Merging Multi-Task Models via Weight-Ensembling Mixture of Experts

Anke Tang [1 2 *] **Li Shen** [3 4 *] **Yong Luo** [1 2] **Nan Yin** [5] **Lefei Zhang** [1 2] **Dacheng Tao** [6]

Abstract

Merging various task-specific Transformer-based vision models trained on different tasks into a unified model can execute all the tasks concurrently. Previous methods, exemplified by task arithmetic, have proven to be both effective and scalable. Existing methods have primarily focused on seeking a static optimal solution within the original model parameter space. A notable challenge is mitigating the interference between parameters of different models, which can substantially deteriorate performance. In this paper, we propose to merge most of the parameters while upscaling the MLP of the Transformer layers to a weight-ensembling mixture of experts (MoE) module, which can dynamically integrate shared and task-specific knowledge based on the input, thereby providing a more flexible solution that can adapt to the specific needs of each instance. Our key insight is that by identifying and separating shared knowledge and task-specific knowledge, and then dynamically integrating them, we can mitigate the parameter interference problem to a great extent. We conduct the conventional multi-task model merging experiments and evaluate the generalization and robustness of our method. The results demonstrate the effectiveness and provide a comprehensive understanding of our method.

1. Introduction

The swift advancement of deep learning has fostered a shift towards fine-tuning large pre-trained models for downstream tasks, rather than training them from scratch. Having been initially trained on large-scale datasets, pre-trained models have outstanding common sense and are adept at recognizing and processing diverse data patterns (Radford et al., 2019; He et al., 2021; Wang et al., 2023). These models can then be fine-tuned on downstream tasks to acquire task-specific knowledge (Chung et al., 2022; Zheng et al., 2023; Cao et al., 2024). In this context, merging multiple task-specific models into a single unified model has emerged as an effective and scalable strategy for knowledge transfer and multi-task learning (Li et al., 2023; Lin et al., 2023).

There are several state-of-the-art algorithms for merging models. A prominent example in this field is task arithmetic (Ilharco et al., 2023), which interpolates the parameters of models linearly. These methods excel in extracting knowledge from various models and synthesizing a unified model cheaply. Such approaches present a promising solution for constructing robust models (Izmailov et al., 2019; Wortsman et al., 2022), especially in the scenarios where the training data is decentralized, limited, or inaccessible due to privacy constraints (Tang et al., 2023a; Jin et al., 2023).

Existing methods predominantly aim to find a static multi-task optimal solution within the original parameter space. These methods do not introduce any new parameters, thus maintaining the original inference cost. This approach, however, imposes limitations on adaptability to the unique requirements of each instance, as the task-specific optimal solution varies. Another significant challenge of merging multi-task models is mitigating the interference between parameters of different models, which can substantially deteriorate the average performance (Yadav et al., 2023; Yu et al., 2023; Tang et al., 2023b). Existing methods, while effective in some scenarios, may not be flexible enough to handle the dynamic nature of multi-task learning, where the optimal solution can vary depending on the input.

To address these challenges, we propose a novel approach to merge vision Transformers (ViTs). Our method merges most of the parameters while upscaling the multilayer perceptron (MLP) of the Transformer layers to a weight-ensembling Mixture of Experts (MoE) module. This module can dynamically integrate shared and task-specific knowledge based on the input sample, thereby providing a more flexible solution that can adapt to the specific needs of each instance.

Our primary realization is that the issue of parameter inter-

[*]Equal contribution [1]National Engineering Research Center for Multimedia Software, School of Computer Science, Wuhan University, China [2]Hubei Luojia Laboratory, Wuhan, China [3]Sun Yat-sen University, Shenzhen, China [4]JD Explore Academy, China [5]Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates [6]Nanyang Technological University, Singapore. Correspondence to: Yong Luo <luoyong@whu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

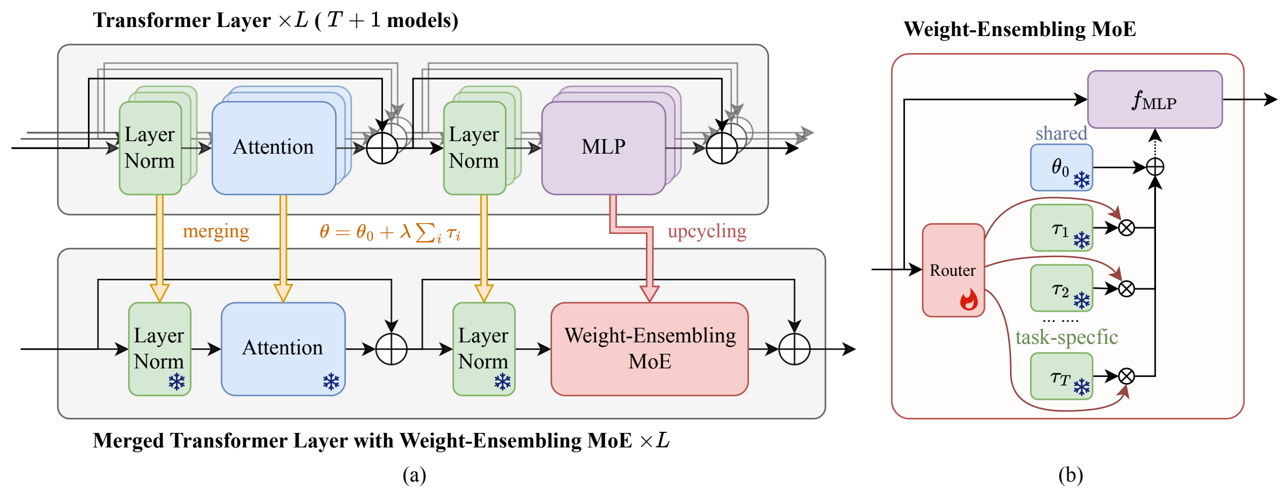


Figure 1. (a) **Framework overview**. This figure shows the overall framework of our proposed method to merge the pre-trained model and fine-tuned task-specific models. We merge weights in the Transformer Layers except for the MLPs. For the MLPs, we upcycle them into weight-assembling MoE modules. (b) **Wieght-Ensembling Mixture of Experts (MoE) Module**. Here we outline the detailed structure of the Weight-Ensembling MoE module, composed of the router, pre-trained MLP weights, and a collection of task vectors. Collaboration between shared weights and task vectors is employed to create input-conditioned weights dynamically. In this way, we separate shared information and task-specific knowledge, which are then combined based on input in time.

ference can be significantly alleviated by identifying and separating shared and task-specific knowledge and then dynamically integrating them. So we can leverage the shared knowledge that is beneficial across all tasks, while also taking into account the unique requirements. By dynamically combining these two types of knowledge based on the specific input data, we can create a more flexible and adaptable model that can effectively handle a wide range of tasks.

We validate our method through conventional multi-task model merging experiments and evaluate its generalization and robustness. The results demonstrate the effectiveness of our method and provide a comprehensive understanding.

To summarize, our contributions are as follows:

- We propose a novel method to merge Transformer-based models. Our method is effective in transferring knowledge from various task-specific fine-tuned models and constructing a unified multi-task model.
- We design a novel Weight-Ensembling MoE (WEMoE) module, which can dynamically integrate shared and task-specific knowledge based on the input sample.
- We conduct extensive experiments, and the results demonstrate the effectiveness of our method and provide a comprehensive understanding of our method.

2. Revisiting Model Merge for MTL

In this section, we first introduce the problem setting and notations. Then we revisit the model merge methods for multi-task learning and discuss their limitations.

2.1. Problem Formulation

We begin with a large pre-trained neural network, which is parameterized by $\theta_0 \in \mathbb{R}^{|\theta|}$, which is adaptable to a wide range of downstream tasks through fine-tuning. Given a set of n downstream tasks, denoted as $\mathcal{S} = \{s_i\}_{i=1}^n$, we fine-tune the pre-trained model f individually for each task s_i, resulting in a series of fine-tuned models, each characterized by its unique parameters θ_i. Our goal is to merge the pre-trained model f and the fine-tuned models $\{f_i\}_{i=1}^n$ into a single model f_{merged} that can handle all the tasks in $\mathcal{S}$.

Before providing an overview of our framework and delving into the details, we first need to introduce the concept of task vector, which is a critical element of our method.

Definition 2.1 (Task Vector (Ilharco et al., 2023)**).** Task vector τ_i is defined as the difference between the parameters of the fine-tuned model and the pre-trained model, i.e.,

$$\tau_i = \theta_i - \theta_0. \tag{1}$$

2.2. Revisiting Model Merge

From a multi-objective optimization perspective, the solution space of the optimal merged models is the Pareto front of the downstream tasks in $\mathcal{S}$. In other words, an optimal merged model should be able to achieve the best performance for all tasks in $\mathcal{S}$ simultaneously. It is challenging to find the optimal merged model, as the solution space is large and complex. What's more, we do not have access to the training data of the downstream tasks but only the pre-trained model and the fine-tuned models, which makes it impossible to train a model from scratch as in the traditional multi-task learning setting. We thus need to find a way to transfer the knowledge from the pre-trained model and the fine-tuned models into a unified merged model.

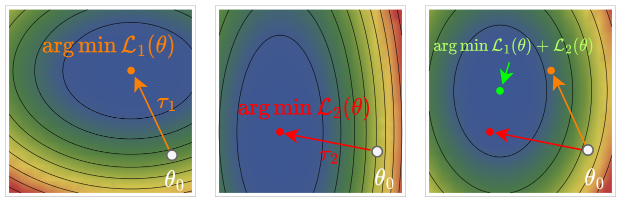
Limitation of static solutions. Current merging methodologies predominantly aim to find a static solution within the original model parameter space. This approach, however, restricts their adaptability to the unique requirements of each instance, as the optimal solution can depend on the input. Finding a Pareto optimal solution in multi-objective optimization can lead to suboptimal performance on individual objectives. As the concept of Pareto optimality is rooted in the idea that a solution is Pareto optimal if there is no static solution that simultaneously improves one objective without degrading at least one other objective. For example, we can not minimize the loss of all tasks better than the global optimal of the joint loss function $\arg\min_\theta \sum_{i=1}^{n} \mathcal{L}_i(\theta)$, where $\mathcal{L}_i$ is the loss function of single downstream task s_i.

This concept is visually represented in Figure 2, where the loss landscapes of s_1, s_2, and $s_1 \cup s_2$ are shown. No static solution θ' that simultaneously satisfies $\mathcal{L}_1(\theta') < \mathcal{L}_1(\theta^*)$ and $\mathcal{L}_2(\theta') < \mathcal{L}_2(\theta^*)$, where $\theta^* = \arg\min_\theta \mathcal{L}_1(\theta)+\mathcal{L}_2(\theta)$. However, θ^* is a suboptimal solution for both tasks individually, as $\mathcal{L}_1(\theta^*) > \min_\theta \mathcal{L}_1(\theta)$ and $\mathcal{L}_2(\theta^*) > \min_\theta \mathcal{L}_2(\theta)$. Visualization of loss landscapes in Figure 7, Appendix A.

It's important to note that there can be instances where two tasks are sufficiently related to positively impact each other, especially in specific multi-task learning and auxiliary-task learning scenarios. However, this heavily depends on the task set and domain. In our study, we primarily focus on situations where negative transfer is a prevalent concern. In our case, tasks from various domains are not closely related, which creates a significant domain gap.

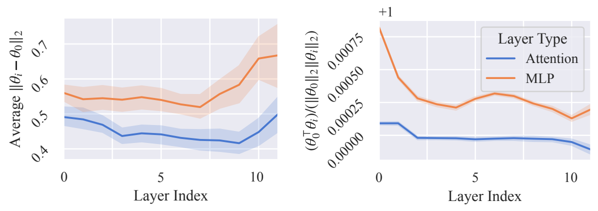
Knowledge separation. Common approaches to knowledge separation include knowledge distillation (Hinton et al., 2015), pruning (Frankle et al., 2021), and feature extraction (Yosinski et al., 2015). It often requires heavy computation to separate or extract the knowledge from deep neural networks. However, knowledge separation is a crucial step for model merging, as it allows us to leverage the shared knowledge that is beneficial across all tasks, while also taking into account the unique requirements of each task. Here, we aim to separate the shared knowledge and task-specific knowledge computationally cheaply and data-freely.

Fortunately, as evident from Eq.(1), the task vector represents the modifications applied to the pre-trained model to optimize it for a specific task s_i. Therefore, it can be interpreted as encapsulating the knowledge specific to that task naturally. On the other hand, the pre-trained model,


$\arg\min \mathcal{L}_1(\theta)$
τ_1
θ_0
$\arg\min \mathcal{L}_2(\theta)$
τ_2
θ_0
$\arg\min \mathcal{L}_1(\theta)+\mathcal{L}_2(\theta)$
θ_0

(a) $\mathcal{L}_1$ (b) $\mathcal{L}_2$ (c) $\mathcal{L}_1 + \mathcal{L}_2$

Figure 2. Illustration of the loss landscapes of s_1, s_2, and $s_1 \cup s_2$. There is no static solution θ' that simultaneously minimizes the loss of both tasks better than $\arg\min_\theta \mathcal{L}_1(\theta) + \mathcal{L}_2(\theta)$.



(a) L_2 length of task vectors (b) cosine similarity

Figure 3. The distance between the parameters of the pre-trained model and the fine-tuned models (CLIP-ViT-B/32 on eight tasks).

characterized by its parameters θ_0, encapsulates the shared knowledge that is relevant across all tasks in the set $\mathcal{S}$. This shared knowledge is derived from the large dataset on which the model was initially trained, and it forms the foundation upon which task-specific adjustments are made.

Similarity in parameter space. Another consideration we must address is the choice of the model segment to undergo knowledge separation. Applying knowledge separation across the entire model might yield a model overly tailored to a specific task or one that is computationally intensive during knowledge reintegration. Conversely, if we limit knowledge separation to a small portion of the model, we may not be able to separate the shared knowledge and task-specific knowledge effectively, leading to a suboptimal merged model. Striking a balance between computational cost and performance necessitates identifying the most valuable segment for knowledge separation.

Empirically, we found that for Transformer models, the weights of the Attention modules in the fine-tuned models have a higher similarity to those in the pre-trained model than MLP modules, consistent with findings in (Lawson & Qureshi, 2023). In Figure 3, we illustrate the distance between the parameters of the pre-trained model and the fine-tuned models (CLIP-ViT-B/32 on eight tasks) using L_2 norm and cosine similarity. As shown in Figure 3(a), the L_2 norm indicates MLPs have higher similarity. However, in Figure 3(b), Attention modules have higher cosine similarity, suggesting a closer relationship. The discrepancy arises because L_2 norm assesses magnitude changes while cosine similarity measures directional alignment. The magnitude of parameters in a neural network plays a crucial role in determining the model's performance and generalization ability. Parameters with large magnitudes can lead to overfitting, where the model learns to fit the noise in the training data, resulting in poor performance on unseen data. On the other hand, parameters with small magnitudes can lead to underfitting, where the model fails to capture the underlying patterns in the data, resulting in suboptimal performance. The L_2 distance directly measures the differences in parameter magnitudes, providing a clear indication of how much the model's performance might change. To this end, we use the L_2 norm to measure the functional similarity between the parameters of fine-tuned and pre-trained models.

3. Methodology

In this section, we provide an overview of our proposed framework before delving into the detailed workings of the Weight-Ensembling Mixture of Experts (MoE) module. Finally, we discuss the test-time adaptation training process.

3.1. Framework Overview

From the previous observations and discussion, our key insight is that we can separate the shared information and task-specific knowledge and combine them dynamically in parameter space based on inputs. Based on this insight, we propose a novel framework to merge the pre-trained model and the fine-tuned models, as shown in Figure 1(a).

So far we have identified the shared information and task-specific knowledge. The next step is to combine them in a way that allows the merged model to handle all the tasks in $\mathcal{S}$. The ability to separate and leverage both knowledge is a key strength of our proposed framework. To achieve this, we propose to dynamically integrate the shared information and task-specific knowledge based on the input samples, rather than seeking a static solution within the original parameter space. This dynamic adjustment allows the model to better adapt to the nuances of each task, thereby improving its performance across source tasks.

To this end, we propose to upcycle the MLPs in the pre-trained model and the fine-tuned models into Weight-Ensembling MoE modules, which are designed to dynamically select task-specific knowledge and combine it with the shared information based on the input samples. We will discuss the details of the Weight-Ensembling MoE module in Section 3.2. For the remaining parts of the model, we utilize the Task Arithmetic method (Ilharco et al., 2023) to merge the weights. Task Arithmetic is a simple yet effective and scalable method for merging models, it operates in the parameter space element-wise without modification to the model structure. The merged model has parameters $\theta = \theta_0 + \lambda \sum_{i=1}^{n} \tau_i$, where λ is a hyperparameter that controls the contribution of the task vectors to the model.

3.2. Weight-Ensembling MoE Module

In this subsection, we delve into the specifics of the Weight-Ensembling Mixture of Experts (MoE) module. As depicted in Figure 1 (b), the Weight-Ensembling MoE module is composed of three main components: the router, the pre-trained MLP weights, and a collection of task vectors.

The router $r : \mathbb{R}^d \to \mathbb{R}^T$, which is essentially a simple MLP, processes the input and subsequently generates a routing weights. These routing weights are used to determine how the knowledge from different tasks is combined. The pre-trained MLP weights θ_0^{MLP} are the weights of the MLPs from the pre-trained model. These weights are crucial as they have been trained to recognize and process a wide range of data patterns, and each finetuned model encompasses this shared information. The task vectors $\{\tau_i^{\text{MLP}} | \theta_i^{\text{MLP}} - \theta_0^{\text{MLP}}\}_{i=1}^{T}$, on the other hand, represent the differences between the MLPs that have been fine-tuned for specific tasks and the pre-trained ones, thus capture the unique adjustments made to the MLPs to optimize them for specific tasks and contain the task-specific knowledge.

To summarize, our proposed Weight-Ensembling MoE module is designed to segregate shared information and task-specific knowledge. This separation allows the module to handle a wide range of tasks without compromising the general applicability of the shared information. The shared information and task-specific knowledge are then dynamically combined based on the input samples.

The mathematical representation of the weight-ensembling MoE module can be expressed as follows:

$$\begin{cases} w = mean(r(\mathbf{h}_{1:N}^{\text{in}})) \in \mathbb{R}^{T\times 1}, \\ \theta^{\text{MLP}} = \theta_0^{\text{MLP}} + \mathbf{D}_\tau w, \\ \mathbf{h}_{1:N}^{\text{out}} = f_{\text{MLP}}(\mathbf{h}_{1:N}^{\text{in}}; \theta^{\text{MLP}}). \end{cases} \tag{2}$$

Where $\mathbf{h}_{1:N}^{\text{in}}$ is the input sequence of tokens, $\mathbf{h}_{1:N}^{\text{out}}$ is the output sequence of tokens, f_{MLP} is the MLP function and $\mathbf{D}_\tau \in \mathbb{R}^{|\theta^{\text{MLP}}|\times T}$ is the dictionary matrix of task vectors, which is made up of fixed task vectors. The $mean$ function averages the routing weights across the tokens in the input sequence. From the perspective of dictionary learning, the Weight-Ensembling MoE module can be viewed as a dictionary look-up operation, where the routing weights are used to select the task vectors from the dictionary matrix $\mathbf{D}_\tau$, which are then added to the pre-trained MLP weights θ_0^{MLP} to create the input-conditioned weights θ^{MLP}.

The structure and initialization of the router. In our weight-ensembling MoE module, the router plays a crucial role. It is responsible for directing the input data to the appropriate combination of expert task vectors. In our research, we employed a straightforward n-layer MLP as the router. During our experiments, we explored two configurations for the MLP: one with no hidden layers ($l = 0$) and another with two hidden layers ($l = 2$). The mathematical representation of these configurations is as follows:

$$r(h) = \mathbf{W}_2\, ReLU(\mathbf{W}_1 h + \mathbf{b}_1) + \mathbf{b}_2, \quad l = 2, \tag{3}$$

and

$$r(h) = \mathbf{b}_0, \quad l = 0. \tag{4}$$

Where $\mathbf{W}$ and $\mathbf{b}$ are the weight and bias, respectively, and $ReLU$ is the rectified linear unit activation function. Unless otherwise specified, we utilize a structure with $l = 2$. In Appendix B.3, we provide an in-depth analysis of the router's structure and its impact on the model's performance.

Initializing the router is crucial to setting up the MoE module, as it can greatly influence its performance. To ensure that the initial routing weights are approximately equal to λ, we initialize $\mathbf{W}_1$ and $\mathbf{W}_2$ by sampling from a Gaussian distribution with a mean of 0 and a variance of 0.01. We then set $\mathbf{b}_1$ and $\mathbf{b}_2$ to zeros and λ, respectively. In the case where $l = 0$, we assign λ to $\mathbf{b}_0$. In fact, for a router with $l = 0$, we can consider it as a partial implementation of AdaMerging (Yang et al., 2023), where only the MLPs undergo adaptive merging weights search.

3.3. Test-Time Adaptation Training

Once the router has been initialized, the next step is to fine-tune its parameters. However, we have no access to the training data of the downstream tasks. To achieve this, we employ test-time adaptation training techniques, which are widely used in the field of semi-supervised learning (Mounsaveng et al., 2023; Liang et al., 2023). Test-time adaptation is a powerful technique that allows the model to adjust its parameters based on the unlabeled test data, thereby improving the performance of the merged model.

In our research, we focus on classification tasks and aim to minimize the multi-task entropy loss of the merged model on the unlabeled test data. Entropy loss is a measure of the uncertainty of the predictions. It is defined as follows:

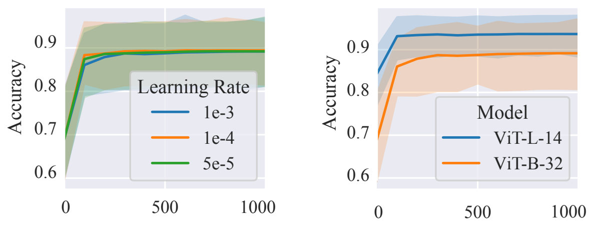
$$\mathcal{L}_{\text{entropy}} = \mathbb{E}_{x\sim\mathcal{D}_{\text{test}}}[-p(\hat{y}|x)\log p(\hat{y}|x)] \tag{5}$$

$$\approx -\frac{1}{|D|}\sum_{i=1}^{|D|}\sum_{c=1}^{C} p(\hat{y}_c|x_i)\log p(\hat{y}_c|x). \tag{6}$$

This is the empirically estimated entropy loss, where $\mathcal{D}_{\text{test}}$ represents the test data distribution, $p(\hat{y}|x)$ is the predicted posterior probability distribution, C is the total number of classes, and $|D|$ is the number of samples in the test dataset. Minimizing it encourages the model to make predictions with higher confidence. This can lead to improved performance on the test data, as the model is more likely to make correct predictions when it is confident in its decisions.

Table 1. Parameter count of the up-scaled models from eight tasks.

MODEL	TRAINABLE	TOTAL	RATIO
	$l = 0$		
CLIP-VIT-B/32	96	566.80M	0.00%
CLIP-VIT-B/16	96	565.15M	0.00%
CLIP-VIT-L/14	192	1.95B	0.00%
	$l = 2$		
CLIP-VIT-B/32	7.16M	573.96M	1.25%
CLIP-VIT-B/16	7.16M	572.31M	1.25%
CLIP-VIT-L/14	25.39M	1.98B	1.28%



(a) Learning rate comparison. (b) Model comparison.

Figure 4. The performance of the merged models with a varying number of steps. (a) CLIP-ViT-B/32 model with different learning rate. (b) Comparison of CLIP-ViT-B/32 and CLIP-ViT-L/14.

4. Experiments

In this section, we conduct experiments to evaluate the performance of our proposed method, including the conventional multi-task model merging experiments, and ablation studies to evaluate the generalization ability and robustness. The code is available at `https://github.com/tanganke/weight-ensembling_MoE`.

4.1. Experimental Setup

We employ CLIP (Radford et al., 2021) as our pre-trained models, which are trained on a large-scale dataset with pairs of images and corresponding textual descriptions, and are able to perform open-vocabulary image classification. Subsequently, we fine-tune the models on eight distinct image classification tasks, namely SUN397 (Xiao et al., 2010), Stanford Cars (Krause et al., 2013), RESISC45 (Cheng et al., 2017), EuroSAT (Helber et al., 2018), SVHN (Netzer et al., 2021), GTSRB (Stallkamp et al., 2012), MNIST (Lecun et al., 1998), and DTD (Cimpoi et al., 2014). The evaluation of model performance is based on top-1 accuracy.

We compare our method with Fisher Merging (Matena & Raffel, 2022), RegMean (Jin et al., 2023), Task Arithmetic (Ilharco et al., 2023), Ties-Merging (Yadav et al., 2023), and AdaMerging/Adamerging++ (Yang et al., 2023). For all methods, unless explicitly specified, we follow the configuration in (Yang et al., 2023) and initialize the scaling coefficient of the task vector, denoted as λ, to 0.3. In

Table 2. Multi-task performance when merging CLIP-ViT-B/32 models on all eight tasks.

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg.
Pre-trained	63.2	59.6	60.2	45.0	31.6	32.6	48.3	44.4	48.1
Individual	**75.3**	**77.7**	**96.1**	**99.9**	**97.5**	98.7	**99.7**	**79.4**	**90.5**
Traditional MTL	73.9	74.4	93.9	98.2	95.8	**98.9**	99.5	77.9	88.9
Multi-Task Model Fusion Methods									
Weight Averaging	65.3	63.3	71.4	73.6	64.2	52.8	87.5	50.1	66.0
Fisher Merging	68.6	69.2	70.7	66.4	72.9	51.1	87.9	59.9	68.3
RegMean	65.3	63.5	75.6	78.6	78.1	67.4	93.7	52.0	71.8
Task Arithmetic	55.3	54.9	66.7	77.4	80.2	69.7	97.3	50.1	69.0
Ties-Merging	65.0	64.3	74.7	76.8	81.3	69.4	96.5	54.3	72.8
AdaMerging (task)	58.3	53.2	71.8	80.1	81.6	84.4	93.4	42.7	70.7
AdaMerging++ (task)	60.8	56.9	73.1	83.4	87.3	82.4	95.7	50.1	73.7
AdaMerging (layer)	64.2	69.5	82.4	92.5	86.5	93.7	97.7	61.1	80.9
AdaMerging++ (layer)	66.6	68.3	82.2	94.2	89.6	89.0	98.3	60.6	81.1
WEMoE (Ours)	**74.1**	**77.4**	**93.7**	**99.1**	**96.2**	**98.9**	**99.6**	**76.4**	**89.4**

Table 3. Multi-task performance when merging CLIP-ViT-L/14 models on all eight tasks.

Method	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg.
Pre-trained	68.2	77.9	71.3	61.3	58.4	50.6	76.4	55.4	64.9
Individual	**82.3**	**92.4**	**97.4**	**99.9**	**98.1**	**99.2**	**99.7**	84.1	**94.1**
Traditional MTL	80.8	90.6	96.3	96.3	97.6	99.1	99.6	**84.4**	93.5
Multi-Task Model Fusion Methods									
Weight Averaging	72.1	81.6	82.6	91.4	78.2	70.6	97.0	62.8	79.5
Fisher Merging	69.2	88.6	87.5	93.5	80.6	74.8	93.3	70.0	82.2
RegMean	73.3	81.8	86.1	97.0	88.0	84.2	98.5	60.8	83.7
Task Arithmetic	74.1	82.1	86.7	92.6	87.9	86.8	98.9	65.6	84.4
Ties-Merging	75.0	84.5	88.0	94.3	85.7	82.1	98.7	67.7	84.5
AdaMerging (layer)	79.0	90.3	90.8	96.2	93.4	98.0	99.0	79.9	90.8
AdaMerging++ (layer)	79.4	90.3	91.6	97.4	93.4	97.5	99.0	79.2	91.0
WEMoE (Ours)	**81.4**	**92.6**	**95.4**	**99.4**	**97.7**	**99.3**	**99.7**	**83.7**	**93.6**

Appendix B.1, we provide a comparison of the baselines.

4.2. Conventional Multi-Task Model Merging

Firstly, we conduct conventional multi-task model merging experiments to evaluate the performance of our proposed method. In Tables 1 and 8, we provides a detailed comparison of the parameter counts of the up-scaled models from eight tasks. The models compared include CLIP-ViT-B/32, CLIP-ViT-B/16, and CLIP-ViT-L/14, with two different router depths ($l = 0$ and $l = 2$). The experimental results are outlined in Table 2 and Table 3, respectively. We have the following key observations:

(1) The performance of fine-tuned models on downstream tasks is significantly better than that of the pre-trained model, indicating that the fine-tuned models have learned task-specific knowledge. (2) For larger models, the fused performance tends to be better across all merging methods. This is because the larger model scale introduces more redundancy among parameters. Consequently, even simple averaging can achieve better performance on larger models. (3) Our proposed approaches, WEMoE, consistently demonstrate the superiority of our approach over SOTA methods across the majority of tasks. (4) In particular, WEMoE outperforms the traditional MTL baseline by 0.5% and 0.1% on average. This is surprising as the traditional MTL baseline is trained on all tasks simultaneously while merging methods are inaccessible to the training data. This indicates that through our approach, we have successfully separated the task-specific knowledge from fine-tuned models. On the other hand, the weight-ensembling MoE module successfully captures the relationship between the input features and knowledge combination, effectively alleviating the mutual influence of parameters, which significantly enhances the overall performance and flexibility of the model.

Convergence and sensitivity analysis. Figure 4 shows the performance of the merged WEMoE models with varying number of steps. In Figure 4a, we merge CLIP-ViT-B/32 models with different learning rate configurations. We observe that the performance of the merged model shows an upward trend with an increase in the number of training steps, and it converges rapidly, reaching a high accuracy level in just 200 steps. Furthermore, the influence of dif-

Table 4. Generalization results on two unseen tasks when merging ViT-B/32 models on six tasks.

Method	Seen Tasks							Unseen Tasks		
	SUN397	Cars	RESISC45	DTD	SVHN	GTSRB	Avg.	MNIST	EuroSAT	Avg.
Task Arithmetic	63.4	62.3	75.3	57.8	84.7	80.4	70.7	77.3	45.6	61.4
Ties-Merging	67.8	66.2	77.0	56.2	77.2	71.0	69.2	75.9	43.1	59.5
AdaMerging	65.2	65.9	88.5	61.1	92.2	91.5	77.4	84.0	**56.1**	70.0
AdaMerging++	68.2	67.6	86.3	63.6	92.6	89.8	78.0	83.9	53.5	68.7
WEMoE (0-layer)	63.8	61.2	78.4	56.4	89.1	92.2	73.5	78.6	49.7	64.2
WEMoE (2-layer)	**74.4**	**78.3**	**94.8**	**75.6**	**96.8**	**99.0**	**86.5**	**86.3**	55.9	**71.1**

Table 5. Ablations of the test data distribution on ViT-B/32 (for all methods, $\lambda = 0.3$).

Method	Cars	EuroSAT	RESISC45	GTSRB	Avg.	Cars	EuroSAT	RESISC45	GTSRB	Avg.
	Clean Test Set					Corrupted Test Set (Motion Blur)				
Task Arithmetic	66.9	94.7	82.6	75.1	79.8	65.3	68.1	80.0	64.2	69.4
Ties-Merging	67.5	83.7	79.8	65.3	74.1	65.6	57.5	77.5	55.4	64.0
AdaMerging	73.7	96.1	85.8	96.3	88.0	71.2	74.6	82.7	94.1	80.6
WEMoE (0-layer)	68.5	94.6	82.0	93.2	84.6	65.3	74.1	79.8	89.0	77.0
WEMoE (2-layer)	**78.8**	**99.5**	**95.4**	**99.1**	**93.2**	**78.1**	**79.7**	**94.6**	**97.8**	**87.6**
	Corrupted Test Set (Impluse Noise)					Corrupted Test Set (Gaussian Noise)				
Task Arithmetic	62.1	**49.1**	72.7	40.4	56.1	63.6	**55.4**	75.9	49.4	61.1
Ties-Merging	62.1	42.0	70.4	34.9	52.3	64.1	50.3	74.5	39.8	57.2
AdaMerging	67.2	30.8	75.9	77.5	62.8	69.9	41.2	80.6	76.0	66.9
WEMoE (0-layer)	63.8	40.7	74.3	66.7	61.4	66.1	45.5	78.3	66.9	64.2
WEMoE (2-layer)	**74.7**	11.6	**91.4**	**91.7**	**67.3**	**76.8**	29.7	**93.2**	**78.2**	**69.5**
	Corrupted Test Set (Pixelate)					Corrupted Test Set (Spatter)				
Task Arithmetic	2.8	41.5	22.8	66.6	33.4	63.3	**60.1**	73.9	54.3	62.9
Ties-Merging	**4.1**	40.8	20.6	57.1	30.6	64.4	50.8	71.4	44.3	57.8
AdaMerging	2.5	**53.8**	22.4	90.6	**42.3**	69.9	43.6	75.4	89.4	69.6
WEMoE (0-layer)	2.1	52.7	**23.4**	84.5	40.7	65.8	49.3	72.9	83.3	67.8
WEMoE (2-layer)	0.4	9.6	2.2	**97.0**	27.3	**76.2**	28.2	**91.2**	**96.0**	**72.9**
	Corrupted Test Set (Contrast)					Corrupted Test Set (JPEG Compression)				
Task Arithmetic	66.0	62.9	75.9	70.6	68.9	66.5	72.3	82.2	60.0	70.3
Ties-Merging	66.8	53.4	75.9	61.5	64.4	67.5	60.4	80.0	50.1	64.5
AdaMerging	71.7	69.8	79.3	95.1	79.0	70.9	75.8	83.6	90.1	80.1
WEMoE (0-layer)	67.3	68.5	74.8	91.4	75.5	66.4	75.3	81.4	83.1	76.5
WEMoE (2-layer)	**77.7**	**77.4**	**93.9**	**98.5**	**86.9**	**78.2**	**80.7**	**95.1**	**96.2**	**87.6**

ferent learning rates is not significant, suggesting that our method is insensitive to the learning rate parameter. This is a desirable property as it reduces the need for hyperparameter tuning. In Figure 4b, we compare the performance of CLIP-ViT-B/32 and CLIP-ViT-L/14 models.

Ablations of the up-scaling strategy. To further investigate the impact of the up-scaling strategy, we conduct experiments on CLIP-ViT-B/32. We compare three different up-scaling strategies: (1) Entire Transformer Block: The MoE up-scaling is applied on the entire Transformer block. (2) Attention + MLP (separately): The MoE up-scaling is applied on the attention weights and MLP weights separately. (3) MLP only: The MoE up-scaling is applied on the MLP weights only, as we have done in the main experiments.

In Table 6, we present the results of this ablation study. When both attention and MLP weights are up-scaled separately, the performance is significantly improved across most tasks, surpassing the other two methods. Recall that the decision to apply the routing network solely to the MLP weights was intentional, driven by the empirical finding that the MLP weights show less similarity between fine-tuned and pre-trained models than the attention weights (Section 2.2). This implies that MLPs may contain more task-specific information, making them an ideal target for the routing mechanism. If the routing network were applied to both MLPs and attention weights, it would result in increased computational load and memory usage, and the marginal performance gain might not warrant the cost.

4.3. Generalization and Robustness Evaluation

When we engage in merging multi-task models, our primary objective is to enhance the model's performance on the seen task. However, it is also crucial to explore the model's ability to generalize across diverse data distributions and assess the robustness of the merging algorithm when faced with shifts in test distribution. This refers to scenarios where the distribution of test data deviates from that of the training

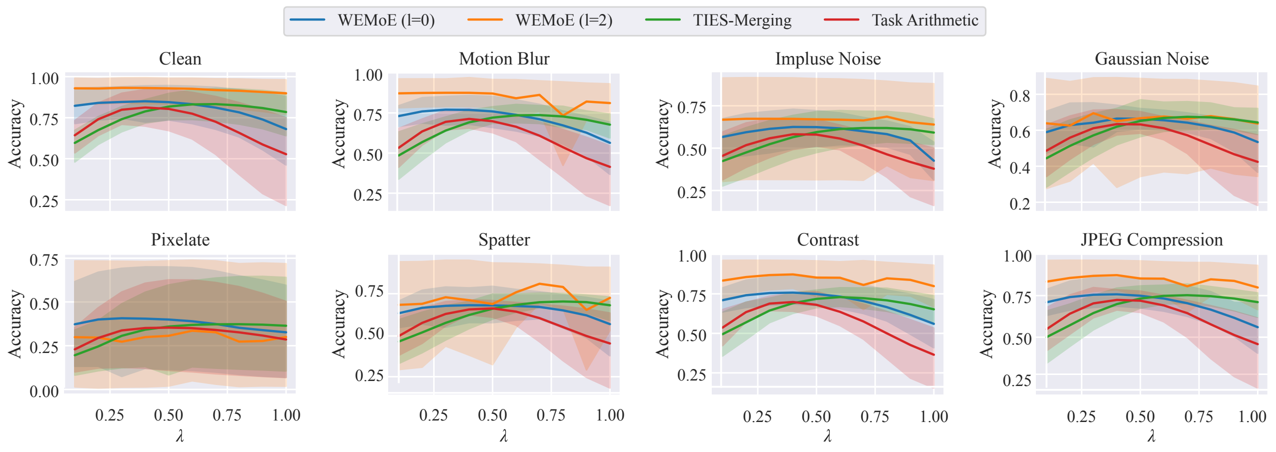


Figure 5. The results for robustness experiment on CLIP-ViT-B/32. The x-axis of each plot represents the scaling coefficient λ of task vectors, while the y-axis shows the accuracy of the merged model on different merged tasks.

Table 6. Comparison of different up-scaling strategies on various tasks.

Method (1-layer router)	SUN397	Cars	RESISC45	EuroSAT	SVHN	GTSRB	MNIST	DTD	Avg.
Entire Transformer Block	70.5	74.0	91.0	97.0	93.6	96.4	99.1	70.0	86.4
Attention + MLP (separately)	**73.5**	**78.4**	**94.0**	98.3	**96.2**	98.4	**99.6**	**75.6**	**89.2**
MLP only	73.2	76.7	93.8	**98.6**	95.7	**98.6**	99.5	74.5	88.3

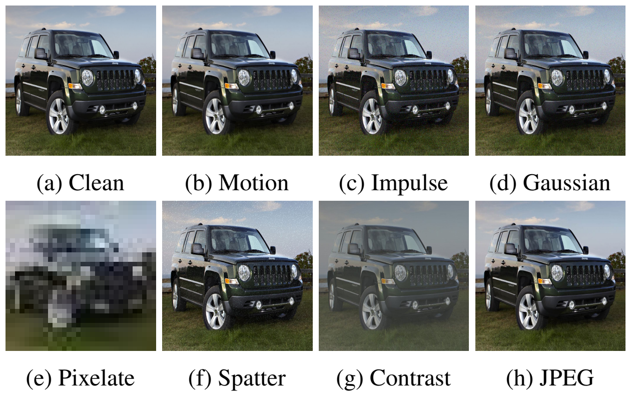


Figure 6. Here are eight instances of distorted images, produced using the method suggested in (Hendrycks & Dietterich, 2019).

data, which is common in real-world applications.

Generalization experiments. To assess the model's generalization, we selected two tasks from the eight downstream tasks as unseen tasks. We utilized the fine-tuned models on the remaining six tasks for merging, constructing a six-task model. This six-task model was then applied to the unseen tasks to evaluate the model's generalization capability. We conducted experiments on both 0-layer routers and 2-layer routers, and the results are presented in Table 4.

Robustness experiments. To evaluate the robustness of the merging algorithm, we utilized the methods suggested in (Hendrycks & Dietterich, 2019) to generate distorted images from the clean test set. We selected seven types of distortions, including motion blur, impulse noise, gaussian noise, pixelate, spatter, contrast, and JPEG compression. We then merge and evaluate the performance of the merged models on the distorted images. We compared routers of two different depths, see Eq. (4) and Eq. (3) for details. We conduct experiments on CLIP-ViT-B/32 and CLIP-VIT-B/16, the results are presented in Table 5 and Table 11, respectively. Figure 6 shows eight instances of distorted images.

Our experimental results show that across various methods, WEMoE ($l = 2$) consistently achieves the highest performance on the clean test set and most of the distorted test sets. This indicates that our method is effective in handling both clean and distorted data. The superior performance suggests its potential for robust multi-task merging. However, in scenarios where there is a significant loss in image quality, such as pixelation, WEMoE ($l = 2$) might overfit some specific tasks, resulting in a performance decline. In contrast, WEMoE ($l = 0$) tends to exhibit more stability in performance under such conditions. This is attributed to its lower parameter count, making it less prone to overfitting.

In addition, we observe that the performance of WEMoE

($l = 0$) is comparable to AdaMerging, recalling that when $l = 0$, WEMoE is essentially equivalent to a partial implementation of AdaMerging, with the scaling factor λ fixed at 0.3 for the majority of the model, except for MLP modules. This observation validates the hypothesis we presented in Section 2.2 regarding parameter similarity.

We also analyze the performance of our method with different scaling coefficients λ of the task vector. The results are presented in Figure 5 and Figure 10, respectively. We observe that the performance of the merged model is relatively stable with respect to the scaling coefficient λ of the task vector. This indicates that our method is also more robust to the scaling coefficient λ of the task vector.

4.4. Routing Analysis

Here we perform a small analysis of the expert weighting by the router. We use the CLIP-ViT-B/32 model as an example, and the results are presented in Figure 11. Our examination reveals a tendency of the router to allocate a greater weight to the task vector corresponding to the source task of the input sample. This indicates that the routers are able to effectively identify the expert with the highest performance depending on the input feature, and assign a higher weight to it. Such behavior aligns with our expectations and intuition. Additional details can be found in Appendix D.

5. Related Work

In this section, we review recent related works on multi-task model fusion and mixture of experts (MoE).

Multi-Task Model Fusion. Weight interpolation proves to be a straightforward yet powerful strategy, facilitating scalable model fusion without the need for extensive computations (Izmailov et al., 2019; Matena & Raffel, 2022; Wortsman et al., 2022; Kaddour, 2022; Ilharco et al., 2023; Yadav et al., 2023; Yang et al., 2023; Wu et al., 2023). However, this strategy also poses challenges, particularly when merging models with diverse structures.

Mode connectivity has been uncovered by insights into the loss landscape (Daniel Freeman & Bruna, 2017; Nagarajan & Kolter, 2019). This phenomenon reveals that different solutions can be connected by a pathway within the parameter space, maintaining low objective function values along the path, thus facilitating model fusion (Draxler et al., 2019; Frankle et al., 2020; Entezari et al., 2022). Different methods such as discovering simple linear paths (Garipov et al., 2018), non-linear trajectories (Tatro et al., 2020), or mappings within a lower dimension (Yunis et al., 2022; Benton et al., 2021) have been proposed to leverage this concept.

Alignment emerges as another effective series of works. This approach involves aligning and interpolating corresponding components from various models to mitigate disparities (Li et al., 2016; Tatro et al., 2020). Techniques include matching activations or weights (George Stoica et al., 2023; Jin et al., 2023), employing graph matching for channel-wise alignment (Liu et al., 2022), or exploiting the principle of permutation invariance (Ainsworth et al., 2023).

Mixture of Experts. The Mixture of Experts (MoE) model, first introduced by (Jacobs et al., 1991), is a machine learning technique that involves training multiple models, each of which specializes in a different part of the input space. Over the years, MoE has garnered considerable attention (Jiang et al., 2024; Dai et al., 2024). Much innovation revolves around the design of more efficient routers. For instance, the Switch Transformer (Fedus et al., 2022b) simplifies the selection process by choosing only the top expert per token, showcasing superior scalability compared to prior approaches. Base Layers (Lewis et al., 2021) introduces a linear assignment that optimizes token-expert affinities, ensuring an equal distribution of tokens among experts. In addition to methods that involve routers selecting experts, there are alternative approaches such as allowing each expert to choose tokens (Zhou et al., 2022). For a comprehensive review of MoE, readers can refer to (Fedus et al., 2022a).

6. Conclusion

In this paper, we propose a novel method to merge Transformer-based vision models from different tasks. Our method is effective in transferring knowledge from various task-specific fine-tuned models and constructing a unified multi-task model. We propose a novel Weight-Ensembling MoE (WEMoE) module, which can dynamically integrate shared and task-specific knowledge based on the input sample. We conduct extensive experiments, the experimental results demonstrate the effectiveness of our method and provide a comprehensive understanding of our method.

In the future, we plan to explore the potential of our method in other scenarios, such as merging transformers from different modality. We also plan to investigate the possibility of applying our method to other architectures, such as CNNs. It's also interesting to combine our method with parameter-efficient fine-tuning methods such as Adapter tuning (Houlsby et al., 2019) and LoRA (Hu et al., 2021) to further improve the efficiency of our method.

Acknowledgements

This work is supported in part by STI 2030—Major Projects (No. 2021ZD0201405), the National Natural Science Foundation of China (Grant No. 62276195 and U23A20318), the Special Fund of Hubei Luojia Laboratory under Grant 220100014, and the Fundamental Research Funds for the Central Universities (No. 2042024kf0039). Dr Tao's research is partially supported by NTU RSR and Start Up Grants.

Impact Statement

This work presents a novel method for merging Transformer-based models, aiming to advance the field of machine learning, particularly in the area of multi-task learning. The proposed method has the potential to significantly improve the efficiency and scalability of multi-task learning systems, thereby enabling more effective knowledge transfer and utilization. The societal implications of this work are manifold. On the positive side, the proposed method could lead to more efficient and effective machine learning systems, which could in turn lead to advancements in various fields where machine learning is applied, such as healthcare, education, and technology. This could potentially result in improved services and products, benefiting society at large.

References

Ainsworth, S. K., Hayase, J., and Srinivasa, S. Git Re-Basin: Merging Models modulo Permutation Symmetries, March 2023. URL http://arxiv.org/abs/2209.04836.

Benton, G. W., Maddox, W. J., Lotfi, S., and Wilson, A. G. Loss Surface Simplexes for Mode Connecting Volumes and Fast Ensembling, November 2021. URL http://arxiv.org/abs/2102.13042.

Cao, B., Lin, H., Han, X., and Sun, L. The life cycle of knowledge in big language models: A survey. *Mach. Intell. Res.*, 21(2):217–238, 2024. doi: 10.1007/S11633-023-1416-X. URL https://doi.org/10.1007/s11633-023-1416-x.

Cheng, G., Han, J., and Lu, X. Remote Sensing Image Scene Classification: Benchmark and State of the Art. *Proceedings of the IEEE*, 105(10):1865–1883, October 2017. ISSN 0018-9219, 1558-2256. doi: 10.1109/JPROC.2017.2675998. URL http://arxiv.org/abs/1703.00121. arXiv:1703.00121 [cs].

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling Instruction-Finetuned Language Models, December 2022. URL http://arxiv.org/abs/2210.11416.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. 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.

Dai, D., Deng, C., Zhao, C., Xu, R. X., Gao, H., Chen, D., Li, J., Zeng, W., Yu, X., Wu, Y., Xie, Z., Li, Y. K., Huang, P., Luo, F., Ruan, C., Sui, Z., and Liang, W. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. 2024.

Daniel Freeman, C. and Bruna, J. Topology and geometry of half-rectified network optimization: 5th International Conference on Learning Representations, ICLR 2017. 2017. URL http://www.scopus.com/inward/record.url?scp=85064823226&partnerID=8YFLogxK.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. A. Essentially No Barriers in Neural Network Energy Landscape, February 2019. URL http://arxiv.org/abs/1803.00885.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The Role of Permutation Invariance in Linear Mode Connectivity of Neural Networks, July 2022. URL http://arxiv.org/abs/2110.06296.

Fedus, W., Dean, J., and Zoph, B. A Review of Sparse Expert Models in Deep Learning, September 2022a. URL http://arxiv.org/abs/2209.01667.

Fedus, W., Zoph, B., and Shazeer, N. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, June 2022b. URL http://arxiv.org/abs/2101.03961.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Linear Mode Connectivity and the Lottery Ticket Hypothesis, July 2020. URL http://arxiv.org/abs/1912.05671.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Pruning Neural Networks at Initialization: Why are We Missing the Mark?, March 2021. URL http://arxiv.org/abs/2009.08576.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D., and Wilson, A. G. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs, October 2018. URL http://arxiv.org/abs/1802.10026.

George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. ZipIt! Merging Models from Different Tasks without Training, May 2023. URL http://arxiv.org/abs/2305.03053.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked Autoencoders Are Scalable Vision Learners, December 2021. URL http://arxiv.org/abs/2111.06377.

Helber, P., Bischke, B., Dengel, A., and Borth, D. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 204–207. IEEE, 2018.

Hendrycks, D. and Dietterich, T. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, March 2019. URL http://arxiv.org/abs/1903.12261.

Hinton, G., Vinyals, O., and Dean, J. Distilling the Knowledge in a Neural Network, March 2015. URL http://arxiv.org/abs/1503.02531.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-Efficient Transfer Learning for NLP, June 2019. URL http://arxiv.org/abs/1902.00751.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. URL http://arxiv.org/abs/2106.09685.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing Models with Task Arithmetic, March 2023. URL http://arxiv.org/abs/2212.04089.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging Weights Leads to Wider Optima and Better Generalization, February 2019. URL http://arxiv.org/abs/1803.05407.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, March 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL https://ieeexplore.ieee.org/document/6797059.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of Experts, January 2024. URL http://arxiv.org/abs/2401.04088.

Jin, X., Ren, X., Preotiuc-Pietro, D., and Cheng, P. Dataless Knowledge Fusion by Merging Weights of Language Models, April 2023. URL http://arxiv.org/abs/2212.09849.

Kaddour, J. Stop Wasting My Time! Saving Days of ImageNet and BERT Training with Latest Weight Averaging, October 2022. URL http://arxiv.org/abs/2209.14981.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3D Object Representations for Fine-Grained Categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pp. 554–561, December 2013. doi: 10.1109/ICCVW.2013.77. URL https://ieeexplore.ieee.org/document/6755945.

Lawson, D. and Qureshi, A. H. Merging Decision Transformers: Weight Averaging for Forming Multi-Task Policies, September 2023. URL http://arxiv.org/abs/2303.07551.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. ISSN 00189219. doi: 10.1109/5.726791. URL http://ieeexplore.ieee.org/document/726791/.

Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., and Zettlemoyer, L. BASE Layers: Simplifying Training of Large, Sparse Models. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6265–6274. PMLR, July 2021. URL https://proceedings.mlr.press/v139/lewis21a.html.

Li, W., Peng, Y., Zhang, M., Ding, L., Hu, H., and Shen, L. Deep Model Fusion: A Survey, September 2023. URL http://arxiv.org/abs/2309.15698.

Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. Convergent Learning: Do different neural networks learn the same representations?, February 2016. URL http://arxiv.org/abs/1511.07543.

Liang, J., He, R., and Tan, T. A Comprehensive Survey on Test-Time Adaptation under Distribution Shifts, March 2023. URL http://arxiv.org/abs/2303.15361.

Lin, Y., Gao, Y., Gong, M., Zhang, S., Zhang, Y., and Li, Z. Federated learning on multimodal data: A comprehensive survey. *Mach. Intell. Res.*, 20(4):539–553, 2023. doi: 10.1007/S11633-022-1398-0. URL https://doi.org/10.1007/s11633-022-1398-0.

Liu, C., Lou, C., Wang, R., Xi, A. Y., Shen, L., and Yan, J. Deep Neural Network Fusion via Graph Matching with

Applications to Model Ensemble and Federated Learning. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 13857–13869. PMLR, June 2022. URL https://proceedings.mlr.press/v162/liu22k.html.

Matena, M. and Raffel, C. Merging Models with Fisher-Weighted Averaging, August 2022. URL http://arxiv.org/abs/2111.09832.

Mounsaveng, S., Chiaroni, F., Boudiaf, M., Pedersoli, M., and Ayed, I. B. Bag of Tricks for Fully Test-Time Adaptation, October 2023. URL http://arxiv.org/abs/2310.02416.

Nagarajan, V. and Kolter, J. Z. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading Digits in Natural Images with Unsupervised Feature Learning. 2021.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language Models are Unsupervised Multitask Learners. 1:9, 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning Transferable Visual Models From Natural Language Supervision, February 2021. URL http://arxiv.org/abs/2103.00020.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, August 2012. ISSN 0893-6080. doi: 10.1016/j.neunet.2012.02.016. URL https://www.sciencedirect.com/science/article/pii/S0893608012000457.

Tang, A., Luo, Y., Hu, H., He, F., Su, K., Du, B., Chen, Y., and Tao, D. Improving Heterogeneous Model Reuse by Density Estimation. In *Thirty-Second International Joint Conference on Artificial Intelligence*, volume 4, pp. 4244–4252, August 2023a. doi: 10.24963/ijcai.2023/472. URL https://www.ijcai.org/proceedings/2023/472.

Tang, A., Shen, L., Luo, Y., Ding, L., Hu, H., Du, B., and Tao, D. Concrete Subspace Learning based Interference Elimination for Multi-task Model Fusion, December 2023b. URL http://arxiv.org/abs/2312.06173.

Tatro, N., Chen, P.-Y., Das, P., Melnyk, I., Sattigeri, P., and Lai, R. Optimizing Mode Connectivity via Neuron Alignment. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15300–15311. Curran Associates, Inc., 2020.

Wang, X., Chen, G., Qian, G., Gao, P., Wei, X., Wang, Y., Tian, Y., and Gao, W. Large-scale multi-modal pre-trained models: A comprehensive survey. *Mach. Intell. Res.*, 20(4):447–482, 2023. doi: 10.1007/S11633-022-1410-8. URL https://doi.org/10.1007/s11633-022-1410-8.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., and Schmidt, L. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, July 2022. URL http://arxiv.org/abs/2203.05482.

Wu, C., Wang, T., Ge, Y., Lu, Z., Zhou, R., Shan, Y., and Luo, P. π-tuning: transferring multimodal foundation models with optimal multi-task interpolation. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. SUN database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492, San Francisco, CA, USA, June 2010. IEEE. ISBN 978-1-4244-6984-0. doi: 10.1109/CVPR.2010.5539970. URL http://ieeexplore.ieee.org/document/5539970/.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. Resolving Interference When Merging Models, June 2023. URL http://arxiv.org/abs/2306.01708.

Yang, E., Wang, Z., Shen, L., Liu, S., Guo, G., Wang, X., and Tao, D. AdaMerging: Adaptive Model Merging for Multi-Task Learning, October 2023. URL http://arxiv.org/abs/2310.02575.

Ye, H. and Xu, D. TaskExpert: Dynamically Assembling Multi-Task Representations with Memorial Mixture-of-Experts, July 2023. URL http://arxiv.org/abs/2307.15324.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. Understanding Neural Networks Through Deep Visualization, June 2015. URL http://arxiv.org/abs/1506.06579.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch, November 2023. URL http://arxiv.org/abs/2311.03099.

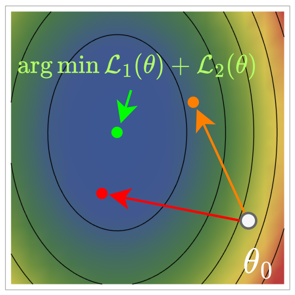
Yunis, D., Patel, K. K., Savarese, P., Vardi, G., Livescu, K., Walter, M., Frankle, J., and Maire, M. On Convexity and Linear Mode Connectivity in Neural Networks. *OPT2022: 14th Annual Workshop on Optimization for Machine Learning*, 2022.

Zheng, H., Shen, L., Tang, A., Luo, Y., Hu, H., Du, B., and Tao, D. Learn From Model Beyond Fine-Tuning: A Survey, October 2023. URL http://arxiv.org/abs/2310.08184.

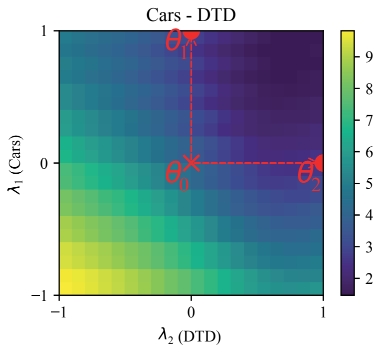
Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A., Chen, Z., Le, Q., and Laudon, J. Mixture-of-Experts with Expert Choice Routing, October 2022. URL http://arxiv.org/abs/2202.09368.

The appendix is organized into several sections, each providing additional insights and details related to different aspects of the main work.

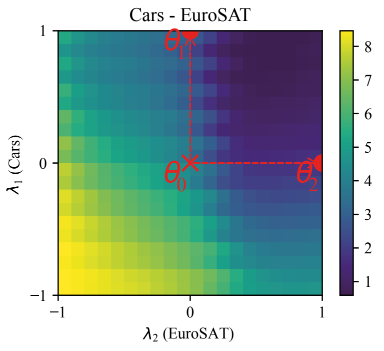
A. Loss Landscape Visualization



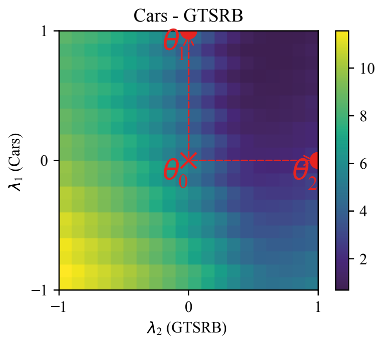
(a) $\mathcal{L}_1 + \mathcal{L}_2$



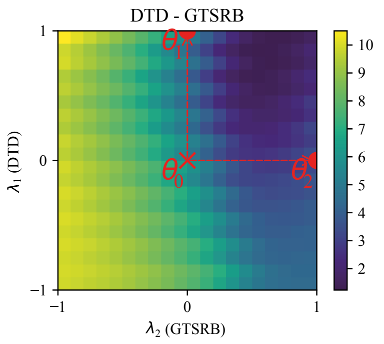
(b) Cars-DTD



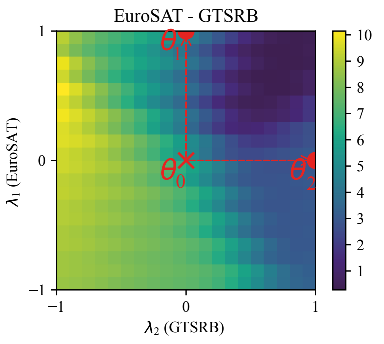
(c) Cars-EuroSAT



(d) Cars-GTSRB



(e) DTD-GTSRB



(f) EuroSAT-GTSRB

Figure 7. Visualization of the joint loss $\mathcal{L}_1 + \mathcal{L}_2$ and five task pairs for CLIP-ViT-B/32 in the loss landscape. We perform interpolations between pre-trained weights and two fine-tuned weights in the weight space on a 2D plane using the formula $\theta = \theta_0 + \lambda_1\tau_1 + \lambda_2\tau_2$, where θ_0 represents pre-trained weights, $\tau_i = \theta_i - \theta_0$ are two task vectors with λ_i in the range [-1, 1].

In Section 2.2, we highlight the inherent limitation of a static solution for multi-task model merging, emphasizing its potential to result in suboptimal performance on individual tasks. To further illustrate this point, we present the visualization of the loss landscape for the joint loss $\mathcal{L}_1 + \mathcal{L}_2$ and five task pairs using CLIP-ViT-B/32 in Figure 7. The visualization involves interpolations between pre-trained weights and two fine-tuned weights in the weight space on a 2D plane, expressed as $\theta = \theta_0 + \lambda_1\tau_1 + \lambda_2\tau_2$, where θ_0 represents pre-trained weights, and $\tau_i = \theta_i - \theta_0$ are two task vectors with λ_i in the range [-1, 1]. The resulting heatmaps demonstrate that task-specific models fine-tuned from the same pre-trained model share the same loss basin when evaluated on the joint task, yet none of them attains the global optimum.

B. Multi-Task Model Fusion

B.1. Baseline methods

Table 7. Comparison of different methods and their data access requirements.

Method	Labeled tasks data	Validation data (labeled)	Test time adaptation
Fisher Merging (Matena & Raffel, 2022)	Yes	No	No
RegMean (Jin et al., 2023)	Yes	No	No
Task Arithmetic (Ilharco et al., 2023)	No	Yes	No
Ties-Merging (Yadav et al., 2023)	No	Yes	No
AdaMerging (Yang et al., 2023)	No	No	Yes
Ours	No	No	Yes

Here we provide a summary of the baseline methods used in the experiments conducted in this study.

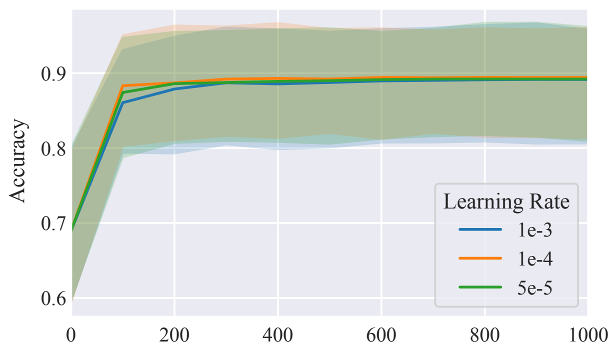
- **Simple Weight Average**: This method involves taking the average of the weights of models that have been fine-tuned on different tasks. It is sometimes referred to as ModelSoups in relevant literature. When applied to fully fine-tuned models, the weights of the models are directly averaged. This is mathematically represented as $\theta = 1/n \sum_{i=1}^{n} \theta_i$, where θ represents the average weight, n is the total number of models, and θ_i is the weight of the i-th model.
- **Task Arithmetic**: This method involves calculating a task vector for each task and then adding these vectors together to create a multi-task vector. This multi-task vector is then scaled by a coefficient λ and added to the initial parameters of the pre-trained model to create a multi-task model. The formula for this is $\theta = \theta_0 + \lambda \sum_i (\theta_i - \theta_0)$, where θ_0 is the initial parameter of the pre-trained model, θ_i is the task vector for the i-th task, and λ is a hyperparameter chosen based on the model's performance on a validation set. In this study, λ was set to 0.3.
- **Ties-Merging**: This algorithm follows three steps (trim, elect sign of parameters, and disjoint merge) to obtain a merged task vector ν. Given the final merged task vector τ, the final model is chosen in a similar way as task arithmetic, i.e. $\theta = \theta_0 + \lambda\tau$, where λ is a hyperparameter that the best-performing model is chosen on the validation set. In our study, λ is chosen to be 0.3.
- **Fisher Merging**: This method requires access to some labeled data from all tasks in order to estimate the Fisher information matrix. The Fisher information matrix is then used to assess the importance of each task and to merge the models by computing a weighted average of the models' weights.
- **RegMean**: This method also requires access to some labeled data from all tasks, but it is used to compute the Gram matrix. The Gram matrix is a matrix that represents the inner products of vectors in a set.
- **AdaMerging**: AdaMerging is an adaptive model merging method where it autonomously learns the coefficients for merging either on a task-wise or layer-wise basis, using entropy minimization on unlabeled test samples as a surrogate objective function to refine the merging coefficients.
  - The task-wise AdaMerging is formulated as $\theta = \theta_0 + \sum_{i=1}^{n} \lambda_i \tau_i$ where λ_k is the merging coefficient for the k-th task and τ_k is the task vector for the k-th task.
  - The layer-wise AdaMerging is formulated as $\theta^l = \theta_0^l + \sum_{i=1}^{n} \lambda_i^l \tau_i^l$. Where the superscript l denotes the layer index.

B.2. Image Classification Tasks

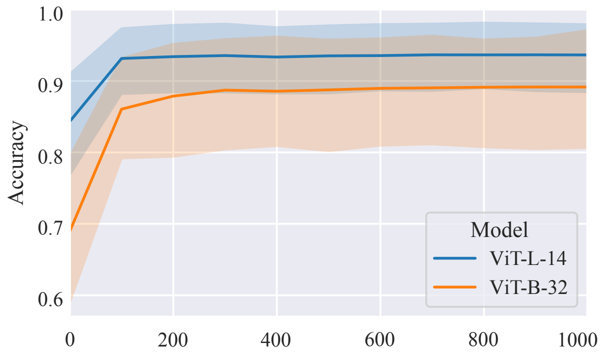
In this section, we conduct experiments to validate the effectiveness of our method. We merge the CLIP-ViT-B/32 and CLIP-ViT-L/14 models on all eight tasks. In Table 8, we compare the parameter counts and reductions in WEMoE models with varying tasks. Where the proposed method has an MoE structure on MLP weights, which introduces additional parameters compared to a single pre-trained model (the MLPs have about 60% of the parameters in a Transformer-based

Table 8. Comparison of parameter counts and reductions in WEMoE models with varying tasks.

Method	Trainable Parameters	Total Parameters	Parameters Reduced by Merging
Single Pre-trained	113.45M (100%)	113.45M	-
WEMoE (2-layer, 2 tasks)	7.11M (3.04%)	233.89M	-6.99M
WEMoE (2-layer, 3 tasks)	7.11M (2.45%)	290.57M	49.78M
WEMoE (2-layer, 4 tasks)	7.12M (2.02%)	347.25M	106.55M
WEMoE (2-layer, 5 tasks)	7.13M (1.77%)	403.93M	163.32M
WEMoE (2-layer, 6 tasks)	7.14M (1.55%)	460.61M	220.09M
WEMoE (2-layer, 7 tasks)	7.15M (1.38%)	517.28M	276.87M
WEMoE (2-layer, 8 tasks)	7.16M (1.25%)	573.96M	333.64M



(a) Learning rate comparison.



(b) Model comparison.

Figure 8. The performance of the merged models with a varying number of steps. (a) CLIP-ViT-B/32 model with different learning rates. (b) Comparison of CLIP-ViT-B/32 and CLIP-ViT-L/14.

model). This is an inherent limitation of all mixture-of-experts based methods. It's an interesting direction to explore techniques to reduce memory overhead in future work, such as using a low-rank approximation or sparse structure.

Tables 2 and 3 present a comprehensive comparison of multi-task performance for CLIP-ViT-B/32 and CLIP-ViT-L/14 models across eight distinct tasks. The table includes results for various merging methods, such as traditional multi-task learning (MTL), Weight Averaging, Fisher Merging (Matena & Raffel, 2022), RegMean (Jin et al., 2023), Task Arithmetic (Ilharco et al., 2023), Ties-Merging (Yadav et al., 2023), AdaMerging/AdaMerging++ (task-wise and layer-wise) (Yang et al., 2023), and our proposed WEMoE method. Additionally, individual task performance, traditional MTL, and several model fusion methods are evaluated to provide a thorough understanding of the effectiveness of each approach.

We have the following key observations, which showcase WEMoE's effectiveness in multi-task model fusion:

1. The performance of fine-tuned models on downstream tasks is significantly better than that of the pre-trained model, indicating that the fine-tuned models have learned task-specific knowledge.
2. For larger models, the fused performance tends to be better across all merging methods. This is because the larger model scale introduces more redundancy among parameters. Consequently, even simple averaging can achieve better performance on larger models.
3. Our proposed approach, WEMoE, consistently demonstrates the superiority of our approach over SOTA methods across the majority of tasks.
4. In particular, WEMoE outperforms the traditional MTL baseline by 0.5% and 0.1% on average. This is surprising as the traditional MTL baseline is trained on all tasks simultaneously while merging methods are inaccessible to the training data.

These observations indicate that through our approach, we have successfully separated the task-specific knowledge from fine-tuned models. On the other hand, the weight-ensembling MoE module successfully captures the relationship between the input features and knowledge combination, effectively alleviating the mutual influence of parameters, which significantly enhances the overall performance and flexibility of the model.

Figure 8 shows the performance of the merged WEMoE models with varying number of steps. In Figure 8a, we merge CLIP-ViT-B/32 models with different learning rate configurations. We observe that the performance of the merged model shows an upward trend with an increase in the number of training steps, and it converges rapidly, reaching a high accuracy level in just 200 steps. Furthermore, the influence of different learning rates is not significant, suggesting that our method is insensitive to the learning rate parameter. This is a desirable property as it reduces the need for hyperparameter tuning. In Figure 8b, we compare the performance of CLIP-ViT-B/32 and CLIP-ViT-L/14 models.

B.3. Ablations of Router Ddepth

Table 9. Parameter comparison of WEMoE (1-layer) and WEMoE (2-layer) on CLIP-ViT-B/32 models. We add AdaMerging as a baseline for comparison.

Method	Number of Trainable Parameters
AdaMerging (layer-wise)	1.3K
WEMoE (1-layer)	73.8K(0.01%)
WEMoE (2-layer)	7.16M(1.25%)

Table 10. Ablation study of the router depth on the performance of the up-scaled CLIP-ViT-B/32 models. We add AdaMerging as a baseline for comparison.

METHOD	SUN397	CARS	RESISC45	EUROSAT	SVHN	GRSRB	MNIST	DTD	AVG.
ADAMERGING (LAYER-WISE)	66.6	68.3	82.4	92.5	86.5	93.7	97.7	61.1	80.9
WEMOE (1-LAYER)	73.2	76.7	93.8	98.6	95.7	98.6	99.5	74.5	88.3
WEMOE (2-LAYER)	74.1	77.4	93.7	99.1	96.2	98.9	99.6	76.4	89.4

To explore the influence of router depth on the performance of the scaled-up model, we perform an ablation study where the router depth is varied. Before we delve into the experimental results, let's clarify the router depth in our WEMoE model, as introduced in Section 3.2. In our WEMoE modules, the router is implemented as a multi-layer perceptron (MLP).

- WEMoE (0-layer) functions as a bias-only model, representing a special case of an MLP with no hidden layers. It generates a constant routing weight for all inputs, captured by the formula as follows

$$r(h) = b_0, \tag{7}$$

  indicating that it does not adjust based on the input. When we only up-scale the MLP modules of the vision Transformers to MoE modules, WEMoE (0-layer) can be considered as a partial implementation of AdaMerging. Add when we up-scale the vision Transformers layer-wisely, WEMoE (0-layer) can be considered equivalent to AdaMerging. For WEMoE (0-layer), the MoE modules can be unloaded, thus no additional parameters and inference cost are introduced.

- For WEMoE (1-layer), each router is a one-layer MLP that takes the input sample h and outputs the routing weight $r(h)$, which is adaptive to the input. The routing weight is calculated as follows

$$r(h) = W_1 h + b_1. \tag{8}$$

- For WEMoE (2-layer), each router is a two-layer MLP and the routing weight is calculated as follows

$$r(h) = W_2 ReLU(W_1 h + b_1) + b_2. \tag{9}$$

In Tables 4 and 5, we note a substantial performance improvement when comparing WEMoE (2-layer) to WEMoE (0-layer). While it may appear that the auxiliary parameters are the primary contributors to the final performance, rather than the proposed MoE design, this is not the case. In fact, the key innovation of our approach lies in the proposed MoE design, which offers the ability to dynamically integrate shared and task-specific knowledge based on input, i.e. the data adaptability of the routing mechanism.

In Tables 9 and 10, we present additional findings to support our argument. We compare the number of trainable parameters and performance between WEMoE (1-layer) and WEMoE (2-layer). The data reveal that WEMoE (1-layer) possesses 73.8K trainable parameters, which constitute only 0.01% of the total parameters in the merged model. Notably, the performance of WEMoE (1-layer) is significantly better than AdaMerging and nearly matches that of WEMoE (2-layer) across all tasks. This evidence underscores our claim that the MoE design is crucial for performance enhancement.

C. Robustness to Out-of-Distribution Data

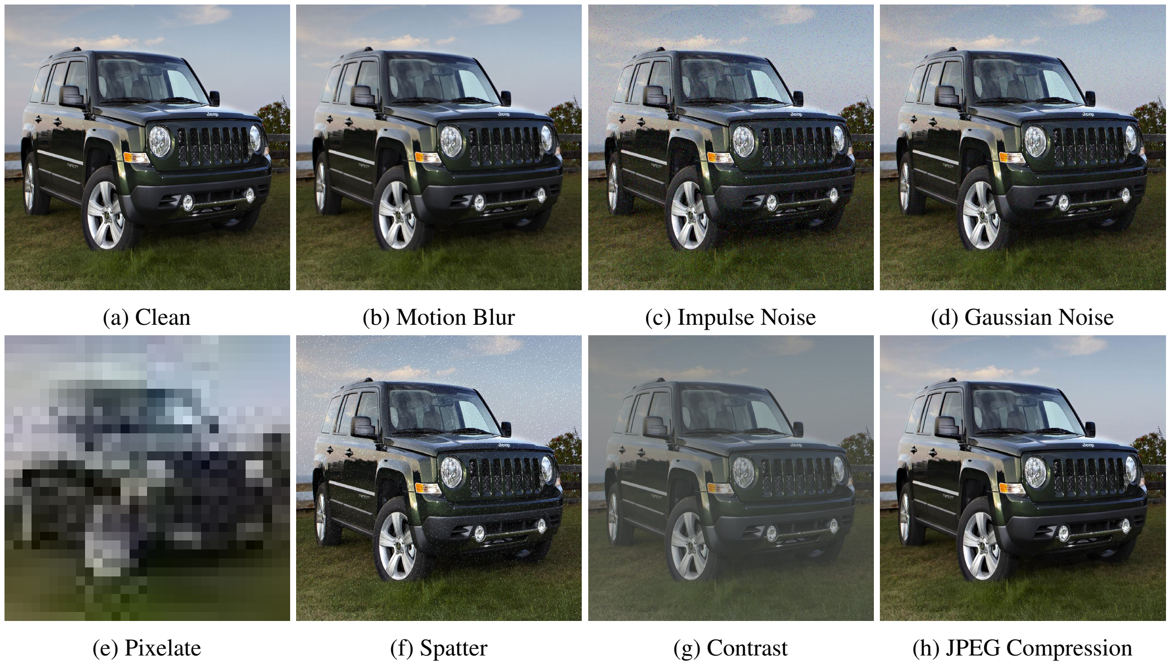


Figure 9. Here are eight instances of distorted images, produced using the method suggested in (Hendrycks & Dietterich, 2019).

In real-world applications, it is common to encounter out-of-distribution (OOD) data, i.e. the unlabeled test data that we seek to generalize may deviate from the distribution of the training data. To evaluate the robustness of our method, we conduct experiments on the clean test dataset and seven corrupted test datasets. The corrupted test datasets are generated using the method suggested in (Hendrycks & Dietterich, 2019). In Figure 9, we show eight example images from the corrupted test datasets. Following (Yang et al., 2023), we conduct experiments on four datasets: Cars (Stallkamp et al., 2012), EuroSAT (Helber et al., 2018), RESISC45 (Cheng et al., 2017), and GTSRB (Stallkamp et al., 2012).

We compare our method with the Task Arithmetic (Ilharco et al., 2023), Ties-Merging (Yadav et al., 2023), and AdaMerging (Yang et al., 2023). We first set the scaling factor $\lambda = 0.3$ and merge the models trained on the clean test dataset. The results are shown in Tables 5 and 11.

The results in Table 5 and Table 11 that among various approaches, WEMoE ($l = 2$) consistently demonstrates the highest performance across both the clean test set and the majority of distorted test sets. This underscores the effectiveness of our method in handling both clean and distorted data, suggesting its potential for robust multi-task merging. We also note that in situations where there is a significant degradation in image quality, such as pixelation, WEMoE ($l = 2$) may exhibit overfitting to some specific tasks, leading to a decline in performance. In contrast, WEMoE ($l = 0$) tends to display greater stability in performance under such conditions. This can be attributed to its lower parameter count, rendering it less susceptible to overfitting.

Furthermore, we note that the performance of WEMoE ($l = 0$) closely aligns with that of AdaMerging. It is worth recalling that when $l = 0$, WEMoE corresponds to a partial implementation of AdaMerging, with the scaling factor λ fixed at 0.3

Table 11. Ablations of the test data distribution on ViT-B/16 (for all methods, $\lambda = 0.3$).

METHOD	CARS	EUROSAT	RESISC45	GTSRB	AVG.	CARS	EUROSAT	RESISC45	GTSRB	AVG.
	CLEAN TEST SET					CORRUPTED TEST SET (MOTION BLUR)				
TASK ARITHMETIC	75.3	96.3	85.3	80.5	84.3	73.5	70.9	83.9	72.2	75.1
TIES-MERGING	74.8	93.4	84.0	65.8	79.5	73.1	65.5	82.3	57.4	69.6
ADAMERGING	83.4	97.2	88.6	97.5	91.7	81.3	75.9	87.4	95.6	85.0
WEMOE (0-LAYER)	78.6	95.3	86.3	95.9	89.0	75.8	73.3	84.8	91.1	81.2
WEMOE (2-LAYER)	**87.3**	**99.3**	**96.2**	**99.3**	**95.5**	**86.3**	**76.8**	**95.2**	**98.2**	**89.1**
	CORRUPTED TEST SET (IMPLUSE NOISE)					CORRUPTED TEST SET (GAUSSIAN NOISE)				
TASK ARITHMETIC	70.4	**59.5**	75.2	54.0	64.8	72.2	**60.8**	78.5	51.0	65.6
TIES-MERGING	70.5	46.2	73.0	42.0	57.9	72.8	47.6	77.0	42.2	59.9
ADAMERGING	77.6	42.1	81.9	90.2	**73.0**	79.1	58.9	81.2	**74.5**	**73.4**
WEMOE (0-LAYER)	72.7	47.0	77.2	80.5	69.4	74.1	58.3	80.1	64.7	69.3
WEMOE (2-LAYER)	**83.2**	11.1	**92.3**	**96.2**	70.7	**84.8**	11.7	**94.4**	73.3	66.1
	CORRUPTED TEST SET (PIXELATE)					CORRUPTED TEST SET (SPATTER)				
TASK ARITHMETIC	3.8	38.0	24.8	71.3	34.5	72.1	58.4	79.9	60.1	67.6
TIES-MERGING	**4.9**	36.3	21.4	57.6	30.1	72.1	50.7	77.8	46.9	61.9
ADAMERGING	4.1	**46.4**	23.6	91.3	**41.3**	79.3	**60.9**	85.8	93.7	**80.0**
WEMOE (0-LAYER)	3.1	42.4	**26.1**	84.3	39.0	73.7	57.	82.0	87.1	74.9
WEMOE (2-LAYER)	0.5	20.6	1.9	**97.3**	30.1	**84.0**	11.9	**93.8**	**97.5**	71.8
	CORRUPTED TEST SET (CONTRAST)					CORRUPTED TEST SET (JPEG COMPRESSION)				
TASK ARITHMETIC	73.4	62.5	81.3	76.9	73.5	75.1	73.1	84.8	64.7	74.4
TIES-MERGING	73.4	58.0	80.0	63.1	68.6	74.8	66.9	83.8	54.1	69.9
ADAMERGING	81.4	**68.1**	85.8	96.8	83.0	81.9	76.0	87.3	91.0	84.1
WEMOE (0-LAYER)	76.2	65.6	82.2	93.6	79.4	77.3	74.2	86.1	84.1	80.4
WEMOE (2-LAYER)	**86.0**	67.8	**95.1**	**98.9**	**87.0**	**86.9**	**82.0**	**96.2**	**95.9**	**90.2**

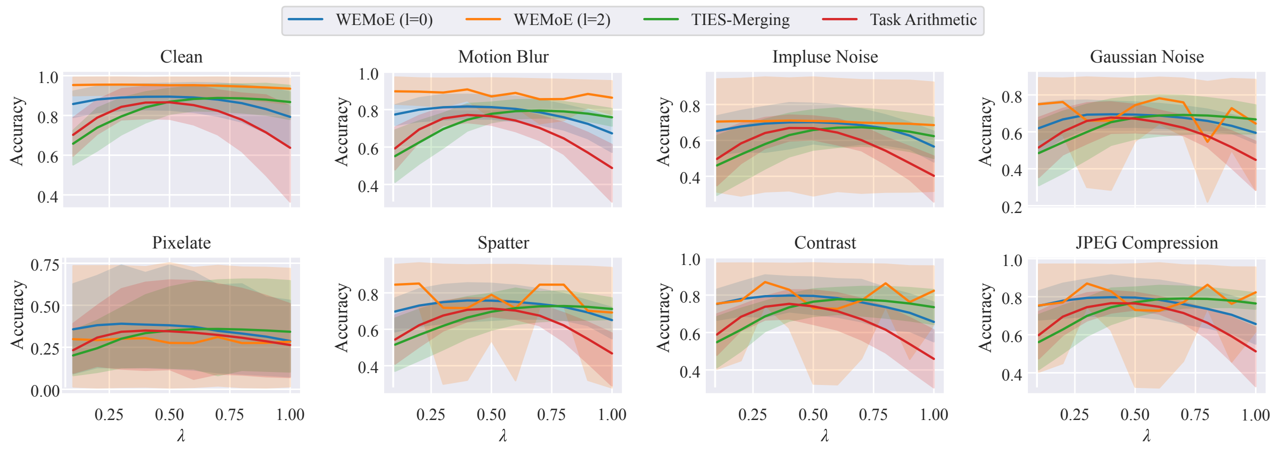


Figure 10. The results for the robustness experiment on CLIP-ViT-B/16. The x-axis of each plot represents the scaling coefficient λ of task vectors, while the y-axis shows the accuracy of the merged model on different merged tasks.

for the majority of the model, except for MLP modules. This observation provides support for the hypothesis outlined in Section 2.2 concerning parameter similarity.

Figures 5 and 10 show more detailed results of the robustness experiments on CLIP-ViT-B/32 and CLIP-ViT-B/16, respectively. The x-axis of each plot represents the scaling coefficient λ of task vectors, while the y-axis shows the accuracy of the merged model on the four merged tasks. The results show that WEMoE ($l = 2$) outperforms other methods across all tasks and datasets in the majority of cases. It also exhibits stability across variations in hyperparameter λ. It achieves nearly optimal performance for all configurations of λ when the test dataset follows the same distribution as the training data. Even when there is a significant disparity between the distributions of test and training data, comparable performance can be achieved with Task Arithmetic and Ties-Merging.

D. Routing Analysis

In this section, we delve into an analysis of the expert weighting as determined by the router. For this analysis, we use the CLIP-ViT-B/32 model that has been merged on eight downstream tasks as a representative example. The results of this analysis are visually represented in Figure 11. In the figure, we demonstrate how routers at different depths in the network allocate routing weights for inputs from different tasks. This visualization helps in understanding how the routing weights vary across different tasks and layers, providing insights into the network's decision-making process.

Upon examining the router's behavior, we notice a distinct pattern. For shallow-layer routers, there is no clear correlation between routing weight allocation and the source task of the input sample. Input samples from different tasks exhibit similar routing weights. In contrast, deep-layer routers show a pronounced correlation between weight allocation and the source task of the input sample. In these cases, the router tends to favor the task vector corresponding to the input sample's source task by assigning it a higher weight. This observation suggests that deeper-layer routers possess the capability to identify the expert likely to perform better based on input features. As a result, they assign a higher weight to this expert, reflecting increased confidence in its performance. This behavior of the router is in line with our expectations and intuition. This ability to discern and assign weights effectively is a key factor in the overall performance of the merged model.

This also indicates that shallow-level features are insufficient to distinguish between different tasks; instead, these features predominantly contain shared information. In contrast, deep-level features contain more task-specific information. This provides us with insights, suggesting that we can further narrow down the scope of information separation and integration. For instance, we might consider applying methods like Task Arithmetic to merge parameters of shallow-level MLPs as well, while constructing MoE for information separation exclusively in deep-level MLPs. This approach can further reduce the inference cost overhead by further reducing additional parameters introduced by MoE while maintaining the performance of the merged model.

Figure 12 presents the first choice matrix of the CLIP-ViT-B/32 model. This matrix indicates the percentage of samples for which the router assigns the highest weight to the corresponding task. At the first layer, the router assigns almost all samples to Cars. As the layer index increases, the router gradually assigns more samples to the correct task. From the 6th layer onwards, the router assigns the highest weight to the correct task for the majority of samples. This is consistent with our previous observation from Figure 11, where we notice that the router at the 6th layer is the first to exhibit a clear correlation between routing weight allocation and the source task of the input sample.

Why not perform a qualitative analysis of features from various experts? Such an analysis is often beneficial for a comprehensive understanding. However, visualizing features from different experts in our study is not straightforward. This difficulty arises because our proposed Mixture of Experts (MoE) design differs from research that analyzes features, such as Ye & Xu (2023). Our method generates input-conditioned weights and infers through the MLP modules only once. In contrast, Ye & Xu (2023) pass the input through each expert. Instead, in this Appendix section, we offer a routing analysis that provides insights into the routing mechanism of Weight-Ensembling MoE (WEMoE).

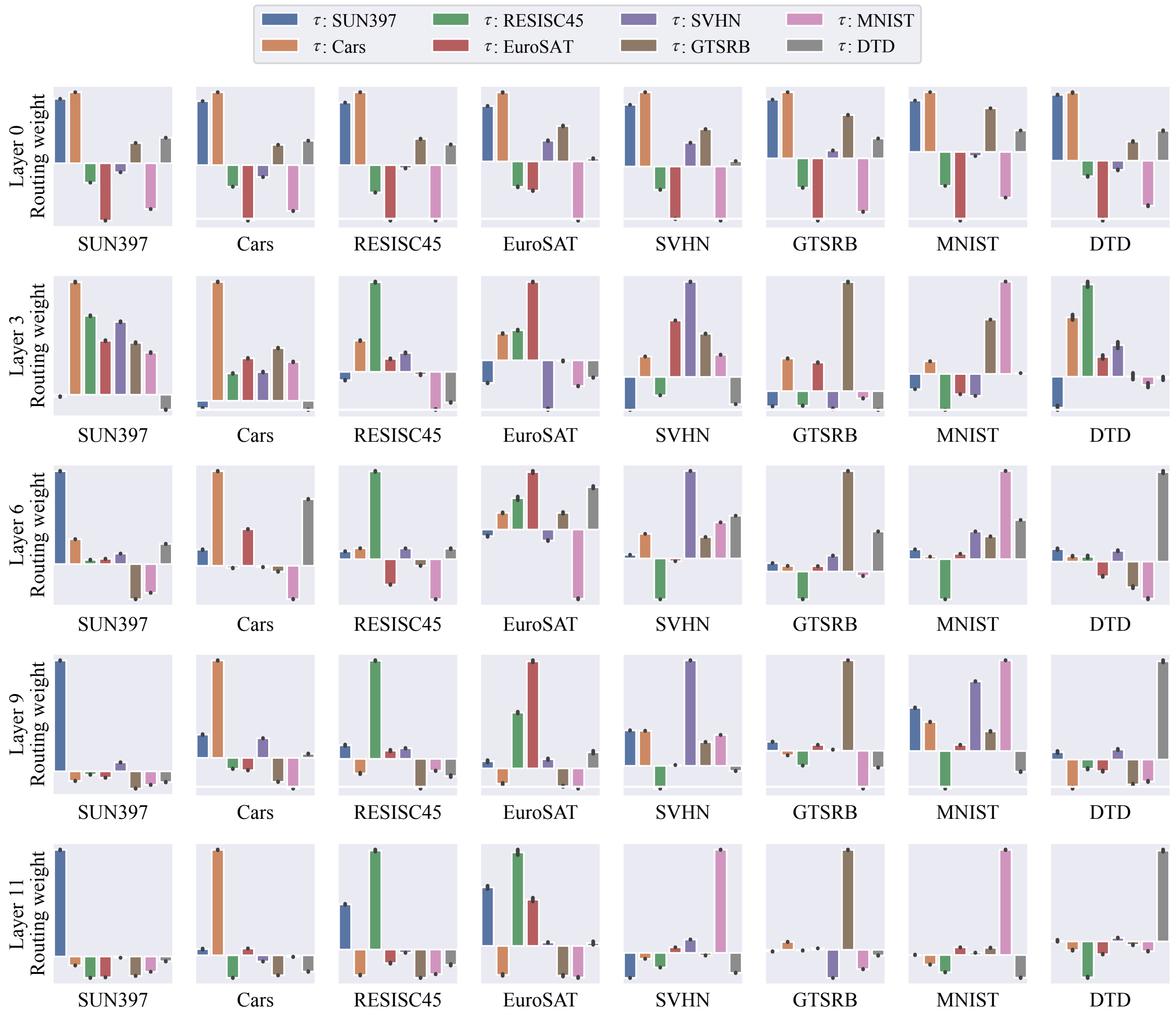


Figure 11. Analysis of expert weighting by the router using the CLIP-ViT-B/32 model at layers 0,3,6,9 and 11. This figure presents the routing weights for different layers and tasks in the neural network. Each subplot corresponds to a specific task, and the y-axis represents the routing weights for that task. The x-axis labels indicate the task names.

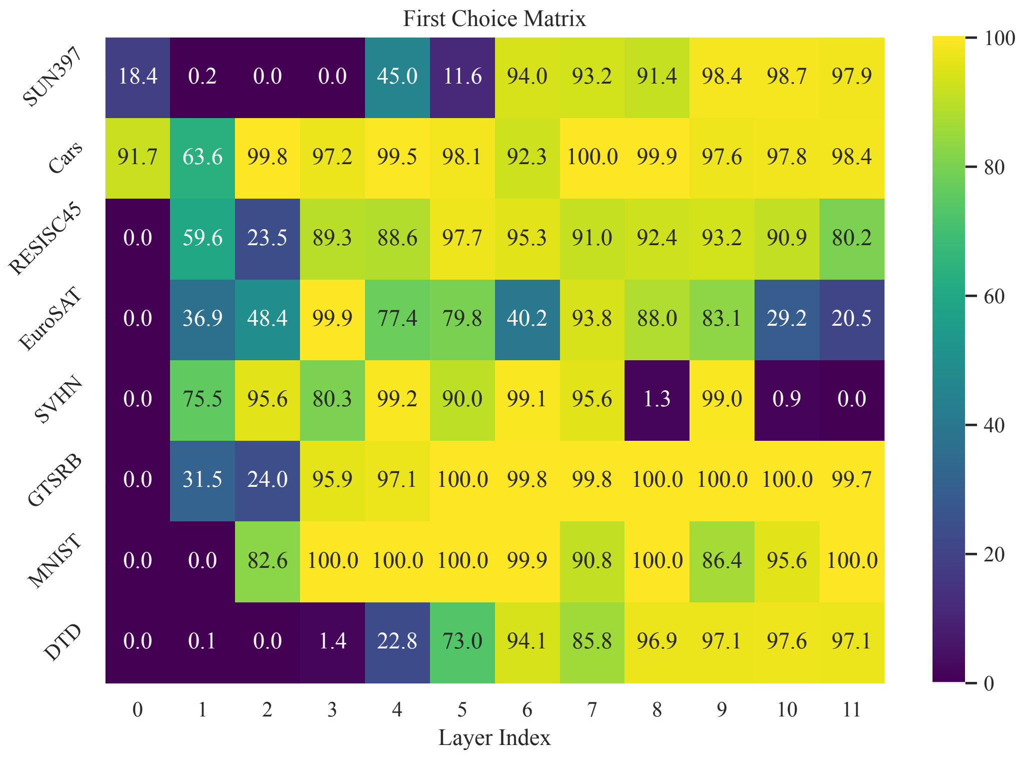


Figure 12. This heatmap presents the first choice matrix of the CLIP-ViT-B/32 model. Each row corresponds to a specific task, and the x-axis labels indicate the layer index. Each entry in the matrix represents the percentage of samples for which the router assigns the highest weight to the corresponding task.