MASS: MoErging through Adaptive Subspace Selection

Anonymous authors
Paper under double-blind review

Abstract

Model merging has recently emerged as a lightweight alternative to ensembling, combining multiple fine-tuned models into a single set of parameters with no additional training overhead. Yet, existing merging methods fall short of matching the full accuracy of separately fine-tuned endpoints. We present `MASS` (*MoErging through Adaptive Subspace Selection*), a new approach that closes this gap by unifying multiple fine-tuned models while retaining near state-of-the-art performance across tasks. Building on the low-rank decomposition of per-task updates, `MASS` stores only the most salient singular components for each task and merges them into a shared model. At inference time, a non-parametric, data-free router identifies which subspace (or combination thereof) best explains an input's intermediate features and activates the corresponding task-specific block. This procedure is fully training-free and introduces only a two-pass inference overhead plus a $\sim 2\times$ storage factor compared to a single pretrained model, irrespective of the number of tasks. We evaluate `MASS` on open-vocabulary image classification using `ViT-B-16`, `ViT-B-32` and `ViT-L-14` for benchmarks of 8, 14 and 20 tasks respectively, establishing a new state-of-the-art. Most notably, `MASS` recovers up to $\sim 98\%$ of the average accuracy of individual fine-tuned models, making it a practical alternative to ensembling at a fraction of the storage cost.

1 Introduction

In the early days of deep learning, the default practice was to train models entirely from scratch. With the rise of massive pretrained networks, research pivoted toward fine-tuning these backbones for specialized tasks (Devlin et al., 2019; Tan et al., 2018; Yosinski et al., 2014; Hu et al., 2022; Radford et al., 2021). Nowadays, with the abundance of fine-tuned models on platforms like HuggingFace[1], we are witnessing a shift toward *no-tuning* methods that leverage both pretrained foundations and diverse fine-tuned endpoints. Among these, *model merging* (Singh & Jaggi, 2020; Ainsworth et al., 2023; Ilharco et al., 2023) has gained significant attention by combining multiple fine-tuned models into a single parameter set, eliminating the need for additional training or data.

An important application of model merging is the combination of models fine-tuned on different tasks that share the same pretrained backbone. Early approaches such as `Task Arithmetic` (Ilharco et al., 2023) and its extensions (Yadav et al., 2023; Yu et al., 2024; Daheim et al.; Wang et al., 2024; Ortiz-Jiménez et al., 2023; Huang et al., 2024; Akiba et al., 2025) define a task vector as the difference between pretrained and fine-tuned weights, and build a multitask model by summing these vectors to the base model. More recent work (Gargiulo et al., 2025; Daniel et al., 2025) demonstrates that preserving the layer-wise structure of these updates leads to stronger results. Instead of flattening the updates into vectors, methods such as `Task Singular Vectors` (TSV) (Gargiulo et al., 2025) exploit their matrix form and uncover a clear low-rank structure, where only a small number of singular vectors per task are needed to recover most of the fine-tuned accuracy.

Despite the progress in performance, most merging methods remain limited by an unrealistic assumption: that the task identity is known at inference time. We challenge this assumption as impractical and argue that under such a setting, the optimal strategy would be compression. Indeed, the low-rank compression method `TSV-C` (Gargiulo et al., 2025) achieves 99.5% normalized accuracy

[1] https://huggingface.co/docs/hub/models-the-hub

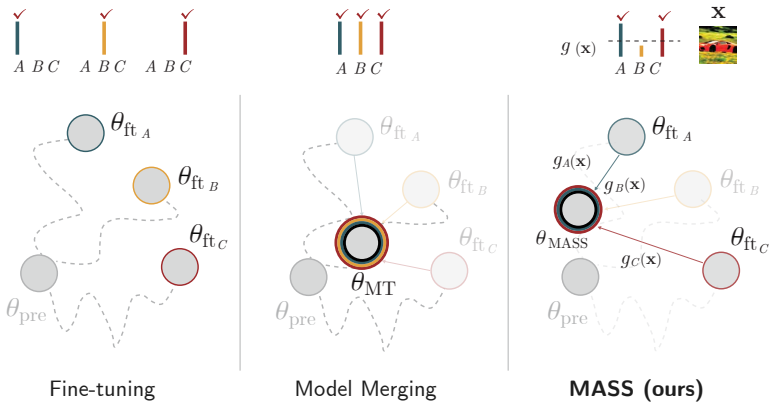


Figure 1: *(left)* Fine-tuning yields three separate models on tasks A, B and C. *(middle)* Model merging produces a single model incorporating the task vectors using a constant function of the input. *(right)* `MASS` stores the pretrained model θ_{pre} and the task singular vectors $V^\top$ across tasks. At test time, `MASS` adaptively performs merging using a routing mechanism that chooses appropriate task vectors for the input $\mathbf{x}$, using a thresholded gating function $g(\mathbf{x})$. The gate is the residual between the activations of $\mathbf{x}$ and their projections onto the span of the right singular vectors $V_\perp$.

across all benchmarks while requiring only $2\times$ storage and no additional inference overhead, effectively solving the problem. Mixture-of-Experts Merging (MoErging) (Yadav et al., 2024) methods such as `SMILE` (Tang et al., 2024a), `WeMoE` (Tang et al., 2024b), and `TwinMerging` (Lu et al., 2024) go a step further by incorporating a router into the merging process.

However, *these methods still assume that the correct classification head is provided at test time*, thereby inheriting the same unrealistic constraint and failing to exploit the full potential of routing-based approaches.

Key assumptions. In this work we assume that the task is *not* known at inference time, and that both the most suitable encoder subspaces and the corresponding classification head must be determined *automatically*. This setting is simultaneously more challenging and more realistic, enabling a single generalist model to handle all tasks encountered during fine-tuning without external supervision.

Contributions. We propose `MASS`, a novel MoErging method that dynamically selects the most relevant tasks and corresponding label spaces via an adaptive routing mechanism that conditions the merging process on the input itself (Fig. 1). `MASS` sidesteps the data and training required to train a router by leveraging a novel weight-space router that identifies the most relevant task subspaces as defined by their task singular vectors. By selectively integrating these subspaces into the pretrained backbone and extending the routing decisions to the classification heads, `MASS` enables a training-free and data-free merging process without relying on oracle knowledge of the task at hand.

We evaluate `MASS` on `ViT-B-32`, `ViT-B-16`, and `ViT-L-14` across up to 20 vision tasks and 8 language tasks, demonstrating substantial gains over existing methods. For a modest increase in computational cost ($\sim 2\times$ forward passes) and storage ($\sim 2\times$ parameter footprint regardless of the number of tasks), our method achieves up to 98% of the average accuracy of individual fine-tuned models while handling the full union of expert label spaces without oracle guidance at test time.

Wrapping up, our contribution is three-fold:

- We introduce `MASS`, a MoErging method that augments singular-vector-based merging with adaptive routing, eliminating the need for test-time task knowledge.
- We design a projection-based router that operates without task data or additional tuning, making it directly applicable to the merging setting.
- We present extensive experiments that establish new state-of-the-art results across benchmarks and further provide an interpretation of task singular vectors in text space.

We release our code, checkpoints, and all relevant logs for research purposes.

2 BACKGROUND

Task Vectors `Task Arithmetic` (TA) (Ilharco et al., 2023) defines task vectors capturing the specific differences in model weights for individual tasks. Formally, the weights θ_{MT} of a multi-task model for T tasks are computed by aggregating the task-specific weight differences, or task vectors, as $\theta_{\text{MT}} = \theta_{\text{pre}} + \alpha \sum_{i=1}^{T} \tau_i$, where θ_{pre} is the set of pretrained model weights, α is a scaling factor, and $\tau_i = \theta_{\text{ft}_i} - \theta_{\text{pre}}$ is the task vector for task i, with θ_{ft_i} being the fine-tuned weights for the task. Following Gargiulo et al. (2025), however, we consider these operations at the layer level. From a layer-wise perspective, task arithmetic can be rewritten as $\theta_{\text{MT}}^{(\ell)} = \theta_{\text{pre}}^{(\ell)} + \alpha \sum_{i=1}^{T} \Delta_i^{(\ell)}$, where $\theta_{\text{pre}}^{(\ell)}$ encodes the pretrained weights for layer ℓ, and $\Delta_i^{(\ell)} = \theta_{\text{ft}_i}^{(\ell)} - \theta_{\text{pre}}^{(\ell)}$ is the task-specific weight difference for task i at layer ℓ. When layer ℓ has a matrix structure, its corresponding $\Delta_i^{(\ell)}$ is called the *per-layer task matrix* for task i. The layer index will be omitted for brevity.

Task Singular Vectors Given a task i, Gargiulo et al. (2025) consider the SVD of the corresponding task matrices Δ_i on a generic layer $\Delta_i = U_i \Sigma_i V_i^\top$. They then perform a low-rank approximation of the Δs and orthogonalize their singular vectors to reduce inter-task interference. In practice, this is equivalent to summing the top-k rank-one matrices for each task, with an added orthogonalization step to prevent singular vectors belonging to different tasks from interfering. The full procedure is detailed in Algorithm 2 (lines 10–20).

3 APPROACH

Our approach is best understood as a pre-processing step followed by an inference-time step. The former consists of a one-time merging procedure to obtain an encoder model θ_{MT}, as detailed in Algorithm 2. We refer to this as the 'fixed' merging step, as it is performed only once and remains independent of the input. During inference, θ_{MT} is used in a dynamic process, outlined in Algorithm 1, consisting of 4 steps:

(i) **First pass**: forward the input through θ_{MT} and extract its embedding $\mathbf{z}_\ell$ at a chosen layer ℓ;

(ii) **Routing**: project $\mathbf{z}_\ell$ onto the task subspaces, selecting those having lowest projection error;

(iii) **Adaptive merge**: merge the selected task subspaces into Δ_{ada};

(iv) **Second pass**: classify the input using the final merged model $\theta_{\text{MT}} = \theta_{\text{pre}} + \alpha\Delta_{\text{ada}}$.

Algorithm 1 Adaptive Merging Step

Require: Pretrained model weights θ_{pre}, task-specific updates $\{\Delta_i\}_{i=1}^{T}$, fixed merged model θ_{MT}, top-k parameter k, threshold η, task-specific classification heads $\{h_i\}_{i=1}^{T}$, sample $\mathbf{x}$
Ensure: Predicted class c^*

$\mathbf{z}_\ell \leftarrow \text{ForwardPass}(\theta_{\text{MT}}, \mathbf{x})$ # first pass
for $i = 1, \ldots, T$ **do**
  $r_i \leftarrow \|\mathbf{z}_\ell - V_i V_i^\top \mathbf{z}_\ell\|_2$ # residual as per Section 3.2.1
end for
$w \leftarrow \text{softmax}(-r)$
$\Omega \leftarrow \{i : w_i \geq \eta\}$ # Select tasks above threshold
$\Omega \leftarrow \text{TopK}(\Omega, w, k)$ # Keep only top-k weighted tasks
Merge selected subspaces: $\Delta_{\text{ada}} \leftarrow \sum_{i \in \Omega} U_i \Sigma_i V_i^\top$
Compute adaptive model: $\theta_{\text{MASS}} \leftarrow \theta_{\text{pre}} + \alpha \Delta_{\text{ada}}$
Classification procedure
$\mathbf{z}_{L-1} \leftarrow \text{ForwardPass}(\theta_{\text{MASS}}, \mathbf{x})$ # Compute shared representation
$\mathbf{z}_i \leftarrow h_i(\mathbf{z}_{L-1})$ # Evaluate each head
$(i^\star, c^\star) \leftarrow \underset{(i,c) \in \Omega \times \{1,\ldots,C_i\}}{\arg\max}\ z_i[c]$ # Highest logit across heads
return $c^\star$

3.1 FIXED MERGING

We begin with a one-time merging step to produce a model capable of task discrimination. This model provides the intermediate activations used by the router in the first pass. For this, we use `TSV-M` (Gargiulo et al., 2025) due to its subspace-aware aggregation. Although one could alternatively rely on the pretrained base, Table 3 shows that it leads to lower task classification accuracy.

3.2 INTEGRATING ROUTING

We extend the aggregation step in Section 2 to include a routing mechanism. Given an input $\mathbf{x}$, the merged model can be adaptively determined by:

$$\theta_{\text{MT}} = \theta_{\text{pre}} + \alpha \sum_{i=1}^{T} \mathbb{1}_{[g_i(\mathbf{x})=1]}(\mathbf{x})\tau_i = \theta_{\text{pre}} + \alpha \sum_{i=1}^{T} \mathbb{1}_{[g_i(\mathbf{x})=1]}(\mathbf{x}) \sum_{j=1}^{k} \sigma_j^i u_j^i v_j^{i\top}, \tag{1}$$

where $g_i(\mathbf{x})$ is a per-task gating function that adaptively selects which task subspaces to activate, and subsequently merge, depending on the input at hand.

Traditionally, however, routers require either task-specific *data* for non-parametric procedures such as nearest-neighbor-based routing, or both data and *training* for parametric routers. This contrasts with the typical merging scenario, which does not assume access to the data used to train the endpoint models, as these could, in principle, be downloaded from any public model repository. We therefore introduce a completely ***data*- and *tuning*-free** approach.

3.2.1 PROJECTION-BASED ROUTING

Given an input intermediate representation $\mathbf{z}_\ell$ for a predetermined layer ℓ, we want to identify which task subspace (or set of subspaces) is the most relevant. Concretely, one way to do this is to compute the Euclidean residual of $\mathbf{z}_\ell$ after projecting onto $\text{span}(V_i^{(\ell)})$:

$$r_i \;=\; \left\| \mathbf{z}_\ell \;-\; \text{Proj}_{V_i^{(\ell)}}(\mathbf{z}_\ell) \right\|_2 \quad ,$$

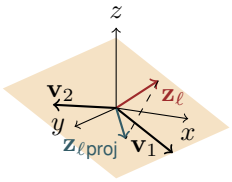


Figure 2: Projection of the activations $\mathbf{z}_\ell$ onto the span of TSVs $\mathbf{v}_1, \mathbf{v}_2$.

where $\text{Proj}_{V_i^{(\ell)}}(\mathbf{z}_\ell) = V_i^{(\ell)}\big(V_i^{(\ell)}\big)^\top \mathbf{z}_\ell$ is the optimal L_2 projector (see Proposition D.1). At this point, the additive inverse of the residuals is normalized through a softmax to obtain the coefficients. The router then picks those exceeding a predetermined threshold η, limiting the selection to the top-k when more tasks surpass it. For details regarding the choice of the layer used to compute the residual, see Section 4.2.

3.2.2 ACCOUNTING FOR REDUNDANT DIRECTIONS

For projection-based routing to be effective, no task should overshadow the others. Consider, for example, three tasks: `MNIST`, `EMNIST`, and `KMNIST`. Being trained on very similar datasets covering the same classes, Δ_{MN} and Δ_{EMN} share a large portion of their right-singular directions: $\text{span}\big(V_{\text{MN}}^{(\ell)}\big) \approx \text{span}\big(V_{\text{EMN}}^{(\ell)}\big)$, while `KMNIST` has some distinct directions $V_{\text{KM}}^{(\ell)}$ capturing more Japanese *kana*-like shapes. However, all three tasks may agree on certain generic "black background, white glyph" features. Because `MNIST` and `EMNIST` partially *reinforce* these directions (they both include them), the *union* of their subspaces can appear "wider" or more dominant in that region of feature space. Consequently, for many test samples $\mathbf{z}_\ell$ with black backgrounds and centered shapes:

$$\left\| \mathbf{z}_\ell - \text{Proj}_{V_{\text{MN}} \cup V_{\text{EMN}}}(\mathbf{z}_\ell) \right\|_2 < \left\| \mathbf{z}_\ell - \text{Proj}_{V_{\text{KM}}}(\mathbf{z}_\ell) \right\|_2.$$

Hence, the router sees a smaller residual for `MNIST`/`EMNIST`, declaring those tasks more suitable even if the glyph belongs to `KMNIST`.

During the fixed merging step, instead of aggregating all task matrices, we only keep those that are *sufficiently distinct*. We first select a single task matrix as the initial element of the merge set. Then, for each remaining task matrix, we determine whether to include it based on its similarity to the

matrices already in the set. A task matrix is added only if its similarity to all previously accepted matrices remains below a predefined threshold.

Formally, let $\{\Delta_{a_1}, \ldots, \Delta_{a_r}\}$ be the set of accepted updates at a given layer. When evaluating a new task update Δ_i, we flatten it as $\delta_i = \mathrm{vec}(\Delta_i)$ and compute its cosine similarity $\mathrm{sim}(\delta_i, \delta_{a_m})$ with each previously accepted update δ_{a_m}. If

$$\max_{1 \le m \le r} \mathrm{sim}(\delta_i, \delta_{a_m}) > \varepsilon\,,$$

where ε is a user-specified threshold (e.g. $\varepsilon = 0.3$), we discard Δ_i and do *not* merge it; else we accept it. This ensures that highly similar task subspaces do not overshadow less common ones. This procedure is performed before merging, see lines 2–8 of Algorithm 2.

3.3 Adaptive Merging and Inference

With the router having selected a subset Ω of relevant tasks, we merge their subspaces via `TSV-M` (Gargiulo et al., 2025) into a single model θ_{MASS}. Crucially, *unlike standard model merging, which assumes an oracle provides the correct task head at inference, we do not know the task in advance.* Instead, after obtaining the shared representation $\mathbf{z}_{L-1} \in \mathbb{R}^d$ from θ_{MASS}, we run each head $h_i : \mathbb{R}^d \to \mathbb{R}^{C_i}$ for every selected task $i \in \Omega$:

$$\mathbf{z}_i = h_i(\mathbf{z}_{L-1}), \quad \mathbf{z}_i \in \mathbb{R}^{C_i}.$$

We then pick the highest logit among all heads in Ω. Formally:

$$(i^\star, c^\star) = \underset{(i,c) \in \Omega \times \{1,\ldots,C_i\}}{\arg\max} z_i[c]\,.$$

In other words, we identify which head $i^\star$ is most "confident" and select its predicted class $c^\star$. *This procedure covers the unknown-task scenario*, allowing the model to determine both head and label space on a per-input basis.

3.4 Residual Minimization as Maximum A Posteriori Estimation

The task selection process in `MASS` can be viewed as a *maximum a posteriori* (MAP) estimation problem. If residuals follow an isotropic Gaussian, the likelihood of a feature vector given a task decays exponentially with its squared ℓ_2 reconstruction error. Thus, choosing the task with minimal residual is equivalent to the MAP estimate. This view parallels probabilistic PCA (Tipping & Bishop, 1999), where squared reconstruction error minimization corresponds to maximum likelihood estimation under the same Gaussian assumption. Residual-based selection is therefore statistically optimal under a simple, least-informative model that is particularly apt for `MASS`, which lacks training data to fit more complex distributions. The isotropic prior treats all directions equally, avoiding bias toward specific tasks.

Proposition 3.1. *Let $\mathbf{z}_\ell \in \mathbb{R}^d$ be a feature vector, and for each task i, decompose it as*

$$\mathbf{z}_\ell = V_i V_i^\top \mathbf{z}_\ell + \varepsilon_i, \qquad \varepsilon_i = \big(I - V_i V_i^\top\big)\mathbf{z}_\ell.$$

Assume $\varepsilon_i \sim \mathcal{N}(0, \sigma^2 I)$ and a uniform prior over tasks: $p(\text{task} = i) = \frac{1}{K}$ for all $i \in \{1, 2, \ldots, K\}$.. Then the maximum a posteriori estimate of the task reduces to

$$\hat{\imath}_{\text{MAP}} = \arg\max_i p(\text{task} = i \mid \mathbf{z}_\ell) = \arg\min_i \|\varepsilon_i\|_2^2.$$

4 Experiments

4.1 Merging performance

Models, baselines and datasets We conduct our experiments on three versions of the `CLIP` (Radford et al., 2021) model, each equipped with a different `ViT` (Dosovitskiy et al., 2021) visual encoder: `ViT-B-32`, `ViT-B-16`, and `ViT-L-14`. As baselines, we compare against multiple training-free merging strategies, notably weight averaging, `Task Arithmetic` (Ilharco et al., 2023), and `Consensus Merging` (Wang et al., 2024). For additional context, zero-shot performance serves as a null reference point, and the mean accuracy of individually fine-tuned models

	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
Base	Zeroshot	48.1$_{(54.8)}$	56.9$_{(64.5)}$	57.5$_{(65.2)}$	55.3$_{(60.6)}$	61.9$_{(67.9)}$	62.5$_{(68.3)}$	64.9$_{(69.2)}$	69.1$_{(73.8)}$	68.2$_{(72.7)}$
	Finetuned	90.3$_{(100)}$	89.0$_{(100)}$	89.5$_{(100)}$	92.4$_{(100)}$	91.3$_{(100)}$	91.9$_{(100)}$	94.2$_{(100)}$	93.4$_{(100)}$	94.0$_{(100)}$
Fixed	Weight Averaging	67.1$_{(74.6)}$	65.5$_{(73.8)}$	64.4$_{(72.6)}$	73.0$_{(78.8)}$	70.8$_{(77.3)}$	69.2$_{(75.3)}$	80.5$_{(85.2)}$	78.5$_{(83.8)}$	76.1$_{(80.9)}$
	Task Arithmetic	68.8$_{(75.7)}$	64.6$_{(72.5)}$	64.0$_{(71.9)}$	73.0$_{(78.3)}$	70.6$_{(77.0)}$	69.0$_{(75.0)}$	84.4$_{(89.3)}$	80.4$_{(85.8)}$	76.9$_{(81.7)}$
	Consensus TA	72.6$_{(80.1)}$	70.3$_{(78.9)}$	68.5$_{(76.8)}$	75.9$_{(81.7)}$	74.9$_{(81.7)}$	72.2$_{(78.4)}$	85.5$_{(90.5)}$	82.0$_{(87.6)}$	78.9$_{(83.8)}$
	TSV-M	83.2$_{(91.8)}$	78.6$_{(88.0)}$	75.6$_{(84.3)}$	85.5$_{(92.2)}$	81.4$_{(88.8)}$	78.8$_{(85.5)}$	91.2$_{(96.7)}$	88.8$_{(94.9)}$	87.5$_{(93.0)}$
	Iso-C	82.8$_{(91.7)}$	78.4$_{(88.0)}$	73.2$_{(81.9)}$	87.5$_{(94.4)}$	79.8$_{(87.0)}$	75.3$_{(81.6)}$	92.6$_{(98.2)}$	89.6$_{(95.8)}$	86.8$_{(92.3)}$
	Iso-CTS	82.0$_{(90.9)}$	80.6$_{(90.4)}$	77.0$_{(86.2)}$	88.7$_{(95.9)}$	84.1$_{(91.8)}$	80.7$_{(87.7)}$	92.8$_{(98.5)}$	**91.1$_{(97.4)}$**	89.2$_{(94.9)}$
MoE	WeMoE	**88.8$_{(97.5)}$**	74.3$_{(82.8)}$	68.2$_{(76.3)}$	89.1$_{(96.4)}$	76.6$_{(83.2)}$	65.0$_{(70.5)}$	88.7$_{(94.2)}$	72.3$_{(76.8)}$	65.0$_{(69.4)}$
	SMILE-1	83.2$_{(92.1)}$	75.4$_{(84.5)}$	72.8$_{(82.3)}$	87.8$_{(94.9)}$	81.7$_{(89.5)}$	79.5$_{(86.7)}$	91.2$_{(96.7)}$	86.6$_{(92.7)}$	84.9$_{(90.5)}$
	SMILE-2	84.4$_{(93.5)}$	76.4$_{(85.6)}$	74.1$_{(83.8)}$	89.0$_{(96.2)}$	82.7$_{(90.7)}$	80.4$_{(87.7)}$	92.0$_{(97.6)}$	87.1$_{(93.4)}$	85.5$_{(91.1)}$
	MASS	87.0$_{(96.5)}$	**82.9$_{(93.2)}$**	**81.1$_{(90.9)}$**	**90.6$_{(98.0)}$**	**87.8$_{(96.1)}$**	**81.1$_{(88.7)}$**	**92.9$_{(98.6)}$**	90.9$_{(97.3)}$	**90.8$_{(96.6)}$**

Table 1: Average absolute accuracy results on model merging benchmarks; subscript (in parentheses) is the normalized average accuracy.

Method	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg.
Finetuned	75.0$_{(100)}$	83.4$_{(100)}$	87.5$_{(100)}$	91.5$_{(100)}$	85.4$_{(100)}$	85.9$_{(100)}$	93.6$_{(100)}$	88.7$_{(100)}$	86.4$_{(100)}$
Weight Averaging	69.1$_{(92.1)}$	62.6$_{(75.1)}$	79.4$_{(90.7)}$	89.8$_{(98.1)}$	83.9$_{(98.2)}$	81.2$_{(94.5)}$	91.7$_{(97.9)}$	73.2$_{(82.5)}$	78.9$_{(91.3)}$
Task Arithmetic	70.5$_{(94.0)}$	57.8$_{(69.3)}$	78.4$_{(89.6)}$	90.2$_{(98.6)}$	83.6$_{(97.9)}$	80.5$_{(93.7)}$	92.3$_{(98.6)}$	77.8$_{(87.7)}$	78.9$_{(91.3)}$
Ties-Merging	70.3$_{(93.7)}$	65.0$_{(77.9)}$	78.9$_{(90.2)}$	90.2$_{(98.6)}$	83.5$_{(97.8)}$	81.6$_{(95.0)}$	91.7$_{(97.9)}$	78.3$_{(88.3)}$	79.9$_{(92.5)}$
WeMoE	72.5$_{(96.6)}$	79.0$_{(94.7)}$	51.9$_{(59.3)}$	89.3$_{(97.5)}$	69.6$_{(81.4)}$	81.5$_{(94.8)}$	88.6$_{(94.6)}$	82.1$_{(92.5)}$	76.8$_{(88.9)}$
SMILE-1	72.0$_{(96.0)}$	84.2$_{(101.0)}$	84.3$_{(96.3)}$	91.3$_{(99.8)}$	84.7$_{(99.2)}$	84.1$_{(97.9)}$	93.3$_{(99.7)}$	87.0$_{(98.1)}$	85.1$_{(98.5)}$
SMILE-2	73.2$_{(97.6)}$	**84.2$_{(101.0)}$**	85.0$_{(97.1)}$	**91.3$_{(99.8)}$**	84.9$_{(99.4)}$	84.8$_{(98.7)}$	93.5$_{(99.9)}$	87.3$_{(98.4)}$	85.5$_{(99.0)}$
MASS	**74.1$_{(98.8)}$**	83.2$_{(99.8)}$	**85.8$_{(98.1)}$**	90.9$_{(99.3)}$	**85.1$_{(99.6)}$**	**84.9$_{(98.8)}$**	**94.2$_{(100.6)}$**	**88.9$_{(100.2)}$**	**85.9$_{(99.4)}$**

Table 2: Accuracy when merging 8 fine-tuned models on the GLUE (Wang et al.) benchmark; Normalized average accuracy in subscript.

marks the upper bound on achievable performance. We evaluate on three collections of tasks, containing 8, 14, and 20 tasks, respectively. The latter is the most extensive setup considered in (Wang et al., 2024; Gargiulo et al., 2025; Daniel et al., 2025). We refer to Section B.3 for the specifics. We quantify results using both average absolute accuracy and average normalized accuracy.

MoErging results Table 1 reports the average absolute accuracy and the corresponding normalized accuracy (subscript, also in percentage) for each method, model size, and number of vision tasks. To adapt SMILE and WeMoE to our general setting, we employ a majority-voting-based heuristic to route the classification head. Details in §B.6.

We first note that MASS sets a new state of the art for MoErging across 8 out of 9 benchmarks, with gains as high as $\approx$ **6**% with respect to the best performing baseline. Figure 3 shows a per-dataset breakdown of the results, indicating that the accuracy is consistently high throughout all the datasets and not skewed on a subset of easier ones. In terms of storage, MASS requires a constant $2\times$ parameter increase, whereas other MoErging baselines range from $\sim 2.5\times$ up to $\sim 14\times$. A breakdown is provided in §B.1.

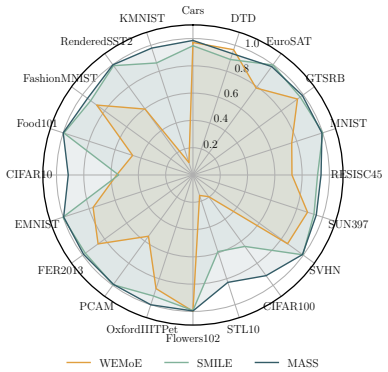
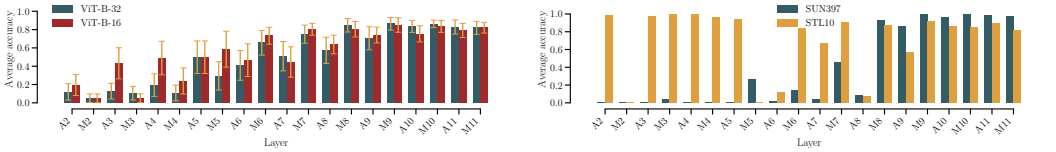


Figure 3: Per-dataset accuracies.

We also report the accuracies obtained by fixed merging methods, *i.e.* those not employing a router. These are evaluated with the ground truth classification head, *i.e.* assuming the task is known at inference time. While our evaluation setting is significantly more challenging, MASS still outperforms these ones by a consistent margin. In particular, when comparing it with the TSV-M baseline it builds upon, we see that routing produces a $\approx$ **5**% increase in accuracy.



(a) Averaged across all tasks for two models. (b) Focusing on SUN397 and STL10 for a ViT-B-32.

Figure 4: Per-layer task accuracies for ViT-B-32 on the 20-task benchmark. Layers starting with 'A' indicate attention layers, while those starting with 'M' refer to MLPs.

We further compare MASS with TwinMerging (Lu et al., 2024) on the 8-task benchmark using ViT-B-32, the only vision setting with reported results in their paper. Despite the lack of evaluation code for other setups, we find that MASS achieves **97.6%** normalized accuracy in this shared setting, exceeding the 95.3% reported for TwinMerging. Notably, the latter assumes oracle knowledge of the correct classification head, whereas MASS selects both the merged experts and the output space **at inference time**. Despite operating in a more general setting, MASS still reaches a higher accuracy. We also report in Table 2 the results on merging 8 Flan-T5-Base models (Chung et al., 2024) finetuned on the language tasks in GLUE (Wang et al.). MASS still yields the best results, showing its benefits to be modality agnostic.

Batched inference In the main experiments, we evaluated our method in the most challenging scenario, where each sample may belong to a different task, forcing the router to operate per input. Yet, in many practical settings (e.g., batched requests from the same domain), several inputs share the same task. In such cases, we can router-select a single merged model once per batch, closing the gap almost entirely: as shown in Figure 5, our approach achieves a mean normalized accuracy of at least 97% in 8 out of 9 settings, effectively matching individually fine-tuned models. This indicates that while our per-sample approach is already effective, batching can further reduce overhead and significantly boost accuracy.

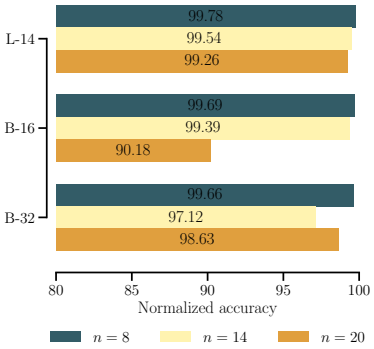


Figure 5: Batched accuracy.

4.2 Choosing a routing layer

We now investigate the effect of routing layer selection on task accuracy. Figure 4a shows the task prediction accuracy obtained by routing at different layers for ViT-B-32 and ViT-B-16 architectures. Interestingly, the best layer is consistent across architectures, with ViT-B-32 and ViT-B-16 both achieving peak performance at layer 9, and MLP layers exhibiting slightly better performance than the self-attention layers. However, it is immediately clear that the best layer varies significantly across tasks. In fact, some layers exhibit a variance of up to 40% in accuracy across tasks. This can be better appreciated by looking at Figure 4b, where we can see the per-layer task accuracies on STL10 and SUN397 for a ViT-B-32. While both tasks can be accurately predicted with the best chosen layer $\ell = 9$, the former shows a marked improvement in accuracy when routing at earlier layers $\ell = 3, 4, 5$, while the latter benefits from routing at later layers $\ell = 9, 10, 11$ and results in poor performance in the earlier ones. This suggests that the best routing layer is task-dependent, and may encourage further research on adaptive ways to determine it. We report an analogous analysis for the ViT-L-14 model in the appendix.

4.3 Comparison with other routers

We compare our router with two alternatives that differently balance accuracy, data, and compute.

Nearest Neighbor (nn) first constructs a small *support set* of representative examples from each task's validation data. For each test sample, we compare its intermediate representation $\mathbf{z}_\ell$ to the stored representations of all support examples. Formally, we compute the cosine similarity to each support sample and select the nearest neighbor among all tasks, inferring the task identity from

the task to which the support sample belongs. This method requires no additional parameters but assumes access to (and storage for) a curated batch of validation embeddings per task.

The second approach (`mlp`) fits an MLP over the union of validation sets from all tasks, with each example labeled by its originating task. After extracting $\mathbf{z}_\ell$, we train an MLP f_θ to predict the task via cross-entropy loss. Full architectural details are in the supplementary materials.

Results Table 3 shows the average normalized accuracy obtained by `MASS` when varying routing strategy. The `nn` approach, which stores and compares with a small validation set from each task, generally performs well but remains slightly below our best results. The MLP router achieves the highest accuracy overall, but it relies on a labeled validation set for training–a requirement that contradicts the core premise of merging methods, where access to original task data is often unavailable. This dependency makes the MLP less broadly applicable in realistic scenarios, such as when downloading fine-tuned models from public hubs without any accompanying data.

MASS	ViT-L-14		
+	8 tasks	14 tasks	20 tasks
nn	94.0	92.1	92.0
mlp	98.9	99.5	98.3
$\texttt{proj}_{\texttt{PRE}}$	99.1	97.7	91.9
$\texttt{proj}_{\texttt{TSV-M}}$	98.6	97.3	96.6

Table 3: Average normalized accuracy for different routers.

Focusing on the projection-based router (`proj`), we observe that starting from the `TSV-M` model ($\texttt{proj}_{\texttt{TSV-M}}$) outperforms routing from the pretrained backbone ($\texttt{proj}_{\texttt{PRE}}$) when scaling the number of tasks to 20. This is further confirmed in Table 7, where the gap widens to $\approx 10\%$ accuracy for a `ViT-B-32`. This gap underscores the core key insight behind our method: `TSV-M` arranges each task's top singular vectors into distinct, orthogonal subspaces, all contained in the final merged model. Therefore, $\texttt{proj}_{\texttt{TSV-M}}$ only needs to measure how well each subspace reconstructs the activations, "finding back" the subspace that was originally embedded for each task. In other words, once `TSV-M` has embedded each task's directions into a single model, a projection is enough to pinpoint the correct subspace, requiring no labels or additional training.

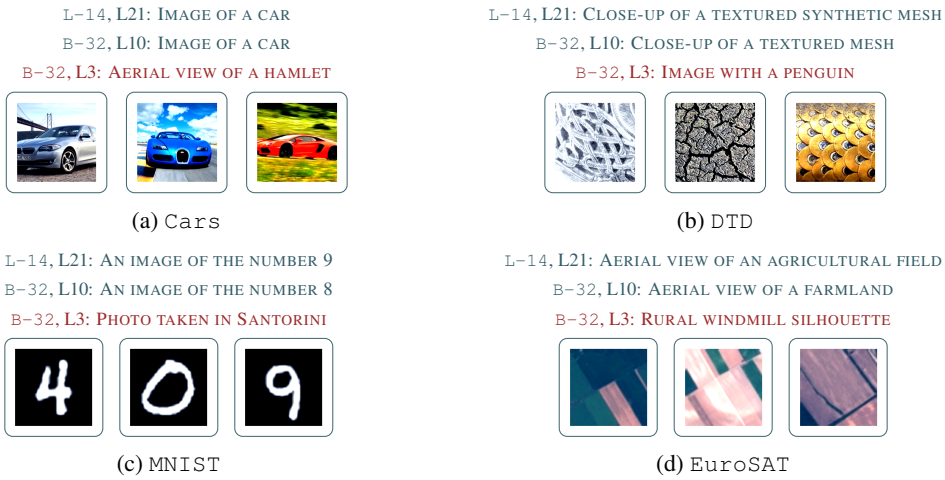
4.4 Interpreting task singular vectors

Finally, we attempt to interpret the task singular vectors that are used for routing in `MASS`. Notably, prior work suggests that mid-layer embeddings in CLIP-like models preserve semantic content, implying that routing at this layer hinges on meaningful features. To validate this, we interpret TSVs using the TextSpan algorithm (Gandelsman et al., 2024; Basile et al., 2024). The latter iteratively identifies and removes the most influential text directions, revealing which concepts best explain how the model's weight changes affect its representation. Importantly, while TextSpan was originally developed to analyze image embeddings, we employ it here to interpret singular vectors derived from weight updates.

Our results, illustrated in Figure 6, confirm that TSVs capture meaningful and interpretable visual-language associations. For example, the singular vectors associated with the `Cars` dataset (Krause et al., 2013) strongly activate the description "Image of a car", whereas those for the `DTD` dataset (Cimpoi et al., 2014) align closely with the phrase "Close-up of a textured mesh". Interestingly, across architectures (`ViT-L-14`, `ViT-B-32`), we find consistent textual concepts emerging from similar singular vectors for the same tasks. For instance, the singular vectors for both `ViT-L-14` at layer 21 and `ViT-B-32` at layer 10 align with almost equal phrases. This consistency hints at the intriguing possibility of transferring semantic information across architectures using text embeddings as an interpretable common ground.

We further note that the interpretability of these embeddings strongly depends on the chosen routing layer. While mid-to-late layers (e.g., layer 10 in `ViT-B-32` and layer 21 in `ViT-L-14`) yield semantically meaningful descriptions aligned closely with each dataset's visual domain, earlier layers (e.g., layer 3 in `ViT-B-32`) produce irrelevant or misleading concepts such as "Image with a penguin" for the texture dataset (`DTD`) or "Photo taken in Santorini" for digit classification (`MNIST`). Such discrepancies suggest that early layers encode generic or low-level features, while mid-to-late layers become progressively specialized toward domain-specific semantic structures.

Overall, these analyses validate our routing strategy: the task singular vectors at mid-layer embeddings capture precisely the semantic differences the router exploits to discriminate tasks. This also points to a deeper insight: the singular vectors derived from weight updates mirror the semantic structure of the data itself, highlighting a direct connection between weights and data.



(a) Cars (b) DTD

(c) MNIST (d) EuroSAT

Figure 6: Captions obtained by decoding task singular vectors as text as described in Section 4.4, accompanied by task representative images. Captions produced by the task singular vectors of predictive layers reflect the task content, those obtained by non-predictive ones do not.

5 Related Work

Model Merging has emerged as a lightweight alternative to ensembling. Early work focused on aligning models trained from scratch with different random seeds by solving permutation and mode-connectivity issues (Frankle et al., 2020; Entezari et al., 2022; Ainsworth et al., 2023; Crisostomi et al., 2025). More recent approaches instead merge multiple fine-tuned models derived from a common backbone (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024; Wang et al., 2024; Gargiulo et al., 2025; Daniel et al., 2025). These methods typically treat task updates as vectors and combine them through rescaling, pruning, or averaging, while newer work shows that respecting the layer-wise structure of updates significantly improves results (Gargiulo et al., 2025; Daniel et al., 2025). Our method builds directly on this line by adding adaptivity through input-driven routing, narrowing the performance gap between merged and fine-tuned models.

MoErging. A parallel line of work incorporates routing into merging, often referred to as "MoErging" (He et al., 2023; Yadav et al., 2024; Tang et al., 2024a; Lu et al., 2024). In the LLM domain, routers are trained to select among LoRAs or adapters (Jang et al., 2023; Chronopoulou et al., 2023; Muqeeth et al., 2024), sometimes requiring additional supervision or fine-tuning. Closer to our setting, Transformer^2 (Sun et al., 2025) introduces a two-step routing pipeline but relies on a modified fine-tuning procedure, while SMILE (Tang et al., 2024a) proposes a data- and training-free approach that leaves the pretrained backbone unchanged and *adds* task-specific low-rank updates at inference through layer-wise routing. In contrast, MASS embeds all updates into a single model, performs task selection once across layers, and *deactivates* irrelevant subspaces via a projection-based router. Finally, TwinMerging (Lu et al., 2024) requires per-task labels to train its router, whereas MASS is fully training- and data-free. We defer further discussion of related work to Section A.

6 Conclusions

In this paper, we introduced MASS (MoErging through Adaptive Subspace Selection), a method that aggregates low-rank task updates with adaptive routing, jointly selecting both encoder subspaces and classification heads without test-time supervision. To address the absence of per-task data in realistic merging scenarios, we designed a projection-based router that is entirely training- and data-free.

Our experiments show that MASS achieves state-of-the-art performance, recovering nearly the full accuracy of fine-tuned experts at a fraction of their combined storage cost. Future work includes refining the router for more precise subspace selection and extending MASS to out-of-distribution settings, where unseen tasks could be composed on the fly from existing singular vectors.

Reproducibility Statement

All implementation details and hyperparameter settings required to reproduce our results are provided in Sections 3, 4 and in Appendix B. Code is provided in the supplementary materials and will be publicly released along with all the checkpoints and logs.

References

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023. 1, 9, 16

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, 2025. ISSN 2522-5839. 1, 16

Lorenzo Basile, Valentino Maiorca, Luca Bortolussi, Emanuele Rodolà, and Francesco Locatello. Residual transformer alignment with spectral decomposition. *arXiv preprint arXiv:2411.00246*, 2024. 8

Joshua Belofsky. Token-level adaptation of lora adapters for downstream task generalization. In *Proceedings of the 2023 6th Artificial Intelligence and Cloud Computing Conference*, pp. 168–172, 2023. 16

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – Mining Discriminative Components with Random Forests. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision – ECCV 2014*, pp. 446–461, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10599-4. doi: 10.1007/978-3-319-10599-4_29. 18

Feng Cheng, Ziyang Wang, Yi-Lin Sung, Yan-Bo Lin, Mohit Bansal, and Gedas Bertasius. Dam: Dynamic adapter merging for continual video qa learning. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6805–6817. IEEE Computer Society, 2025. 16

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote Sensing Image Scene Classification: Benchmark and State of the Art. *Proceedings of the IEEE*, 105(10):1865–1883, October 2017. ISSN 1558-2256. doi: 10.1109/JPROC.2017.2675998. URL https://ieeexplore.ieee.org/document/7891544/?arnumber=7891544. Conference Name: Proceedings of the IEEE. 18

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*, 2023. 9, 16

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024. 7

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing Textures in the Wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613, Columbus, OH, USA, June 2014. IEEE. ISBN 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.461. URL https://ieeexplore.ieee.org/document/6909856. 8, 18

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep Learning for Classical Japanese Literature, November 2018. URL http://arxiv.org/abs/1812.01718. arXiv:1812.01718 [cs, stat]. 18

Adam Coates, Andrew Ng, and Honglak Lee. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, June 2011. URL https://proceedings.mlr.press/v15/coates11a.html. ISSN: 1938-7228. 18

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, May 2017. doi: 10.1109/IJCNN.2017.7966217. URL https://ieeexplore.ieee.org/document/7966217/?arnumber=7966217. ISSN: 2161-4407. 18

Donato Crisostomi, Marco Fumero, Daniele Baieri, Florian Bernard, and Emanuele Rodolà. C^2M^3: Cycle-consistent multi-model merging. In *Advances in Neural Information Processing Systems*, volume 37, 2025. 9, 16

Nico Daheim, Thomas Möllenhoff, Edoardo Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *The Twelfth International Conference on Learning Representations*. 1, 16

Marczak Daniel, Magistri Simone, Cygert Sebastian, Twardowski Bartłomiej, D Bagdanov Andrew, and Joost van de Weijer. No task left behind: Isotropic model merging with common and task-specific subspaces. *ArXiv preprint*, 2025. 1, 6, 9, 16

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*. Springer, 2025. 16

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019. 1

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929. 5

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022. 9, 16

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, 2020. 9, 16

Yossi Gandelsman, Alexei A Efros, and Jacob Steinhardt. Interpreting clip's image representation via text-based decomposition. In *ICLR*, 2024. 8

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. In *Proc. CVPR*, 2025. 1, 3, 4, 5, 6, 9, 16, 17, 19, 20

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P. Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018. 16

Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in Representation Learning: A Report on Three Machine Learning Contests. In Minho Lee, Akira Hirose, Zeng-Guang Hou, and Rhee Man Kil (eds.), *Neural Information Processing*, pp. 117–124, Berlin, Heidelberg, 2013. Springer. ISBN 978-3-642-42051-1. doi: 10.1007/978-3-642-42051-1_16. 18

Fidel A. Guerrero-Peña, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, and Marco Pedersoli. Re-basin via implicit sinkhorn differentiation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, 2023. 16

Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. Merging experts into one: Improving computational efficiency of mixture of experts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14685–14691, 2023. 9, 16

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. 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*, 12(7):2217–2226, July 2019. ISSN 2151-1535. doi: 10.1109/JSTARS.2019.2918242. URL https://ieeexplore.ieee.org/document/8736785/?arnumber=8736785. Conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 18

Stefan Horoi, Albert Manuel Orozco Camacho, Eugene Belilovsky, and Guy Wolf. Harmony in Diversity: Merging Neural Networks with Canonical Correlation Analysis. 2024. 16

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 1

Chenyu Huang, Peng Ye, Tao Chen, Tong He, Xiangyu Yue, and Wanli Ouyang. Emr-merging: Tuning-free high-performance model merging, 2024. 1, 16

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023. 1, 3, 5, 9, 16, 17

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Exploring the benefits of training expert language models over instruction tuning. In *International Conference on Machine Learning*, pp. 14702–14729. PMLR, 2023. 9, 16

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: renormalizing permuted activations for interpolation repair. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023. 16

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D Object Representations for Fine-Grained Categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pp. 554–561, Sydney, Australia, December 2013. IEEE. ISBN 978-1-4799-3022-7. doi: 10.1109/ICCVW.2013.77. URL http://ieeexplore.ieee.org/document/6755945/. 8, 18

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. 18

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. 18

Zhenyi Lu, Chenghao Fan, Wei Wei, Xiaoye Qu, Dangyang Chen, and Yu Cheng. Twin-merging: Dynamic integration of modular expertise in model merging. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2, 7, 9, 16

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. 16

Tommaso Mencattini, Adrian Robert Minut, Donato Crisostomi, Andrea Santilli, and Emanuele Rodolà. Merge^3: Efficient evolutionary merging on consumer-grade gpus. *arXiv preprint arXiv:2502.10436*, 2025. 16

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Dilan Görür, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. 16

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. Learning to route among specialized experts for zero-shot generalization. In *International Conference on Machine Learning*, pp. 36829–36846, 2024. 9, 16

Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant deep weight space alignment, 2023. 16

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf. 18

Maria-Elena Nilsback and Andrew Zisserman. Automated Flower Classification over a Large Number of Classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729, December 2008. doi: 10.1109/ICVGIP.2008.47. URL https://ieeexplore.ieee.org/document/4756141. 18

Guillermo Ortiz-Jiménez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. 1, 16

Oleksiy Ostapenko, Zhan Su, Edoardo Ponti, Laurent Charlin, Nicolas Le Roux, Lucas Caccia, and Alessandro Sordoni. Towards modular llms by building and reusing a library of loras. In *International Conference on Machine Learning*, pp. 38885–38904. PMLR, 2024. 16

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505, June 2012. doi: 10.1109/CVPR.2012.6248092. URL https://ieeexplore.ieee.org/document/6248092. ISSN: 1063-6919. 18

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021. 1, 5

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 1, 16

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170. 18

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pp. 1453–1460, July 2011. doi: 10.1109/IJCNN.2011.6033395. URL https://ieeexplore.ieee.org/document/6033395/?arnumber=6033395. ISSN: 2161-4407. 18

George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*, a. 16

George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. In *The Thirteenth International Conference on Learning Representations*, b. 16

Qi Sun, Edoardo Cetin, and Yujin Tang. Transformer-squared: Self-adaptive llms. In *The Thirteenth International Conference on Learning Representations*, 2025. 9, 16

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, pp. 270–279. Springer, 2018. 1

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. Parameter-efficient multi-task model fusion with partial linearization. In *The Twelfth International Conference on Learning Representations*. 16

Anke Tang, Li Shen, Yong Luo, Shuai Xie, Han Hu, Lefei Zhang, Bo Du, and Dacheng Tao. Smile: Zero-shot sparse mixture of low-rank experts construction from pre-trained foundation models. *arXiv preprint arXiv:2408.10174*, 2024a. 2, 9, 16

Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging multi-task models via weight-ensembling mixture of experts. In *International Conference on Machine Learning*, pp. 47778–47799. PMLR, 2024b. 2, 16

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622, 1999. 5

Bastiaan S. Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation Equivariant CNNs for Digital Pathology. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger (eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, pp. 210–218, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00934-2. doi: 10.1007/978-3-030-00934-2_24. 18

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*. 6, 7

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2024. 1, 5, 6, 9, 16, 17

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo-Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models, 2021. 16

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, September 2017. URL http://arxiv.org/abs/1708.07747. arXiv:1708.07747 [cs, stat]. 18

Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. SUN Database: Exploring a Large Collection of Scene Categories. *International Journal of Computer Vision*, 119(1):3–22, August 2016. ISSN 1573-1405. doi: 10.1007/s11263-014-0748-y. URL https://doi.org/10.1007/s11263-014-0748-y. 18

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. 1, 9, 16

Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning, 2024. 2, 9, 16

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2024a. 16

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*, 2024b. 16

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014. 1

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2024. 1, 9, 16

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4447–4462, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.263. URL https://aclanthology.org/2024.findings-acl.263/. 16

Luca Zhou, Daniele Solombrino, Donato Crisostomi, Maria Sofia Bucarelli, Fabrizio Silvestri, and Emanuele Rodolà. Atm: Improving model merging by alternating tuning and merging, 2024. URL https://arxiv.org/abs/2411.03055. 16

A Extended related work

A.1 Model merging

Model Merging has recently gained traction as a computationally efficient alternative to ensembling. Early methods, motivated by linear mode connectivity (Frankle et al., 2020; Entezari et al., 2022; Mirzadeh et al., 2021; Garipov et al., 2018), primarily aligned models trained with different optimization seeds. This was achieved by finding neuron permutations matching these ones before the aggregation (Ainsworth et al., 2023; Jordan et al., 2023; Crisostomi et al., 2025; Singh & Jaggi, 2020; Stoica et al., a; Guerrero-Peña et al., 2023; Navon et al., 2023; Horoi et al., 2024). More recent merging approaches aggregate instead multiple fine-tuned models derived from a shared pretrained backbone (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024; Matena & Raffel, 2022; Wortsman et al., 2021; Davari & Belilovsky, 2025; Wang et al., 2024; Zhou et al., 2024; Gargiulo et al., 2025; Ortiz-Jiménez et al., 2023; Mencattini et al., 2025; Huang et al., 2024; Daheim et al.; Stoica et al., b; Yang et al., 2024a; Tang et al.). These methods incorporate various strategies to improve the merging, such as finding the optimal combination of task vectors (Yang et al., 2024b), mitigating sign disagreement (Yadav et al., 2023), randomly dropping a fraction of the updates (Yu et al., 2024), or employing evolutionary strategies (Akiba et al., 2025; Mencattini et al., 2025). The most recent line of work considers task vectors at a layer level, accounting this way for the natural structure of the layers and significantly improving the merging outcome (Stoica et al., b; Gargiulo et al., 2025; Daniel et al., 2025). Our approach builds upon the latter methodologies by introducing adaptivity through input-driven routing, significantly narrowing the performance gap between the finetuned endpoints and their resulting merge.

A.2 MoErging

Model Merging with Mixture-of-Experts, often referred to as "MoErging" (He et al., 2023; Yadav et al., 2024; Jang et al., 2023; Chronopoulou et al., 2023; Belofsky, 2023; Zhao et al., 2024; Muqeeth et al., 2024; Tang et al., 2024b; Ostapenko et al., 2024; Cheng et al., 2025; Tang et al., 2024a; Lu et al., 2024; Sun et al., 2025), explores how independently trained experts–potentially contributed by a decentralized community–can be combined within a single adaptive model by dynamically selecting which expert(s) should handle a given input. In the LLM domain, specialized modules (e.g., LoRAs or adapters) are merged via parametric or data-driven routers that match the incoming prompt to the most relevant module (Jang et al., 2023; Chronopoulou et al., 2023; Belofsky, 2023; Zhao et al., 2024; Muqeeth et al., 2024; Tang et al., 2024b; Ostapenko et al., 2024; Cheng et al., 2025); some approaches, such as PHATGOOSE (Muqeeth et al., 2024), require fine-tuning additional routing parameters, whereas others, like weight-ensembling MoE (Tang et al., 2024b), may need to train the router at test time. Sharing a similar two-pass pipeline, `Transformer`^{2} (Sun et al., 2025) modifies the finetuning procedure to yield task-aligned singular vectors, allowing for expert routing at test time. Differently from the latter, `MASS` works on any independently fine-tuned models finetuned with no ad hoc routines. A similar strategy was independently introduced by `SMILE` (Tang et al., 2024a), which, like our method, enables data-free merging of multiple experts. However, `SMILE` leaves the pretrained backbone intact and selectively *adds* task-specific low-rank updates at inference, whereas `MASS` begins with a single model containing all updates and *deactivates* any irrelevant subspaces via a router. The fact that both lines of work arrived at a similar subspace-activation concept, despite differing motivations, highlights the versatility and broad appeal of such an approach. Lastly, `TwinMerging` (Lu et al., 2024) similarly merges a shared expert and multiple task-specific experts via a gating function. However, `TwinMerging` relies on flat task arithmetic (Ilharco et al., 2023) and requires per-task labeled data to train its router, reducing its applicability. In contrast, `MASS` operates in a fully data-free and training-free regime by design.

B Additional details

We here describe the details required to implement and reproduce our results. The code is provided in the supplementary materials. In particular, Section B.2 describes implementation details, Section B.4 specifies the employed evaluation metrics, Section B.5 reports the employed architecture and Section C.1 specificies the hyperparameters and how they were chosen.

	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
MoE	WeMoE	5.06	8.06	11.05	5.12	8.16	11.20	5.78	9.31	12.84
MoE	SMILE-1	**1.61**	2.07	2.52	**1.62**	2.09	2.55	**1.47**	**1.82**	2.18
MoE	SMILE-2	3.05	4.60	6.14	3.09	4.67	6.24	2.59	3.78	4.96
	MASS	2.00	**2.00**	**2.00**	2.00	**2.00**	**2.00**	2.00	2.00	**2.00**

Table 4: Relative parameters increase with respect to the base model.

B.1 Storage and compute overhead

The $\sim 2\times$ **compute** figure is an approximation: we run the backbone twice (fixed TSV-M (Gargiulo et al., 2025) pass + routed pass) and the router itself adds $< 1\%$ FLOPs. Since model-merging saves training and storage rather than inference time, this modest overhead is acceptable. MASS routes per sample, so each incurs this cost; however, batching over the same task can reduce it. For example, with 32 samples, routing costs only $1.06\times$ forward equivalents. The **storage overhead** is exactly $2\times$: alongside the merged model, we store $1/T$ TSVs per task. Summed across tasks and layers, these form a second full set of weights, effectively doubling storage. A full comparison with the overhead incurred by the baselines is provided in Table 4.

B.2 Implementation

We used the same model checkpoints as Consensus TA (Wang et al., 2024), except for the one for the EMNIST dataset which we had to re-finetune due to an inconsistency in image orientation between EMNIST and MNIST. Shortly, the *torchvision*[2] version yields rotated and flipped images, spuriously yielding extremely similar models (same classes, roughly same dataset statistics) that performed very poorly when interchanged. Simply re-rotating and flipping the EMNIST images to match the orientation of MNIST solves the issue. We further used a single classification head for STL10 and CIFAR10 due to their 9 shared classes. The final head has the shared classes plus the two dataset-specific ones, *i.e.* monkey and frog.

B.3 Benchmarks and datasets

The 8-task benchmark, introduced in (Ilharco et al., 2023), comprises the following datasets: Cars, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397, and SVHN. Moving to 14 tasks, we add CIFAR100, STL10, Flowers102, OxfordIIITPet, PCAM, and FER2013. The 20-task suite further includes EMNIST, CIFAR10, Food101, FashionMNIST, RenderedSST2, and KMNIST. We provide the specific dataset details in Table 5.

B.4 Evaluation measures

To account for differences in task difficulty, we report both *absolute* and *normalized* accuracy in our results. The normalized accuracy serves as a relative performance measure by comparing the multi-task model's accuracy to that of individual fine-tuned models. It is computed as:

$$\text{Normalized Accuracy} = \frac{1}{T}\sum_{i=1}^{T}\frac{\text{accuracy}(\theta_{MT}, t_i)}{\text{accuracy}(\theta_{ft_i}, t_i)} \tag{2}$$

where T represents the total number of tasks, θ_{MT} is the multi-task model, and θ_{ft_i} corresponds to the fine-tuned model for task t_i. By normalizing accuracy in this way, we ensure a fairer comparison that accounts for variations in baseline task performance.

[2] https://pytorch.org/vision/stable/index.html

Dataset	image size	# train	# val	# test
Cars (Krause et al., 2013)	varies	7330	814	8041
DTD (Cimpoi et al., 2014)	varies	1692	188	1880
EuroSAT (Helber et al., 2019)	64×64	21600	2700	2700
GTSRB (Stallkamp et al., 2011)	varies	23976	2664	12630
MNIST (Lecun et al., 1998)	28×28	55000	5000	10000
RESISC45 (Cheng et al., 2017)	256×256	17010	1890	6300
SUN397 (Xiao et al., 2016)	varies	17865	1985	19850
SVHN (Netzer et al., 2011)	32×32	68257	5000	26032
CIFAR100 (Krizhevsky & Hinton, 2009)	32×32	45000	5000	10000
STL10 (Coates et al., 2011)	96×96	4500	500	8000
Flowers102 (Nilsback & Zisserman, 2008)	varies	918	102	6149
OxfordIIITPet (Parkhi et al., 2012)	varies	3312	368	3669
PCAM (Veeling et al., 2018)	96×96	257144	5000	32768
FER2013 (Goodfellow et al., 2013)	48×48	25839	2870	7178
EMNIST (Cohen et al., 2017)	28×28	235000	5000	40000
CIFAR10 (Krizhevsky & Hinton, 2009)	32×32	45000	5000	10000
Food101 (Bossard et al., 2014)	512×512	70750	5000	25250
FashionMNIST (Xiao et al., 2017)	28×28	55000	5000	10000
RenderedSST2 (Socher et al., 2013)	varies	6228	692	1821
KMNIST (Clanuwat et al., 2018)	28×28	55000	5000	10000

Table 5: Image sizes, and numbers of train, validation, and test samples for the considered datasets.

B.5 Architectures

We use CLIP from the *OpenClip* library[3], using the three different versions described in Table 6. For the router, we use a small two-layer MLP with a hidden dimension of 1024. It accepts a 512-dimensional embedding vector, applies a linear transformation, a ReLU activation, and dropout with a probability of 0.5, and outputs logits corresponding to task selection probabilities.

Model	**Layers**	**Hidden Dimension**	**Heads**	**Patch Size**	**Parameters**
ViT-B-32	12	768	12	32×32	∼86M
ViT-B-16	12	768	12	16×16	∼86M
ViT-L-14	24	1024	16	14×14	∼307M

Table 6: Comparison of ViT-B-32, ViT-B-16, and ViT-L-14 architectures.

[3] https://github.com/mlfoundations/open_clip/

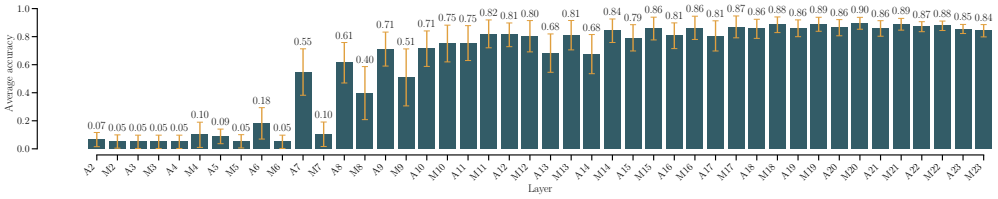


Figure 7: `ViT-L-14` per-layer task accuracies.

B.6 Adapting oracle MoErging methods

As extensively discussed, we argue that MoErging methods should not rely on an oracle head, since their pipelines implicitly assume that the task label is unknown at inference time (otherwise, merging the correct task vector would be trivial). To respect this assumption, we introduced a routing procedure that exploits the routers already implemented in each method. For SMILE, we extracted the mode of the tokens at each layer and applied a naïve majority-voting scheme across layers to select the head for each sample, which was then used for the final classification. Analogously, for WeMoE, we applied the same logic, implementing a straightforward head-selection strategy that leverages the existing gates and coefficients. All results reported for MoE methods were obtained under this unified setting.

Algorithm 2 Fixed Merging Step

Require: Pretrained model weights θ_{pre}, task-specific updates $\{\Delta_i\}_{i=1}^{T}$, user-specified threshold ε
Ensure: Fixed merged model weights θ_{MT}

Accounting for redundant directions (Section 3.2.2)
$\mathcal{M} = \{\}$
for $i = 1, \ldots, T$ **do**
  $\delta_i \leftarrow \text{vec}(\Delta_i)$
  if $\max_{\{j \in \mathcal{M}\}} \text{sim}(\delta_i, \delta_j) < \varepsilon$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{i\}$
  end if
end for
Merging step using `TSV-M` Gargiulo et al. (2025) **on the** $\{\Delta_i\}_{i \in \mathcal{M}}$
for $i \in \mathcal{M}$ **do**
  $\Delta_i = U_i \, \Sigma_i \, V_i^{\top}$
  $\tilde{U}_i \leftarrow U_{i[:,1:k]}$, $\tilde{\Sigma}_i \leftarrow \Sigma_{i[1:k,1:k]}$, $\tilde{V}_i \leftarrow V_{i[:,1:k]}$
end for
  $U \leftarrow [\tilde{U}_1 \,|\, \tilde{U}_2 \,|\, \cdots \,|\, \tilde{U}_T]$
  $\Sigma \leftarrow \text{block\_diag}(\tilde{\Sigma}_1, \tilde{\Sigma}_2, \ldots, \tilde{\Sigma}_T)$
  $V \leftarrow [\tilde{V}_1 \,|\, \tilde{V}_2 \,|\, \cdots \,|\, \tilde{V}_T]$
  $U_{\perp} \leftarrow \text{orthogonalize}(U)$
  $V_{\perp} \leftarrow \text{orthogonalize}(V)$
  $\hat{\Delta} \leftarrow U_{\perp} \, \Sigma \, V_{\perp}^{\top}$
  $\theta_{\text{MT}} \leftarrow \theta_{\text{pre}} + \alpha \, \hat{\Delta}$
return θ_{MT}

Radar charts on 8- and 14-task benchmarks. In section Section 4, we present comprehensive results for the approach using the 20 task benchmark. Figure 9 and Figure 10 respectively display the normalized accuracies for our method on 8 and 14 tasks across all three model sizes (`ViT-B-32`, `ViT-B-16`, and `ViT-L-14`). In both cases, the approach retains a high fraction of each fine-tuned model's performance, with normalized accuracies often above 80–90%. Notably, the method scales gracefully as the number of tasks increases from 8 to 14.

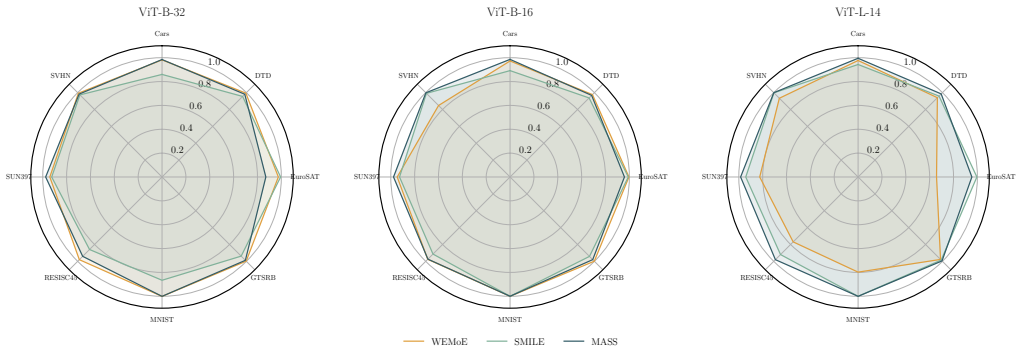


Figure 9: Normalized task accuracies over models `ViT-B-32`, `ViT-B-16` and `ViT-L-14` for the 8 tasks benchmark.

C Additional Experiments and Results

In this section, we provide detailed per-task and per-layer accuracy plots, along with further examples of decoded task vectors, to complement the results presented in the main paper.

C.1 Hyperparameter Settings

Following the recommendation of TSV (Gargiulo et al., 2025), we use α as a single scaling factor with the suggested value of 1.0 for the TSV-M merging configurations in both Algorithm 2 and Algorithm 1. Consistent with TSV, the compression rate assigned to each task space is set to $\frac{1}{T}$. We optimized the similarity threshold ε over the range $\{0.1, 0.2, ..., 0.9\}$ and determined the router's selection threshold η via a Bayesian search within the interval [0.05, 0.5]. As illustrated in Figures 4a and 7, we identify the optimal projection layer by focusing on those revealing the highest task accuracy. Specifically, for `ViT-B-32` and `ViT-B-16` models, we select the attention and MLP layers within the range $\{7, ..., 11\}$, while for the `ViT-L-14` model, the chosen layers fall in the range $\{19, ..., 23\}$. The temperature parameter for tuning the behavior of the softmax function at line 6 in Algorithm 1 is set to 1.

Sensitivity to Hyperparameters The merging coefficient is set to 1 as in `TSV-M` (Gargiulo et al., 2025). Figure 8 shows how accuracy changes with routing threshold η and top-K. At low η (0.05) and large K, too many tasks are merged, causing interference. At high η (≥ 0.3), only the top task is selected, making performance insensitive to K (it's effectively an argmax). The best accuracy occurs in a broad middle range, peaking at $\eta = 0.2$, $K = 3$, where the router balances selectivity and coverage. We also vary the cosine threshold ε used to discard similar task updates before merging. Due to the high dimensionality of the Δs, large thresholds ($\varepsilon \geq 0.4$) retain all updates, leaving redundancy unaddressed (accuracy 93.5). Small ones ($\varepsilon \leq 0.05$) instead remove even distinct directions, significantly harming accuracy (≤ 88.6). Intermediate values ($\varepsilon \approx 0.2$) offer a robust filtering, improving performance (≥ 93.9) by suppressing redundancy.

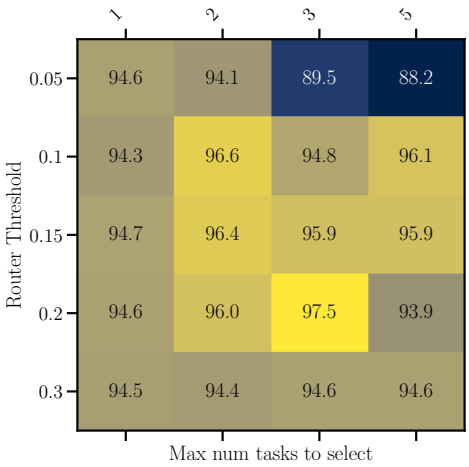


Figure 8: Hyperparameter sensitivity.

Layer-wise accuracies for individual datasets. Figures 11 and 12 show per-layer accuracies for `ViT-B-32` on different subsets of the 8-task benchmark:

- Figure 11 focuses on `Cars`, `DTD`, `EuroSAT`, and `GTSRB`.
- Figure 12 displays results for `MNIST`, `RESISC45`, `SUN397`, and `SVHN`.

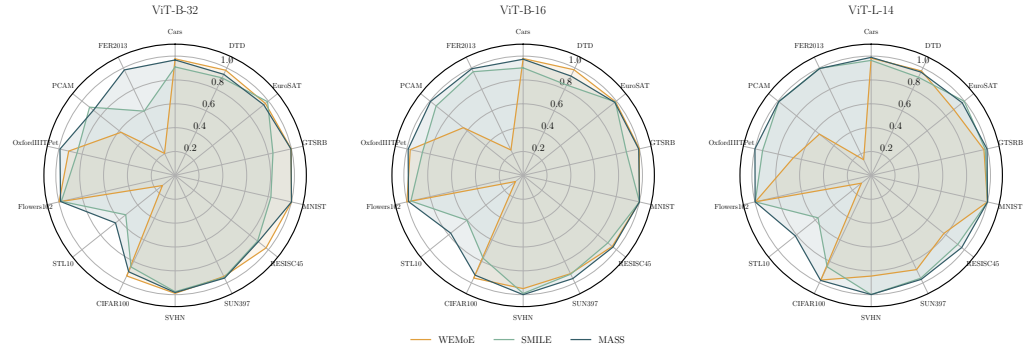
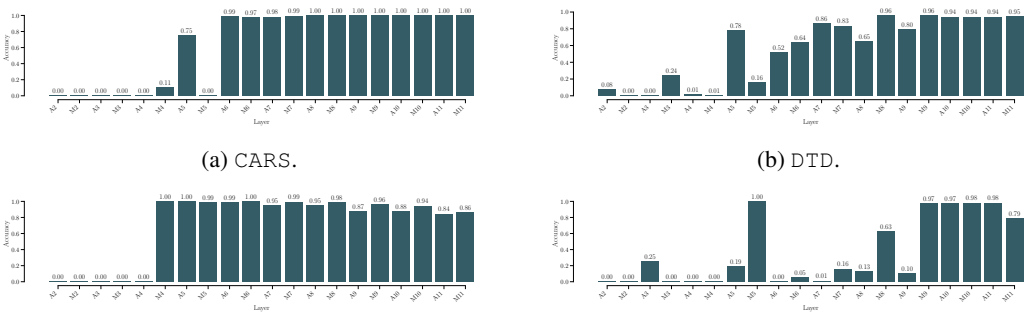


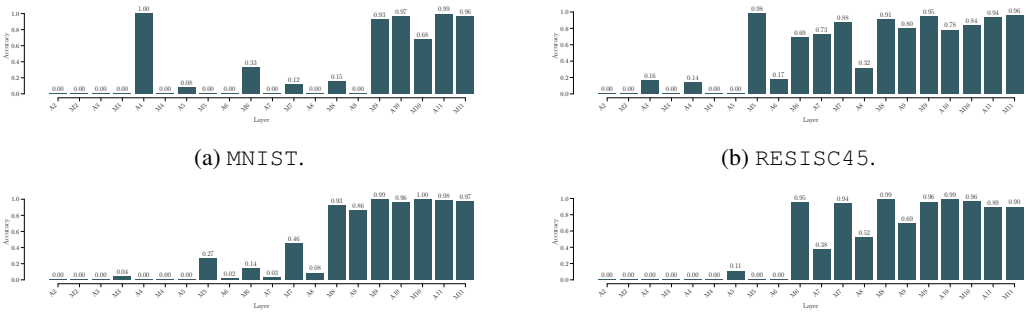
Figure 10: Normalized task accuracies over models `ViT-B-32`, `ViT-B-16` and `ViT-L-14` for the 14 tasks benchmark.



(a) `CARS`. (b) `DTD`.

(c) `EuroSAT`. (d) `GTSRB`.

Figure 11: Per-layer task accuracies for `ViT-B-32` on `Cars`, `DTD`, `EuroSAT`, and `GTSRB`.



(a) `MNIST`. (b) `RESISC45`.

(c) `SUN397`. (d) `SVHN`.

Figure 12: Per-layer task accuracies for `ViT-B-32` on `MNIST`, `RESISC45`, `SUN397`, and `SVHN`.

MASS	ViT-B-32			ViT-B-16		
+	8 tasks	14 tasks	20 tasks	8 tasks	14 tasks	20 tasks
nn	92.7	89.6	89.4	92.7	90.0	90.2
mlp	96.8	95.8	95.8	97.1	94.8	96.5
proj_{PRE}	98.2	88.6	79.4	98.7	92.0	81.2
$\text{proj}_{\text{TSV-M}}$	96.5	93.2	90.9	98.0	96.1	88.7

Table 7: Average normalized accuracy for different routers.

Again, the top-performing layer is not shared across the tasks, confirming what we observed in the main paper.

Layer-wise accuracies for ViT-L-14. Figure 7 reports average per-layer task accuracy for the larger ViT-L-14, showing that in this case the most predictive layer for routing is $\ell = 20$. Since ViT-L-14 has 24 layers (compared to the 12 layers in ViT-B-32 and ViT-B-16), the most predictive layer is roughly at the same relative depth.

Visualizations for additional datasets. Following the approach in Section 4.4, Figure 13 shows examples of decoded task vectors for datasets like SVHN, GTSRB, SUN397, and RESISC45. Here, we see textual prompts such as "*An image of the number 4*" or "*Aerial view of an industrial area*", which align with each dataset's distinct domain. This reaffirms that our singular vectors capture domain-specific transformations while preserving high-level semantic alignment to the pretrained model.

(a) SVHN (b) GTSRB

(c) SUN397 (d) RESISC45

Figure 13: Captions obtained by decoding task singular vectors as text for datasets SVHN, GTSRB, SUN397, and RESISC45 as described in Section 4.4, accompanied by three representative images for each dataset.

D PROPOSITIONS AND PROOFS

Proposition D.1 (Optimality of Orthogonal Projection). *Let $V \in \mathbb{R}^{d\times k}$ have orthonormal columns spanning a subspace $\mathcal{S} \subseteq \mathbb{R}^d$, and let $\mathbf{a} \in \mathbb{R}^d$. Then the unique minimizer of $\|\mathbf{a} - \mathbf{w}\|_2^2$ over all $\mathbf{w} \in \mathcal{S}$ is*

$$\widehat{\mathbf{w}} \;=\; V\,V^\top \mathbf{a}.$$

Proof. Any $\mathbf{w} \in \mathcal{S}$ can be written as $V\,\boldsymbol{\alpha}$ for some $\boldsymbol{\alpha} \in \mathbb{R}^k$. The problem

$$\min_{\mathbf{w} \in \mathcal{S}} \|\mathbf{a} - \mathbf{w}\|_2^2 \quad \Longleftrightarrow \quad \min_{\boldsymbol{\alpha} \in \mathbb{R}^k} \|\mathbf{a} - V\,\boldsymbol{\alpha}\|_2^2$$

has a strictly convex objective, so its global minimizer is found by setting the gradient to zero. A short calculation shows

$$\boldsymbol{\alpha} \;=\; V^\top \mathbf{a} \quad \Longrightarrow \quad \widehat{\mathbf{w}} \;=\; V\,(V^\top \mathbf{a}) \;=\; V\,V^\top\,\mathbf{a}.$$

Uniqueness follows from the strict convexity, and $\|\mathbf{a} - \widehat{\mathbf{w}}\|_2$ is necessarily the smallest possible distance in $\mathcal{S}$. Equivalently, $\mathbf{a} - \widehat{\mathbf{w}}$ is orthogonal to $\mathcal{S}$, so no further reduction in norm is possible. □

Proposition D.2 (§3.1)**.** *Let $z_\ell \in \mathbb{R}^d$ be a feature vector, and for each task i, decompose it as*

$$z_\ell = V_i V_i^\top z_\ell + \varepsilon_i, \qquad \varepsilon_i = \big(I - V_i V_i^\top\big) z_\ell.$$

Assume $\varepsilon_i \sim \mathcal{N}(0, \sigma^2 I)$. Then the maximum a posteriori estimate of the task reduces to

$$\hat{\imath}_{\mathrm{MAP}} = \arg\max_i p(\textit{task} = i \mid z_\ell) = \arg\min_i \|\varepsilon_i\|_2^2.$$

Thus, under these assumptions, selecting the task with the smallest squared Euclidean residual is exactly equivalent to maximizing the posterior.

Proof. By assumption, $p(\varepsilon_i) = (2\pi\sigma^2)^{-d/2} \exp\!\Big(-\|\varepsilon_i\|_2^2/(2\sigma^2)\Big)$, so $-\log p(\varepsilon_i) \propto \|\varepsilon_i\|_2^2$. Since $V_i V_i^\top z_\ell$ is a deterministic shift, the likelihood $p(z_\ell \mid \text{task} = i)$ depends only on ε_i. With a uniform prior over tasks,

$$\hat{\imath}_{\mathrm{MAP}} = \arg\max_i p(z_\ell \mid \text{task} = i) = \arg\max_i p(\varepsilon_i) = \arg\min_i \|\varepsilon_i\|_2^2.$$

Hence, minimizing the ℓ_2 residual is exactly equivalent to maximizing the posterior. □