to 1.0, and η with step size 0.05 from 0.15 to 0.5.

First, in Figs. 3(c) and 3(d), STF achieves optimal performance with $\gamma = 0.8$ and $\eta = 0.2$ on T5-base. Second, performance varies with both hyperparameters. For sparsity ratio η, performance first improves then declines as it increases. When η is too low, important features are filtered out as most parameters are set to zero. When η is too high, noise in the task matrices affects the quality of singular vectors. For scaling factor γ, we observe a similar trend - performance initially improves then deteriorates with increasing values. A small γ diminishes the magnitude of singular vectors, while a large γ may amplify interference between tasks.

Model Merging Time We analyze the time required for model merging with STF on different models. As shown in Table 4, the merging time increases with model size, primarily due to the SVD computation on larger matrices. For smaller models, TIES and PCB are more efficient than Knots and STF. However, with larger models, STF becomes more efficient because TIES and PCB require storing all parameters in memory simultaneously and run only on CPU, while STF processes one task matrix at a time and leverages GPU acceleration. The relatively short merging times demonstrate that STF is practical for real-world applications, despite its theoretical time complexity.

Method	T5-base	GPT-2 LoRA	Llama-2-7B
TIES	34	0.08	1280
PCB	113	0.09	1593
Knots	–	8	–
STF (Ours)	127	5	623

Table 4: Merging time for different models (unit: s). We run PCB and TIES on CPU because of the large memory requirement.

5 Conclusion

In this paper, we present STF, a novel model merging approach that preserves task-specific features in linear transformations through feature superposition. Extensive experiments demonstrate that STF consistently outperforms existing methods across different architectures and tasks. The success of STF demonstrates the value of leveraging working mechanisms of deep neural networks in model merging, rather than treating them at the parameter level.

Limitations

While STF effectively preserves features during merging, it does not explicitly identify task-specific features corresponding to semantic meanings. Future work could leverage mechanistic interpretability techniques like superposition analysis and sparse autoencoders to better isolate and preserve task-specific features in linear transformations. Additionally, STF currently relies on a simple scaling

approach to manage interference between tasks. More sophisticated methods for analyzing feature importance and selectively removing interference from less critical features could further improve performance. These advances would help develop a more principled approach to preserving task capabilities during model merging. Another major limitation is the time complexity of SVD and solving the linear system in STF, which can be improved by using more efficient method for feature extraction in the future.

Acknowledgements

This work is supported by National Key Research and Development Program of China (Grant No.2023YFB2903904), National Natural Science Foundation of China (Grant No. 92270106), Beijing Natural Science Foundation (Grant No. 4242039), and CCF-Huawei Populus Grove Fund.

References

Guillaume Alain and Yoshua Bengio. 2018. Understanding intermediate layers using linear classifier probes. 2018. *arXiv preprint arXiv:1610.01644*.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, and 1 others. 2022. Promptsource: An integrated development environment and repository for natural language prompts. *arXiv preprint arXiv:2202.01279*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. 2017. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)*.

Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. 2022. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. 2014. Describing textures in the wild. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

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

Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim Kuan Goh, Ho-Kin Tang, Daojing He, and Min Zhang. 2024. Parameter competition balancing for model merging. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, and 1 others. 2022. Toy models of superposition. *arXiv preprint arXiv:2209.10652*.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from rdf data. In *10th International Conference on Natural Language Generation*, pages 124–133. ACL Anthology.

Yifei He, Yuzheng Hu, Yong Lin, Tong Zhang, and Han Zhao. 2024. Localize-and-stitch: Efficient model merging via sparse task arithmetic. *arXiv preprint arXiv:2408.13656*.

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

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics*, pages 9318–9333.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2391–2401.

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

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

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3d object representations for fine-grained categorization. In *International Conference on Computer Vision Workshops*.

Yann LeCun. 1998. The mnist database of handwritten digits.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. Cmmlu: Measuring massive multitask language understanding in chinese. *arXiv preprint arXiv:2306.09212*.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, and 8 others. 2025. On the biology of a large language model. *Transformer Circuits Thread*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, and 1 others. 2020. Dart: Open-domain structured data record to text generation. *arXiv preprint arXiv:2007.02871*.

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

nostalgebraist. 2020. interpreting gpt: the logit lens. *LessWrong*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001.

Pramuditha Perera, Matthew Trager, Luca Zancato, Alessandro Achille, and Stefano Soatto. 2023. Prompt algebra for task composition. *arXiv preprint arXiv:2306.00310*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. *2011 AAAI Spring Symposium Series*.

Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. 2020. Getting closer to AI complete question answering: A set of prerequisite real tasks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8722–8731. AAAI Press.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, and 7 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, and 1 others. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473.

Rishi Sharma, James Allen, Omid Bakhshandeh, and Nasrin Mostafazadeh. 2018. Tackling the story ending biases in the story cloze test. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 752–757.

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

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. 2024. Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019. Quartz: An open-domain dataset of qualitative relationship questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5941–5946.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, and 3 others. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Matthew Trager, Pramuditha Perera, Luca Zancato, Alessandro Achille, Parminder Bhatia, and Stefano Soatto. 2023. Linear spaces of meanings: compositional structures in vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15395–15404.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. 2024. Localizing task information for improved model merging and compression. *arXiv preprint arXiv:2405.07813*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and 1 others. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

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

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2023. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

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 *Forty-first International Conference on Machine Learning*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

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

Jinghan Zhang, Junteng Liu, Junxian He, and 1 others. 2023. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36:12589–12610.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase Adversaries from Word Scrambling. In *Proc. of NAACL*.

Yuyan Zhou, Liang Song, Bingning Wang, and Weipeng Chen. 2024. Metagpt: Merging large language models using model exclusive task arithmetic. *arXiv preprint arXiv:2406.11385*.

A Measure Feature Preservation

We measure feature preservation by $\left|\langle \sigma_i^{(k)}\mathbf{u}_i^{(k)}, \bar{\mathbf{M}}\mathbf{v}_i^{(k)} - \sigma_i^{(k)}\mathbf{u}_i^{(k)}\rangle\right|$, where $\sigma_i^{(k)}\mathbf{u}_i^{(k)}$ and $\bar{\mathbf{M}}\mathbf{v}_i^{(k)}$ and $\mathbf{v}_i^{(k)}$ are the task-specific features identified in Section 3.2, and $\bar{\mathbf{M}}$ is the merged task matrix for various merging methods without scaling. This measurement is based on the superposition objective in (1). Higher values indicate that the output of merged task matrix $\bar{\mathbf{M}}$ is less similar to the output feature vectors of original task matrix $\mathbf{M}_i$, and thus the feature preservation is worse.

We calculate the preservation for every feature vector in the original fine-tuned models across all datasets, and then average all these individual preservation values. In Fig. 1(a), we choose the input linear transformation matrices of all FFN layers in T5-base model. We also measured the feature preservation for other layers, including the attention layers. Fig. 4 shows the feature preservation for the output layers of FFN in T5-base model, which shows the same trend with Fig. 1(a). We observe that STF exactly preserves the magnitudes and directions of output feature vectors, while Ties-Merging and PCB-Merging have much lower feature preservation. The results indicate that better feature preservation strongly correlates with better performance and validate our hypothesis that feature superposition is effective for model merging.

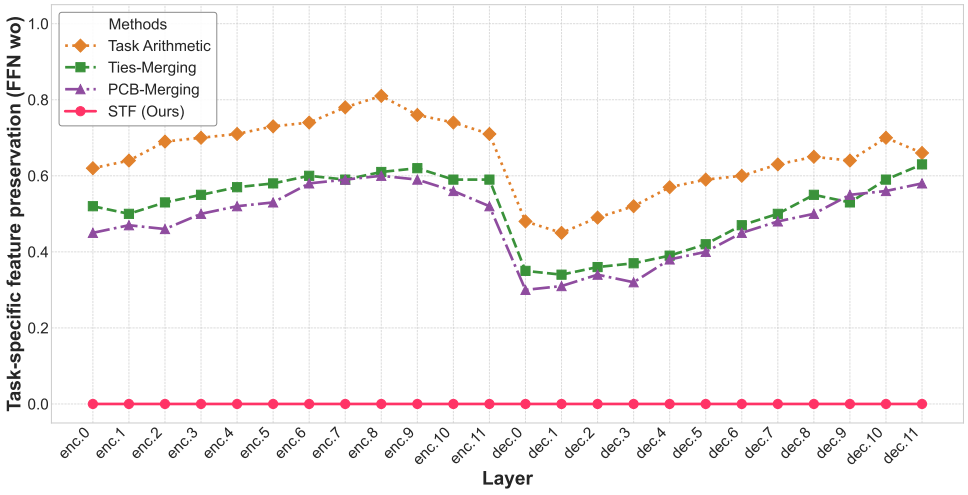


Figure 4: Feature preservation for FFN output layer in T5-base model

B Singular Value Decomposition of Task Matrices

Applying SVD to task matrix, we have $\mathbf{M}_i = \sum_{k=1}^{r_i} \sigma_i^{(k)}\mathbf{u}_i^{(k)}\mathbf{v}_i^{(k)\top}$, where $\sigma_i^{(k)}$ is the k-th singular value, and $\mathbf{u}_i^{(k)}$ and $\mathbf{v}_i^{(k)}$ are the k-th left and right singular vectors of $\mathbf{M}_i$, respectively. These singular vectors serve as orthogonal basis for output $\mathbf{M}_i\mathbf{x}$ and input $\mathbf{x}$. Specifically, any input representation $\mathbf{x} \in \mathbb{R}^n$ can be decomposed as $\mathbf{x} = \sum_{k=1}^{r_i} w_{i,j}^{(k)}\mathbf{v}_i^{(k)} + \mathbf{b}$, where $\mathbf{b}$ is orthogonal to all right singular vectors. When $\mathbf{M}_i$ transforms $\mathbf{x}$, we get $\mathbf{M}_i\mathbf{x} = \sum_{k=1}^{r_i} \sigma_i^{(k)} w_{i,j}^{(k)}\mathbf{u}_i^{(k)}$ since $\mathbf{b}$ lies in the null space of $\mathbf{M}_i$. This shows that the input $\mathbf{x}$ and output $\mathbf{M}_i\mathbf{x}$ can be represented as a weighted sum of right and left singular vectors, respectively.

C Proof of Theorem 1

Here we use $\mathbf{X} = [\mathbf{x}_i^{(k)\top}\mathbf{x}_{i'}^{(k')}]_{i,k;i',k'}$ to denote the matrix formed by enumerating over i, k for the row index and i', k' for the column index.

Proof. The objective of feature superposition can be converted to

$$\left\langle \mathbf{u}_i^{(k)}, \mathbf{M}\mathbf{v}_i^{(k)} \right\rangle = \left\langle \mathbf{u}_i^{(k)}, \mathbf{M}_i\mathbf{v}_i^{(k)} \right\rangle, \quad \forall i, k.$$

We bring $\mathbf{M} = \sum_{i'=1}^{T}\sum_{k'=1}^{r_i} \alpha_{i'}^{(k')}\mathbf{u}_{i'}^{(k')}\mathbf{v}_{i'}^{(k')\top}$ and $\mathbf{M}_i = \sum_{k'=1}^{r_i} \alpha_i^{(k')}\mathbf{u}_i^{(k')}\mathbf{v}_i^{(k')\top}$ to the left side of the equation, and move $\sigma_i^{(k)}$ to the right side.

Then, the objective becomes

$$\mathbf{u}_i^{(k)\top}\left(\sum_{i'=1}^{T}\sum_{k'=1}^{r_i}\alpha_{i'}^{(k')}\mathbf{u}_{i'}^{(k')}\mathbf{v}_{i'}^{(k')\top}\right)\mathbf{v}_i^{(k)}=\sigma_i^{(k)} \Leftrightarrow \sum_{i'=1}^{T}\sum_{k'=1}^{r_i}\mathbf{u}_i^{(k)\top}\mathbf{u}_{i'}^{(k')}\mathbf{v}_{i'}^{(k')\top}\mathbf{v}_i^{(k)}\alpha_{i'}^{(k')}=\sigma_i^{(k)} \tag{4}$$

which is one linear equation with $\mathbf{u}_i^{(k)\top}\mathbf{u}_{i'}^{(k')}\mathbf{v}_{i'}^{(k')\top}\mathbf{v}_i^{(k)}$ as the coefficient of variable $\alpha_{i'}^{(k')}$. We can convert all equations in (1) to a linear system for all tasks in $\{\mathsf{T}_1,\ldots,\mathsf{T}_T\}$:

$$\begin{bmatrix} \mathbf{u}_1^{(1)\top}\mathbf{u}_1^{(1)}\mathbf{v}_1^{(1)\top}\mathbf{v}_1^{(1)} & \cdots & \mathbf{u}_1^{(1)\top}\mathbf{u}_T^{(r_T)}\mathbf{v}_T^{(r_T)\top}\mathbf{v}_1^{(1)} \\ \vdots & \ddots & \vdots \\ \mathbf{u}_T^{(r_T)\top}\mathbf{u}_1^{(1)}\mathbf{v}_1^{(1)\top}\mathbf{v}_T^{(r_T)} & \cdots & \mathbf{u}_T^{(r_T)\top}\mathbf{u}_T^{(r_T)}\mathbf{v}_T^{(r_T)\top}\mathbf{v}_T^{(r_T)} \end{bmatrix}\begin{bmatrix}\alpha_1^{(1)}\\ \vdots \\ \alpha_T^{(r_T)}\end{bmatrix}=\begin{bmatrix}\sigma_1^{(1)}\\ \vdots \\ \sigma_T^{(r_T)}\end{bmatrix}.$$

With Kronecker product, we can write the linear system in a more compact form:

$$\mathbf{\Sigma}\circ\mathbf{U}\circ\mathbf{V}\boldsymbol{\alpha}=\boldsymbol{\sigma},$$

where $\mathbf{U}=[\mathbf{u}_i^{(k)\top}\mathbf{u}_{i'}^{(k')}]_{i,k;i',k'}=\bar{\mathbf{U}}^\top\bar{\mathbf{U}}\in\mathbb{R}^{r\times r}$; $\mathbf{V}=[\mathbf{v}_{i'}^{(k')\top}\mathbf{v}_i^{(k)}]_{i',k';i,k}=\bar{\mathbf{V}}^\top\bar{\mathbf{V}}\in\mathbb{R}^{r\times r}$; $\boldsymbol{\alpha}$ is the vector of variables $\alpha_{i'}^{(k')}$; and $\boldsymbol{\sigma}$ is the vector of singular values $\sigma_i^{(k)}$. The index i,k,i',k' share consistent order across $\mathbf{U}$ and $\mathbf{V}$. □

D More Experimental Results

Table 5: Out-of-Distribution Generalization Performance of merged T5-base model

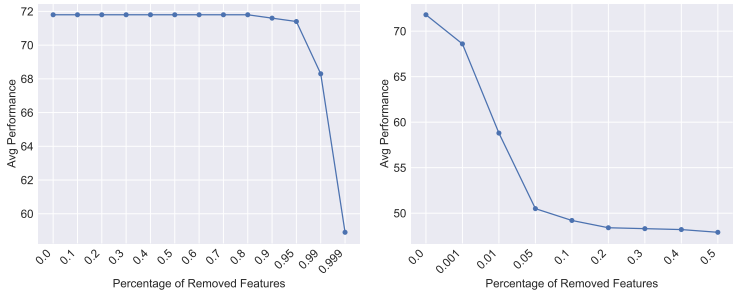
Method	Average	Out-of-Distribution Performance					
		cosmos_qa	social_iqa	quail	wic	copa	h-swag
Pre-trained	31.1	21.9	18.8	24.1	65.6	43.8	12.5
Average	36.3	23.5	37.0	25.4	50.2	54.5	27.2
Task Arithmetic	37.0	21.9	36.8	25.5	49.5	61.4	26.6
Ties-Merging	37.1	22.2	37.8	24.9	51.6	62.2	26.5
PCB-Merging	<u>37.3</u>	23.0	38.1	24.6	52.2	59.0	27.2
STF (ours)	**38.2**	22.4	38.0	26.0	51.6	64.1	27.0

D.1 More Analysis

Removing Singular Vectors We investigate removing singular vectors from task matrices before merging, analyzing on T5-base: removing smallest versus largest singular vectors. As shown in Figs. 5(a) and 5(b), removing up to 80% of smallest singular vectors has minimal impact on performance, while removing just 5% of largest singular vectors causes significant degradation. This indicates that while most singular features are redundant, preserving those with largest singular values is crucial for maintaining model capabilities. This insight could guide future work on selective feature preservation during model merging.

Fully Fine-tuned Vision Model: ViT Merging We evaluate STF on vision tasks using two CLIP models (Radford et al., 2021) with ViT-B/32 and ViT-L/14 architectures (Dosovitskiy, 2020) as visual encoders. We use the released checkpoints from Ilharco et al. (2022) that were fine-tuned on eight diverse classification tasks spanning remote sensing, traffic signs, and satellite imagery domains. These tasks include: Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2011), MNIST (LeCun, 1998), RESISC45 (Cheng et al., 2017), SUN397 (Xiao et al., 2016), and SVHN (Netzer et al., 2011). For all experiments, we keep the text encoder fixed and only merge the parameters of visual encoders.

As shown in Table 6, STF achieves state-of-the-art performance when merging fully fine-tuned ViT models, outperforming existing methods by 1.1% and 0.3% on average across 8 tasks for ViT-B/32



(a) Remove smallest singular vectors (b) Remove biggest singular vectors

Figure 5: Performance of STF with different percentage of removed features.

and ViT-L/14 architectures respectively. For ViT-B/32, STF shows particularly strong performance on RESISC45 and EuroSAT, with 2.9% and 11.1% improvements over the best baseline. For ViT-L/14, STF maintains high performance across all tasks, demonstrating effective preservation of task-specific features during merging.

Table 6: Test set performance when merging ViT-B/32 and ViT-L/14 models on 8 vision tasks.

Model	Task(→) Method(↓)	Average	Test Set Performance							
			SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD
ViT-B/32	Pre-trained	46.3	61.7	54.7	58.5	51.2	29.1	27.4	45.6	42.1
	Fine-tuned	90.5	75.3	77.7	96.1	99.7	97.5	98.7	99.7	79.4
	Averaging	65.8	65.3	63.4	71.4	71.7	64.2	52.8	87.5	50.1
	Fisher Merging	68.3	**68.6**	**69.2**	70.7	66.4	72.9	51.1	87.9	59.9
	RegMean	71.8	65.3	63.5	75.6	78.6	78.1	67.4	93.7	52.0
	Task Arithmetic	70.1	63.8	62.1	72.0	77.6	74.4	65.1	94.0	52.2
	Ties-Merging	73.6	64.8	62.9	74.3	78.9	83.1	71.4	97.6	56.2
	PCB-Merging	76.3	66.7	65.5	78.5	79.3	**86.4**	**77.1**	**98.2**	59.1
	STF (ours)	**77.4**	65.4	62.0	**81.4**	**90.4**	84.1	**77.1**	95.6	**62.9**
ViT-L/14	Pre-trained	64.1	68.2	76.4	69.7	64.7	60.4	49.4	67.9	56.3
	Fine-tuned	94.2	82.3	92.4	97.4	100	98.1	99.2	99.7	84.1
	Averaging	79.6	72.1	81.6	82.6	91.9	78.2	70.7	97.1	62.8
	Fisher Merging	82.2	69.2	**88.6**	87.5	93.5	80.6	74.8	93.3	70.0
	RegMean	83.7	73.3	81.8	86.1	**97.0**	88.0	84.2	98.5	60.8
	Task Arithmetic	84.5	74.1	82.1	86.7	93.8	87.9	86.8	98.9	65.6
	Ties-Merging	86.0	76.5	85.0	89.4	95.9	90.3	83.3	99.0	68.8
	PCB-Merging	87.5	**76.8**	86.2	89.4	96.5	88.3	91.0	98.6	**73.6**
	STF (ours)	**87.8**	75.4	85.8	**90.3**	96.6	**91.7**	**91.2**	**99.2**	72.4

D.2 Hyperparameter Settings

Fully Fine-tuned NLP Model: T5 Merging STF is evaluated with a sparsity ratio $\eta = 20\%$ and a scaling factor $\gamma = 0.8$. To reduce variance and ensure reliable evaluation, we report the experimental results of the average performance over different templates for each task, i.e., paws has 11 templates, qasc has 5 templates, quartz has 8 templates, story_cloze has 5 templates, wiki_qa has 5 templates, winogrande has 5 templates, and wsc has 10 templates.

PEFT of NLP Model: LORA Adapters Merging STF is evaluated with a sparsity ratio $\eta = 30\%$ and a scaling factor $\gamma = 0.5$.

Large Model: LLM Merging We follow the setup from Du et al. (2024) to merge these LLMs. For STF, we does not apply trimming step and set the scaling factor $\gamma = 0.8$.

Fully Fine-tuned Vision Model: ViT Merging With sparse ratio $\eta = 20\%$, we grid search over the scaling factor γ in $[0.5, 1.2]$ with step size 0.1 and find the best performance with $\gamma = 0.6$ for ViT-B/32 and $\gamma = 0.7$ for ViT-L/14.