Learning to Recover Task Experts from a Multi-Task Merged Model

Jinwook Jung[1], Taegyu Kim[1], Kumju Jo[1], and Sungyong Baik[1,2⋆]

[1]Department of Artificial Intelligence, [2]Department of Data Science
Hanyang University, Republic of Korea
jjw970517@hanyang.ac.kr, sa090180@gmail.com, juice0630@hanyang.ac.kr,
dsybaik@hanyang.ac.kr

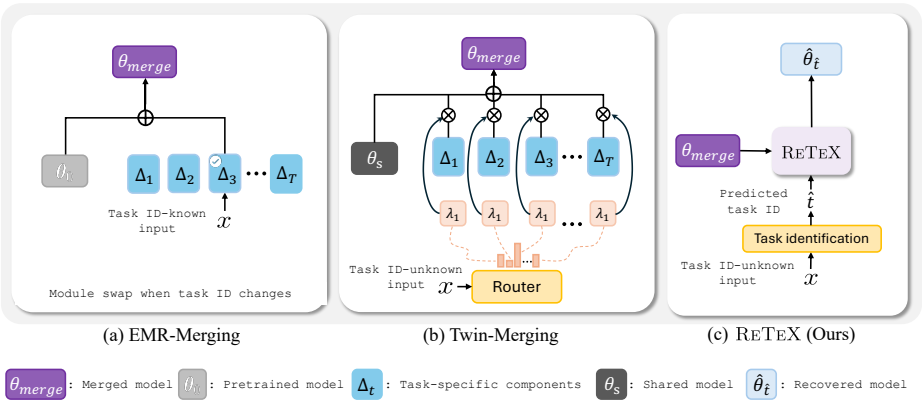
Abstract. Multi-task model merging aims to consolidate several task-specific experts into a unified model, yet static merging consistently suffers from parameter interference. While dynamic merging models aim to bridge this gap, many works rely on the costly storage and loading of redundant expert components at inference. In this work, from the perspective of task expert, we view parameter interference as parameter perturbation introduced to each expert during merging process. We show that such parameter perturbations can be modeled as affine transformation, which can be approximated as additive offsets. Motivated by these, we propose **Re**cover **T**ask **eX**pert (**ReTeX**), a framework that predicts those offsets, in order to *undo* parameter interference and recover task-expert performance from a single merged checkpoint. To recover the appropriate expert when task identity is unknown, we introduce a router-free task identifier based on SVD subspace signatures computed offline before inference. At inference, the identifier selects the task whose subspace yields the smallest projection residual for a given input. As a result, ReTeX recovers over 95% of individual-expert performance in both vision and NLP domains, while significantly improving generalization to unseen tasks. Crucially, we also show that the parameter offset prediction leads to emergent adaptive interpolation of expert knowledge for out-of-distribution (OOD) tasks. ReTeX adaptively interpolates seen expert knowledge to handle unseen tasks. Our code is available at https://github.com/BAIKLAB/ReTeX

Keywords: Multi-task model merging · Task expert recovery

1 Introduction

Since the advent of foundation models [1, 26, 60] pre-trained on large-scale data, deep learning has achieved remarkable success across diverse domains by fine-tuning such large pre-trained models for individual downstream tasks. This paradigm naturally yields a growing collection of task-specific experts. However, storing an expert for each task separately is costly to deploy at scale due to storage, memory traffic, and management overhead.

⋆ Corresponding author.



(a) EMR-Merging (b) Twin-Merging (c) ReTeX (Ours)

Fig. 1: Component requirements for multi-task deployment. (a) *EMR-Merging* [33] assumes that the task ID is known and activates task-specific components on top of a shared backbone $\boldsymbol{\theta}_0$. (b) *Twin-Merging* [47] handles task ID-unknown inputs by using a router to predict input-dependent mixing coefficients over task components, combined with a shared backbone $\boldsymbol{\theta}_s$. (c) ReTeX handles task ID-unknown inputs by first predicting the task ID $\hat{t}$ and then recovering $\hat{\boldsymbol{\theta}}_{\hat{t}}$ from a single merged checkpoint $\boldsymbol{\theta}_{\text{merge}}$ using the predicted offset $\hat{\boldsymbol{\beta}}_{\hat{t}}$, without storing expert checkpoints at inference.

Multi-task model merging [34, 75, 76] has emerged as a promising solution to consolidate multiple experts into a single model by directly manipulating their parameters. Most early approaches construct a single merged model by weighted combinations of parameters or updates, where task coefficients are often selected by heuristics or hyperparameter tuning [34, 36, 48, 67, 75, 76, 79]. However, static merging methods frequently underperform task experts due to parameter interference: task-specific updates collide in weight space, producing model parameters that are suboptimal for multiple tasks.

Recently, dynamic merging methods [47, 49, 53, 68] have been proposed to improve accuracy by selecting or composing task-specific components conditioned on each input, but typically require storing and reading additional task-specific parameters at inference. Moreover, several existing methods assume that the task identity is known at inference [33, 34, 75], or train a separate router using substantial labeled task data [47, 68, 78], which can introduce additional storage overhead and privacy issues.

We take an alternative view: we interpret parameter interference as parameter perturbation introduced to each task expert during merging process. Upon this perspective, we aim to recover task expert performance from a single merged checkpoint. We empirically find that the dominant effect of interference can be modeled as an additive offset. Thus, we aim to "undo" parameter interference and recover task-expert performance by predicting and removing additive offsets.

Based on this insight, we propose ReTeX, a plug-and-play expert recovery framework. ReTeX augments a merged model with a lightweight recovery module that predicts a task-conditioned parameter offset and applies it to the merged

weights to reconstruct task-specialized behavior. Importantly, ReTeX is trained using parameter supervision only: the supervision signal is provided by existing expert checkpoints (targets are expert parameters), and recovery training does not require access to task datasets or input–output examples. This design keeps recovery training simple and avoids additional data collection or storage.

A key challenge is determining which expert to recover when task identity is unknown. To this end, we introduce a router-free task identifier based on SVD subspace signatures (singular values and vectors). Notably, SVD subspace signatures are computed offline before inference. At inference, we select the task whose subspace yields the smallest projection residual for each input, removing the need for training a router or storing task experts. Surprisingly, adaptive interpolation of expert knowledge emerges from the proposed parameter offset prediction, when faced with unseen or out-of-distribution tasks.

We further note that our proposed method is a flexible plug-and-play framework that can be seamlessly applied on top of various merged models. Extensive experiments on computer vision and NLP benchmarks demonstrate that ReTeX substantially enhances the performance of various model merging methods, greatly improving the performance on both seen and unseen tasks and achieving 95% of individual-expert performance.

We summarize our contributions as follows:

1. We introduce ReTeX, an expert recovery framework that reconstructs task expert performance from a single merged checkpoint by "undoing" parameter interference via offset prediction.
2. We propose a router-free task identification method based on SVD subspace projection residuals, removing the need for labeled router training and reducing data storage requirements.
3. We show that ReTeX is plug-and-play on top of various merged models across vision and NLP, recovering up to 95% of expert performance while improving robustness to unseen/OOD tasks.

2 Related work

Model merging. Model merging aims to consolidate multiple task experts into a single merged model by directly combining or editing expert parameters, avoiding joint multi-task retraining. Early works have employed weighted combination of expert parameters [36, 48] or corresponding task vectors [34], where the goal becomes finding a set of coefficients such that a resulting merged model achieve strong performance on all tasks. However, simple weighted averaging often results in parameter interference and thus performance degradation, which few works have attempted to tackle by sparsifying or masking task vectors to mitigate conflicts [71, 75]. A complementary line of work addresses this issue before merging by modifying the fine-tuning stage itself, encouraging task experts to become more mergeable or less interfering in parameter space [35, 43, 54]. Despite efforts, a single shared model still often falls short of per-task experts.

Task-conditioned adaptation and meta-learning. HyperNetworks and FiLM generate task- or input-conditioned weights and feature-wise affine modulations [30, 56]. Few-shot conditional adaptation methods, such as CNAPs and its follow-up variants, infer task-conditioned parameters from support-set statistics or limited supervision [10, 11, 46, 59, 69]. Related meta-learning methods learn task-adaptive initialization, attenuation, hyperparameters, losses, or online/test-time adaptation mechanisms [4–8, 16–18, 27]. Although these works are related in that they produce task-adaptive behavior, they primarily adapt from support examples, task observations, or gradient-based updates. Although similarly task-adaptive, ReTeX has a different goal: it recovers task-expert behavior from an already merged checkpoint using expert-parameter supervision, rather than adapting from support examples.

Dynamic model merging. Dynamic model merging aims to reduce parameter interference by formulating input-adaptive weighted combination by predicting coefficients for each input via routing at inference time [47,49,53,68]. While these methods have greatly improved performance, they require large task datasets to train a routing module or a large number of forward passes to estimate the task ID of each sample. A related training-free direction, SiM [64], shows that SVD-based task manifolds constructed from a small support set can provide reliable task signals by routing each test input through projection residuals, without additional router training. Building on this observation, ReTeX uses the SVD residual-based task signal not merely to select among stored components, but to condition task-expert recovery from a single merged checkpoint. However, these dynamic or routing-based approaches still require storing and loading multiple task components during inference, increasing memory traffic and complexity as the number of tasks grows. Meanwhile, other works have proposed storing lightweight task-expert components by compressing full parameters via masking or sparsification [28,33,71]. These approaches reduce storage for task experts, but they assume knowledge of the task ID of each input, limiting their applicability in practice. In contrast to previous dynamic model merging methods that require either task ID of each input or multiple experts during inference, our proposed approach directly recovers task-expert performance from a single merged model.

3 Background

Problem setting. Given a pre-trained model $f : \mathcal{X} \times \Theta \to \mathcal{Y}$ with parameters $\boldsymbol{\theta}_0 \in \Theta$, we consider T downstream tasks indexed by $t \in \{1, \ldots, T\}$. Since many finetuned models have become available in HuggingFace with the advent of pretraining-finetuning paradigm, we make the following assumption, similar to previous works [34, 47, 49, 53, 68]: for each task t, a task expert $f_{\boldsymbol{\theta}_t}$ is assumed to be available prior to merging process. Prior to model merging process, we assume each task expert has been obtained by fine-tuning $f_{\boldsymbol{\theta}_0}$ on each task dataset $\mathcal{D}_t = \{(\boldsymbol{x}_t^i, y_t^i)\}_{i=1}^{N_t}$ with $\boldsymbol{x}_t^i \in \mathcal{X}$ and $y_t^i \in \mathcal{Y}$.

Given task experts, multi-task model merging aims to find a single merged model $\boldsymbol{\theta}_{\text{merge}}$ that can generalize across all tasks by directly operating on the

expert parameters $\{\boldsymbol{\theta}_t\}_{t=1}^{T}$:

$$\boldsymbol{\theta}_{\text{merge}} = \sum_{t=1}^{T} \alpha_t \boldsymbol{\theta}_t. \tag{1}$$

Task arithmetic. To better facilitate the management of task expert knowledge, Task Arithmetic (TA) [34] proposes to isolate task-specific knowledge from prior knowledge; $\boldsymbol{\tau}_t := \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$. Using task vectors, TA and subsequent works formulate merging as

$$\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \lambda_t \boldsymbol{\tau}_t, \tag{2}$$

where λ_t is a task coefficient.

Dynamic model merging. Despite previous efforts, static model merging is susceptible to parameter interference [34], as it is forced to decide which task-specific model parameter will contribute to the final merged model parameters, which are then fixed for all input data. However, each data sample can come from varying tasks and thus require varying model expertise. Recently, several works have shifted their attention to dynamic model merging [37, 45, 47, 49, 53]. Dynamic model merging aims to find input-wise model expertise; in other words, the goal is to find the input-adaptive task coefficients $\lambda_t(\boldsymbol{x})$:

$$\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \lambda_t(\boldsymbol{x}) \boldsymbol{\tau}_t. \tag{3}$$

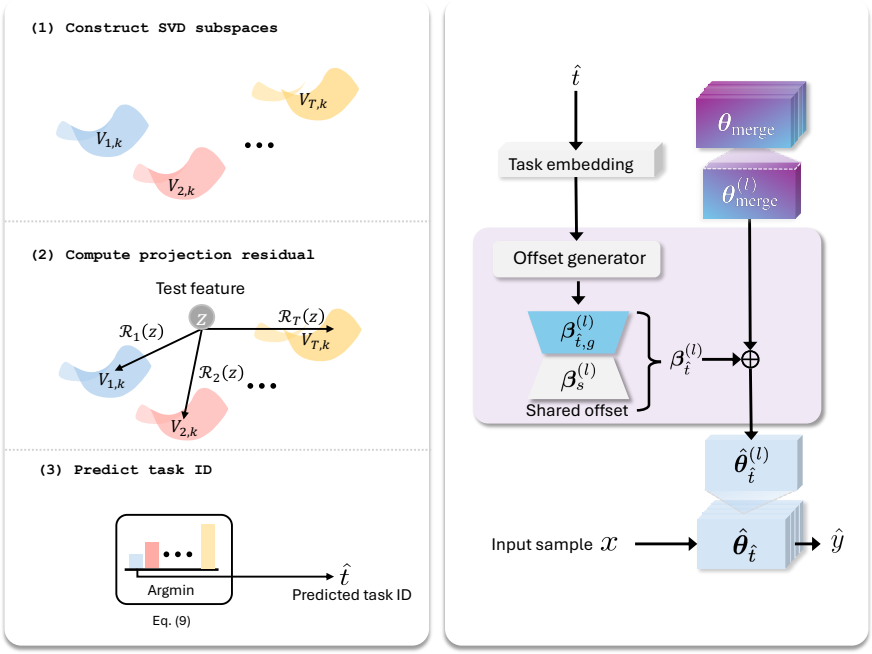
4 Learning to Recover Task Experts

Previous dynamic model merging methods have greatly improved the performance of a merged model by dynamically composing experts with input-adaptive coefficients at inference time [47, 49, 53, 68]. While dynamic model merging methods greatly mitigated parameter interference, previous methods often require the knowledge of task ID of each input example [28, 33, 71] or require additional training with abundant amount of training examples [47, 49, 68].

In this work, we approach model merging from the perspective of each task expert: each task expert experiences parameter perturbations that other experts inject during merging, resulting in parameter interference. Thus, in this work, we aim to "undo" such parameter perturbations and recover a task expert from a single merged model. As delineated in Appendix Sec. A, we show that each task expert parameter $\boldsymbol{\theta}_t$ can be expressed as an affine transformation of $\boldsymbol{\theta}_{\text{merge}}$:

$$\boldsymbol{\theta}_t = \gamma_t \, \boldsymbol{\theta}_{\text{merge}} \; + \; \boldsymbol{\beta}_t, \tag{4}$$

where γ_t and $\boldsymbol{\beta}_t$ denote task-conditioned scaling and shift terms, which are composed of task coefficients and other task expert parameters. This form suggests that expert recovery reduces to predicting $(\gamma_t, \boldsymbol{\beta}_t)$ conditioned on task identity.



(a) Task identification (b) Task expert recovery

Fig. 2: Overall ReTeX framework. (a) Task identification constructs a task memory bank by fitting a low-dimensional SVD subspace to merged-model representations for each task. Given a test representation $\mathbf{z}$, ReTeX computes projection residuals $\{\mathcal{R}_t(\mathbf{z})\}_{t=1}^{T}$ and outputs a task ID estimate $\hat{t}$ by an argmin rule. (b) Task expert recovery conditions a task embedding on $\hat{t}$ and predicts low-rank offsets across layers. Adding the offsets to merged parameters yields inference parameters for prediction.

However, our empirical analysis indicates that γ_t concentrates near an identity 1, which motivates an offset-only recovery rule:

$$\boldsymbol{\theta}_t \approx \boldsymbol{\theta}_{\text{merge}} + \boldsymbol{\beta}_t, \tag{5}$$

where we interpret $\boldsymbol{\beta}_t$ as "undoing" parameter perturbations introduced by other task experts to reduce parameter interference and recover task-expert performance for task t.

Thus, as illustrated in Fig. 2, ReTeX operates in two stages: (1) task identification that estimates a task ID $\hat{t}$ by nearest-subspace matching using a compact task memory bank (Sec. 4.1), and (2) task expert recovery that reconstructs inference parameters from $\boldsymbol{\theta}_{\text{merge}}$ conditioned on $\hat{t}$ using offset prediction (Sec. 4.2).

4.1 Task identification

To estimate task ID of each, we take inspirations from the nearest-subspace viewpoint used in the Nearest Subspace Classifier (NSC) [15]. In particular,

ReTeX represents each task with a low-dimensional subspace estimated from merged-model representations. At inference, ReTeX selects the task that minimizes the projection residual. We note that these subspaces incur much less memory overhead, compared to full task expert parameters or low-rank experts, as demonstrated in experimental results.

Constructing task SVD subspaces. Let $\mathbf{z}_t^i = f_{\text{merge}}(\boldsymbol{x}) \in \mathbb{R}^d$ denote an intermediate representation extracted by a merged model $f_{\boldsymbol{\theta}_{\text{merge}}}$. For each task $t \in \{1, \ldots, T\}$, we utilize N reference inputs from original training set $\mathcal{D}_t^{\text{mb}} = \{\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, \ldots, \boldsymbol{x}_t^N\}$ and compute the empirical mean of its representations $\mathbf{z}_t^i$:

$$\boldsymbol{\mu}_t = \frac{1}{N} \sum_{i=1}^{N} \mathbf{z}_t^i. \tag{6}$$

Then, we compute the feature matrix $\mathbf{X}_t \in \mathbb{R}^{N \times d}$ whose i-th row equals $(\mathbf{z}_t^i - \boldsymbol{\mu}_t)^\top$, and compute the SVD

$$\mathbf{X}_t = \mathbf{U}_t \boldsymbol{\Sigma}_t \mathbf{V}_t^\top. \tag{7}$$

ReTeX keeps the top-k right singular vectors as the task subspace basis $\mathbf{V}_{t,k} \in \mathbb{R}^{d \times k}$ with $k \ll d$. The set $\boldsymbol{\mu}_t + \text{span}(\mathbf{V}_{t,k})$ defines an affine subspace for task t. ReTeX stores $\{\boldsymbol{\mu}_t, \mathbf{V}_{t,k}\}_{t=1}^T$ as task signatures. Unless otherwise specified, ReTeX uses $N = 64$ reference inputs per task.

Task identification. Given a test input $\boldsymbol{x}$, and its feature $\mathbf{z} = f_{\text{merge}}(\boldsymbol{x})$. ReTeX measures alignment to each task subspace using the projection residual

$$\mathcal{R}_t(\mathbf{z}) = \left\| \left(\mathbf{I} - \mathbf{V}_{t,k} \mathbf{V}_{t,k}^\top\right) (\mathbf{z} - \boldsymbol{\mu}_t) \right\|_2, \tag{8}$$

where $\mathbf{I} \in \mathbb{R}^{d \times d}$ denotes the identity matrix. The task ID estimate satisfies

$$\hat{t} = \underset{t \in \{1, \ldots, T\}}{\arg\min}\ \mathcal{R}_t(\mathbf{z}). \tag{9}$$

4.2 Task expert recovery

Given the task ID estimate $\hat{t}$ from Sec. 4.1, ReTeX recovers the corresponding task-expert performance by predicting low-rank offsets that would "undo" parameter interference as follows: ReTeX defines inference parameters as

$$\boldsymbol{\theta}_{\hat{t}} \ := \ \boldsymbol{\theta}_{\text{merge}} + \boldsymbol{\beta}_{\hat{t}}, \tag{10}$$

where $\boldsymbol{\beta}_{\hat{t}}$ denotes the offset predicted, conditioned on estimated task ID $\hat{t}$.

Task embedding. Each task t is represented with a learnable embedding $\boldsymbol{e}_t \in \mathbb{R}^{d_{\text{emb}}}$, where d_{emb} is the dimension of task embedding. A one-hot task indicator for $\hat{t}$ maps to the embedding $\boldsymbol{e}_{\hat{t}}$, which serves as input to the offset generator.

Low-rank offset prediction. ReTeX adopts a low-rank factorization for offsets independently for each layer l. Let $\boldsymbol{\theta}_{\text{merge}}^{(l)} \in \mathbb{R}^{a \times b}$ denote merged parameters

at layer l, and choose rank $r < \min(a, b)$. For the further efficiency, we decompose an offset as follows:

$$\boldsymbol{\beta}_{\hat{t}}^{(l)} = \boldsymbol{\beta}_{\hat{t},g}^{(l)} \boldsymbol{\beta}_{s}^{(l)}, \tag{11}$$

where $\boldsymbol{\beta}_{s}^{(l)} \in \mathbb{R}^{r \times b}$ is a learnable offset shared across all task experts, and $\boldsymbol{\beta}_{\hat{t},g}^{(l)} \in \mathbb{R}^{a \times r}$ is a task-specific offset predicted by single-layer offset generator $h^{(l)}$ conditioned on $\boldsymbol{e}_{\hat{t}}$. Thus, for each layer, task-expert parameters at layer l are approximated as $\boldsymbol{\theta}_{\hat{t}}^{(l)} = \boldsymbol{\theta}_{\text{merge}}^{(l)} + \boldsymbol{\beta}_{\hat{t}}^{(l)}$. Stacking layers yields $\boldsymbol{\theta}_{\hat{t}} = \boldsymbol{\theta}_{\text{merge}} + \boldsymbol{\beta}_{\hat{t}}$, consistent with Eq. (10).

Optimization. We note that the training of the offset prediction module only uses parameter supervision, without the need for task datasets of input-output samples or test samples. The training process of ReTeX only requires a randomly sampled task ID t and its corresponding task expert parameters. In other words, the training objective for ReTeX is

$$\mathcal{L} = \mathbb{E}_{t \sim \mathcal{U}(1,T)} \left[\left\| (\boldsymbol{\theta}_{\text{merge}} + \hat{\boldsymbol{\beta}}_t) - \boldsymbol{\theta}_t \right\|_2^2 \right], \tag{12}$$

where $\mathcal{U}(1, T)$ denotes a uniform distribution over task indices; $\hat{\boldsymbol{\beta}}_t$ denotes the predicted task-specific offset for the sampled task t; and $\boldsymbol{\theta}_t$ acts as ground-truth.

At inference, ReTeX does not require storage of task experts: ReTeX predicts a task ID $\hat{t}$ and recovers $\boldsymbol{\theta}_{\hat{t}}$ on demand from $\boldsymbol{\theta}_{\text{merge}}$.

5 Experiments

We evaluate ReTeX in a *task-agnostic* multi-task deployment, where test inputs arrive as a mixed-task stream and per-sample task identity is not provided. Accordingly, we do not assume task-specific validation or calibration at inference; ReTeX predicts task identity and recovers the corresponding expert behavior on demand from a *single* merged checkpoint. We report results on vision, NLP, and additionally study unseen-task generalization and efficiency.

Baselines and comparison scope. We group baselines by inference-time component requirements. *Static model merging* produces a single merged checkpoint and applies it to all inputs, without storing expert checkpoints at inference. This group includes Weight Averaging, Task Arithmetic [34], TIES-Merging [75], Consensus TA [71] and AdaMerging [76]. *Dynamic model merging* performs input-adaptive merging at inference and typically requires access to task-specific components (e.g., expert checkpoints or equivalent representations) during inference. This group includes Twin-Merging [47], DaWin [53], MoW-Merging [78], and WEMoE [68]. We also report Zero-shot (pre-trained model) and Individual experts (original task experts) as lower and upper reference points. Detailed baseline configurations and protocols are provided in Appendix Sec. C.

Implementation details. Unless specified otherwise, the base merged model $\boldsymbol{\theta}_{\text{merge}}$ is constructed via simple Weight Averaging of task experts, to highlight

Table 1: Multi-task performance of merged models across different CLIP backbones and numbers of tasks. Values in parentheses $_{(\cdot)}$ indicate normalized accuracy (merged / individual). All methods are evaluated on 8, 14, and 20 computer vision tasks.

Method	ViT-B/32			ViT-B/16			ViT-L/14		
	8 tasks	14 tasks	20 tasks	8 tasks	14 tasks	20 tasks	8 tasks	14 tasks	20 tasks
Zero-shot	48.3	57.2	56.1	55.3	61.3	59.7	64.7	68.2	65.2
Individual	92.9	90.9	91.4	94.7	92.8	92.8	95.9	94.3	94.8
(Static model merging)									
Weight Averaging	$66.3_{(72.1)}$	$64.3_{(71.1)}$	$61.0_{(67.5)}$	$72.2_{(76.6)}$	$69.5_{(74.8)}$	$65.3_{(70.4)}$	$79.6_{(83.2)}$	$76.7_{(81.1)}$	$71.6_{(75.6)}$
Task Arithmetic [34]	$70.6_{(76.3)}$	$65.3_{(72.0)}$	$60.5_{(66.7)}$	$75.9_{(74.4)}$	$70.5_{(75.9)}$	$65.8_{(70.7)}$	$85.0_{(88.6)}$	$79.4_{(83.9)}$	$74.1_{(78.1)}$
TIES-Merging [75]	$74.5_{(80.3)}$	$65.1_{(71.9)}$	$62.3_{(68.8)}$	$79.4_{(83.8)}$	$70.1_{(75.5)}$	$66.6_{(71.7)}$	$85.8_{(89.5)}$	$77.3_{(81.6)}$	$72.9_{(76.9)}$
Consensus TA [71]	$75.0_{(80.8)}$	$70.4_{(77.4)}$	$65.4_{(72.0)}$	$79.4_{(83.9)}$	$74.4_{(79.9)}$	$69.8_{(74.9)}$	$86.3_{(90.1)}$	$82.2_{(86.9)}$	$79.0_{(83.2)}$
AdaMerging [76]	$75.9_{(81.7)}$	$70.5_{(77.3)}$	$66.2_{(72.7)}$	$79.5_{(83.8)}$	$73.8_{(79.2)}$	$69.6_{(74.6)}$	$87.4_{(94.6)}$	$82.6_{(87.2)}$	$77.8_{(82.0)}$
(Dynamic model merging)									
Twin-Merging [47]	$84.0_{(90.3)}$	$70.0_{(76.7)}$	$57.5_{(61.8)}$	$91.4_{(96.2)}$	$78.4_{(83.9)}$	$63.1_{(67.0)}$	$93.7_{(97.7)}$	$86.2_{(91.2)}$	$74.8_{(78.6)}$
DaWin [53]	$89.0_{(95.3)}$	$73.8_{(80.3)}$	$52.8_{(57.7)}$	$87.1_{(91.9)}$	$77.8_{(83.5)}$	$62.8_{(67.3)}$	$91.6_{(95.5)}$	$82.6_{(87.2)}$	$77.5_{(81.8)}$
MoW-Merging [78]	$88.1_{(94.8)}$	$83.2_{(91.5)}$	$79.3_{(86.8)}$	$93.7_{(98.9)}$	$79.3_{(85.5)}$	$78.2_{(84.3)}$	$94.9_{(99.0)}$	$78.8_{(83.6)}$	$81.8_{(86.3)}$
WEMoE [68]	$90.4_{(97.3)}$	$83.1_{(90.7)}$	$74.4_{(81.2)}$	$93.1_{(98.2)}$	$85.6_{(92.9)}$	$78.8_{(84.1)}$	$94.8_{(98.8)}$	$87.0_{(91.6)}$	$75.7_{(79.5)}$
ReTeX (Ours)	$\mathbf{92.2}_{(99.3)}$	$\mathbf{89.8}_{(98.8)}$	$\mathbf{89.8}_{(98.3)}$	$\mathbf{94.2}_{(99.4)}$	$\mathbf{91.9}_{(99.0)}$	$\mathbf{91.7}_{(98.4)}$	$\mathbf{95.4}_{(99.5)}$	$\mathbf{93.5}_{(99.1)}$	$\mathbf{93.6}_{(98.9)}$

that ReTeX can recover high-fidelity experts even from a simple static merge. We train the task-conditioned offset predictor for 5000 iterations using Adam [39] with learning rate 2×10^{-4} and a cosine schedule with 600 warm-up steps. We note that the training of the offset predictor in ReTeX does not require original task datasets of input-output examples. ReTeX training requires parameter supervision only: given any task ID, it predicts a task-conditioned parameter offset to approximate the task expert parameters, which are used as ground-truth.

5.1 Vision tasks

Setting. We follow the multi-task CLIP merging protocol in [71]. We fine-tune a separate expert per dataset on three CLIP [58] backbones (ViT-B/32, ViT-B/16, ViT-L/14), and evaluate merged models under task-agnostic inference. The 8-task suite includes SUN397 [74], Cars [40], RESISC45 [14], EuroSAT [31], SVHN [51], GTSRB [65], MNIST [25], and DTD [19]. The 14-task suite further adds CIFAR100 [41], STL10 [21], Flowers102 [52], Oxford-IIIT-Pet [55], PCAM [70], and FER2013 [29]. The 20-task suite further adds EMNIST [22], CIFAR10 [41], Food101 [12], FashionMNIST [73], RenderedSST2 [63], and KMNIST [20]. To study scalability in the number of tasks, we additionally evaluate a 30-task suite on ViT-B/32 by augmenting the 20-task suite with ten more datasets: Vegetables [2], Kvasir-v2 [57], Intel Images [9], Weather [72], Cats and dogs [23], MangoLeafBD [3], Beans [42], Landscape Recognition [24], Garbage Classification [13], and Fruits-360 [50]. We report classification accuracy (%) on all datasets.

Results on 8/14/20 tasks. Tab. 1 reports multi-task accuracy across three CLIP backbones and 8/14/20 task suites. Static merging baselines degrade substantially as the number of tasks increases, consistent with parameter inter-

Table 2: Multi-task performance on ViT-B/32 across different numbers of vision tasks. Values in parentheses $_{(\cdot)}$ indicate normalized accuracy (merged / individual).

Method	8 tasks	14 tasks	20 tasks	30 tasks
Zero-shot	48.3	57.2	56.1	55.5
Individual	92.9	90.9	91.4	93.1
(Static model merging)				
Weight Averaging	$66.3_{(72.1)}$	$64.3_{(71.1)}$	$61.0_{(67.5)}$	$59.1_{(64.2)}$
Task Arithmetic [34]	$70.6_{(76.3)}$	$65.3_{(72.0)}$	$60.5_{(66.7)}$	$58.0_{(62.8)}$
TIES-Merging [75]	$74.5_{(80.3)}$	$65.1_{(71.9)}$	$62.3_{(68.8)}$	$59.6_{(64.7)}$
Consensus TA [71]	$75.0_{(80.8)}$	$70.4_{(77.4)}$	$65.4_{(72.0)}$	$63.4_{(68.5)}$
AdaMerging [76]	$75.9_{(81.7)}$	$70.5_{(77.3)}$	$66.2_{(72.7)}$	$56.8_{(61.3)}$
(Dynamic model merging)				
Twin-Merging [47]	$84.0_{(90.3)}$	$70.0_{(76.7)}$	$57.5_{(61.8)}$	$60.1_{(65.2)}$
DaWin [53]	$89.0_{(95.3)}$	$73.8_{(80.3)}$	$52.8_{(57.7)}$	$40.3_{(42.9)}$
MoW-Merging [78]	$88.1_{(94.8)}$	$83.2_{(91.5)}$	$79.3_{(86.8)}$	$56.4_{(60.6)}$
WEMoE [68]	$90.4_{(97.3)}$	$83.1_{(90.7)}$	$74.4_{(81.2)}$	$67.1_{(72.4)}$
ReTeX (Ours)	$\mathbf{92.0}_{(99.1)}$	$\mathbf{89.8}_{(98.8)}$	$\mathbf{89.4}_{(97.9)}$	$\mathbf{91.2}_{(97.9)}$

ference under a single fixed checkpoint. Dynamic merging baselines mitigate interference via input-adaptive composition, but still exhibit degradation under larger suites and require inference-time access to task-specific components. Across all backbones and task counts, ReTeX achieves the strongest performance and closely matches Individual experts, reaching around 98–99% normalized accuracy while using a single merged checkpoint plus lightweight recovery parameters.

Results on 30 tasks. Tab. 2 shows that performance of both static and dynamic merging baselines deteriorates as the task suite scales to 30 tasks. In contrast, ReTeX remains stable and continues to closely match Individual experts, indicating that offset-based expert recovery scales favorably to larger and more diverse task collections.

5.2 Unseen-task generalization robustness

Setting. This section evaluates whether ReTeX maintains performance when evaluation inputs include tasks that are not used to build the merged checkpoint. Following the protocol in AdaMerging [76], the evaluation uses eight vision

Table 3: Evaluation under OOD scenarios.

Method	Seen Task Accuracy	Unseen Task Accuracy
Weight Averaging	63.7	57.8
Weight Averaging + ReTeX	**90.4**	**63.4**
Task Arithmetic [34]	72.2	58.5
Task Arithmetic + ReTeX	**90.4**	**66.0**
TIES-Merging [75]	75.7	57.9
TIES-Merging + ReTeX	**89.8**	**68.7**
AdaMerging [76]	76.6	66.8
AdaMerging + ReTeX	**90.2**	**69.1**

datasets. MNIST [25] and EuroSAT [31] serve as unseen tasks, while the remaining six datasets (Cars [40], DTD [19], GTSRB [65], RESISC45 [14], SUN397 [74], and SVHN [51]) serve as seen tasks. The merged checkpoint construction and ReTeX recovery training use only seen-task experts, and no unseen-task expert checkpoint is available. Appendix Sec. C provides further details.

Soft recovery beyond one-hot task selection. The task identifier in Sec. 4.1 produces a hard task ID by nearest-subspace matching. For unseen tasks, a hard assignment to a single seen task becomes a brittle approximation. To enable interpolation across seen experts, this section replaces the one-hot task indicator with a soft mixing vector over seen tasks. Concretely, projection residuals $\{\mathcal{R}_t(\mathbf{z})\}_{t=1}^{T}$ are converted into mixture weights $\mathbf{p}(\mathbf{z}) \in \mathbb{R}^T$ (for example, by applying a softmax to the negative residuals), and ReTeX conditions offset prediction on $\mathbf{p}(\mathbf{z})$ instead of a one-hot task vector. To support this behavior without using any task data, recovery training uses soft targets in parameter space: a mixing vector is sampled from a Dirichlet distribution and defines a convex combination of seen experts as the supervision target.

Results. Tab. 3 reports performance when ReTeX is applied on top of multiple merged checkpoints. ReTeX consistently improves accuracy on both seen tasks and unseen tasks across all merging baselines. Seen-task accuracy increases to around 90% for every merged model. Unseen-task accuracy also improves for every baseline, and the best unseen-task performance is obtained by combining ReTeX with AdaMerging [76], reaching 69.1%. These results support the claim in the introduction that offset-based recovery enables interpolation over seen expertise and improves unseen-task generalization robustness.

5.3 NLP tasks

Setting. This section evaluates ReTeX on a multi-task NLP suite using T5-large as a common pre-trained backbone. A separate task expert is obtained by fine-tuning the backbone on each task, and merged methods are evaluated under task-agnostic inference. The 7-task scenario comprises: (1) QASC [38], (2) WikiQA [77], (3) QuaRTz [66] for question answering, (4) PAWS [80] for paraphrase

Table 4: Multi-task performance of merged T5-large across seven NLP tasks. Numbers in parentheses $_{(\cdot)}$ denote normalized accuracy (merged / fine-tuned). Bold values indicate the best performance among merged methods (excluding fine-tuned).

Method	PAWS	QASC	QuaRTz	StoryCloze	WikiQA	Winogrande	WSC	Avg.
Fine-tuned	94.4	98.9	87.8	90.8	96.0	74.7	79.2	88.8
(Static model merging)								
Weight Averaging	$61.3_{(64.9)}$	$82.6_{(83.5)}$	$70.5_{(80.3)}$	$53.7_{(59.1)}$	$63.2_{(65.8)}$	$49.7_{(66.5)}$	$36.1_{(45.6)}$	$59.6_{(67.1)}$
Task Arithmetic [34]	$77.8_{(82.4)}$	$96.0_{(97.1)}$	$78.6_{(89.5)}$	$86.4_{(95.2)}$	$59.1_{(61.6)}$	$62.3_{(83.4)}$	$52.8_{(66.7)}$	$73.3_{(82.5)}$
TIES-Merging [75]	$81.5_{(86.3)}$	$96.2_{(97.3)}$	$80.1_{(91.2)}$	$83.6_{(92.1)}$	$64.9_{(67.6)}$	$66.5_{(89.0)}$	$65.3_{(82.4)}$	$76.9_{(86.6)}$
Consensus TA [71]	$77.7_{(82.3)}$	$67.4_{(68.1)}$	$65.5_{(74.6)}$	$79.9_{(88.0)}$	$90.3_{(94.1)}$	$78.9_{(105.6)}$	$56.0_{(70.7)}$	$73.7_{(83.0)}$
AdaMerging [76]	$58.5_{(62.0)}$	$76.0_{(76.8)}$	$75.5_{(86.0)}$	$82.1_{(90.4)}$	$93.5_{(97.4)}$	$65.0_{(87.0)}$	$64.4_{(81.3)}$	$73.6_{(82.9)}$
(Dynamic model merging)								
Twin-Merging [47]	$93.2_{(98.7)}$	$96.0_{(97.1)}$	$87.1_{(99.2)}$	$85.6_{(94.3)}$	$77.1_{(80.3)}$	$70.8_{(94.8)}$	$69.7_{(88.0)}$	$82.8_{(93.2)}$
DaWin [53]	$85.6_{(90.7)}$	$96.1_{(97.2)}$	$81.4_{(92.7)}$	$73.2_{(80.6)}$	$70.2_{(73.1)}$	$68.7_{(92.0)}$	$57.9_{(73.1)}$	$76.2_{(85.8)}$
WEMoE [68]	$93.4_{(98.9)}$	$95.4_{(96.5)}$	$74.5_{(84.9)}$	$79.6_{(87.7)}$	$94.1_{(98.0)}$	$90.4_{(121.0)}$	$57.0_{(72.0)}$	$83.4_{(93.9)}$
MoW-Merging [78]	$94.7_{(100.3)}$	$90.2_{(91.2)}$	$79.0_{(90.0)}$	$83.1_{(91.5)}$	$56.6_{(59.0)}$	$93.8_{(125.6)}$	$51.0_{(64.4)}$	$78.3_{(88.2)}$
ReTeX (Ours)	$\mathbf{96.1}_{(101.8)}$	$\mathbf{97.4}_{(98.5)}$	$\mathbf{83.5}_{(95.1)}$	$\mathbf{90.3}_{(99.4)}$	$\mathbf{95.7}_{(99.7)}$	$\mathbf{95.0}_{(127.2)}$	$\mathbf{48.0}_{(60.6)}$	$\mathbf{86.6}_{(97.5)}$

identification, (5) Story Cloze [62] for sentence completion, (6) Winogrande [61], and (7) WSC [44] for coreference resolution. All numbers are reported as percentages, and values in parentheses denote normalized accuracy (merged / fine-tuned), similar to Tab. 4.

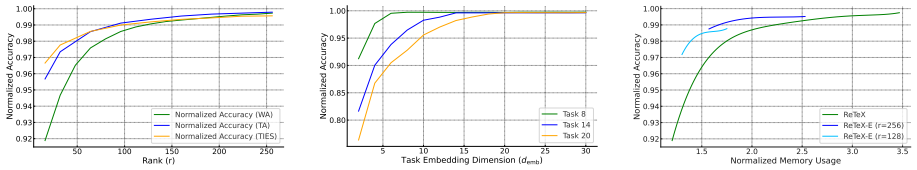
Results. Tab. 4 reports multi-task performance of merged T5-large across the seven NLP tasks. Static model merging baselines, including Task Arithmetic [34], TIES-Merging [75], and TSV-M [28], show clear degradation relative to fine-tuned experts under multi-task interference. Dynamic model merging baselines, including Twin-Merging [47], DaWin [53], WEMoE [68], and MoW-Merging [78], improve over static merging by using input-adaptive composition, but they still fall short of expert-level performance on average.

Across the full suite, ReTeX achieves the strongest average performance among merged methods, reaching 86.6 average accuracy and 97.5% normalized accuracy relative to fine-tuned experts. ReTeX matches or exceeds fine-tuned experts on several tasks such as PAWS and Winogrande, while WSC remains challenging and some baselines obtain higher scores on that task. Overall, the results indicate that ReTeX scales to multi-task NLP settings and recovers expert-level performance from a single merged model with high fidelity.

5.4 Ablation study

We conduct ablation studies for further analysis. Unless specified otherwise, this subsection follows the ViT-B/32 protocol from [71].

Integration of ReTeX with different base merged models. This ablation evaluates whether recovery quality depends on the base merged checkpoint. ReTeX is trained on top of Weight Averaging, Task Arithmetic [34], and TIES-Merging [75], while keeping the base merge rule fixed during recovery training.



(a) Effect of the base merged model across different ranks r.

(b) Effect of task embedding dimension d_{emb} for 8/14/20 tasks.

(c) Memory–accuracy trade-off of ReTeX and ReTeX-E at outer rank r.

Fig. 3: Performance and efficiency trade-offs of ReTeX. (a) Normalized accuracy across ranks r for three base merged models (WA, TA, TIES). (b) Normalized accuracy as a function of task embedding dimension d_{emb} for 8, 14, and 20 tasks with ViT-B/32 and fixed r=256. (c) Normalized accuracy vs. normalized memory for ReTeX and ReTeX-E at fixed outer rank r while varying r_g. Normalized memory is computed as the storage of recovery parameters divided by the storage of all task experts.

Fig. 3a shows that using a stronger base merge improves normalized accuracy, with larger gains at smaller recovery ranks r. This indicates that expert recovery benefits from a better initialization in weight space, especially under tight low-rank budgets.

Task embedding dimension. This ablation studies the effect of task embedding dimension d_{emb} on recovery. We investigate the influence of the task embedding dimension, d_{emb}, on the recovery performance of ReTeX. For this analysis, we fix the rank r=256 and vary d_{emb} while evaluating on configurations with 8, 14, and 20 tasks. Fig. 3b shows that normalized accuracy increases with d_{emb} and then saturates. High recovery is achieved once d_{emb} is comparable to the number of tasks, and increasing d_{emb} beyond this point yields diminishing returns.

5.5 Efficient low-rank offset factorization

Two-stage offset generation. ReTeX uses a low-rank offset per layer, $\boldsymbol{\beta}_t^{(l)} = \boldsymbol{\beta}_{t,g}^{(l)}\boldsymbol{\beta}_s^{(l)}$, where $\boldsymbol{\beta}_{t,g}^{(l)} \in \mathbb{R}^{a\times r}$ is task-conditioned and $\boldsymbol{\beta}_s^{(l)} \in \mathbb{R}^{r\times b}$ is shared. The dominant parameter cost comes from producing $\boldsymbol{\beta}_{t,g}^{(l)}$.

ReTeX-E reduces this cost by factorizing the generator output: $\boldsymbol{\beta}_{t,g}^{(l)} = \boldsymbol{\beta}_{t,gA}^{(l)}\boldsymbol{\beta}_{gB}^{(l)}$ with $\boldsymbol{\beta}_{t,gA}^{(l)} \in \mathbb{R}^{a\times r_g}$, $\boldsymbol{\beta}_{gB}^{(l)} \in \mathbb{R}^{r_g\times r}$, and $r_g < r$. The final offset becomes $\boldsymbol{\beta}_t^{(l)} = (\boldsymbol{\beta}_{t,gA}^{(l)}\boldsymbol{\beta}_{gB}^{(l)})\boldsymbol{\beta}_s^{(l)}$.

Results. Fig. 3c fixes the outer rank r and varies r_g. ReTeX-E improves the memory–accuracy trade-off and maintains high recovery quality over a wide range of r_g. The corresponding inference memory usage is reported in Tab. 5.

Table 5: Inference cost with CLIP ViT-B/32. Per-input inference latency and peak VRAM usage in the 8-task vision setting with task-ID unknown, together with average accuracy (Avg.) on ViT-B/32, measured on an NVIDIA GeForce RTX 3090.

Method	Inference cost (per input)	VRAM (GB)	Avg.
(Static model merging)			
Weight Averaging	0.0004s	1.3	66.3
Task Arithmetic [34]	0.0004s	1.3	70.8
TIES-Merging [75]	0.0004s	1.3	75.1
Consensus TA [71]	0.0004s	1.3	75.0
(Dynamic model merging)			
Twin-Merging [47]	0.03s	3.2	84.0
DaWin [53]	0.63s	5.5	89.0
WEMoE [68]	0.02s	4.3	90.4
MoW-Merging [78]	0.008s	4.9	88.1
(Component selection)			
ID-LoRA [32]	0.0009s	1.9	89.3
ID-EMR-Merging [33]	0.004s	1.9	90.5
ReTeX (Sample-wise)	0.05s	2.5	92.2
ReTeX (Group: $B = 64$)	0.0015s	2.7	92.2
ReTeX-E (Sample-wise)	0.07s	1.8	92.0
ReTeX-E (Group: $B = 64$)	0.0015s	2.0	92.0

5.6 Computational cost

Batch inference. ReTeX supports efficient batched execution by grouping samples that share the same predicted task ID $\hat{t}$ (Sec. 4.1). Given a batch of B test inputs, task predictions $\{\hat{t}_i\}_{i=1}^{B}$ are computed and then: (i) partition indices by task ID, $\mathcal{I}_t = \{i \in \{1, \ldots, B\} : \hat{t}_i = t\}$, and for each t with $|\mathcal{I}_t| > 0$ recover the corresponding expert once from the single merged checkpoint; (ii) run each sub-batch $X_{\mathcal{I}_t}$ through the recovered expert and scatter outputs back to the original order. This reduces the number of recoveries to the number of unique task IDs in the batch, $K \leq \min(B, T)$.

Component selection. Component-based multi-task deployment stores a shared backbone together with task-specific components and selects components conditioned on task identity. To test whether task identification plus component selection is sufficient for a task-unknown stream, per-task LoRA adapters and EMR-Merging components are evaluated by selecting components using predicted task IDs from Sec. 4.1. For a fair latency comparison, ID-LoRA and ID-EMR-Merging are also evaluated under the same grouped inference protocol as ReTeX. After predicting task IDs, samples with the same predicted task ID are grouped and processed together with batch size $B = 64$. The reported latency is the per-input inference cost, obtained by normalizing the measured grouped latency by the batch size. In Tab. 5, these baselines use the prefix ID-.

This comparison evaluates an alternative deployment path that combines task identification with compact components, and clarifies when expert recovery provides additional value beyond component selection.

Results. Tab. 5 reports latency and peak VRAM in the 8-task vision setting without task IDs with CLIP ViT-B/32. Latency includes task identification, offset generation, and the forward pass with the recovered parameters. Static merging baselines are fastest (0.0004s per input) with low VRAM (1.3GB), but accuracy remains low (66.3–75.1 Avg.). Dynamic merging baselines improve accuracy (84.0–90.4) but incur higher VRAM (4.3–5.6GB) and additional latency (0.008–0.63s) due to inference-time access to task components.

Component selection baselines reduce VRAM to 1.9GB and run quickly (0.0009–0.004s), but accuracy remains below ReTeX (89.3–90.5 vs. 92.2 Avg.) and deployment still depends on storing and selecting task-specific components.

ReTeX achieves the best accuracy (92.2 Avg.) with 2.5GB VRAM in the sample-wise setting. With grouped inference at B=64, ReTeX reduces latency to 0.0015s per input while maintaining accuracy, with a small VRAM increase to 2.7GB. ReTeX-E further reduces VRAM (1.8GB sample-wise; 2.0GB grouped) with minor accuracy change.

6 Conclusion

In this work, instead of storing task experts during inference, we aim to recover task experts directly from a merged model, retrieving task-expert-level performance while removing the need for storing experts during inference. We note that the task-specific offset between the task-expert parameters and the merged model parameters can be recovered from the merged model. Building upon this, we introduce a new model merging approach that learns to **Re**cover **T**ask **eX**perts (**ReTeX**) from a merged model by predicting these offsets. Particularly, our framework first estimates the task identity for a given input. Conditioned on the estimated task identity, our framework generates a low-rank, task-dependent offset that maps the merged parameters to the corresponding expert for that input. Experimental results across vision and NLP domains highlight the effectiveness of ReTeX in recovering task-expert-level performance while reducing memory overhead, compared to previous input-adaptive merging methods. We hope that this work encourages future research on the relationship between a merged model and task-specific models.

Acknowledgements

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant (No. RS-2025-25422680, No. RS-2020-II201373, Artificial Intelligence Graduate School Program (Hanyang University)) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2025-24533064).

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Ahmed, M.I., Mamun, S.M., Asif, A.U.Z.: Dcnn-based vegetable image classification using transfer learning: A comparative study. In: ICCCSP (2021)
3. Ahmed, S.I., Ibrahim, M., Nadim, M., Rahman, M.M., Shejunti, M.M., Jabid, T., Ali, M.S.: Mangoleafbd: A comprehensive image dataset to classify diseased and healthy mango leaves. Data in Brief (2023)
4. Baik, S., Choi, J., Kim, H., Cho, D., Min, J., Lee, K.M.: Meta-learning with task-adaptive loss function for few-shot learning. In: ICCV (2021)
5. Baik, S., Choi, M., Choi, J., Kim, H., Lee, K.M.: Meta-learning with adaptive hyperparameters. In: NeurIPS (2020)
6. Baik, S., Choi, M., Choi, J., Kim, H., Lee, K.M.: Learning to learn task-adaptive hyperparameters for few-shot learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
7. Baik, S., Hong, S., Lee, K.M.: Learning to forget for meta-learning. In: CVPR (2020)
8. Baik, S., Hong, S., Lee, K.M.: Learning to forget for meta-learning via task-and-layer-wise attenuation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)
9. Bansal, P.: Intel image classification. Available on https://www. kaggle. com/puneet6060/intel-image-classification, Online (2019), accessed: 2026-06-30
10. Bateni, P., Barber, J., Van de Meent, J.W., Wood, F.: Enhancing few-shot image classification with unlabelled examples. In: WACV (2022)
11. Bateni, P., Goyal, R., Masrani, V., Wood, F., Sigal, L.: Improved few-shot visual classification. In: CVPR (2020)
12. Bossard, L., Guillaumin, M., Gool, L.V.: Food-101 – mining discriminative components with random forests. In: ECCV (2014)
13. CCHANG: Garbage classification. https://www.kaggle.com/ds/81794 (2018). https://doi.org/10.34740/KAGGLE/DS/81794, accessed: 2026-06-30
14. Cheng, G., Han, J., Lu, X.: Remote sensing image scene classification: Benchmark and state of the art. Proceedings of the IEEE (2017)
15. Chi, Y., Porikli, F.: Connecting the dots in multi-class classification: From nearest subspace to collaborative representation. In: CVPR. pp. 3602–3609 (2012)
16. Choi, J., Baik, S., Choi, M., Kwon, J., Lee, K.M.: Visual tracking by adaptive continual meta-learning. IEEE Access (2022)
17. Choi, M., Choi, J., Baik, S., Kim, T.H., Lee, K.M.: Scene-adaptive video frame interpolation via meta-learning. In: CVPR (2020)
18. Choi, M., Choi, J., Baik, S., Kim, T.H., Lee, K.M.: Test-time adaptation for video frame interpolation via meta-learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)
19. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: CVPR (2014)
20. Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., Ha, D.: Deep learning for classical japanese literature. In: NeurIPS (2018)
21. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: AISTATS (2011)

22. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: Emnist: Extending mnist to handwritten letters. In: IJCNN (2017)
23. Cukierski, W.: Dogs vs. cats, 2013. URL https://kaggle.com/competitions/dogs-vs-cats Accessed: 2026-06-30
24. DeepNets: Landscape recognition. https://www.kaggle.com/datasets/utkarshsaxenadn/landscape-recognition-image-dataset-12k-images, accessed: 2026-06-30
25. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine (2012)
26. Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.M., Chen, W., et al.: Parameter-efficient fine-tuning of large-scale pre-trained language models. Nature Machine Intelligence (2023)
27. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICLR (2017)
28. Gargiulo, A.A., Crisostomi, D., Bucarelli, M.S., Scardapane, S., Silvestri, F., Rodolà, E.: Task singular vectors: Reducing task interference in model merging. In: CVPR (2025)
29. Goodfellow, I.J., Erhan, D., Carrier, P.L., Courville, A., Mirza, M., Hamner, B., Cukierski, W., Tang, Y., Thaler, D., Lee, D.H., Zhou, Y., Ramaiah, C., Feng, F., Li, R., Wang, X., Athanasakis, D., Shawe-Taylor, J., Milakov, M., Park, J., Ionescu, R., Popescu, M., Grozea, C., Bergstra, J., Xie, J., Romaszko, L., Xu, B., Chuang, Z., Bengio, Y.: Challenges in representation learning: A report on three machine learning contests. In: ICONIP (2013)
30. Ha, D., Dai, A.M., Le, Q.V.: Hypernetworks. In: ICLR (2017)
31. 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 (2019)
32. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. In: ICLR (2022)
33. Huang, C., Ye, P., Chen, T., He, T., Yue, X., Ouyang, W.: Emr-merging: Tuning-free high-performance model merging. In: NeurIPS (2024)
34. Ilharco, G., Ribeiro, M.T., Wortsman, M., Schmidt, L., Hajishirzi, H., Farhadi, A.: Editing models with task arithmetic. In: ICLR (2023)
35. Iurada, L., Ciccone, M., Tommasi, T.: Efficient model editing with task-localized sparse fine-tuning. In: ICLR (2025)
36. Jin, X., Ren, X., Preotiuc-Pietro, D., Cheng, P.: Dataless knowledge fusion by merging weights of language models. In: ICLR (2023)
37. Kang, J., Karlinsky, L., Luo, H., Wang, Z., Hansen, J., Glass, J., Cox, D., Panda, R., Feris, R., Ritter, A.: Self-moe: Towards compositional large language models with self-specialized experts. In: ICLR (2025)
38. Khot, T., Clark, P., Guerquin, M., Jansen, P., Sabharwal, A.: Qasc: A dataset for question answering via sentence composition. In: AAAI (2020)
39. Kingma, D.P.: Adam: A method for stochastic optimization. In: ICLR (2015)
40. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: ICCV Workshop (2013)
41. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
42. Lab, M.A.: Bean disease dataset (2020), https://github.com/AI-Lab-Makerere/ibean/, accessed: 2026-06-30
43. Lee, Y., Jung, J., Baik, S.: Mitigating parameter interference in model merging via sharpness-aware fine-tuning. In: ICLR (2025)

44. Levesque, H., Davis, E., Morgenstern, L.: The winograd schema challenge. In: KR (2012)
45. Li, P., Zhang, Z., Yadav, P., Sung, Y.L., Cheng, Y., Bansal, M., Chen, T.: Merge, then compress: Demystify efficient smoe with hints from its routing policy. In: ICLR (2024)
46. Li, W.H., Liu, X., Bilen, H.: Cross-domain few-shot learning with task-specific adapters. In: CVPR (2022)
47. Lu, Z., Fan, C., Wei, W., Qu, X., Chen, D., Cheng, Y.: Twin-merging: Dynamic integration of modular expertise in model merging. In: NeurIPS (2024)
48. Matena, Michael S. and Raffel, C.A.: Merging models with fisher-weighted averaging. In: NeurIPS (2022)
49. Muqeeth, M., Liu, H., Raffel, C.: Soft merging of experts with adaptive routing. Transactions on Machine Learning Research (2023)
50. Muresan, H., Oltean, M.: Fruit recognition from images using deep learning. Acta Universitatis Sapientiae, Informatica (2018)
51. 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: NIPS Workshop (2011)
52. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: ICVGIP (2008)
53. Oh, C., Li, Y., Song, K., Yun, S., Han, D.: Dawin: Training-free dynamic weight interpolation for robust adaptation. In: ICLR (2025)
54. Ortiz-Jimenez, G., Favero, A., Frossard, P.: Task arithmetic in the tangent space: Improved editing of pre-trained models. In: NeurIPS (2023)
55. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Cats and dogs. In: CVPR (2012)
56. Perez, E., Strub, F., De Vries, H., Dumoulin, V., Courville, A.: Film: Visual reasoning with a general conditioning layer. In: AAAI (2018)
57. Pogorelov, K., Randel, K.R., Griwodz, C., Eskeland, S.L., de Lange, T., Johansen, D., Spampinato, C., Dang-Nguyen, D.T., Lux, M., Schmidt, P.T., et al.: Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection. In: ACM MMSys (2017)
58. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: ICML (2021)
59. Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., Turner, R.E.: Fast and flexible multi-task classification using conditional neural adaptive processes. In: NeurIPS (2019)
60. Saab, K., Tu, T., Weng, W.H., Tanno, R., Stutz, D., Wulczyn, E., Zhang, F., Strother, T., Park, C., Vedadi, E., et al.: Capabilities of gemini models in medicine. arXiv preprint arXiv:2404.18416 (2024)
61. Sakaguchi, K., Bras, R.L., Bhagavatula, C., Choi, Y.: Winogrande: An adversarial winograd schema challenge at scale. In: AAAI (2020)
62. Sharma, R., Allen, J., Bakhshandeh, O., Mostafazadeh, N.: Tackling the story ending biases in the story cloze test. In: ACL (2018)
63. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: EMNLP (2013)
64. Son, J., Jung, J., Baik, S.: Training-free task classification for multi-task model merging. arXiv preprint arXiv:2606.22589 (2026)

65. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The german traffic sign recognition benchmark: a multi-class classification competition. In: IJCNN (2011)
66. Tafjord, O., Gardner, M., Lin, K., Clark, P.: Quartz: An open-domain dataset of qualitative relationship questions. In: EMNLP (2019)
67. Tang, A., Shen, L., Luo, Y., Ding, L., Hu, H., Du, B., Tao, D.: Concrete subspace learning based interference elimination for multi-task model fusion. arXiv preprint arXiv:2312.06173 (2023)
68. Tang, A., Shen, L., Luo, Y., Yin, N., Zhang, L., Tao, D.: Merging multi-task models via weight-ensembling mixture of experts. In: ICML (2024)
69. Triantafillou, E., Larochelle, H., Zemel, R., Dumoulin, V.: Learning a universal template for few-shot dataset generalization. In: ICML (2021)
70. Veeling, B.S., Linmans, J., Winkens, J., Cohen, T., Welling, M.: Rotation equivariant cnns for digital pathology. In: MICCAI (2018)
71. Wang, K., Dimitriadis, N., Ortiz-Jimenez, G., Fleuret, F., Frossard, P.: Localizing task information for improved model merging and compression. In: ICML (2024)
72. Xiao, H., Zhang, F., Shen, Z., Wu, K., Zhang, J.: Classification of weather phenomenon from images by using deep convolutional neural network. Earth and Space Science (2021)
73. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
74. Xiao, J., Ehinger, K.A., Hays, J., Torralba, A., Oliva, A.: Sun database: Exploring a large collection of scene categories. International Journal of Computer Vision (2016)
75. Yadav, P., Tam, D., Choshen, L., Raffel, C., Bansal, M.: Ties-merging: Resolving interference when merging models. In: NeurIPS (2023)
76. Yang, E., Wang, Z., Shen, L., Liu, S., Guo, G., Wang, X., Tao, D.: Adamerging: Adaptive model merging for multi-task learning. In: ICLR (2024)
77. Yang, Y., tau Yih, W., Meek, C.: Wikiqa: A challenge dataset for open-domain question answering. In: ACL (2015)
78. Ye, P., Huang, C., Shen, M., Chen, T., Huang, Y., Ouyang, W.: Dynamic model merging with mixture of weights. IEEE Transactions on Circuits and Systems for Video Technology (2025)
79. 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: ICML (2024)
80. Zhang, Y., Baldridge, J., He, L.: Paws: Paraphrase adversaries from word scrambling. In: NAACL (2019)