Stay Unique, Stay Efficient: Preserving Model Personality in Multi-Task Merging

Kuangpu Guo[1,2], Aijing Yu[4], Jian Liang[2,3⋆], Yuhe Ding[5], Zilei Wang[1], Ran He[2,3], and Tieniu Tan[2,3,6]

[1] University of Science and Technology of China, China
[2] NLPR & MAIS, Institute of Automation, Chinese Academy of Sciences, China
[3] School of Artificial Intelligence, University of Chinese Academy of Sciences, China
[4] Institute of Information Engineering, Chinese Academy of Sciences, China
[5] Anhui University, China
[6] Nanjing University, China
gkp@mail.ustc.edu.cn, liangjian92@gmail.com

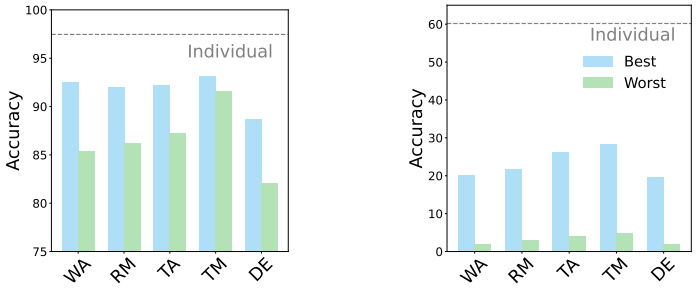
Abstract. Model merging has emerged as a promising paradigm for enabling multi-task capabilities without additional training. However, traditional basic merging methods often experience performance degradation due to parameter conflicts, even when applied to similar tasks. While recent personalized merging frameworks successfully preserve task-specific information to maintain performance, they typically incur storage overhead. In this paper, we propose **D**ecomposition, **T**hresholding, and **S**caling (**DTS**), an approximation-based personalized merging framework that pushes task-specific storage efficiency. DTS first applies singular value decomposition to the task-specific information and retains only a small subset of singular values and vectors. It then introduces a novel thresholding strategy that partitions singular vector elements into groups and assigns a scaling factor to each group. To enable generalization to unseen tasks, we further extend DTS with a variant that fuses task-specific information in a data-free manner based on the semantic similarity of task characteristics. Extensive experiments demonstrate that DTS consistently outperforms state-of-the-art baselines while requiring only 1% extra storage per task. Furthermore, experiments on unseen tasks show that the DTS variant achieves significantly better generalization performance. Our code is available at `https://github.com/krumpguo/DTS`.

Keywords: Model Merging · Multi-task Learning · Efficiency · Personalization · Generalization

1 Introduction

Adapting pre-trained models to diverse tasks via fine-tuning [30, 56, 64, 65] often leads to prohibitive deployment costs. Unlike data-intensive multi-task learning [3, 68], model merging [21, 29, 42, 69, 72] offers a data-free and optimization-free

⋆ Corresponding author.



(a) SVHN **(b)** CoLA

Fig. 1: Performance of the pairwise merged model: we pairwise merge models fine-tuned on SVHN [38] or CoLA [54] with those fine-tuned on the other seven datasets, reporting the best and worst performance across these combinations.WA, RM, TA, TM, and DE refer to Weight-Averaging [57], RegMean [23], Task-Arithmetic [22], Ties-Merging [60], and DARE [66], respectively.

alternative by directly fusing parameters from multiple task-specific models into a unified representation.

Traditional basic model merging methods [22] strive to compress all capabilities into a single, unified parameter body. However, these methods inherently suffer from severe parameter conflicts [41, 60]. To investigate the depth of this issue, we examine the performance degradation from the perspective of task similarity. Specifically, we conduct pairwise model merging by combining the model fine-tuned on the SVHN dataset with models fine-tuned on each of the other seven datasets in the visual benchmark. Fig. 1 (a) shows the best and worst performance of the merged models when evaluated on SVHN, corresponding to merging with the most similar task (MNIST) and the most dissimilar task (EuroSAT), respectively. Surprisingly, we observe significant performance drops even when merging models trained on the most similar tasks. These results highlight the persistent nature of cross-task parameter conflicts and underscore the fundamental limitation of enforcing a single shared parameter set across all tasks.

To address this issue, a second paradigm, personalized model merging (e.g., WEMOE [50], EMR-Merging [20]), has recently gained traction. By recognizing the necessity of preserving task-specific information, these approaches retain a subset of task-specific parameters or employ dynamic routing modules conditioned on task IDs. While they successfully mitigate interference and achieve performance strictly bounded by individual fine-tuning, existing personalized merging frameworks often incur prohibitive storage overhead. Storing dense expert components or complex routing networks negates the primary advantage of parameter efficiency in model merging, severely limiting their deployment in resource-constrained environments.

To bridge this gap, we propose **DTS**, an approximation-based personalized model merging method that leverages **D**ecomposition, **T**hresholding, and **S**caling to efficiently preserve task-specific information while minimizing storage overhead. DTS applies singular value decomposition (SVD) [27] to each parameter matrix

in the task-specific information, retaining only a small subset of the filtered singular values and vectors. This selective retention significantly reduces storage requirements while maintaining essential task information. To further reduce storage costs, we introduce a novel thresholding strategy to approximate the singular vectors. Unlike prior works [25, 41], which either retain a subset of elements using a binary mask or directly quantize the parameters, we argue that such coarse processing may discard important fine-grained information. Instead, we propose a novel thresholding strategy that thresholds the elements of each singular vector into four groups and assigns a scaling factor for each group, as illustrated in Fig. 3. This strategy improves the approximation for the original task-specific information. In addition to applying DTS to task vectors (DTS-T), we introduce a new form of task-specific information termed the difference vector, which captures the parameter difference between individually fine-tuned and merged models. We apply DTS to this difference vector, referring to it as DTS-D.

To enable generalization to unseen tasks, we propose a variant of DTS that adaptively integrates task-specific information from seen tasks. Unlike prior methods that require additional data or trainable routers to determine merging weights, our approach computes weights based on the average semantic similarity between seen and unseen task characteristics. This design is entirely data-free, offering both efficiency and scalability. Experimental results demonstrate that DTS consistently outperforms state-of-the-art baselines on standard multi-task model merging benchmarks, achieving strong performance with only 1% extra storage per task. Furthermore, on unseen tasks, the DTS variant delivers superior generalization performance compared to all competing methods. Our contributions can be summarized as follows:

- We propose **DTS**, a lightweight approximation-based personalized merging method that addresses performance degradation from parameter conflicts by preserving task-specific information.
- We introduce DTS-T and DTS-D variants, which apply DTS on task vectors and difference vectors, respectively. Additionally, we present a variant of DTS that fuses task-specific information based on the semantic similarity of task characteristics, enabling effective generalization to unseen tasks.
- DTS consistently outperforms state-of-the-art baselines on standard benchmarks with only $\approx$1% extra storage per task. Furthermore, experiments on unseen tasks show that the DTS variant achieves superior generalization performance compared to all baseline methods, highlighting its effectiveness in both seen and unseen task settings.

2 Related Work

Model merging [1, 5, 11, 22, 51, 67], which aims to fuse multiple fine-tuned models into a single comprehensive model, has attracted increasing attention with the release of numerous publicly available model checkpoints [12, 24, 56]. Model merging significantly reduces storage and deployment cost by unifying multiple models into a single one, without additional training [17, 32, 49, 66]. Depending

on whether the merged model is static or tailored per task, existing methods can be categorized into basic and personalized model merging methods.

2.1 Basic Model Merging

Basic model merging [9, 15, 22, 23, 35, 36, 57, 60, 66] focuses on universal strategies for merging fine-tuned models into a single unified model. A canonical example is Weight-Averaging [57], which simply averages parameters from different tasks. Task-Arithmetic [22] introduces task vectors that yield a better merged model. Building on this idea, DARE [66] and Ties-Merging [60] propose pruning and scaling task vectors, assuming that not all parameters contribute equally. AdaMerging [63] introduces adaptive learning to automate coefficient selection but incurs additional training cost. Other approaches like Fisher-Merging [23] and RegMean [36] compute merging coefficients via Fisher information or inner product matrices. However, basic model merging often results in parameter conflicts and a loss of task-specific information, leading to a substantial performance gap compared to individual models [28, 48, 62, 71]. Therefore, we explore the personalized model merging in this work.

2.2 Personalized Model Merging

Personalized model merging [20, 34, 37, 50, 53, 74] enhances the merged model with task-specific components to boost performance across a variety of tasks. SMEAR [37] and Twin-Merging [34] store full model parameters for each task and employ routing mechanisms to perform weighted parameter fusion based on expert distributions. WEMOE [50] introduces test-time adaptation by merging most weights while converting MLP layers into a mixture-of-experts (MoE) module. DaWin [39] similarly retains full models and uses entropy over unlabeled test samples to assess task relevance. While effective, these methods typically require access to training data or incur high storage costs. In contrast, EMR-Merging [20] selects a unified base model and generates lightweight, data-free modulators for each task. T-Switch [41] further reduces memory by storing task vectors in a binarized form. FREE-Merging [70] proposes a lightweight task-specific expert module that dynamically compensates for information loss during merging. Nonetheless, existing personalized methods often suffer from either data dependency or excessive storage demands. In this work, we aim to retain essential task-specific information with minimal memory overhead, mitigating parameter conflicts without relying on additional data or training.

3 Method

3.1 Problem Formulation

In this paper, following the setup of prior model merging works [22, 41, 60, 66], we consider a scenario involving N tasks with corresponding datasets $\{\mathcal{D}_n\}_{n=1}^{N}$,

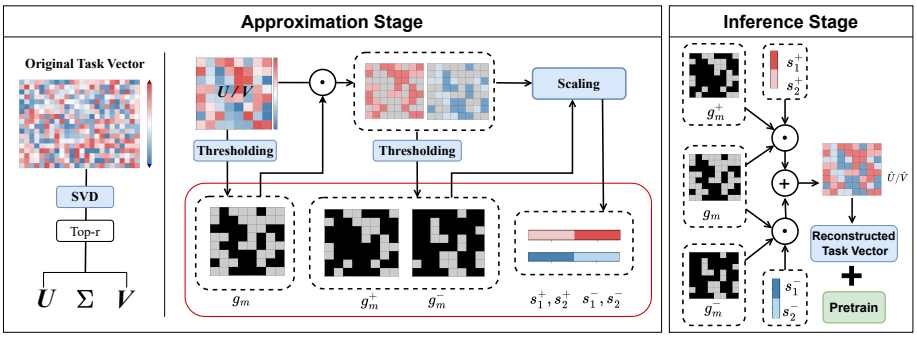


Fig. 2: Framework overview. In the approximation stage, we first apply singular value decomposition to the task-specific information. Next, a novel thresholding strategy is used to group the elements of each singular vector, followed by computing a scaling factor for each group. During inference, the task-specific information is reconstructed using the approximated singular vectors for each task.

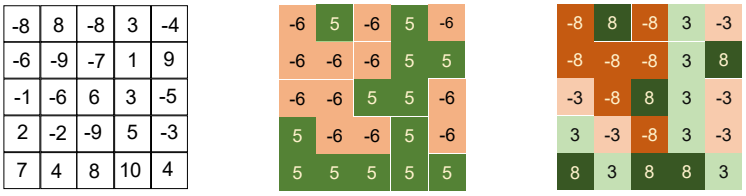
where each sample $(\boldsymbol{x}_n, y_n) \in \mathcal{D}_n$ belongs to the n-th task. Let $f_{\boldsymbol{\theta}_0}$ denote a pre-trained model with parameters $\boldsymbol{\theta}_0 \in \boldsymbol{\Theta}$, and let $f_{\boldsymbol{\theta}_1}, \ldots, f_{\boldsymbol{\theta}_N}$ represent task-specific models fine-tuned on each $\mathcal{D}_n$, where $\boldsymbol{\theta}_n$ represents the fine-tuned weights for the n-th task. Following Task-Arithmetic [22], the task vector $\boldsymbol{\tau}_n$ for the n-th task is defined as $\boldsymbol{\tau}_n = \boldsymbol{\theta}_n - \boldsymbol{\theta}_0$, which serves as a widely adopted representation for capturing task-specific parameter information.

While the standard task vector $\boldsymbol{\tau}_n$ effectively isolates task-specific parameter shifts, it strictly relies on the availability of the original pre-trained weights $\boldsymbol{\theta}_0$. In many practical open-source scenarios, practitioners might only have access to a collection of fine-tuned models or a coarsely merged baseline $\boldsymbol{\theta}_m$. To address this constraint, we introduce a new form of task-specific information termed the **difference vector**, defined as $\boldsymbol{d}_n = \boldsymbol{\theta}_n - \boldsymbol{\theta}_m$, where $\boldsymbol{\theta}_m$ is the parameter set obtained via a basic merging method such as Ties-Merging [60]. Beyond serving as a practical alternative when $\boldsymbol{\theta}_0$ is inaccessible, $\boldsymbol{d}_n$ captures the personalized residual that deviates from the shared common knowledge embedded in $\boldsymbol{\theta}_m$. By explicitly modeling these residuals relative to a task-agnostic baseline, this formulation provides a robust anchor for generalizing to unseen tasks.

As an illustrative example, we consider task vectors as task-specific information. The goal of model merging is to combine the set $\{\boldsymbol{\tau}_n\}_{n=1}^N$ with the pre-trained model to produce a merged model that performs well on the union of all task datasets, $\mathcal{D} = \bigcup_{n=1}^N \mathcal{D}_n$, formulated as:

$$\min \mathbb{E}_{(\boldsymbol{x},y)\in\mathcal{D}} \mathcal{L}\left(f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}), y\right), \ \boldsymbol{\theta}_m^* = \text{Merge}(\boldsymbol{\theta}_0, \{\boldsymbol{\tau}_n\}_{n=1}^N). \tag{1}$$

Here, Merge$(\cdot)$ denotes a general merging function. For instance, Task Arithmetic [22] takes the form $\text{Merge}(\boldsymbol{\theta}_0, \{\boldsymbol{\tau}_n\}_{n=1}^N) = \boldsymbol{\theta}_0 + \sum_{n=1}^N \gamma_n \boldsymbol{\tau}_n$. More advanced personalized strategies, such as generating task-specific modulators [20], can also be employed to merge models across diverse tasks. *Following prior stud-*



(a) Singular vector **(b)** Binarization **(c)** Thresholding

Fig. 3: A toy example of our thresholding strategy. Thresholding provides a more fine-grained approximation by further partitioning positive and negative values into large/small groups.

ies [20, 22, 34, 41, 53, 63, 66], we also assume that the information of the test task (e.g., task ID) is available during evaluation.

3.2 Decomposition, Thresholding, and Scaling

Previous studies [35, 58, 61, 73] have shown that the merged model often suffers from notable performance degradation compared to its fine-tuned counterparts, primarily due to parameter conflicts among the fine-tuned models. In this work, we examine this issue through the lens of task similarity. Specifically, we perform pairwise model merging by combining a model fine-tuned on SVHN [38] with models fine-tuned on each of the other seven datasets in the benchmark.

Fig. 1 (a) shows the best and worst SVHN performance among the merged models, corresponding to merging with the most similar task (MNIST [10]) and the most dissimilar one, respectively. Surprisingly, substantial performance drops are observed even when merging models from similar tasks. A similar pattern emerges in natural language processing tasks, as illustrated in Fig. 1 (b). These findings suggest that parameter conflicts are inherent, even across semantically related tasks, and underscore the importance of preserving personalized information to maintain model performance. This insight motivates our objective: to retain complete task-specific information while minimizing extra storage overhead.

In this paper, we propose preserving the task-specific information of individual models. However, storing full model parameters incurs substantial memory overhead, making this approach impractical in resource-constrained settings. To address this issue, we introduce **D**ecomposition, **T**hresholding, and **S**caling (**DTS**), a personalized method that approximates task-specific information while preserving its effectiveness, as illustrated in Fig. 2. To establish a unified compression pipeline for both variants (DTS-T and DTS-D), let $\boldsymbol{W}_n \in \{\boldsymbol{\tau}_n, \boldsymbol{d}_n\}$ denote the target task-specific deviation matrix that requires storage. For layers whose parameters have more than two dimensions (e.g., convolutional kernels), we first reshape the tensor into a 2D matrix before processing. Specifically, we first apply truncated singular value decomposition to the target matrix and retain only the top-r singular values, as follows:

$$\boldsymbol{U}_n, \boldsymbol{\Sigma}_n, \boldsymbol{V}_n = \mathrm{SVD}_r(\boldsymbol{W}_n), \tag{2}$$

where $\boldsymbol{U}_n$ and $\boldsymbol{V}_n$ denote the left and right singular vector matrices, and $\boldsymbol{\Sigma}_n$ contains the retained singular values. The sparsity coefficient r is the proportion of singular values preserved.

To further reduce storage, we threshold the decomposed components. Unlike previous work [41], which uses a simple binary mask to retain a subset of elements as shown in Fig. 3 (b), we argue that this coarse masking approach may result in the loss of fine-grained information. Instead, we propose a novel thresholding strategy that partitions the elements of each singular vector into four groups and computes a scaling factor for each group as shown in Fig. 3 (c). Specifically, taking the left singular vector matrix $\boldsymbol{U}_n$ as an example, we first mark the sign of each element as follows:

$$g(U_{n,j}) = \begin{cases} 1, & \text{if } U_{n,j} > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where $U_{n,j}$ is the j-th element of $\boldsymbol{U}_n$. This thresholding function encodes the sign of each parameter using only 1 bit, dividing the parameters into positive and negative groups. To preserve more fine-grained information, we further threshold the positive and negative values into two subgroups, respectively. Taking the positive values as an example:

$$g^{+}(U_{n,j}) = \begin{cases} 1, & \text{if } U_{n,j} > \lambda, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where λ is the median of the positive entries in $\boldsymbol{U}_n$. This extended thresholding partitions positive elements into "large" and "small" groups. The same strategy is applied to the negative values. During the inference stage, we reconstruct the positive portion of $\boldsymbol{U}_n$ as:

$$\hat{\boldsymbol{U}}_n^{+} = s_1^{+} \cdot g(\boldsymbol{U}_n) \odot g^{+}(\boldsymbol{U}_n) + s_2^{+} \cdot g(\boldsymbol{U}_n) \odot (1 - g^{+}(\boldsymbol{U}_n)), \tag{5}$$

where $\odot$ denotes element-wise multiplication. The scaling factors s_1^{+} and s_2^{+} align the magnitude of the approximated vectors with the original components.

The design of DTS is theoretically grounded in optimal low-rank approximation and highly efficient scalar quantization. According to the Eckart-Young-Mirsky theorem [14], the truncated SVD reconstruction provides the theoretically optimal low-rank approximation. The truncation error is strictly bounded by the discarded singular values, effectively filtering out unstructured parameter noise. Furthermore, our 4-group thresholding serves as a median-based piecewise constant approximation. Given these fixed median splits, we aim to achieve the conditional minimum Mean Squared Quantization Error (MSQE) within each subgroup $\mathcal{S}_k$. The optimal scalar s_k that minimizes this L_2 residual is the arithmetic mean of the absolute values of the elements within $\mathcal{S}_k$. For efficient engineering implementation using tensor operations, we compute this arithmetic mean using the L_1 norm:

$$s_1^{+} = \frac{\|\boldsymbol{U}_n \odot g(\boldsymbol{U}_n) \odot g^{+}(\boldsymbol{U}_n)\|_1}{\|g(\boldsymbol{U}_n) \odot g^{+}(\boldsymbol{U}_n)\|_1}, \; s_2^{+} = \frac{\|\boldsymbol{U}_n \odot g(\boldsymbol{U}_n) \odot (1 - g^{+}(\boldsymbol{U}_n))\|_1}{\|g(\boldsymbol{U}_n) \odot (1 - g^{+}(\boldsymbol{U}_n))\|_1}. \tag{6}$$

The numerator calculates the sum of absolute values within the subgroup, and the denominator calculates the exact cardinality. This formulation leverages the L_1 norm to compute the exact arithmetic mean, ensuring optimal local magnitude alignment under the median-based partition. In Fig. 3, the computed values of s_1^+ and s_2^+ are 8 and 3, respectively. We apply the identical procedure to the negative values and to obtain the approximated $\hat{\boldsymbol{V}}_n$. For one-dimensional parameters (e.g., biases), we skip SVD and directly apply thresholding and scaling.

Crucially, this 4-group configuration allocates precisely 1 bit for the sign and 1 bit for the magnitude level per element, equating to a 2-bit representation (a 16× compression ratio). Finally, during the inference phase, the task-specific parameters are dynamically reconstructed by adding the approximated deviation back to their corresponding reference base model:

$$\hat{\boldsymbol{\theta}}_n = \boldsymbol{\theta}_{\text{ref}} + \hat{\boldsymbol{U}}_n \boldsymbol{\Sigma}_n \hat{\boldsymbol{V}}_n^T, \tag{7}$$

where $\boldsymbol{\theta}_{\text{ref}} = \boldsymbol{\theta}_0$ for the DTS-T variant, and $\boldsymbol{\theta}_{\text{ref}} = \boldsymbol{\theta}_m$ for the DTS-D variant. As discussed, this process preserves essential task-specific information while mitigating parameter conflicts, requiring only $\approx 1\%$ extra storage per task.

3.3 Extending DTS for Unseen Tasks

In practical deployments, merged models frequently encounter unseen tasks. While the base merged model can serve as a generic initialization, it does not account for task similarities, which limits its generalization performance. The objective here is to adaptively transfer the preserved task-specific residuals (e.g., the approximated difference vectors $\hat{\boldsymbol{d}}_n$) to novel tasks without requiring additional training data.

To achieve this, we utilize the semantic meta-information of tasks as a training-free metric for parameter transferability. Unlike existing adaptation methods that rely on target-domain data to compute merging weights via gradient updates, our approach leverages semantic similarity to enable data-free generalization.

For clarity, we illustrate the approach using the DTS-D variant. Taking classification tasks as an example, where class names serve as task-specific characteristics, we encode them with a pretrained text encoder to generate class name embeddings. The task-level semantic representation is obtained by averaging the embeddings of all class names within a given task. For an unseen task, we compute the mean embedding E_u over its class names. Given the embeddings $\{E_n\}_{n=1}^N$ of the seen tasks, the dynamic merging process is formulated as:

$$\hat{\boldsymbol{\theta}}_u = \boldsymbol{\theta}_m + \sum_{n=1}^{N} \gamma_n \cdot \hat{\boldsymbol{d}}_n,\ \gamma_n = \frac{\cos(E_u, E_n)}{\sum_{k=1}^{N} \cos(E_u, E_k)}, \tag{8}$$

where $\cos(\cdot,\cdot)$ denotes the cosine similarity. The identical strategy applies to the DTS-T variant by substituting $\hat{\boldsymbol{d}}_n$ with $\hat{\boldsymbol{\tau}}_n$ and replacing $\boldsymbol{\theta}_m$ with $\boldsymbol{\theta}_0$. For generation tasks, task characteristics are obtained by encoding the corresponding task instructions or dataset descriptions with a language model.

Table 1: Multi-task performance (%) when merging ViT-B/32 models on eight tasks. ADR refers to the Accuracy Drop Rate, and AMR denotes the Additional Memory Rate. The best result is **highlighted**, and the second-best result is underlined. **Per** indicates whether the merging is a personalized method.

Method	Per	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg. ↑	ADR ↓	AMR ↓
Individual	-	74.49	77.73	98.22	99.80	97.46	98.73	99.69	79.36	90.69	–	–
Weight-Averaging [57]	✗	65.35	63.41	71.42	71.69	64.20	52.82	87.56	50.18	65.83	27.42	0.00
Fisher-Merging [36]	✗	68.69	69.21	70.73	66.41	72.91	51.17	87.94	59.99	68.38	24.61	0.00
RegMean [23]	✗	65.35	63.53	75.61	78.66	78.10	67.49	93.75	52.02	71.81	20.71	0.00
Task-Arithmetic [22]	✗	54.78	54.98	67.69	78.70	80.21	69.68	97.34	50.37	69.22	23.68	0.00
Ties-Merging [60]	✗	64.17	64.43	76.31	76.62	81.28	69.37	96.53	54.52	72.90	19.62	0.00
DARE [66]	✗	64.76	63.08	71.02	70.70	62.04	50.68	86.17	50.64	64.89	28.45	0.00
AdaMerging [63]	✗	64.44	68.05	79.31	93.80	87.06	91.93	97.56	59.11	80.16	11.62	0.00
AdaMerging++ [63]	✗	66.61	68.34	82.28	94.11	89.54	89.01	98.18	60.66	81.09	10.58	0.00
SVD	✓	71.18	71.60	96.95	99.64	97.24	98.02	99.66	77.58	88.98	1.88	5.87
Twin-Merging [34]	✓	71.56	68.78	89.97	72.11	96.65	93.35	99.66	72.50	83.07	8.41	100.0
WEMOE [50]	✓	73.92	77.36	93.58	99.11	96.25	**98.64**	99.57	76.43	89.36	1.47	49.95
EMR-Merging [20]	✓	71.02	72.75	93.49	99.24	96.86	98.12	99.58	74.36	88.18	2.77	15.62
FREE-Merging [70]	✓	74.00	77.23	93.48	99.55	96.30	98.21	99.57	75.47	89.74	1.05	10.00
TALL-Mask [53]	✓	73.02	**77.38**	97.63	99.38	97.15	98.46	99.66	77.61	90.04	0.71	15.62
T-Switch [41]	✓	74.05	77.32	96.47	99.52	97.33	98.41	99.56	78.55	90.15	0.59	6.25
DTS-T	✓	**74.15**	76.85	97.92	99.66	97.00	98.34	99.63	**79.03**	90.32	0.40	3.68
DTS-D	✓	**74.15**	76.87	**97.98**	**99.78**	**97.36**	98.63	**99.67**	78.78	**90.40**	**0.39**	3.68
DTS-T*	✓	74.01	76.13	97.74	99.52	96.96	98.15	99.60	78.51	90.08	0.67	0.98
DTS-D*	✓	73.97	76.42	97.58	99.56	97.13	98.58	99.47	78.63	90.17	0.57	0.98

Unlike existing adaptation mechanisms that require newly collected labeled data and iterative backpropagation, our proposed variant operates in a strictly zero-shot and data-free manner. The computational overhead introduced by the text encoder is minimal, as all semantic task embeddings are extracted offline prior to deployment. Consequently, this strategy provides a highly efficient adaptation mechanism for unseen tasks without introducing any additional training cost or online inference latency.

4 Experiments

In this section, we evaluate both the effectiveness and efficiency of the proposed method through conventional multi-task model merging experiments, and further assess its ability to generalize to unseen tasks.

4.1 Conventional Multi-task Model Merging

Backbones and datasets. We conduct experiments on visual classification, natural language processing, and natural language generation tasks. For visual classification, we evaluate our method using three variants of CLIP [43]: ViT-B/32, ViT-L/14, and ViT-B/16. Following prior work [34, 41], the ViT-B/32 and ViT-L/14 backbones are tested across eight benchmark datasets: SUN397 [59], Cars [26], RESISC45 [7], EuroSAT [18], SVHN [38], GTSRB [47], MNIST [10], and DTD [8]. Additionally, we extend our evaluation to a broader suite of 30 tasks using the ViT-B/16 architecture. For natural language processing tasks, we adopt RoBERTa [33] and GPT-2 [44] as backbones, and evaluate on eight tasks from the GLUE benchmark [52]: CoLA [54], SST-2 [46], MRPC [13], STS-B [4], QQP [6],

Table 2: Multi-task performance (%) when merging ViT-L/14 models on eight tasks. ADR refers to the Accuracy Drop Rate, and AMR denotes the Additional Memory Rate. The best result is highlighted in **bold**, and the second-best result is underlined. **Per** indicates whether the merging is a personalized method.

Method	**Per**	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg. ↑	ADR ↓	AMR ↓
Individual	-	81.72	92.39	98.85	99.88	98.11	99.24	99.69	84.15	94.25	–	–
Weight-Averaging [57]	✗	72.19	81.42	82.55	91.93	78.08	70.76	97.14	62.95	79.63	15.52	0.00
Fisher-Merging [36]	✗	69.24	88.61	87.50	93.53	80.66	74.82	93.32	70.07	82.22	12.77	0.00
RegMean [23]	✗	73.38	81.80	86.10	97.01	88.12	84.27	98.54	60.82	83.76	11.14	0.00
Task-Arithmetic [22]	✗	73.92	82.13	87.64	92.82	87.91	86.77	98.94	65.64	84.47	10.38	0.00
Ties-Merging [60]	✗	74.74	84.50	89.00	94.18	85.66	82.07	98.65	67.71	84.56	10.28	0.00
DARE [66]	✗	73.03	82.70	86.19	93.41	85.26	83.48	98.58	65.69	83.54	11.36	0.00
AdaMerging [63]	✗	79.03	90.34	90.86	96.19	93.44	98.05	99.12	79.94	90.87	3.59	0.00
AdaMerging++ [63]	✗	79.46	90.38	91.66	97.47	93.42	97.55	99.05	79.20	91.02	3.43	0.00
SVD	✓	78.49	89.73	98.02	99.86	98.08	98.91	99.71	81.91	93.09	1.23	4.08
Twin-Merging [34]	✓	**81.92**	91.59	96.87	99.72	98.03	92.42	99.57	83.94	93.01	1.31	100.0
WEMOE [50]	✓	81.42	92.10	95.46	99.48	97.73	99.13	99.70	83.74	93.60	0.70	58.80
EMR-Merging [20]	✓	80.47	90.71	98.55	99.54	97.94	99.10	99.69	82.71	93.59	0.71	15.62
FREE-Merging [70]	✓	81.60	92.03	96.54	99.61	98.10	98.79	99.72	82.80	93.65	0.63	10.00
TALL-Mask [53]	✓	80.58	91.61	98.68	99.76	98.08	99.24	99.74	83.14	93.85	0.42	15.62
T-Switch [41]	✓	81.84	**92.38**	**98.89**	99.74	98.03	99.08	99.63	83.72	94.16	0.10	6.25
DTS-T	✓	81.82	91.93	98.82	**99.90**	98.05	99.21	**99.77**	83.93	94.18	0.08	2.95
DTS-D	✓	81.75	92.09	**98.89**	99.88	**98.27**	**99.25**	99.72	**84.15**	**94.24**	**0.02**	2.95
DTS-T*	✓	81.72	92.17	98.83	99.82	98.03	99.15	99.71	83.87	94.16	0.10	0.99
DTS-D*	✓	81.69	91.73	98.77	99.86	98.15	99.22	99.72	83.99	94.14	0.12	0.99

MNLI [55], QNLI [45], and RTE [16]. For natural language generation, we use Qwen-14B [2] as the backbone and evaluate on MMLU [19], TruthfulQA [31], and BBQ [40], following prior work [34]. Additional details are provided in the supplementary material.

Baselines. We compare our method against both basic and personalized model merging methods. Basic methods—including Weight-Averaging [57], Fisher-Merging [36], RegMean [23], Task-Arithmetic [22], Ties-Merging [60], DARE [66], and AdaMerging [63]—produce a single merged model without storing any task-specific information. In contrast, personalized approaches retain additional task-specific parameters and include simple SVD, Twin-Merging [34], WEMOE [50], EMR-Merging [20], TALL-Mask [53], T-Switch [41], and FREE-Merging [70]. Unless otherwise noted, we adopt the settings from T-Switch [41] for all baselines. We report results for both **DTS-T** and **DTS-D** using a default sparsity coefficient of $r = 0.3$. To further evaluate efficiency under strict memory constraints, we also provide results for **DTS-T*** and **DTS-D***, where r is adaptively adjusted to ensure that the additional storage overhead remains below 1% across all backbones. Additional details can be found in the supplementary material.

Metrics. We report both absolute performance (accuracy) and relative performance, measured by the *accuracy drop rate* (**ADR**), defined as the ratio between the accuracy difference of the merged model and its individually fine-tuned counterpart (the upper bound). A lower drop rate indicates reduced performance degradation. To assess memory efficiency, we also report the *additional memory rate* (**AMR**), which quantifies the extra memory required to store task-specific information per task, beyond the storage of the merged model itself. Lower values correspond to more memory-efficient approaches.

Table 3: Multi-task performance (%) when merging RoBERTa models on eight tasks. ADR refers to the Accuracy Drop Rate, and AMR denotes the Additional Memory Rate. The best result is **highlighted**, and the second-best result is underlined. **Per** indicates whether the merging is a personalized method.

Method	Per	CoLA	SST2	MRPC	STSB	QQP	MNLI	QNLI	RTE	Avg. ↑	ADR ↓	AMR ↓
Individual	-	60.18	94.04	89.22	90.63	91.41	87.20	92.71	79.06	85.56	–	–
Weight-Averaging [57]	✗	13.96	64.11	69.36	31.84	75.36	42.19	58.70	55.23	51.34	40.00	0.00
RegMean [23]	✗	36.67	90.60	75.74	62.68	83.55	70.02	82.35	58.48	70.01	18.17	0.00
Task-Arithmetic [22]	✗	18.78	85.89	79.90	74.03	83.78	59.08	69.67	62.09	66.65	22.10	0.00
Ties-Merging [60]	✗	20.48	84.40	81.13	58.19	85.70	64.65	74.81	42.96	64.04	25.15	0.00
DARE [66]	✗	9.28	77.87	77.94	30.77	79.25	39.35	71.48	62.09	56.00	34.54	0.00
SVD	✓	58.31	93.92	88.48	**90.65**	87.56	85.80	92.26	65.25	82.78	3.24	4.14
Twin-Merging [34]	✓	59.12	93.53	88.65	72.36	89.17	84.30	92.32	73.89	81.67	4.55	100.0
EMR-Merging [20]	✓	39.96	93.35	86.27	82.73	89.72	85.45	89.57	74.37	80.18	6.29	15.62
FREE-Merging [70]	✓	54.50	93.69	88.46	67.04	88.03	80.60	89.90	79.06	80.16	6.31	10.00
TALL-Mask [53]	✓	45.81	93.81	88.73	88.87	88.51	80.29	92.37	75.09	81.69	4.52	15.62
T-Switch [41]	✓	53.12	**94.04**	89.22	90.15	**91.16**	87.08	**92.57**	**77.26**	84.33	1.44	6.25
DTS-T	✓	59.33	93.66	89.68	90.59	90.55	86.50	91.80	**77.26**	84.93	0.73	3.81
DTS-D	✓	59.66	93.69	89.71	90.62	90.80	**87.09**	92.11	76.17	**84.98**	**0.67**	3.81
DTS-T*	✓	**59.71**	93.35	89.95	90.54	88.87	85.33	91.69	76.90	84.54	1.18	0.88
DTS-D*	✓	59.26	93.81	**89.96**	90.58	89.63	86.45	92.18	76.17	84.75	0.94	0.88

Experimental results. Individual models require storing a fully fine-tuned model per task, and we omit their additional memory usage in comparisons. Table 1 and Table 2 present detailed comparisons of model performance and additional memory overhead for visual classification tasks. In addition, we evaluate our method on 30 datasets using ViT-B/16 as the backbone, with detailed results provided in Table 10 of the supplementary material. The following key observations can be made: (1) Basic model merging methods perform significantly worse than individual models. (2) Recent personalized merging approaches improve per-task performance by incorporating task-specific parameters. However, these methods often incur substantial memory costs. For instance, WEMOE [50] requires an extra 58.80% of the model size per task. In contrast, our method achieves comparable or superior performance with only $\sim 1\%$ additional memory per task. (3) Our method offers flexibility in balancing performance and memory usage through a single tunable sparsity coefficient, allowing it to adapt to varying deployment constraints. In comparison, methods such as EMR-Merging [20] rely on fixed storage budgets and lack adaptability, limiting practical applicability in real-world scenarios.

For natural language processing tasks, as shown in Table 3 and Table 11 in the supplementary material, the results on RoBERTa and GPT-2 follow trends similar to those observed in visual classification tasks. Notably, our method achieves 99.06% of the individual model's performance on RoBERTa, requiring only 0.88% additional memory per task, demonstrating a favorable trade-off between efficiency and effectiveness. For natural language generation tasks, as shown in Table 4, the results align with those observed in the visual classification and natural language processing tasks. These results confirm that the task-specific information extracted by DTS is both compact and effective, capturing key task characteristics with minimal storage.

Table 4: Multi-task performance (%) when merging Qwen-14B models on three tasks. AMR denotes the Additional Memory Rate. The best result is **highlighted**, and the second-best result is highlighted in underlined. **Per** indicates whether the merging is a personalized method.

Method	**Per**	MMLU	TruthfulQA	BBQ	Avg. ↑	AMR ↓
Individual	-	68.36	54.35	93.53	72.08	–
Weight-Averaging [57]	✗	67.11	50.02	82.32	66.48	0.00
DARE [66]	✗	67.23	51.31	83.74	67.43	0.00
Task-Arithmetic [22]	✗	66.63	53.38	78.24	66.08	0.00
Ties-Merging [60]	✗	67.28	50.02	84.10	67.13	0.00
SVD	✓	67.99	52.45	91.71	70.72	5.39
Twin-Merging [34]	✓	68.07	52.38	90.73	70.39	100.0
EMR-Merging [20]	✓	67.94	52.50	91.02	70.49	15.62
FREE-Merging [70]	✓	68.13	53.91	92.54	71.52	10.00
T-Switch [41]	✓	68.05	53.72	92.50	71.42	6.25
DTS-T	✓	68.30	**54.12**	92.97	71.80	3.57
DTS-D	✓	**68.32**	54.11	**92.99**	**71.81**	3.57
DTS-T*	✓	68.20	53.99	92.90	71.70	0.92
DTS-D*	✓	68.18	53.99	92.91	71.70	0.92

4.2 Generalization on Unseen Tasks

Baselines and datasets. To evaluate the generalization ability of our method, we conduct experiments on unseen tasks using the same set of baselines as in previous sections, with a few differences. The details can be found in Sec. 7.2 of the supplementary material. The dataset setup remains consistent with previous experiments, with minor modifications to assess generalization. For visual classification tasks, we designate RESISC45 [7] and SVHN [38] from the original set of eight datasets as unseen tasks, following the protocol in [50]. This means their fine-tuned models are not accessed during the merging process. The remaining six datasets are treated as seen tasks, and their corresponding fine-tuned models are used in merging. For natural language processing tasks, we consider QQP [6], RTE [16], and SST-2 [46] as unseen tasks, while the remaining four tasks are treated as seen tasks. For natural language generation tasks, we treat BBQ as the unseen task and the remaining tasks as seen.

Experimental results. The results for the ViT-B/32 and GPT-2 backbones are presented in Table 5 and Table 6, while those for ViT-L/14 and Qwen-14B backbones can be found in Table 12 and Table 13 in the supplementary material. Our method consistently outperforms all baselines on both seen and unseen tasks. Notably, while personalized merging methods generally perform better than basic approaches on seen tasks, they do not necessarily guarantee better generalization. For example, on the ViT-B/32 backbone, all personalized baselines underperform Ties-Merging [60] on unseen tasks. This highlights a key limitation: personalized methods tend to optimize for task-specific performance on seen tasks, which may limit their transferability to unseen tasks. In contrast, our method demonstrates strong performance on both seen and unseen tasks. Crucially, this gain incurs negligible overhead: Our approach achieves consistent performance improvements in a zero-shot and data-free manner. Besides, the semantic task embeddings are extracted entirely offline, requiring only a single forward pass of the CLIP

Table 5: Generalization results (%) on two unseen tasks for ViT-B/32 models merged from six tasks. The best result is **highlighted**, and the second-best result is underlined. **Per** indicates whether the merging is a personalized method.

		Seen Tasks							Unseen Tasks		
Method	**Per**	SUN397	Cars	EuroSAT	GTSRB	MNIST	DTD	Avg.	RESISC45	SVHN	Avg.
Individual	-	74.49	77.73	99.80	98.73	99.69	79.36	88.30	98.22	97.46	97.84
Fisher-Merging [36]	✗	68.19	67.41	86.47	67.23	81.64	58.69	71.61	60.25	42.51	51.38
RegMean [23]	✗	69.45	70.53	97.06	86.99	98.35	67.12	81.58	50.22	51.50	50.86
Task-Arithmetic [22]	✗	65.28	63.68	87.17	76.18	94.24	56.47	73.84	52.43	45.27	48.85
Ties-Merging [60]	✗	68.27	65.93	81.22	70.01	89.07	56.02	71.75	60.36	47.34	53.85
DARE [66]	✗	69.99	69.32	72.16	55.39	84.52	56.81	68.03	51.60	49.36	50.48
AdaMerging [63]	✗	69.84	72.45	95.18	95.53	98.16	70.71	83.65	48.75	**60.72**	54.74
SVD	✓	71.18	71.60	99.64	98.02	99.66	77.58	86.28	60.60	23.50	42.05
Twin-Merging [34]	✓	71.76	69.20	73.24	93.37	99.64	72.25	79.91	52.43	45.27	48.85
WEMOE [50]	✓	**74.32**	**78.16**	98.71	**98.64**	99.57	75.13	87.42	47.39	51.37	49.38
EMR-Merging [20]	✓	71.81	74.61	99.32	98.40	99.63	75.85	86.60	28.95	49.80	39.38
TALL-Mask [53]	✓	73.02	77.38	99.38	98.46	99.66	77.61	87.58	52.43	45.27	48.85
T-Switch [41]	✓	74.05	77.32	99.52	98.41	99.56	78.55	87.90	60.60	23.50	42.05
DTS-T	✓	74.15	76.85	99.66	98.34	99.63	**79.03**	87.94	61.35	49.11	55.23
DTS-D	✓	74.15	76.87	**99.78**	98.63	**99.67**	78.78	**87.98**	60.91	49.91	55.41
DTS-T*	✓	74.14	76.26	99.70	98.25	99.61	78.74	87.78	61.32	48.92	55.12
DTS-D*	✓	74.07	76.62	99.76	98.58	**99.67**	78.83	87.92	**61.42**	49.57	**55.50**

text encoder prior to deployment. By securing significant improvements without additional training costs or online inference latency, DTS demonstrates a superior performance-efficiency trade-off, confirming its practical viability for cold-start multi-task scenarios.

4.3 Analysis of DTS

Sensitivity to sparsity factor r. In our approach, we approximate the parameters of the individual model by retaining only a partial rank of the SVD decomposition, controlled by the sparsity coefficient r. As shown in Fig. 4, we plot the relationship between performance and model storage overhead for different values of r, and compare our method against several baselines. Additionally, we present the performance of our method for varying values of r in Table 17 of the supplementary material. Our method consistently achieves superior performance with minimal additional storage across all settings.

Effectiveness of the task-specific information extracted by DTS. To further validate the effectiveness of the task-specific information extracted by our method, we compare it with several existing personalized merging approaches, integrated with our adaptive weighting strategy on unseen tasks. Specifically, for EMR-Merging [20], we use the variant of DTS to combine the reconstructed personalized parameters, resulting in a model adapted for unseen tasks. For WEMOE [50] and T-Switch [41], we apply our method to the personalized MLP layers and binarized parameters obtained from seen tasks, respectively. Using the same experimental setup as in the GPT-2 experiments, we report results on unseen tasks in Table 7. Under the same adaptive merging mechanism, our method consistently achieves the best performance, further demonstrating the effectiveness and generalization of the task-specific information.

Table 6: Generalization results (%) on three unseen tasks when merging GPT-2 models on four tasks. The best result is **highlighted**, and the second-best result is underlined. **Per** indicates whether the merging is a personalized method.

		Seen Tasks					Unseen Tasks			
Method	**Per**	CoLA	MNLI	MRPC	QNLI	Avg.	QQP	RTE	SST-2	Avg.
Individual	-	76.80	81.99	80.39	88.27	81.86	89.64	65.34	91.17	82.05
Task-Arithmetic [22]	✗	68.26	68.44	72.54	60.69	67.48	67.12	43.68	50.71	53.84
Ties-Merging [60]	✗	63.08	79.93	34.06	71.16	62.06	70.36	57.03	51.26	59.55
DARE [66]	✗	63.37	65.51	69.11	57.00	63.75	68.07	42.96	51.56	54.20
SVD	✓	76.22	81.26	79.90	87.84	81.31	63.16	52.70	50.91	55.59
Twin-Merging [34]	✓	76.27	80.03	79.65	87.53	80.87	69.73	55.29	51.14	58.72
EMR-MERGING [20]	✓	73.63	81.74	80.14	87.04	80.64	69.94	54.87	51.12	58.64
TALL-Mask [53]	✓	74.78	78.59	78.43	88.15	79.98	67.12	43.68	50.71	53.84
WEMOE [50]	✓	72.30	80.24	77.59	83.06	78.30	69.16	56.93	51.02	58.37
T-Switch [41]	✓	76.27	**81.87**	79.90	88.57	81.65	63.16	52.70	50.91	55.59
DTS-T	✓	**76.98**	81.72	79.94	87.99	81.66	70.00	56.73	50.45	59.06
DTS-D	✓	76.69	81.58	80.33	**88.64**	**81.81**	70.03	**57.65**	51.03	59.57
DTS-T*	✓	76.72	81.84	79.94	88.15	81.66	70.11	56.49	51.45	59.35
DTS-D*	✓	76.51	81.62	80.26	88.41	81.70	**71.12**	56.93	**52.08**	**60.04**

Table 8: The ablation results for Decomposition (D), Thresholding (T), and Scaling (S) when merging ViT-B/32 models.

D	T	S	DTS-T ($r = 0.3$)		DTS-T* ($r = 0.08$)	
			Avg. Acc ↑	AMR ↓	Avg. Acc ↑	AMR ↓
–	–	–	90.69	–	90.69	–
✓	–	–	90.57	33.73	90.15	10.03
–	✓	✓	90.41	9.37	90.41	2.52
✓	✓	✓	90.32	3.68	90.08	0.98
✓	✓	–	4.98	3.69	4.80	0.99

Table 9: The average accuracy (%) and switching time (ms) of our method against rank-1 LoRA.

Backbone	Method	Avg. Acc ↑	Time ↓
ViT/B-32	rank-1 LoRA	68.61	1.01
	Reconstruction	90.32	1.06
ViT/L-14	rank-1 LoRA	81.93	1.83
	Reconstruction	94.18	1.97
Qwen-14B	rank-1 LoRA	68.03	67.61
	Reconstruction	71.80	69.83

Effectiveness of each component in DTS. Our method is composed of three key components: decomposition, thresholding, and scaling. As shown in Table 8, without decomposition, storing task-specific information would require significant extra memory. To demonstrate the effectiveness of thresholding and scaling, we conducted an ablation study, with results presented in Table 8 and Table 16 of the supplementary material. Compared to simple binarization, our thresholding approach yields an accuracy improvement of approximately 2%. Increasing the number of thresholding groups to eight would double the storage cost while providing virtually no additional performance gain. Additionally, without the scaling strategy, model performance drops to around 5%. These results underscore the effectiveness of each component in our method.

Reconstruction Efficiency. The reconstruction operation defined in Eq. (7) is executed only once upon task switching, strictly conditioned on the task ID. Crucially, the reconstructed weights are cached for all subsequent forward passes. Therefore, DTS introduces absolutely no additional FLOPs or latency during the inference phase. To evaluate the switching overhead, we compare our

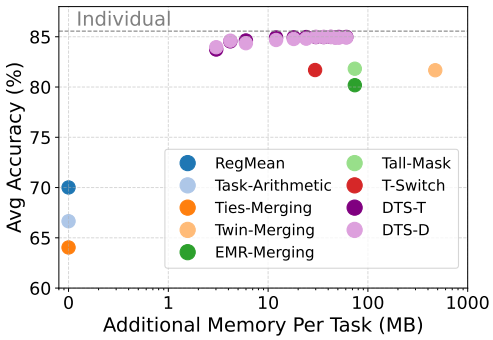


Fig. 4: Performance (%) of merged model and additional memory requirement for different methods with RoBERTa as the backbone.

Method	QQP	RTE	SST-2	Avg.
EMR-Merging [20]	71.48	52.70	50.80	58.33
WEMOE [50]	**72.33**	53.01	50.72	58.69
Twin-Merging [34]	71.84	56.05	50.27	59.38
T-Switch [41]	72.15	53.42	50.91	58.83
DTS-T	70.00	56.73	50.45	59.06
DTS-D	70.03	**57.65**	51.03	59.57
DTS-T*	70.11	56.49	51.45	59.35
DTS-D*	71.12	56.93	**52.08**	**60.04**

Table 7: Effectiveness of the task-specific information extracted by DTS. Even with the same weighting mechanism, our method consistently achieves the best performance on unseen tasks.

reconstruction cost against the standard practice of loading rank-1 LoRA weights. As detailed in Table 9, our method demonstrates a highly favorable trade-off. In return, it unlocks substantial performance gains, boosting average accuracy by 21.71%, 12.25%, and 3.77 % on the respective backbones. This demonstrates that our approach can facilitate robust model merging and dynamic adaptation without compromising deployment efficiency.

5 Conclusion

In this work, we revisited the challenge of model merging from the perspective of task similarity and demonstrated that significant performance degradation persists even when merging models fine-tuned on highly similar tasks. To address this, we introduced DTS—a compact and effective model merging method based on decomposition, thresholding, and scaling. DTS efficiently preserves essential task-specific information by decomposing task vectors into low-rank approximations, achieving high performance with minimal memory overhead. To support generalization to unseen tasks, we further proposed a variant of DTS, a data-free adaptive merging strategy that weights task-specific information based on the semantic similarity of task characteristics. Experimental results on standard multi-task model merging benchmarks demonstrate that our method consistently outperforms state-of-the-art baselines, requiring only 1% extra storage per task. Moreover, experiments on unseen tasks show that the adaptive variant of DTS achieves superior generalization performance.

Acknowledgments

This work was funded by the National Natural Science Foundation of China under Grants (62276256, U2441251), the Beijing Natural Science Foundation (Z260008), and the National Key Research and Development Program of China (2026ZD1500301).

References

1. Ainsworth, S.K., Hayase, J., Srinivasa, S.: Git re-basin: Merging models modulo permutation symmetries. In: Proc. ICLR (2023)
2. Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., et al.: Qwen technical report. arXiv preprint arXiv:2309.16609 (2023)
3. Caruana, R.: Multitask learning. Machine learning pp. 41–75 (1997)
4. Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., Specia, L.: Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In: Proceedings of the 11th International Workshop on Semantic Evaluation (2017)
5. Chen, H.M., Hu, S.X., Luk, W., Hospedales, T., Fan, H.: Fw-merging: Scaling model merging with frank-wolfe optimization. In: Proc. ICCV (2025)
6. Chen, Z., Zhang, H., Zhang, X., Zhao, L.: Quora question pairs (2018)
7. Cheng, G., Han, J., Lu, X.: Remote sensing image scene classification: Benchmark and state of the art. Proceedings of the IEEE pp. 1865–1883 (2017)
8. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: Proc. CVPR (2014)
9. Davari, M., Belilovsky, E.: Model breadcrumbs: scalable upcycling of finetuned foundation models via sparse task vectors merging. In: Proc. ICML (2024)
10. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine pp. 141–142 (2012)
11. Ding, Y., Liang, J., Jiang, B., Wang, Z., Zheng, A., Luo, B.: Harmonizing and merging source models for clip-based domain generalization. arXiv preprint arXiv:2506.09446 (2025)
12. Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., Smith, N.: Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. arXiv:2002.06305 (2020)
13. Dolan, B., Brockett, C.: Automatically constructing a corpus of sentential paraphrases. In: Third International Workshop on Paraphrasing (2005)
14. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. Psychometrika pp. 211–218 (1936)
15. Gargiulo, A.A., Crisostomi, D., Bucarelli, M.S., Scardapane, S., Silvestri, F., Rodola, E.: Task singular vectors: Reducing task interference in model merging. In: Proc. CVPR (2025)
16. Giampiccolo, D., Magnini, B., Dagan, I., Dolan, W.B.: The third pascal recognizing textual entailment challenge. In: Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing (2007)
17. He, J.: Gradient reweighting: Towards imbalanced class-incremental learning. In: Proc. CVPR (2024)
18. Helber, P., Bischke, B., Dengel, A., Borth, D.: Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing pp. 2217–2226 (2019)
19. Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., Steinhardt, J.: Measuring massive multitask language understanding. In: Proc. ICLR (2020)
20. Huang, C., Ye, P., Chen, T., He, T., Yue, X., Ouyang, W.: Emr-merging: Tuning-free high-performance model merging. In: Proc. NeurIPS (2024)
21. Hyun, J., Hwang, S., Han, S.H., Kim, T., Lee, I., Wee, D., Lee, J.Y., Kim, S.J., Shim, M.: Multi-granular spatio-temporal token merging for training-free acceleration of video llms. In: Proc. ICCV (2025)

22. Ilharco, G., Ribeiro, M.T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., Farhadi, A.: Editing models with task arithmetic. In: Proc. ICLR (2023)
23. Jin, X., Ren, X., Preotiuc-Pietro, D., Cheng, P.: Dataless knowledge fusion by merging weights of language models. In: Proc. ICLR (2023)
24. Jordan, K., Sedghi, H., Saukh, O., Entezari, R., Neyshabur, B.: Repair: Renormalizing permuted activations for interpolation repair. In: Proc. ICLR (2023)
25. Kim, Y., Lee, S., Jung, A., Ryu, B., Hong, S.: Task vector quantization for memory-efficient model merging. In: Proc. ICCV (2025)
26. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: Proc. ICCV (2013)
27. Lange, K., Lange, K.: Singular value decomposition. Numerical Analysis for Statisticians pp. 129–142 (2010)
28. Lee, Y., Jung, J., Baik, S.: Mitigating parameter interference in model merging via sharpness-aware fine-tuning. In: Proc. ICLR (2025)
29. Li, L., Zhang, T., Bu, Z., Wang, S., He, H., Fu, J., Wu, Y., Bian, J., Chen, Y., Bengio, Y.: Map: Low-compute model merging with amortized pareto fronts via quadratic approximation. In: Proc. ICLR (2025)
30. Li, Y., Ma, Y., Yan, S., Zhang, C., Liu, J., Lu, J., Xu, Z., Chen, M., Wang, M., Zhan, S., et al.: Model merging in pre-training of large language models. In: Proc. NeurIPS (2025)
31. Lin, S., Hilton, J., Evans, O.: Truthfulqa: Measuring how models mimic human falsehoods. In: Proc. ACL (2022)
32. Liu, S., Wu, H., He, B., Liu, Z., Han, X., Yuan, M., Song, L.: 1bit-merging: Dynamic quantized merging for large language models. arXiv:2502.10743 (2025)
33. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv:1907.11692 (2019)
34. Lu, Z., Fan, C., Wei, W., Qu, X., Chen, D., Cheng, Y.: Twin-merging: Dynamic integration of modular expertise in model merging. In: Proc. NeurIPS (2024)
35. Marczak, D., Magistri, S., Cygert, S., Twardowski, B., Bagdanov, A.D., van de Weijer, J.: No task left behind: Isotropic model merging with common and task-specific subspaces. In: Proc. ICML (2025)
36. Matena, M.S., Raffel, C.A.: Merging models with fisher-weighted averaging. In: Proc. NeurIPS (2022)
37. Muqeeth, M., Liu, H., Raffel, C.: Soft merging of experts with adaptive routing. arXiv:2306.03745 (2023)
38. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y., et al.: Reading digits in natural images with unsupervised feature learning. In: Proc. NeurIPS (2011)
39. Oh, C., Li, Y., Song, K., Yun, S., Han, D.: Dawin: Training-free dynamic weight interpolation for robust adaptation. In: Proc. ICLR (2025)
40. Parrish, A., Chen, A., Nangia, N., Padmakumar, V., Phang, J., Thompson, J., Htut, P.M., Bowman, S.R.: Bbq: A hand-built bias benchmark for question answering. In: Proc. ACL (2022)
41. Qi, B., Li, F., Wang, Z., Gao, J., Li, D., Ye, P., Zhou, B.: Less is more: Efficient model merging with binary task switch. In: Proc. CVPR (2025)
42. Qiu, Z., Xu, Y., He, C., Meng, F., Xu, L., Wu, Q., Li, H.: Mingle: Mixtures of null-space gated low-rank experts for test-time continual model merging. In: Proc. NeurIPS (2025)

43. 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: Proc. ICML (2021)
44. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog p. 9 (2019)
45. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. In: Proc. EMNLP (2016)
46. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proc. EMNLP (2013)
47. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The german traffic sign recognition benchmark: a multi-class classification competition. In: Proc. IJCNN (2011)
48. Stoica, G., Ramesh, P., Ecsedi, B., Choshen, L., Hoffman, J.: Model merging with svd to tie the knots. In: Proc. ICLR (2025)
49. Sun, W., Li, Q., Geng, Y., Li, B.: Cat merging: A training-free approach for resolving conflicts in model merging. In: Proc. ICML (2025)
50. Tang, A., Shen, L., Luo, Y., Yin, N., Zhang, L., Tao, D.: Merging multi-task models via weight-ensembling mixture of experts. In: Proc. ICML (2024)
51. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288 (2023)
52. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: Glue: A multi-task benchmark and analysis platform for natural language understanding. In: Proc. ICLR (2018)
53. Wang, K., Dimitriadis, N., Ortiz-Jimenez, G., Fleuret, F., Frossard, P.: Localizing task information for improved model merging and compression. In: Proc. ICML (2024)
54. Warstadt, A., Singh, A., Bowman, S.R.: Neural network acceptability judgments. In: Proc. ACL (2019)
55. Williams, A., Nangia, N., Bowman, S.R.: A broad-coverage challenge corpus for sentence understanding through inference. In: Proc. ACL (2018)
56. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Huggingface's transformers: State-of-the-art natural language processing. arXiv:1910.03771 (2019)
57. 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: Proc. ICML (2022)
58. Wu, H., Xu, J., Le, H., Samaras, D.: Importance-based token merging for efficient image and video generation. In: Proc. ICCV (2025)
59. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: Proc. CVPR (2010)
60. Yadav, P., Tam, D., Choshen, L., Raffel, C.A., Bansal, M.: Ties-merging: Resolving interference when merging models. In: Proc. NeurIPS (2023)
61. Yan, K., Zhang, M., Cui, S., Zikun, Q., Jiang, B., Liu, F., Zhang, C.: Calm: Consensus-aware localized merging for multi-task learning. In: Proc. ICML (2025)
62. Yang, E., Tang, A., Shen, L., Guo, G., Wang, X., Cao, X., Zhang, J.: Continual model merging without data: Dual projections for balancing stability and plasticity. In: Proc. NeurIPS (2025)
63. Yang, E., Wang, Z., Shen, L., Liu, S., Guo, G., Wang, X., Tao, D.: Adamerging: Adaptive model merging for multi-task learning. In: Proc. ICLR (2024)

64. Yang, J., Jin, D., Tang, A., Shen, L., Zhu, D., Chen, Z., Zhao, Z., Wang, D., Cui, Q., Zhang, Z., et al.: Mix data or merge models? balancing the helpfulness, honesty, and harmlessness of large language model via model merging. In: Proc. NeurIPS (2025)
65. Ye, P., Huang, C., Shen, M., Chen, T., Huang, Y., Zhang, Y., Ouyang, W.: Merging vision transformers from different tasks and domains. arXiv:2312.16240 (2023)
66. Yu, L., Yu, B., Yu, H., Huang, F., Li, Y.: Language models are super mario: Absorbing abilities from homologous models as a free lunch. In: Proc. ICML (2024)
67. Zhang, Y., Zhao, Z., Chen, Z., Ding, Z., Yang, X., Sun, Y.: Beyond training: Dynamic token merging for zero-shot video understanding. In: Proc. ICCV (2025)
68. Zhang, Y., Yang, Q.: A survey on multi-task learning. IEEE Transactions on Knowledge and Data Engineering pp. 5586–5609 (2021)
69. Zhao, Z., Shen, T., Zhu, D., Li, Z., Su, J., Wang, X., Wu, F.: Merging loras like playing lego: Pushing the modularity of lora to extremes through rank-wise clustering. In: Proc. ICLR (2025)
70. Zheng, S., Wang, H.: Free-merging: Fourier transform for efficient model merging. In: Proc. CVPR (2025)
71. Zhong, Y., Liu, Z., Li, Y., Wang, L.: Aim: Adaptive inference of multi-modal llms via token merging and pruning. In: Proc. ICCV (2025)
72. Zhou, Y., Wu, X., Wu, J., Feng, L., Tan, K.C.: Hm3: Hierarchical multi-objective model merging for pretrained models. In: Proc. NeurIPS (2025)
73. Zhou, Y., Song, L., Wang, B., Chen, W.: Metagpt: Merging large language models using model exclusive task arithmetic. In: Proc. EMNLP (2024)
74. Zhu, D., Song, Y., Shen, T., Zhao, Z., Yang, J., Zhang, M., Wu, C.: Remedy: Recipe merging dynamics in large vision-language models. In: Proc. ICLR (2025)