LiNeS: Post-training Layer Scaling Prevents Forgetting and Enhances Model Merging

Ke Wang[*]
EPFL
k.wang@epfl.ch

Nikolaos Dimitriadis[*]
EPFL
nikolaos.dimitriadis@epfl.ch

Alessandro Favero
EPFL
alessandro.favero@epfl.ch

Guillermo Ortiz-Jimenez
Google DeepMind
gortizj@google.com

François Fleuret
University of Geneva
francois.fleuret@unige.ch

Pascal Frossard
EPFL
pascal.frossard@epfl.ch

Abstract

Large pre-trained models exhibit impressive zero-shot performance across diverse tasks, but fine-tuning often leads to catastrophic forgetting, where improvements on a target domain degrade generalization on other tasks. To address this challenge, we introduce LiNeS, Layer-increasing Network Scaling, a post-training editing technique designed to preserve pre-trained generalization while enhancing fine-tuned task performance. LiNeS scales parameter updates linearly based on their layer depth within the network, maintaining shallow layers close to their pre-trained values to preserve general features while allowing deeper layers to retain task-specific representations. We further extend this approach to multi-task model merging scenarios, where layer-wise scaling of merged parameters reduces negative task interference. LiNeS demonstrates significant improvements in both single-task and multi-task settings across various benchmarks in vision and natural language processing. It enhances out-of-distribution generalization, integrates seamlessly with existing multi-task model merging baselines enhancing their performance across benchmarks and model sizes, and can boost generalization when merging LLM policies aligned with different rewards via RLHF. Importantly, our method is simple to implement and complementary to many existing techniques.

1 Introduction

Pre-trained models have become the backbone of modern machine learning pipelines (Bommasani et al., 2021; Touvron et al., 2023). Their introduction has shifted the paradigm from end-to-end training to fine-tuning (Zhuang et al., 2020), leading to the proliferation of thousands of fine-tuned checkpoints derived from a few foundation models (Rombach et al., 2022; Team et al., 2023). To improve downstream performance across multiple tasks or align with multiple preferences (Singh & Jaggi, 2020; Matena & Raffel, 2022; Ilharco et al., 2023; Yadav et al., 2023; Rame et al., 2024), model merging techniques combine available checkpoints, avoiding the costly process of joint fine-tuning (Ilharco et al., 2023; Yadav et al., 2023). However, specializing models introduces trade-offs, such as the forgetting of previously acquired knowledge (Aghajanyan et al., 2020) – a phenomenon known as *catastrophic forgetting* (McCloskey & Cohen, 1989). Furthermore, merging checkpoints fine-tuned on different tasks can lead to significant performance degradation due to task interference (Yadav et al., 2023; Wang et al., 2024).

To mitigate catastrophic forgetting, many works propose regularizing the fine-tuning process (Aghajanyan et al., 2020; Kumar et al., 2022; Gouk et al., 2020; Razdaibiedina et al., 2022).

[*] Equal Contribution

Leveraging the insight that shallow layers capture generalizable representations (Yosinski et al., 2014; Neyshabur et al., 2020), Howard & Ruder (2018); Dong et al. (2022) apply lower learning rates to the shallow layers to retain general features. However, modifying the fine-tuning process can be complex and computationally expensive. This motivated the development of post-training model editing and model merging methods that directly edit the checkpoints in weight space. For instance, Wortsman et al. (2022b); Rame et al. (2022) mitigate catastrophic forgetting by interpolating weights between pre-trained and fine-tuned models. In multi-task settings, Yadav et al. (2023); Wang et al. (2024) proposed methods to reduce interference among tasks when merging multiple checkpoints. Yet, significant performance degradation persists when merging multiple models, leaving this as an open challenge.

Most model merging methods, however, treat all layers equally, overlooking the earlier insight that shallow layers should remain close to their pre-trained weights to avoid losing the general representations they encode. In this paper, we explore whether this insight can be leveraged post-training. We find that reducing the magnitude of shallow-layer updates after fine-tuning can retain single-task performance gains while significantly mitigating forgetting.

We propose `LiNeS`, **L**ayer-**i**ncreasing **Ne**twork **S**caling, a post-training, plug-and-play method that directly edits the residual, i.e., the difference between the fine-tuned and pre-trained checkpoint, by applying a scaling coefficient that linearly increases with layer depth. This scaling effectively preserves the general features captured in the shallow layers of the pre-trained model while retaining task-specific features in the deep layers of the fine-tuned model. Moreover, we extend `LiNeS` to the multi-task model merging setting, where contributions from one task distort the general features also required by other tasks.

`LiNeS` demonstrates remarkable performance on diverse test scenarios and is orthogonal to existing post-training merging algorithms. It modifies the fine-tuned checkpoint to consistently retrieve nearly full performance on the fine-tuned task while significantly restoring generalization on other tasks. Furthermore, it can be seamlessly integrated with existing weight interpolation methods for improving out-of-distribution generalization (Wortsman et al., 2022b). When merging multiple models, `LiNeS` improves baseline methods for merging checkpoints fine-tuned on multiple tasks in both computer vision and natural language processing benchmarks (Ilharco et al., 2023; Yadav et al., 2023; Wang et al., 2024) and also enhances performance when merging checkpoints fine-tuned on the same task (Wortsman et al., 2022a) and merging LLM policies aligned with different rewards (Rame et al., 2024) via Reinforcement Learning with Human Feedback (RLHF) (Christiano et al., 2017).

Our contributions are as follows:

- We propose `LiNeS`, a post-training editing technique that preserves the zero-shot generalization of pre-trained models while retaining fine-tuned knowledge by applying layer-wise scaling on parameter updates. For example, in image-classification tasks with CLIP ViT-B/32 checkpoints, `LiNeS` maintains on average 99.8% of performance on the fine-tuned task while preserving 97.9% performance of the pre-trained model on other control tasks, effectively mitigating catastrophic forgetting.
- We demonstrate that `LiNeS` significantly enhances multi-task model merging baselines, consistently improving performance across benchmarks and architectures in both vision and NLP domains. For instance, we observe a 3.1% and 4.0% improvement over Task Arithmetic (Ilharco et al., 2023) and Ties-merging (Yadav et al., 2023) respectively, for a 20-task computer vision benchmark with ViT-L/14.
- We show that `LiNeS` can be applied to enhance existing weight interpolation methods across various scenarios, improving out-of-distribution generalization, merging multiple checkpoints fine-tuned on the same task with different hyper-parameter configurations, and merging LLM policies aligned with different rewards.

Our proposed method is simple to implement[1], orthogonal to many existing approaches, and improves performance in a wide variety of settings.

[1] PyTorch pseudo-code is provided in Appendix A; The source code can be found at `https://github.com/wang-kee/LiNeS`

2 Related Works

Representation collapse and regularized fine-tuning Pre-trained models such as CLIP exhibit strong zero-shot performance across diverse data distributions due to the robust and transferable feature representations learned during pre-training (Radford et al., 2021; Jia et al., 2021). However, fine-tuning on specific tasks often harms the zero-shot generalization performance on distributions different from the fine-tuning domain (Wortsman et al., 2022b; Goyal et al., 2023; Aghajanyan et al., 2020). This degradation arises from the distortion of pre-trained features during fine-tuning (Kumar et al., 2022), a phenomenon referred to as *representation collapse* by Aghajanyan et al. (2020). To mitigate representation collapse, many works have proposed to regularize the fine-tuning process to preserve the general pre-trained features (Aghajanyan et al., 2020; Kumar et al., 2022; Goyal et al., 2023; Gouk et al., 2020; Zhang et al., 2022; Razdaibiedina et al., 2022; Shen et al., 2021; Lee et al., 2022). Some of these approaches take into account that different layers of a model learn distinct features, with the shallower layers capturing more general features and deeper layers specializing in task-specific representations (Neyshabur et al., 2020; Yosinski et al., 2014; Raghu et al., 2019b;a). Specifically, they apply layer-wise learning rate decay, preserving more of the pre-trained features in the shallow layers while allowing deeper layers to specialize for the target domain (Clark, 2020; Bao et al., 2021; Dong et al., 2022; Howard & Ruder, 2018; Zhang et al., 2020). However, modifying the fine-tuning process is orders of magnitude more computationally expensive compared to post-training merging methods.

Weight interpolation and model merging Garipov et al. (2018); Draxler et al. (2018) showed that two solutions derived from separate training runs can be connected by nonlinear paths of low loss, while *linear mode connectivity* (Frankle et al., 2020) extended the paths to the linear case. These insights enabled the transfer of the benefits regarding robustness of (traditional) output ensembles (Hansen & Salamon, 1990; Lakshminarayanan et al., 2017) to weight ensembles, reconciling the bias-variance trade-off (Belkin et al., 2019) while eliminating the computational cost of multiple inferences (Fort et al., 2020). These findings can be leveraged to improve performance on single-task (Izmailov et al., 2018; Wortsman et al., 2021; Rame et al., 2022; Wortsman et al., 2022a), out-of-distribution (Wortsman et al., 2022b; Ramé et al., 2023), multi-task (Ilharco et al., 2022; Dimitriadis et al., 2023) and multi-objective alignment (Zhong et al., 2024; Ramé et al., 2024) settings. Furthermore, model merging can also applied as a scalable approach to unify multiple task-specific models into a single model with multi-task capabilities (Ilharco et al., 2023; Yadav et al., 2023), despite performance loss compared to individual models. Several methods have tried to improve multi-task model merging by preserving the important parameters defined via the Fisher Information Matrix (Matena & Raffel, 2022; Tam et al., 2024), using heuristics (Davari & Belilovsky, 2023; Luo et al., 2023; Jin et al., 2022), randomly dropping and rescaling the task vector parameters (Yu et al., 2024) or by focusing on resolving weight interference caused by sign disagreements and redundant parameters (Yadav et al., 2023; Wang et al., 2024). Recent works use gradient descent to learn the layer-specific merging coefficients per task, e.g., Ada-merging (Yang et al., 2024) minimizes entropy in unlabeled test data while aTLAS (Zhang et al., 2024) optimizes using cross-entropy loss on validation data. Compared to `LiNeS`, these methods do not incorporate any prior knowledge on early vs. deep layers and require training, resulting in significant computational overheads.

3 Post-training Layer-wise Scaling Mitigates Forgetting

In this section, we present the key insight of our work: Scaling down the updates of shallow layers after fine-tuning can mitigate catastrophic forgetting and restore zero-shot generalization while preserving performance on the target task.

Notation We consider a pre-trained model $\boldsymbol{\theta}_0 \in \mathbb{R}^N$ with N parameters. Fine- tuning on a specific task t results in the fine-tuned weights $\boldsymbol{\theta}_t$. The difference between these two sets of weights, $\boldsymbol{\tau}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$, is referred to as the *task vector* or *residual* for task t (Ilharco et al., 2023) and represents the updates made during fine-tuning.

Fine-tuning leads to catastrophic forgetting We quantitatively demonstrate the phenomenon of catastrophic forgetting with the following experiments. Consider the 8-task image classification benchmark studied in Ilharco et al. (2023). We fine-tune a CLIP ViT-B/32 model on each task,

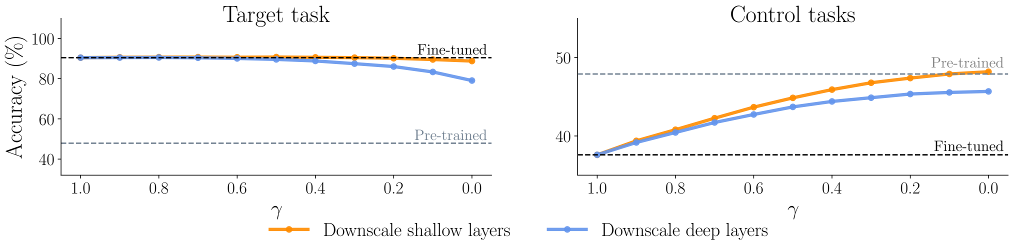


Figure 1: Downscaling the shallow layers maintains the fine-tuned performance on target tasks (orange line, left), while restoring zero-shot performance from pre-trained model on control tasks (orange line, right). The performance for downscaling deep layers instead is presented in blue lines, which underperforms downscaling shallow layers in both cases. γ represents the minimum scaling factor applied to the layers, where a smaller γ leads to stronger downscaling strength, with $\gamma = 1$ restoring the original fine-tuned model.

measuring performance on the fine-tuned task – referred to as the *target task* – and the remaining 7 tasks – the *control tasks*. The results, shown in Table 1, demonstrate that while fine-tuning significantly improves accuracy on the target task, it drastically reduces accuracy on the control tasks, underscoring the loss of the model's zero-shot generalization abilities.

Table 1: Fine-tuning harms generalization on control tasks. Our proposed post-training edition leads to a superior trade-off between performance on target and control tasks.

Model / Accuracy	Target	Control
Pre-trained	48.3	48.3
Fine-tuned	90.5	38.0
Fine-tuned+LiNeS (ours)	90.3	48.0

Shallow-layer updates impact minimally on target task accuracy Most parameter updates during the fine-tuning process are redundant, as similar performance is achievable without updating most pre-trained parameters (Yadav et al., 2023; Wang et al., 2024; He et al., 2024). Moreover, prior work shows that task-specific features are often concentrated in deeper layers of the network (Neyshabur et al., 2020; Yosinski et al., 2014; Raghu et al., 2019b). Based on these observations, we hypothesize that updates to the shallow layers contribute minimally to target tasks. To test this, we progressively downscale the updates to shallow layers after fine-tuning. Specifically, we apply a scaling factor to the updates to the ℓ-th layer $\boldsymbol{\tau}^{(\ell)}$, defined as: $\lambda^{(\ell)} = \gamma + (1-\gamma)\frac{\ell-1}{L-1}, \ \forall \ell \in [L]$, This linearly scales the updates from a factor of γ for the first layer to 1 for the last one. As a result, fine-tuning updates to the shallow layers are scaled down more aggressively, with later layers experiencing progressively smaller reductions. We then reintroduce the scaled task vector into the pre-trained model and measure its performance on the fine-tuned task. Figure 1 shows the results of this experiment for the CLIP ViT-B/32 checkpoint fine-tuned across the 8 tasks, where γ is progressively decreased to strengthen the downscaling effect. We observe that, even with strong downscaling of shallow layers, the target task accuracy remains nearly unaffected. In contrast, when we downscale the deeper layers, target task accuracy drops significantly. These results support our hypothesis that shallow-layer updates are largely unnecessary for maintaining accuracy on the target task.

Shallow-layer updates undermine zero-shot generalization While shallow-layer updates have minimal impact on target-task accuracy, they distort the general features learned during pre-training, which reside primarily in the shallow layers (Neyshabur et al., 2020; Yosinski et al., 2014; Raghu et al., 2019b). We hypothesize that the degradation of performance on control tasks is largely due to these distortions in the shallow layers. Using the same experimental setup, we now evaluate the zero-shot performance on the control tasks, i.e., the other 7 unseen tasks. As shown in Figure 1 (right), as the strength of the shallow-layer downscaling increases, the accuracy on control tasks approaches the original pre-trained models performance. This shows that by reducing the shallow-layer updates, we can restore much of the zero-shot performance lost during fine-tuning.

Improved trade-off between target and control performance To optimize the trade-off between target and control task performance, we select a scaling coefficient γ for each model that maximizes a weighted balance between these two objectives (detailed in Appendix C.1). After selecting the optimal scaling coefficient, the test results are shown in the final row of Table 1. Our post-training method preserves target task accuracy with a minimal 0.2% difference while improving control task performance by 10%, compared to the fine-tuned model.

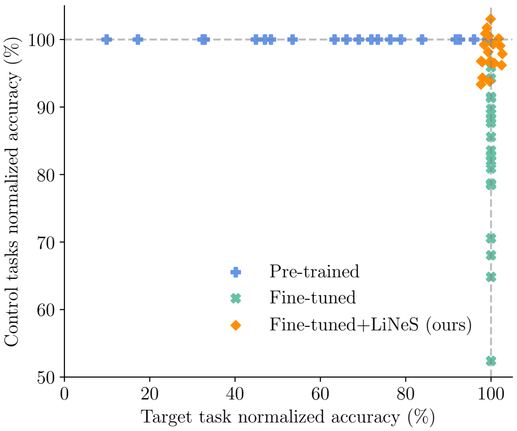


Figure 2: Our linear scaling (LiNeS) retains performance on both control and fine-tuned target tasks. Each scatter dot represents a fine-tuned checkpoint.

We further apply the same method to a 20-task computer vision benchmark (Wang et al., 2024). For evaluation, we report both the *target task normalized accuracy* and the *control task normalized accuracy* on the 19 tasks, where accuracy is normalized by the performance of the fine-tuned model for the target task and the zero-shot accuracy of the pre-trained model for the control tasks. We compare to fine-tuned models on each task and the pre-trained model as baselines. Figure 2 shows that fine-tuning degrades zero-shot generalization, as indicated by the performance drop on control tasks. In contrast, our post-training scaling method significantly improves generalization while maintaining near-full target task accuracy. On average, our method achieves a target task normalized accuracy of 99.8% and a control task normalized accuracy 97.9%. This demonstrates its effectiveness in preserving both task-specific knowledge from fine-tuned checkpoints and the generalization capabilities of the pre-trained model. The full breakdown of results by task is available in Figure 6 in Appendix.

4 Method

Based on the results of the previous section, we propose LiNeS for **L**ayer-**i**ncreasing **Ne**twork **S**caling, a simple post-training technique that linearly rescales the updates of different layers in the task vector based on their depth in the network. LiNeS is designed to retain general features in the shallow layers while preserving the task-specific adaptations in the deeper layers.

Given a task vector $\boldsymbol{\tau}$ with L layer blocks we apply the layer-wise linear scaling to adjust the contributions of shallow and deep layers using the following formulation:

$$\boldsymbol{\tau} = \texttt{concat}\left(\lambda^{(1)}\boldsymbol{\tau}^{(1)}, \dots, \lambda^{(L)}\boldsymbol{\tau}^{(L)}\right), \quad \text{where } \lambda^{(\ell)} = \alpha + \beta\frac{\ell-1}{L-1}, \quad \forall \ell \in [L]. \tag{1}$$

As a result, the layers in $\boldsymbol{\tau}$ are progressively scaled with a factor between α for the first layer and $\alpha+\beta$ for the last layer, with intermediate layers scaled with a linearly increasing schedule depending on their depth. The final model is then obtained by summing the pre-trained model weights and the edited task vector. Notice that, in Equation 1, $\boldsymbol{\tau}$ can correspond to either a single-task residual or, in the context of model merging, a multi-task vector obtained by merging the residuals of multiple checkpoints fine-tuned starting from a common initialization. Additional details on this process are provided in the next section. Setting $\alpha = \beta = 0$ corresponds to the pre-trained model, while $\alpha = 1, \beta = 0$ is the fine-tuned model in the case that $\boldsymbol{\tau}$ is a single-task vector.

In practice, we find that tuning just one hyper-parameter (either α or β) is often sufficient to achieve a good balance between target task performance and generalization. Specific details on hyper-parameter tuning for different applications are provided in the experimental sections. The linear scaling method introduced in Section 3 corresponds to LiNeS by setting $\alpha = \gamma$ and $\beta = 1 - \gamma$. This formulation generalizes our previous approach, offering a flexible way to adjust the contributions of different layers based on the task requirements.

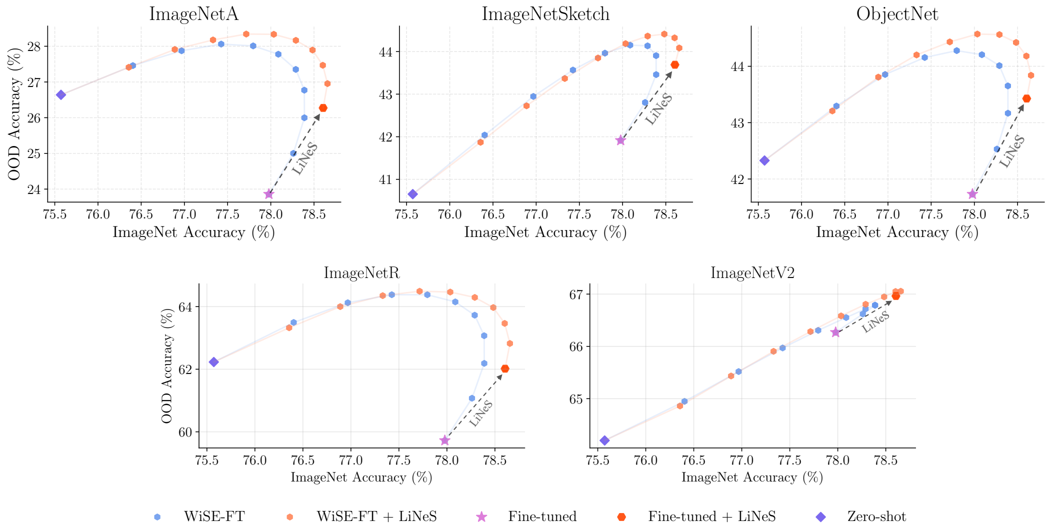


Figure 3: Application of `LiNeS` to WiSE-FT (Wortsman et al., 2022b) improves performance on ImageNet and five different distribution shifts, resulting in a dominating Pareto Front over WiSE-FT.

5 Model Merging Experiments

We empirically verify the effectiveness of applying `LiNeS` across diverse application domains. Section 5.1 presents results for improving robust fine-tuning (Wortsman et al., 2022b) for OOD generalization; Section 5.2 focuses on improving existing multi-task merging methods (Ilharco et al., 2023; Yadav et al., 2023; Wang et al., 2024) in both vision and NLP benchmarks. In Section 5.3, we apply `LiNeS` and improve the merging of single-task fine-tuned models within the setting of Model Soups (Wortsman et al., 2022a), and finally, we enhance merging foundation models fine-tuned on different rewards (Rame et al., 2024) in Section 5.4.

5.1 Improving Robust Fine-tuning for OOD Generalization

We first consider the setting of robust fine-tuning or WiSE-FT (Wortsman et al., 2022b), where linearly interpolating between the pre-trained and the fine-tuned weights improves model performance on OOD datasets. The interpolation is equivalent to scaling the residual $\boldsymbol{\tau}$: $(1-\gamma)\boldsymbol{\theta}_0+\gamma\boldsymbol{\theta} = \boldsymbol{\theta}_0+\gamma\boldsymbol{\tau}$, for $\gamma \in [0, 1]$. We apply `LiNeS` to the residual $\boldsymbol{\tau}$. Following Wortsman et al. (2022b), we evaluate CLIP models fine-tuned on ImageNet (Deng et al., 2009), considering 5 OOD datasets, namely ImageNetSketch (Wang et al., 2019), ImageNet-A (Hendrycks et al., 2021), ImageNet-R (Hendrycks et al., 2020), ObjectNet (Barbu et al., 2019), ImageNet-V2 (Recht et al., 2019).

We apply this `LiNeS` to each of the 70 fine-tuned checkpoints[2] provided by Wortsman et al. (2022a) setting $\alpha = \beta = 0.5$. We present the average results in Figure 3, comparing the performance of WiSE-FT with and without applying `LiNeS` on the 5 OOD datasets. Without applying WiSE-FT, `LiNeS` already enhances both the ID and OOD performance of the fine-tuned models by a noticeable margin. Starting from this edited model and applying the WiSE-FT interpolation with the pre-trained weights leads to a Pareto Front (Caruana, 1997) that consistently dominates the one by WiSE-FT across all distribution shifts, illustrating the applicability of the proposed method across various distribution shifts.. A granular result for applying `LiNeS` to each of the 70 checkpoints is provided in Appendix C.6.2, further highlighting its universal effectiveness across models. We also report similar findings in Figure 8 in Appendix for a CLIP ViT-B/16 checkpoint fine-tuned on ImageNet, using the same hyper-parameters as Wortsman et al. (2022b).

[2] The checkpoints are CLIP ViT-B/32 models fine-tuned on ImageNet with different hyper-parameters.

Table 2: Results for multi-task model merging in vision classification benchmarks of 8 tasks (Ilharco et al., 2023), 14 tasks, and 20 tasks (Wang et al., 2024) for different vision transformer architectures. Applying `LiNeS` improves baseline performance for all benchmark/architecture combinations.

Method	with `LiNeS`	ViT-B/32			ViT-L/14		
		8 tasks	14 tasks	20 tasks	8 tasks	14 tasks	20 tasks
Zero-shot		48.3	57.3	56.1	64.8	68.3	65.3
Fine-tuned		90.5	89.5	90.4	94.0	93.3	94.0
Task Arithmetic	✗	69.7	65.0	60.3	84.0	79.2	74.0
	✓	**74.2** (+4.5)	**69.1** (+4.1)	**63.4** (+3.1)	**86.5** (+2.5)	**82.2** (+3.0)	**77.1** (+3.1)
Ties-Merging	✗	73.6	67.6	63.1	85.6	79.3	75.6
	✓	**77.2** (+3.6)	**72.1** (+4.5)	**67.2** (+4.1)	**88.0** (+2.4)	**82.5** (+3.2)	**79.6** (+4.0)
Consensus Merging	✗	74.5	70.1	65.3	85.2	81.9	78.7
	✓	**77.6** (+3.1)	**73.6** (+3.5)	**68.6** (+3.3)	**87.3** (+2.1)	**84.0** (+2.1)	**81.0** (+2.3)

5.2 Improving Multi-task Model Merging

In this section, we extend `LiNeS` to improve multi-task merging algorithms, aiming to combine multiple models fine-tuned independently on different tasks into a single model (Matena & Raffel, 2022; Ilharco et al., 2023; Ortiz-Jimenez et al., 2023; Yadav et al., 2023). Task arithmetic (Ilharco et al., 2023) proposed to decouple the contributions of the pre-trained model and individual task vectors, first generating a multi-task vector $\boldsymbol{\tau}_{\text{MTL}} = g(\boldsymbol{\tau}_1, \dots, \boldsymbol{\tau}_T)$ with a merging function $g : \mathbb{R}^N \times \cdots \times \mathbb{R}^N \mapsto \mathbb{R}^N$, and then adding back to the pre-trained checkpoint with a scaling factor to construct a multi-task model $\boldsymbol{\theta} = \boldsymbol{\theta}_0 + \lambda \cdot \boldsymbol{\tau}_{\text{MTL}}$. The scalar coefficient λ is tuned using a held-out validation set. Recent works (Yadav et al., 2023; Wang et al., 2024) follow the same protocol while improving the merging function g for retaining more task information. We refer to Appendix B.1 for a more detailed explanation of these methods.

However, significant performance loss occurs between the merged multi-task model and the original fine-tuned checkpoints. This performance decrease partially stems from interference (Yadav et al., 2023; Wang et al., 2024) among task vectors, where the contribution of one task negatively impacts performance on others, leading to overall degradation. Task interference is linked to catastrophic forgetting, as the individual task vectors lose a significant amount of generalization ability to other tasks after fine-tuning and merging them leads to interference among each other. Therefore, we can edit each task vector with `LiNeS` before merging to restore the generalization to other tasks, or for simplicity, edit directly the merged multi-task vector to preserve the shallow and general features that are beneficial across tasks.

We enhance the merging methods by applying `LiNeS` on the merged multi-task vector $\boldsymbol{\tau}_{\text{MTL}}$. For the linear scaling schedule, we tune only β and set α using a heuristic that adjusts based on both the number of merged models and the merging method. Specifically, for task arithmetic which aggregates the individual task vectors through a simple summation operation: $\boldsymbol{\tau}_{\text{sum}} = \sum_{i=1}^{N_{\text{models}}} \boldsymbol{\tau}_i$, we set $\alpha = 1/N_{\text{models}}$. For other merging strategies which result in $\boldsymbol{\tau}_{\text{MTL}}$ with different magnitudes of norms, e.g., aggregation with summation leads to a norm $\times N_{\text{models}}$ larger compared to averaging, we further multiply by a scaling term $\|\boldsymbol{\tau}_{\text{sum}}\| / \|\boldsymbol{\tau}_{\text{MTL}}\|$ to normalize their norm to simple summation. Overall, we set the intercept α to:

$$\alpha = \frac{1}{N_{\text{models}}} \frac{\|\boldsymbol{\tau}_{\text{sum}}\|}{\|\boldsymbol{\tau}_{\text{MTL}}\|}, \text{ where } \tau_{\text{sum}} = \sum_{i=1}^{N_{\text{models}}} \boldsymbol{\tau}_i \tag{2}$$

Therefore, we only tune β and search over the same range as the constant scaling λ used by the aforementioned merging techniques. As a result, `LiNeS` shares the same computational requirements as the baseline merging methods; we provide more details for the hyper-parameters in Appendix B.2.1. Specifically, we consider various multi-task model merging baselines. namely Task Arithmetic (Ilharco et al., 2023), Ties-merging (Yadav et al., 2023), Consensus Merging (Wang et al., 2024), enhancing them with `LiNeS` and evaluate on both computer vision and NLP benchmarks.

5.2.1 Computer vision

We experiment with the 8-task image classification benchmark proposed by Ilharco et al. (2023), as well as the more challenging 14-task and 20-task benchmarks from Wang et al. (2024). Detailed

Table 3: Results for multi-task model merging methods in three NLP benchmarks with T5-large model. LiNeS improves baseline performance across merging methods and benchmarks.

Method	with LiNeS	T5-large (Lester et al., 2021)		
		7 NLP tasks (Yadav et al., 2023)	8 QA tasks (Zhou et al., 2022)	11 NLP tasks (Wang et al., 2024)
Zero-shot		44.9	33.1	36.9
Fine-tuned		85.9	80.7	78.7
Task Arithmetic	✗	71.9	63.8	63.6
	✓	**76.4** (+4.5)	**67.6** (+3.8)	**66.2** (+2.6)
Ties-Merging	✗	71.6	63.0	64.0
	✓	**72.0** (+0.4)	**66.0** (+3.0)	**66.4** (+2.4)
Consensus Merging	✗	73.5	68.6	**67.5**
	✓	**75.4** (+1.9)	**69.3** (+0.7)	**67.5** (+0.0)

descriptions of task composition appear in Appendix B.2.2. We also examine the efficacy of LiNeS across the model scale axis, studying three vision transformer (Dosovitskiy et al., 2021), namely ViT-B/32, ViT-B/16 and ViT-L/14, as CLIP visual encoders (Radford et al., 2021) .

Table 2 presents the results for ViT-B/32 and ViT-L/14, while Appendix C.4 contains the ViT-B/16 experiments. We observe that LiNeS provides a significant improvement to *all* baseline merging methods across *all* tested scenarios, regardless of model sizes and total number of tasks. For example, for the 8-task benchmark with ViT-B/32, LiNeS improves task arithmetic by 4.5%, Ties-merging by 3.6% and consensus merging by 3.1%. For the challenging 20-task benchmark with ViT-L/14, LiNeS leads to consistent and significant improvements, improving task arithmetic by 3.1%, Ties-merging by 4.0% and consensus merging by 2.3%. The detailed performance on individual tasks for each tested scenario is presented in Appendix C.7.

5.2.2 Natural Language Processing

We also evaluate the effectiveness of LiNeS in NLP domain, including a 7-task NLP benchmark (Yadav et al., 2023), an 8-task Question-Answering benchmark (Zhou et al., 2022), and their combined 11-task benchmark (Wang et al., 2024). Appendix B details the experimental settings. Following Tam et al. (2024), we adopt a variant of T5-large model (Raffel et al., 2020), namely T5-large-LM-Adapt (Lester et al., 2021), and use their provided checkpoints. While T5-large contains both encoder and decoder networks, we apply LiNeS only to the decoder, as our findings in Appendix C.5 indicate that applying the edition to the decoder leads to similar observations to vision.

The performance of applying LiNeS to baseline methods with T5-large across various NLP tasks is summarized in Table 3. LiNeS consistently improves multi-task performance across baseline merging methods and benchmarks with a notable margin. For example, on the 7 NLP tasks benchmark, LiNeS improves task arithmetic by 4.5 points, and consensus merging by 1.9 points. Meanwhile, LiNeS outperforms Ties-merging by 3.0% and 2.4% for the 8-QA benchmark and 11-NLP benchmark, respectively.

5.3 Improving Model Soups for Merging single-task models

Averaging in weight space multiple models fine-tuned on the same task derived from the same pre-trained model has been shown to increase target performance (Wortsman et al., 2022a; Rame et al., 2022). In this section, we investigate whether LiNeS can enhance the test performance when merging single-task models.

We follow the setting in Model Soups (Wortsman et al., 2022a) and merge 70 CLIP ViT-B/32 checkpoints fine-tuned on ImageNet (Deng et al., 2009) using different hyper-parameters, plus the pre-trained checkpoint. We consider both variants introduced in Wortsman et al. (2022a), namely uniform and greedy soup. We refer to Appendix B.3 for details regarding these methods and experimental settings. For both cases, the weight-averaging process can be decomposed as follows:

$$\boldsymbol{\theta}_{\text{soup}} = \boldsymbol{\theta}_0 + \boldsymbol{\tau}_{\text{soup}}, \text{ where } \boldsymbol{\tau}_{\text{soup}} = \frac{1}{N_{\text{models}}} \sum_{i=1}^{N_{\text{models}}} (\boldsymbol{\theta}_i - \boldsymbol{\theta}_0) \tag{3}$$

Table 4: LiNeS improves performance over Model Soups (Wortsman et al., 2022a), for both uniform and greedy soup in merging multiple checkpoints fine-tuned on ImageNet with different hyper-parameter configurations.

Method	Enhancements	ImageNet Acc.
Averaged accuracy	/	77.98
Best individual model	/	80.36
Uniform soup	/	79.99
	Task Arithmetic	80.17
	LiNeS	**80.47** (+0.48)
Greedy soup	/	81.01
	Task Arithmetic	81.01
	LiNeS	**81.16** (+0.15)

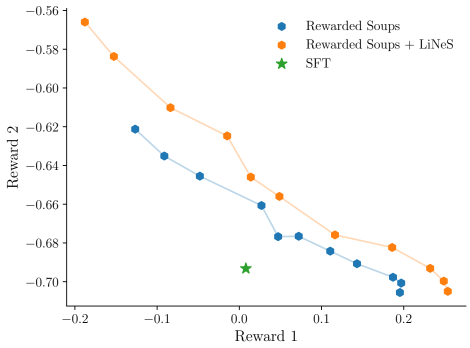


Figure 4: Applying LiNeS to Rewarded Soups (Ramé et al., 2023) improves merging of LLM policies RL fine-tuned on different rewards with a dominating Pareto Front.

We apply LiNeS to $\boldsymbol{\tau}_{\text{soup}}$ fixing $\alpha = 1$ and searching over β. As a baseline, we also consider task arithmetic, where we search for a constant scaling factor on $\boldsymbol{\tau}_{\text{soup}}$. Note that both settings introduce one hyper-parameter to vanilla model soups, and refer to Appendix B.3 for a detailed description of the modifications. Table 4 summarizes the results and shows that LiNeS improves over vanilla soups and task arithmetic for both uniform and greedy soup by 0.48% and 0.15% on ImageNet, respectively. We report the best-performing model and the average performance as baselines. Finally, our proposed method compounds the gains from the greedy soup and leads to the best-performing model.

5.4 Improving Rewarded Soups

In this section, we explore the effectiveness of LiNeS for merging foundation models fine-tuned on different rewards. We consider the Rewarded Soups setting (Ramé et al., 2023), which interpolates the weights $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ of two LLM policies, each optimized for a distinct reward R_1 and R_2, respectively.

Starting with an LLM parameterized by weights $\boldsymbol{\theta}_0$, we first fine-tune it using supervised fine-tuning (SFT) on labeled demonstrations. From the resulting weights $\boldsymbol{\theta}_{\text{SFT}}$, we then apply Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017; Ouyang et al., 2022), training two independent policies via Proximal Policy Optimization (PPO) (Schulman et al., 2017) to maximize the rewards R_1 and R_2 respectively. To merge these policies, we linearly interpolate the residuals $\boldsymbol{\tau}_1 = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_{\text{SFT}}$ and $\boldsymbol{\tau}_2 = \boldsymbol{\theta}_2 - \boldsymbol{\theta}_{\text{SFT}}$. This interpolation defines a continuous set of rewarded policies:

$$\boldsymbol{\theta}_{RS} = \boldsymbol{\theta}_{\text{SFT}} + \lambda\boldsymbol{\tau}_1 + (1-\lambda)\boldsymbol{\tau}_2, \quad \lambda \in [0,1], \tag{4}$$

where the coefficient λ models the user's preferences. We apply LiNeS to the weighted-sum residual: $\lambda\boldsymbol{\tau}_1 + (1-\lambda)\boldsymbol{\tau}_2$, fixing $\alpha = \beta = 1$ for computational reasons.

In our experiment, we use LLaMA-2 7B (Touvron et al., 2023) and the Reddit Summary task (Stiennon et al., 2020), which consists of 14.9k post-summary pairs. We employ two reward models: GPT2-reward-summarization – which scores summaries based on human preferences – and BART-faithful-summary-detector (Chen et al., 2021) – which evaluates the faithfulness of the generated summary to the source post. To evaluate the models, we use a subset of 1k samples from the test set, generate the responses, and compute the average score for each reward dimension.

In Table 4, we present the empirical Pareto Fronts for both Rewarded Soups and Rewarded Soups+LiNeS. LiNeS consistently outperforms the vanilla Rewarded Soups across the full preference space, Pareto dominating the baseline. This result highlights the generality of LiNeS.

6 Discussion

We compare LiNeS with prior work that optimizes the scaling coefficients via backpropagation. Specifically, Ada-merging (Yang et al., 2024) minimizes the entropy loss of the predictions on the test set, while aTLAS (Zhang et al., 2024) minimizes a cross entropy loss on validation samples. Both methods operate on a more fine-grained level and introduce coefficients per layer *and* per task, requiring all $T + 1$ checkpoints, for task vectors and pre-trained model respectively, to be stored in memory during their fine-tuning process.

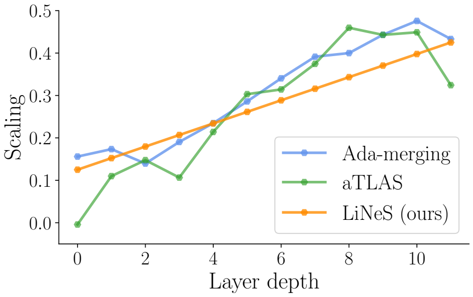


Figure 5: Comparison of the scalings obtained by different methods on 8-task merging benchmark with CLIP ViT-B/32.

We consider the 8-task computer vision benchmark with the ViT-B/32 visual encoder, and present the per-layer scalings in Figure 5. For aTLAS and Ada-merging, we report the average optimized scaling coefficients for attention and linear layers in each block across tasks. Without requiring training, LiNeS leverages the inductive bias of neural networks to achieve scaling very close to aTLAS or Ada-merging, but with much less computational cost. Apart from the excessive memory overhead, both aTLAS and Ada-merging require multiple training epochs, making it challenging to scale for large models. As we demonstrate in Section 5.4, LiNeS efficiently scales to large models like LLaMA (Touvron et al., 2023).

7 Conclusion

In this work, we presented LiNeS, a novel method designed to mitigate catastrophic forgetting after fine-tuning process. By reducing the magnitude for parameter updates in the shallower layers, LiNeS improves the generalization performance of the edited model on control tasks while almost fully preserving performance on the fine-tuned tasks. Furthermore, we demonstrated the versatility of LiNeS in addressing task interference in multi-task model merging, where it consistently improves the baseline model merging methods across vision and NLP benchmarks. Our experiments confirmed the broad applicability of LiNeS across various scenarios, from improving OOD generalization to enhancing multi-task and single-task model merging strategies, as well as improving merging LLM policies aligned with different rewards. Given its simplicity and ease of integration with existing methods, LiNeS offers a practical and inexpensive solution for boosting the generalization and robustness of fine-tuned models in diverse application domains.

Acknowledgments

The authors thank Adam Hazimeh for constructive discussions and comments.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. Better fine-tuning by reducing representational collapse. *arXiv preprint arXiv:2008.03156*, 2020.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://proceedings.neurips.cc/paper/2019/file/97af07a14cacba681feacf3012730892-Paper.pdf.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the Opportunities and Risks of Foundation Models, 2021. URL http://arxiv.org/abs/2108.07258v3.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – Mining Discriminative Components with Random Forests. In *IEEE European Conference on Computer Vision (ECCV)*, 2014.

Rich Caruana. Multitask Learning. *Machine Learning*, 28(1):41–75, 1997.

Sihao Chen, Fan Zhang, Kazoo Sone, and Dan Roth. Improving Faithfulness in Abstractive Summarization with Contrast Candidate Generation and Selection. In *NAACL*, 2021.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 2017. URL http://arxiv.org/abs/1703.00121v1.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. URL http://arxiv.org/abs/1311.3618v2.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv*, 2018. URL http://arxiv.org/abs/1812.01718v1.

K Clark. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. https://proceedings.mlr.press/v15/coates11a.html.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. EMNIST: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks (IJCNN)*, 2017.

MohammadReza Davari and Eugene Belilovsky. Model Breadcrumbs: Scaling Multi-Task Model Merging with Sparse Masks. *arXiv*, 2023. URL http://arxiv.org/abs/2312.06795v1.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. https://ieeexplore.ieee.org/abstract/document/5206848.

Nikolaos Dimitriadis, Pascal Frossard, and François Fleuret. Pareto Manifold Learning: Tackling multiple tasks via ensembles of single-task models. In *International Conference on Machine Learning (ICML)*, 2023.

Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Shuyang Gu, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, and Nenghai Yu. Clip itself is a strong fine-tuner: Achieving 85.7% and 88.0% top-1 accuracy with vit-b and vit-l on imagenet. *arXiv preprint arXiv:2212.06138*, 2022.

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. In *International Conference on Learning Representations (ICLR)*, 2021. URL http://arxiv.org/abs/2010.11929v2.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning (ICML)*, 2018. URL http://arxiv.org/abs/1803.00885v5.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL http://arxiv.org/abs/2010.15110v1.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020. URL http://arxiv.org/abs/1912.05671v4.

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 (NeurIPS)*, 2018. URL http://arxiv.org/abs/1802.10026v4.

Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *International Conference on Neural Information Processing (ICONIP)*, 2013. URL http://arxiv.org/abs/1307.0414v1.

Henry Gouk, Timothy M Hospedales, and Massimiliano Pontil. Distance-based regularisation of deep networks for fine-tuning. *arXiv preprint arXiv:2002.08253*, 2020.

Sachin Goyal, Ananya Kumar, Sankalp Garg, Zico Kolter, and Aditi Raghunathan. Finetune like you pretrain: Improved finetuning of zero-shot vision models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19338–19347, 2023.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

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

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 J. Sel. Top. Appl. Earth Obs. Remote. Sens.*, 12(7):2217–2226, 2019. URL https://doi.org/10.1109/JSTARS.2019.2918242.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Lixuan Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. 2021 ieee. In *CVF International Conference on Computer Vision (ICCV)*, volume 2, 2020.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15262–15271, 2021.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. *arXiv*, 2019. URL http://arxiv.org/abs/1909.00277v2.

Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL http://arxiv.org/abs/2208.05592v2.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023. https://arxiv.org/abs/2110.08207.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. URL http://arxiv.org/abs/1803.05407v3.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning (ICML)*, 2021. URL http://arxiv.org/abs/2102.05918v2.

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

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. QASC: A Dataset for Question Answering via Sentence Composition, 2020. URL http://arxiv.org/abs/1910.11473v2.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=UYneFzXSJWh.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL http://arxiv.org/abs/1612.01474v3.

Yann LeCun. The MNIST database of handwritten digits, 1998. http://yann.lecun.com/exdb/mnist/.

Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv*, 2021. URL http://arxiv.org/abs/2104.08691v2.

Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. Reasoning over paragraph effects in situations. *arXiv*, 2019. URL http://arxiv.org/abs/1908.05852v2.

Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. *arXiv*, 2023.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL http://arxiv.org/abs/2111.09832v2.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*. 1989. https://www.sciencedirect.com/science/article/abs/pii/S0079742108605368.

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.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL http://arxiv.org/abs/2008.11687v2.

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*, 2008.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task Arithmetic in the Tangent Space: Improved Editing of Pre-Trained Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL http://arxiv.org/abs/2305.12827v3.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback, 2022. URL http://arxiv.org/abs/2203.02155v1.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners, 2019. https://openai.com/blog/better-language-models/.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning (ICML)*, 2021. URL http://arxiv.org/abs/2103.00020v1.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 21(140):1–67, 2020.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019a.

Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. *Advances in neural information processing systems*, 32, 2019b.

Alexandre Rame, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 35:10821–10836, 2022.

Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization. In *International Conference on Machine Learning (ICML)*, 2023.

Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 36, 2024.

Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*, 2024.

Anastasia Razdaibiedina, Ashish Khetan, Zohar Karnin, Daniel Khashabi, Vishaal Kapoor, and Vivek Madan. Representation projection invariance mitigates representation collapse. *arXiv preprint arXiv:2205.11603*, 2022.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pp. 5389–5400. PMLR, 2019.

Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. Getting closer to AI complete question answering: A set of prerequisite real tasks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *Commun. ACM*, 64(9):99106, 2021. URL http://arxiv.org/abs/1907.10641v2.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv*, 2019. URL http://arxiv.org/abs/1904.09728v3.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Rishi Sharma, James Allen, Omid Bakhshandeh, and Nasrin Mostafazadeh. Tackling the Story Ending Biases in The Story Cloze Test. In *Association for Computational Linguistics (ACL)*, 2018. URL https://aclanthology.org/P18-2119.

Zhiqiang Shen, Zechun Liu, Jie Qin, Marios Savvides, and Kwang-Ting Cheng. Partial is better than all: Revisiting fine-tuning strategy for few-shot learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 9594–9602, 2021.

Sidak Pal Singh and Martin Jaggi. Model Fusion via Optimal Transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. https://proceedings.neurips.cc/paper/2020/hash/fb2697869f56484404c8ceee2985b01d-Abstract.html.

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 *Empirical Methods in Natural Language Processing (EMNLP)*, 2013. https://aclanthology.org/D13-1170/.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German traffic sign recognition benchmark: a multi-class classification competition. In *International Joint Conference on Neural Networks (IJCNN)*, 2011. https://ieeexplore.ieee.org/document/6033395.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. QuaRTz: An Open-Domain Dataset of Qualitative Relationship Questions. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019. URL https://aclanthology.org/D19-1608.

Derek Tam, Mohit Bansal, and Colin Raffel. Merging by Matching Models in Task Parameter Subspaces. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=qNGo6ghWFB.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023. URL http://arxiv.org/abs/2307.09288v2.

Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2018. URL http://arxiv.org/abs/1806.03962v1.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Mitchell Wortsman, Maxwell C Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning Neural Network Subspaces. In *International Conference on Machine Learning (ICML)*, 2021. URL http://arxiv.org/abs/2102.10472v3.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022a. URL http://arxiv.org/abs/2203.05482v3.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7959–7971, 2022b.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv*, 2017. URL http://arxiv.org/abs/1708.07747v2.

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: 3–22, 2016.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving Interference When Merging Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL http://arxiv.org/abs/2306.01708v2.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive Model Merging for Multi-Task Learning. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=nZP6NgD3QY.

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2015. URL https://aclanthology.org/D15-1237.

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.

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 *Forty-first International Conference on Machine Learning*, 2024.

Frederic Z Zhang, Paul Albert, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. Knowledge composition using task vectors with learned anisotropic scaling. *arXiv preprint arXiv:2407.02880*, 2024.

Haojie Zhang, Ge Li, Jia Li, Zhongjin Zhang, Yuqi Zhu, and Zhi Jin. Fine-tuning pre-trained language models effectively by optimizing subnetworks adaptively. *Advances in Neural Information Processing Systems*, 35:21442–21454, 2022.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*, 2020.

Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase Adversaries from Word Scrambling. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. URL https://aclanthology.org/N19-1131.

Yifan Zhong, Chengdong Ma, Xiaoyuan Zhang, Ziran Yang, Qingfu Zhang, Siyuan Qi, and Yaodong Yang. Panacea: Pareto alignment via preference adaptation for llms. *arXiv preprint arXiv:2402.02030*, 2024.

Jing Zhou, Zongyu Lin, Yanan Zheng, Jian Li, and Zhilin Yang. Not All Tasks Are Born Equal: Understanding Zero-Shot Generalization. In *International Conference on Learning Representations (ICLR)*, 2022.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 2020. URL http://arxiv.org/abs/1911.02685v3.

Appendix

Table of Contents

A LiNeS Pseudocode

We provide here a python pseudocode for the scaling the task vectors.

```
def line_scaling(task_vector, alpha=0.0, beta=1.0, num_blocks=12):
    """
    Progressively scales the task vector based on layer depth.

    Parameters:
    -----------
    task_vector : dict
        A dictionary representing the residual between the fine-tuned checkpoint
        and the pre-trained checkpoint.
    alpha : float
         The minimum scaling factor for the blocks.
    beta : float
        The maximum scaling coefficient difference between the last and first block.
    num_blocks : int
        The total number of layer blocks in the model.
    Returns:
    --------
    scaled_task_vector : dict
        A copy of `task_vector` where each key is scaled based on the layer depth.
    """

    import copy

    # Deep copy the task vector to avoid modifying the original
    scaled_task_vector = copy.deepcopy(task_vector)

    # Generate the key blocks corresponding to the layers of the model
    key_blocks = [f".layer{i}." for i in range(num_blocks)]

    # Create a scaling dictionary to store the scaling factor for each key
    scaling_dic = {}
    for k in task_vector.keys():
        # Find the layer block in the key and assign scaling factor based on layer depth
        for layer, block in enumerate(key_blocks):
            if block in k:
                scaling_dic[k] = alpha + beta * (layer / (num_blocks - 1))
                break

    # Scale the task vector based on the scaling dictionary
    scaled_task_vector.vector = {
        # Use alpha if layer is outside residual blocks
        k: task_vector.vector[k] * scaling_dic.get(k, alpha)
        for k in task_vector.keys()
    }

    return scaled_task_vector

# example: scale single-task fine-tuned residual
task_vector = {k:theta_t[k] - theta_0[k] for k in theta_0.keys()}
scaled_task_vector = line_scaling(task_vector, alpha=gamma, beta=1.0-gamma, num_blocks=12)

# example: Scale the multi-task vectors
mtv = {k: sum(theta_ft[k] - theta_0[k] for theta_ft in ft_models) for k in theta_0.keys()}
scaled_mtv = line_scaling(mtv, alpha=1/len(ft_models), beta=beta, num_blocks=12)
```

B Experimental Details

B.1 Descriptions of baseline model merging methods

- **Task Arithmetic** (Ilharco et al., 2023) generates a multi-task vector by summing the individual task vectors for each task. This multi-task vector is then added to the pre-trained checkpoint, with a scaling factor chosen based on validation set performance.
- **Ties-Merging** (Yadav et al., 2023) resolves parameter conflicts during model merging by first pruning parameters with lower magnitudes from the individual task vectors, followed by addressing sign mismatches, and finally merging parameters with consistent signs with averaging operation. The resulting multi-task vector is then added to the pre-trained checkpoint using a scaling factor determined from the validation set.

- **Consensus Merging** (Wang et al., 2024) enhances existing model merging techniques by eliminating redundant weights in the multi-task vector. It first identifies the relevant subset of parameters for each task, then filters out weights that are relevant to either none or only one task. After removing these redundant weights, the refined multi-task vector is added to the pre-trained checkpoint with a scaling factor selected from the validation set.

While consensus merging can be applied to various merging methods, in all our experiments, we evaluate only its application to task arithmetic.

B.2 Experimental details for multi-task model merging

B.2.1 Hyper-parameters tuning

We list here the hyper-parameter search space for each model merging method in Table B.2.1, while we suggest the authors to the original papers for a detailed description of these hyper-parameters. We highlight that applying `LiNeS` does not introduce extra computational cost in hyper-parameter search for the baseline merging methods.

Method	With `LiNeS`	Hyper-parameter search space
Task Arithmetic	✗	constant scaling term for multi-task vector: [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]
	✓	scaling term β in Eq. 1 for the multi-task vector: [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]
Ties-Merging	✗	constant scaling term for multi-task vector: [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.1,1.2,1.3,1.4,1.5]
	✓	scaling term β in Eq. 1 for the multi-task vector: [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.1,1.2,1.3,1.4,1.5]
Consensus Merging	✗	constant scaling term for multi-task vector: [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]; weight-pruning threshold: [1, 2]
	✓	scaling term β in Eq. 1 for the multi-task vector: [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]; weight-pruning threshold: [1, 2]

B.2.2 Benchmarks

Image classification For the benchmarks used in image classification, we utilized the 8-task benchmark initially proposed by Ilharco et al. (2023), as well as the 14-task and 20-task benchmarks expanded by Wang et al. (2024).

- The **8-task benchmark** comprises the following tasks: Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2011), MNIST (LeCun, 1998), RESISC45 (Cheng et al., 2017), SUN397 (Xiao et al., 2016), and SVHN (Netzer et al., 2011).
- The **14-task benchmark** includes the original eight tasks plus additional ones: CIFAR100 (Krizhevsky et al., 2009), STL10 (Coates et al., 2011), Flowers102 (Nilsback & Zisserman, 2008), OxfordIIITPet (Parkhi et al., 2012), PCAM (Veeling et al., 2018), and FER2013 (Goodfellow et al., 2013).
- The **20-task benchmark** builds on the 14-task benchmark with the addition of: EMNIST (Cohen et al., 2017), CIFAR10 (Krizhevsky et al., 2009), Food101 (Bossard et al., 2014), FashionMNIST (Xiao et al., 2017), RenderedSST2 (Socher et al., 2013; Radford et al., 2019), and KMNIST (Clanuwat et al., 2018).

Natural Language Processing For our NLP experiments, we utilized benchmarks established by Yadav et al. (2023), Tam et al. (2024), and Wang et al. (2024).

- The **7 NLP Tasks** benchmark, as explored in Yadav et al. (2023), includes the following datasets: QASC (Khot et al., 2020), QuaRTz (Tafjord et al., 2019), PAWS (Zhang et al., 2019), Story Cloze (Sharma et al., 2018), WikiQA (Yang et al., 2015), Winogrande (Sakaguchi et al., 2021), and WSC (Levesque et al., 2012).
- The **8 QA Tasks** (Tam et al., 2024) comprises the following datasets: CosmosQA Huang et al. (2019), QASC (Khot et al., 2020), QuAIL Rogers et al. (2020), QuaRTz (Tafjord et al., 2019), PAWS (Zhang et al., 2019), ROPES Lin et al. (2019), SocialIQA Sap et al. (2019), and WikiQA (Yang et al., 2015).

- The **11 NLP Tasks** benchmark is a union of these two benchmarks, as studied in Wang et al. (2024). It contains the following tasks: QASC (Khot et al., 2020), QuaRTz (Tafjord et al., 2019), PAWS (Zhang et al., 2019), Story Cloze (Sharma et al., 2018), WikiQA (Yang et al., 2015), Winogrande (Sakaguchi et al., 2021), WSC (Levesque et al., 2012), CosmosQA Huang et al. (2019), QuAIL Rogers et al. (2020), ROPES Lin et al. (2019), and SocialIQA Sap et al. (2019).

B.3 Experimental details for single-task model merging

B.3.1 Description of Model Soups

Model soups (Wortsman et al., 2022a) is a model merging method which averages the weights of multiple fine-tuned models with different hyper-parameter configurations, improving accuracy of the merged model without increasing inference or memory costs. The authors of model soups proposed two methods:

- **Uniform soup**: Averages the weights of all fine-tuned checkpoints, providing a simple and efficient way to improve performance.
- **Greedy soup**: Starting with the best-performing checkpoint, greedily and iteratively adds the next best-performing checkpoint to the soup, keeping those that improve accuracy of current collection of model checkpoints.

B.3.2 Experimental details for modifications to model soups

We describe in detail the modifications to model soups, namely task arithmetic and our proposed `LiNeS`. For reference, model soups merges the checkpoints by averaging the weights of the individual checkpoints:

$$\boldsymbol{\theta}_{\text{soup}}^{\text{vanilla}} = \boldsymbol{\theta}_0 + \boldsymbol{\tau}_{\text{soup}} \tag{5}$$

Enhancing Model Soups with Task Arithmetic We enhance model soups with task arithmetic, by introducing a scaling factor λ_{ta} to $\boldsymbol{\tau}_{\text{soup}}$ in Equation 5. We search for this hyper-parameter within the range of $[1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0]$. Note that $\lambda_{\text{ta}} = 1.0$ yields the vanilla model soups.

$$\boldsymbol{\theta}_{\text{soup}}^{\text{ta}} = \boldsymbol{\theta}_0 + \lambda_{\text{ta}} \cdot \boldsymbol{\tau}_{\text{soup}} \tag{6}$$

Enhancing Model Soups with `LiNeS` We enhance model soups with `LiNeS`, by applying `LiNeS` to $\boldsymbol{\tau}_{\text{soup}}$ in Equation 5. For the scaling, we apply directly the scaling introduced in Equation 1 to create a scaled task vector $\boldsymbol{\tau}_{\text{soup}}^{\text{LiNeS}}$, fixing α to 1 while searching the value for β within the range of [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]. Note that $\beta = 0.0$ yields the vanilla model soups.

$$\boldsymbol{\theta}_{\text{soup}}^{\text{LiNeS}} = \boldsymbol{\theta}_0 + \boldsymbol{\tau}_{\text{soup}}^{\text{LiNeS}} \tag{7}$$

We further note that, both Task Arithmetic and our proposed method introduce only one hyper-parameter to model soups, while the computational cost for hyper-parameter search is the same. When applying the modifications to greedy soup, we only apply them directly to the selected subset of checkpoints after the greedy selection process.

We search for the hyper-parameter within the validation set and report the performance on test set with the best hyper-parameter based on validation performance.

C Additional Results

C.1 Different trade-offs for retention of task and control task performance

In Section 3, we need to balance two competing objectives: maximizing accuracy on the target task while preserving performance on the control task. To account for different user preferences, we scalarize these objectives by assigning varying weights to the target task accuracy. This weighting scheme can be adjusted depending on the scenario to reflect different priorities. Let w_{target} represent the weight assigned to the target task accuracy, and M_{target} and $\text{M}_{\text{control}}$ denote the normalized accuracies for the target and control tasks, respectively. The optimal value of γ is selected to maximize the following weighted trade-off on the validation set:

$$w_{\text{target}}\text{M}_{\text{target}} + \text{M}_{\text{control}}$$

To account for the high variance in control task performance and to emphasize the target task, we assign it a weight of 2, signifying that its accuracy is prioritized twice as much as the control task's accuracy.

Table 5: Validation results on the target vs control performance benchmark, presented in Section 3, averaged over the 8 tasks. We balance two competing objectives with various scalarization weights w_{target}. In the main text, we use $w_{\text{target}} = 2$.

w_{target}	Averaged normalized accuracy (%)	
	Target task	Control tasks
1	99.8	101.9
2	100.0	101.5
5	100.2	100.8

C.2 Detailed labels for Figure 2

We provide in Figure 6 the detailed labels corresponding to each scatter dot. Each scatter dot corresponds to applying a specific model (FT for fine-tuned model; PT for pre-trained model; LS for fine-tuned model edited with `LiNeS`) on different tasks.

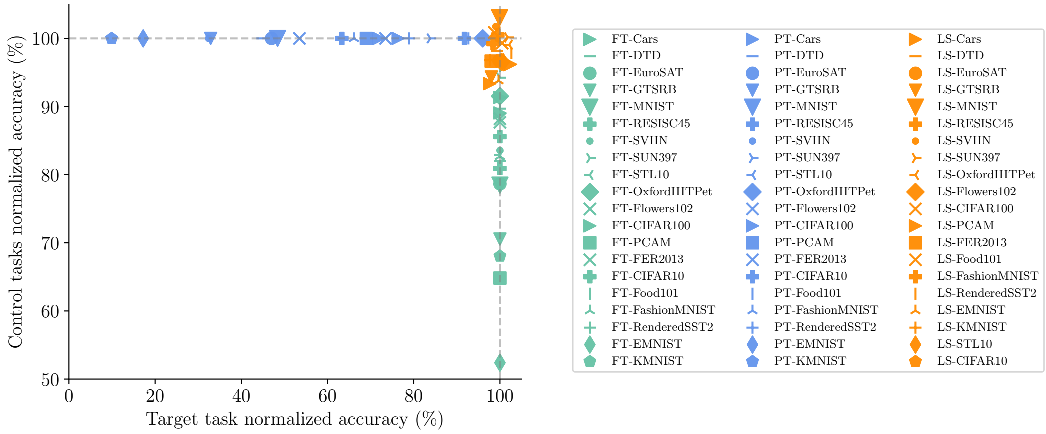


Figure 6: Figure 2 with detailed label information. Each scatter dot corresponds to applying a specific model (FT for fine-tuned model; PT for pre-trained model; LS for fine-tuned model edited with `LiNeS`) on different task.

C.3 Ablations of different choices of scaling function

We provide in this section an ablation study on applying different scaling functions for `LiNeS`. In `LiNeS` we used directly $\lambda^{(\ell)} = \alpha + \beta \cdot \frac{\ell-1}{L-1}$ to scale different layers. Here we test the performance on multi-task model merging in vision benchmarks with the following choices for scaling functions $f(\cdot)$: linear scaling, quadratic scaling and square root scaling:

- linear scaling: $\lambda^{(\ell)} = \alpha + \beta \cdot \frac{\ell-1}{L-1}$
- square root scaling: $\lambda^{(\ell)} = \alpha + \beta \cdot \left(\frac{\ell-1}{L-1}\right)^{\frac{1}{2}}$
- quadratic scaling: $\lambda^{(\ell)} = \alpha + \beta \cdot \left(\frac{\ell-1}{L-1}\right)^{2}$

We provide in Table C.3 the performance of different choices of scaling on vision benchmarks with ViT-B/32. While using quadratic scaling sometimes outperforms using identify function, especially with a larger number of tasks during merging, the improvement is not substantial. Therefore, we choose the linear scaling to keep the method simple and general.

Table 6: Ablation study for applying different scalings for `LiNeS` on vision benchmarks with ViT-B/32.

Method	Scaling function	ViT-B/32		
		8 tasks	14 tasks	20 tasks
Task Arithmetic	linear	**74.2**	69.1	63.4
	square root	73.9	67.5	62.4
	quadratic	73.8	**69.2**	**64.6**
Ties-Merging	linear	**77.2**	**72.1**	67.2
	square root	76.9	70.4	65.6
	quadratic	76.1	71.6	**67.4**
Consensus Merging	linear	**77.6**	73.6	68.6
	square root	77.1	72.5	67.3
	quadratic	77.1	**73.9**	**69.0**

C.4 Results for ViT-B/16 for multi-task merging

We provide in Table C.4 the results complementary to Table 2 for using ViT-B/16 as the image encoder, where we observe similar performance gains and observations by using `LiNeS` as in Table 2.

Table 7: Complementary to Table 2, for results obtained with ViT-B/16 as image encoder.

Method	with `LiNeS`	ViT-B/16		
		8 tasks	14 tasks	20 tasks
Zero-shot		55.5	61.4	59.8
Fine-tuned		92.6	91.6	92.3
Task Arithmetic	✗	74.6	70.4	65.7
	✓	**77.6** (+3.0)	**72.7** (+2.3)	**67.7** (+2.0)
Ties-Merging	✗	79.1	73	68.1
	✓	**79.9** (+0.8)	**75.2** (+2.2)	**71.2** (+3.1)
Consensus Merging	✗	78.9	73.9	70.2
	✓	**79.5** (+0.6)	**75.8** (+1.9)	**72.0** (+1.8)

C.5 Results for editing T5 with `LiNeS`

We repeat here similar experiments we performed on ViT-B/32 model in Section 3 with T5-large (Raffel et al., 2020). T5-large contains both encoder and decoder structure, with sequential residual blocks in both structures. We investigate separately how the shallow-layer updates in the encoder and decoder infect the target and control task accuracy.

We consider the 8-question-answering benchmark (Zhou et al., 2022), and plot in Figure 7 the averaged target and control task accuracy after applying `LiNeS` to

1. only the decoder part (left),
2. only the encoder part (middle),
3. both the encoder and decoder part (right).

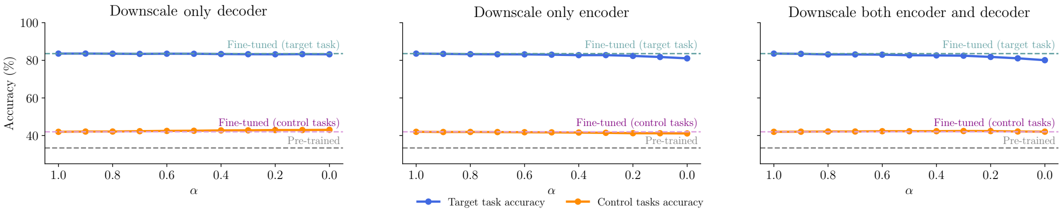


Figure 7: The impact of downscaling the shallower-layer parameter updates on T5-large model within the 8-question-answering benchmark (Raffel et al., 2020). Downscaling only on the decoder (left) architecture preserves full target performance, while slightly improving the control tasks performance. Downscaling on the encoder leads to performance degradation on target tasks.

We observe that, only downscaling on the decoder architecture fully preserves full target performance, while downscaling on the encoder, or on both encoder and decoder, leads to performance drop on the target tasks. On the other hand, downscaling on the decoder part slight improves control generalization of the fine-tuned model, which we do not observe from downscaling on the encoder, or simultaneously on the encoder and decoder. We also note that, unlike the case in vision, the fine-tuned checkpoints on this NLP benchmark actually improve over the zero-shot performance of the pre-trained model on control tasks.

These results motivate us to apply LiNeS to only the decoder part of T5-large when merging multiple checkpoints, which preserves full target task accuracy while slightly improving control task performance, leading to similar observation in applying LiNeS to the ViT-B/32 architecture in vision,

C.6 Additional results for improving WiSE-FT with LiNeS

C.6.1 Results for using ViT-B/16 as visual encoder

We provide in Figure 8 the results for applying LiNeS for improving WiSE-FT, using ViT-B/16 as visual encoder. The ViT-B/16 checkpoint obtained through fine-tuning the CLIP checkpoint on ImageNet with the same hyper-parameter configurations in Wortsman et al. (2022b). From Figure 8 we observe that LiNeS improves over WiSE-FT for both ID and OOD accuracies, leading to similar observations as the results obtained with ViT-B/32.

C.6.2 Individual results for 70 checkpoints

We provide in Figure 9 individual results separately for the experiments on the 70 individual model checkpoints. Note that here y-axis represents the averaged accuracy over 5 OOD datasets. From the figure, we observe that LiNeS consistently improves WiSE-FT in terms of both ID and OOD accuracies for most of the individual checkpoints.

C.7 Detailed performance on individual tasks for multi-task model merging

Image Classification We provide the detailed performance on each individual task for multi-task model merging in image classification benchmarks, complementary to the results in Table 2 and Table C.4 where the accuracies are averaged on the individual tasks.

The single-task performance is presented in Figure 10 for ViT-B/32, Figure 11 for ViT-B/16, and Figure 12 for ViT-L/14. From the results we observe that our method demonstrates a noticeable improvement over baseline merging techniques across individual tasks in all test scenarios.

Natural Language Processing We provide in Figure 13 the detailed single-task performance for the three NLP benchmarks using T5-large, complementary to the results in Table 3. Similar to the

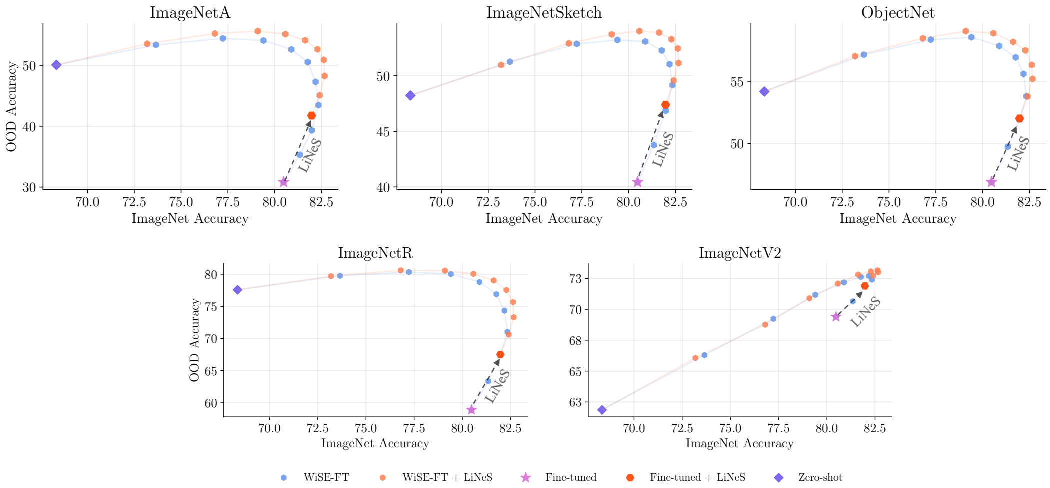


Figure 8: Results for improving WiSE-FT with LiNeS on with ViT-B/16 model fine-tuned on ImageNet.

observation in vision, our method provides a consistent improvement over baselines across individual tasks.

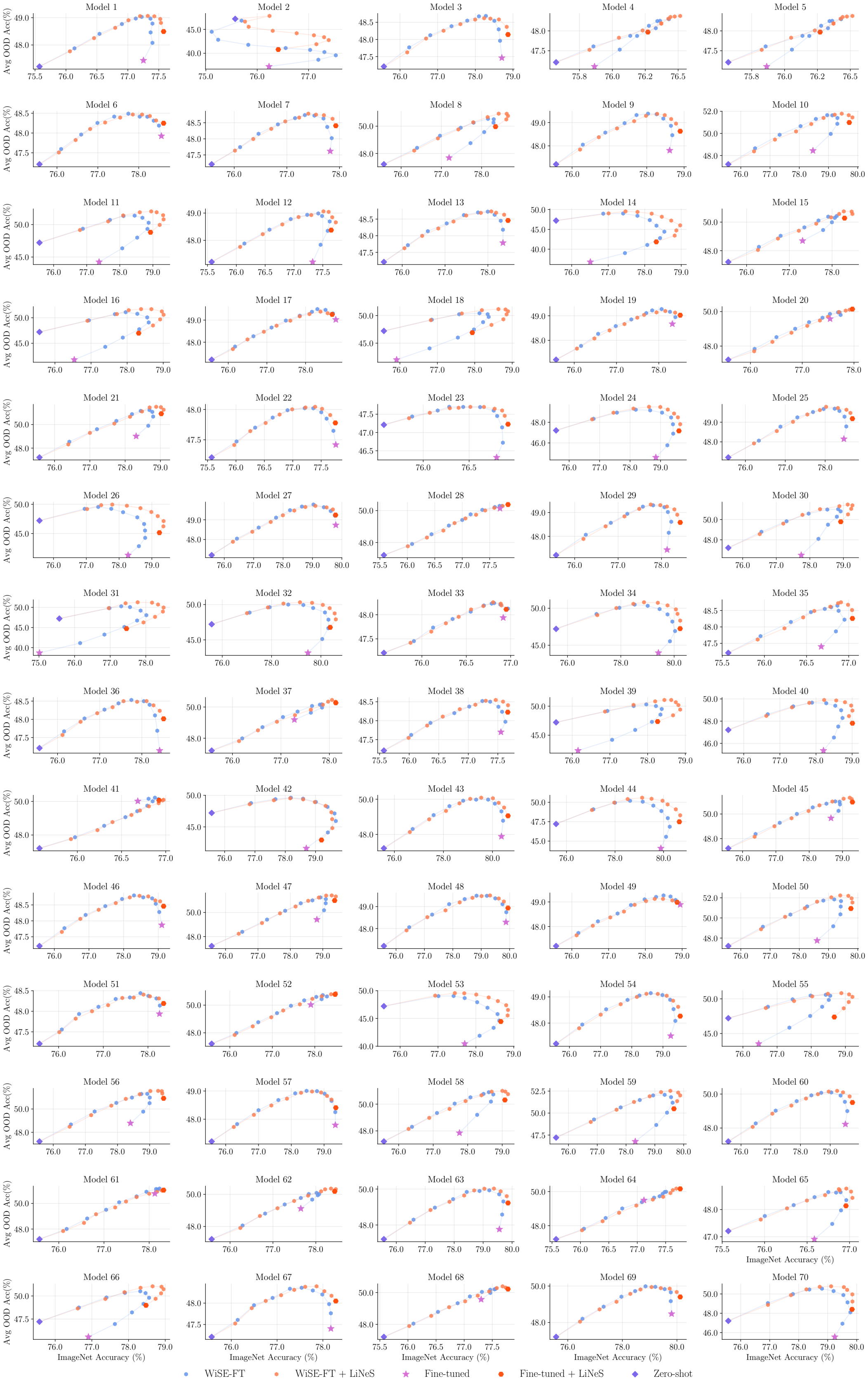
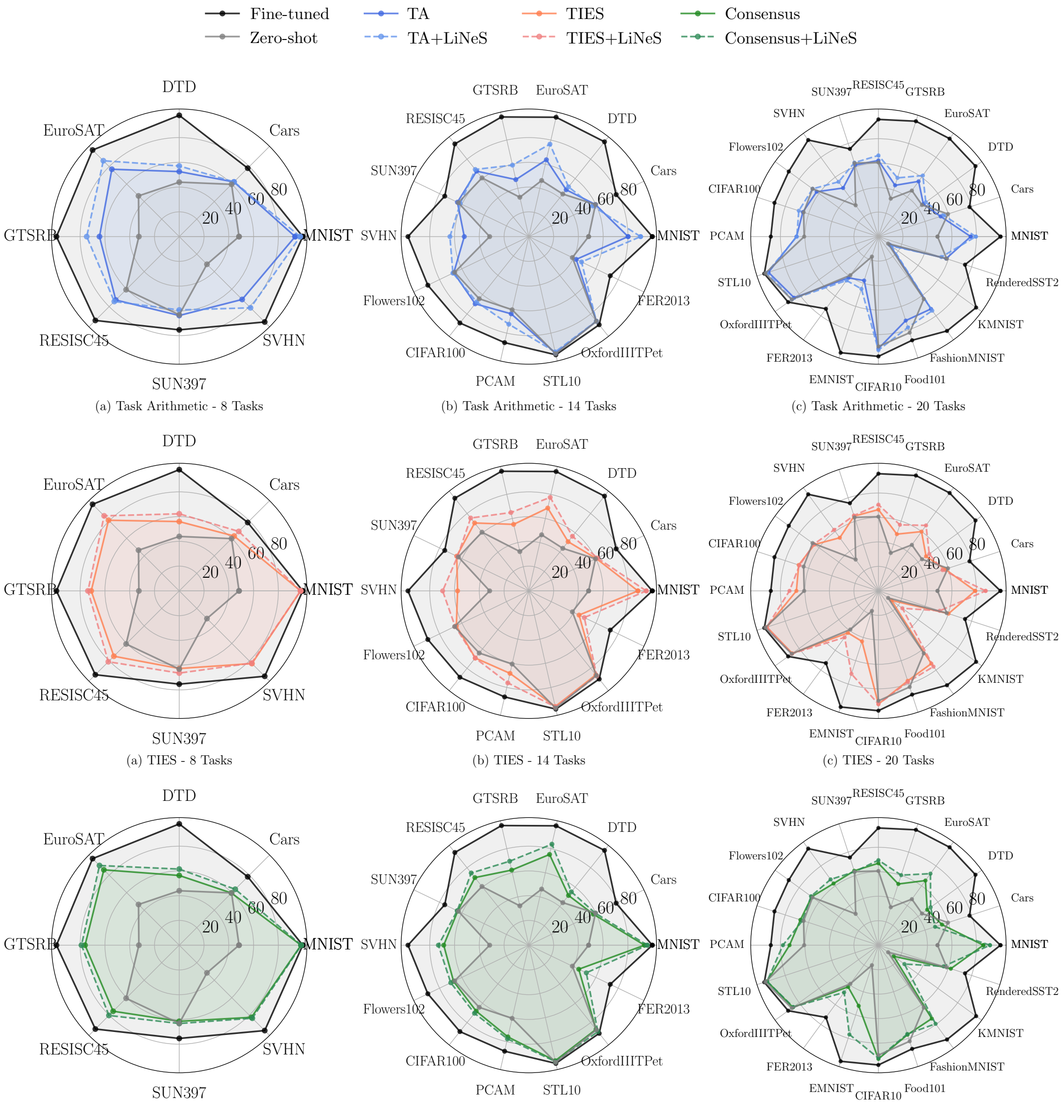


Figure 9: Performance of applying `LiNeS` to WiSE-FT to each ViT-B/32 checkpoint fine-tuned on ImageNet.



(a) Task Arithmetic - 8 Tasks

(b) Task Arithmetic - 14 Tasks

(c) Task Arithmetic - 20 Tasks

(a) TIES - 8 Tasks

(b) TIES - 14 Tasks

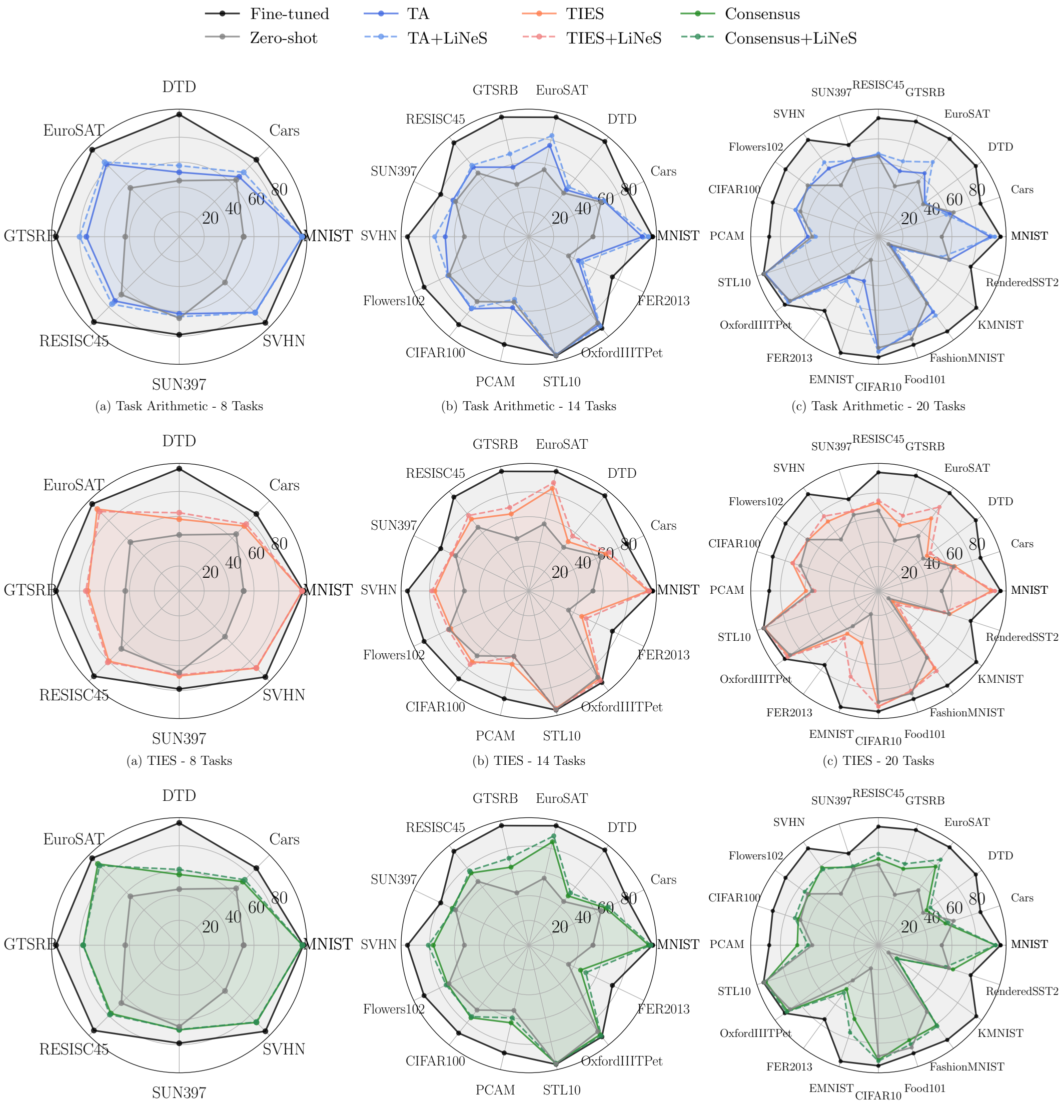
(c) TIES - 20 Tasks

(a) Consensus - 8 Tasks

(b) Consensus - 14 Tasks

(c) Consensus - 20 Tasks

Figure 10: Single-task accuracies for multi-task merging on image classification benchmarks for ViT-B/32.



(a) Task Arithmetic - 8 Tasks

(b) Task Arithmetic - 14 Tasks

(c) Task Arithmetic - 20 Tasks

(a) TIES - 8 Tasks

(b) TIES - 14 Tasks

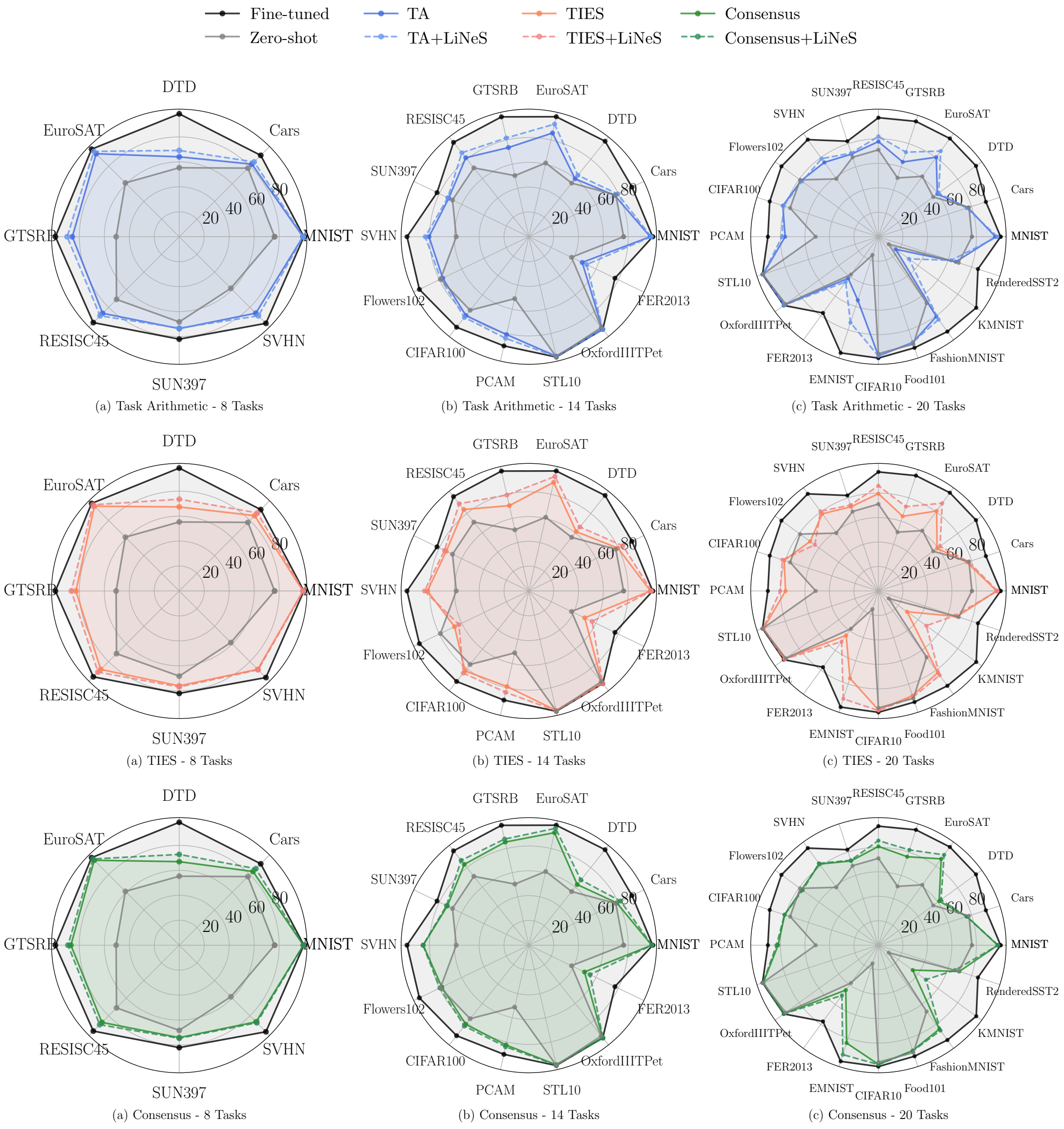
(c) TIES - 20 Tasks

(a) Consensus - 8 Tasks

(b) Consensus - 14 Tasks

(c) Consensus - 20 Tasks

Figure 11: Single-task accuracies for multi-task merging on image classification benchmarks for ViT-B/16.



(a) Task Arithmetic - 8 Tasks

(b) Task Arithmetic - 14 Tasks

(c) Task Arithmetic - 20 Tasks

(a) TIES - 8 Tasks

(b) TIES - 14 Tasks

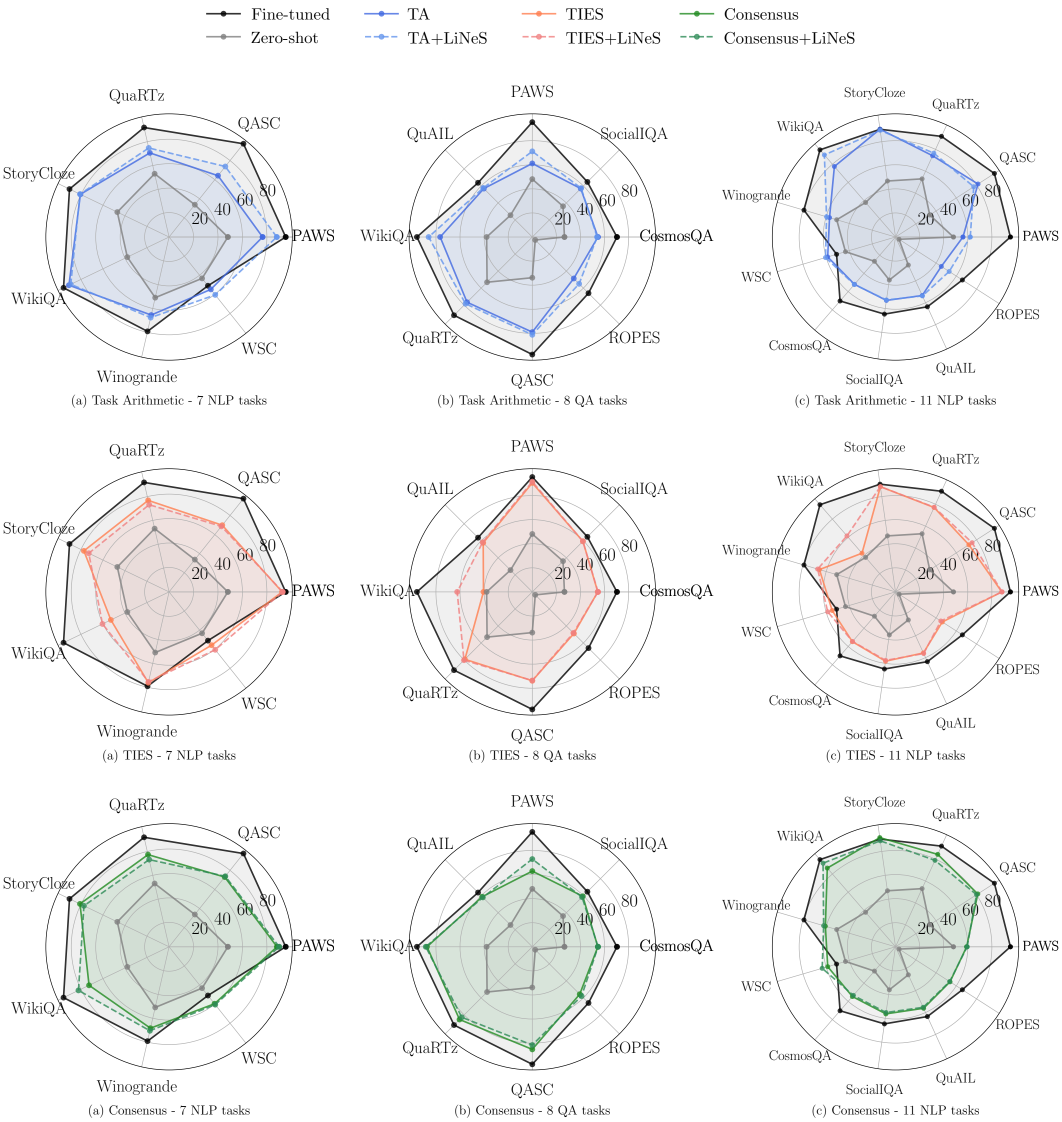
(c) TIES - 20 Tasks

(a) Consensus - 8 Tasks

(b) Consensus - 14 Tasks

(c) Consensus - 20 Tasks

Figure 12: Single-task accuracies for multi-task merging on image classification benchmarks for ViT-L/14.



(a) Task Arithmetic - 7 NLP tasks

(b) Task Arithmetic - 8 QA tasks

(c) Task Arithmetic - 11 NLP tasks

(a) TIES - 7 NLP tasks

(b) TIES - 8 QA tasks

(c) TIES - 11 NLP tasks

(a) Consensus - 7 NLP tasks

(b) Consensus - 8 QA tasks

(c) Consensus - 11 NLP tasks

Figure 13: Single-task accuracies for multi-task merging on NLP benchmarks for T5-large.