MITIGATING PARAMETER INTERFERENCE IN MODEL MERGING VIA SHARPNESS-AWARE FINETUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale deep learning models with a pretraining-finetuning paradigm have led to a surge of numerous task-specific models finetuned from a common pretrained model. Recently, several research efforts have been made on merging these large models into a single multi-task model, particularly with simple arithmetic on parameters. Such merging methodology faces a central challenge: interference between model parameters finetuned on different tasks. Few recent works have focused on desiging a new finetuning scheme that can lead to small parameter interference, however at the cost of the performance of each task-specific finetuned model and thereby limiting that of a merged model. To improve the performance of a merged model, we note that a finetuning scheme should aim for (1) smaller parameter interference and (2) better performance of each finetuned model on the corresponding task. In this work, we aim to design a new finetuning objective function to work towards these two goals. In the course of this process, we find such objective function to be strikingly similar to sharpness-aware minimization (SAM) objective function, which aims to achieve generalization by finding flat minima. Drawing upon our observation, we propose to finetune pretrained models via SAM or its variants. The experimental and theoretical results showcase the effectiveness and orthogonality of our proposed approach, improving performance upon various merging and finetuning methods.

1 INTRODUCTION

Foundation model, a large deep learning model pretrained on large-scale datasets, has shown great advancement across a wide range of downstream tasks, after finetuning on each task (Achiam et al., 2023; Saab et al., 2024; Ding et al., 2023). Recent successes of the pretraining-finetuning paradigm have given rise to a burst of task-specific open-source models in communities, such as Hugging Face. Diversity yet ready availability of large task-specific models have naturally elicited a question from researchers: Can we combine these large models into one, while retaining the performance on each task?

Traditionally, a single multi-task model is obtained by jointly training on data across all tasks (Caruana, 1997; Crawshaw, 2020; Vandenhende et al., 2022). However, given the size of foundation models and the number of tasks, joint training on all tasks incurs significant computational costs. Motivated by the accessibility, variety, abundance, and common origin of task-specific models, several research efforts have focused on merging multiple finetuned models into a single model via simple arithmetic on parameters of these models, thereby removing the need for joint training (Ilharco et al., 2023; Yadav et al., 2023; Yang et al., 2024b; Matena & Raffel, 2022; Jin et al., 2023; Daheim et al., 2024; Li et al., 2023; Yang et al., 2024a). However, a central challenge remains: parameters of different task-specific models interfere or conflict with each other, leading to the performance degradation of a merged multi-task model on each task.

To bridge such performance gap, several works have tried to reduce the parameter interference during the process of merging (Yadav et al., 2023; Jin et al., 2023; Yang et al., 2024b; Wang et al., 2024; Yu et al., 2024). Another line of works focuses on finding a new finetuning scheme that results in task-specific models whose parameters have lower parameter interference (also often referred to as better weight disentanglement with respect to model outputs) (Ortiz-Jimenez et al., 2023; Tang et al., 2024; Jin et al., 2024) and thus less performance degradation after merging. Few studies (Wortsman et al., 2022a; Ilharco et al., 2023; Wortsman et al., 2022b) suggest that the effectiveness of linear arithmetic on parameters in the process of merging may be owed to the linearity of finetuning process. Conversely, Ortiz-Jimenez et al. (2023) have refuted such hypothesis by showing that there is a huge performance drop with approximation of finetuned models with a linearized pretrained model. Another observation they make is that such post-hoc linearized models led to less parameter interference. Based on this observation, few recent works Ortiz-Jimenez et al. (2023); Tang et al. (2024); Jin et al. (2024) have tried to explicitly linearize finetuning processes in order to induce weight disentanglement.

In this work, we note that we need to simultaneously work towards two goals for effective model merging: (1) reducing parameter interference between finetuned models while (2) maintaining the performance of task-specific finetuned models on respective datasets. Therefore, during finetuning process, we aim to directly optimize for both performance on each task and weight disentanglement with respect to performance. In the course of designing a finetuning objective function that aligns with our goals, we find striking resemblances between our goals and sharpness-aware minimization (SAM) (Foret et al., 2021), which aims for better generalization by finding flat minima via minimization of both loss values and loss sharpness. In particular, we find the similarities between the minimization of both loss values and loss sharpness in SAM and joint optimization for performance and weight disentanglement of finetuned models in our goal.

Drawing upon our observations, we propose to finetune pretrained models via SAM or its improved variants (particularly, ASAM in this work), in order to achieve better performance on each task, lower parameter interference, and thus better overall performance of a merged multi-task model. Our extensive experimental results demonstrate that our proposal greatly improves the overall performance of a merged model. The effectiveness of our proposed method is owed to achieving better performance of each task-specific model and less performance gap between task-specific models and a merged model. We further highlight the generalizability and orthogonality of our approach by demonstrating performance improvements when applied together with various merging methods and finetuning methods for model merging.

2 RELATED WORKS

Model Merging. The recent emergence of large foundation models and pretraining-finetuning paradigm has motivated researchers to explore ways of merging multiple finetuned models into a single model without retraining. Model merging, the merging of models with simple arithmetic on parameters, has garnered a significant amount of attention for its flexibility and simplicity. However, parameters of different task-specific models may interfere with each other during merging process, resulting in performance degradation on each task, compared to task-specific models.

To address the parameter interference issue, researchers focus on either designing a merging process (Utans, 1996; Ilharco et al., 2023; Yadav et al., 2023) or designing a finetuning process to mitigate the parameter interference. Initiated with simple averaging (Utans, 1996; Shoemake, 1985), research works on merging process focus on representing task-specific models as task vectors for easier manipulation of knowledge (Ilharco et al., 2023), or weighting parameters (Matena & Raffel, 2022; Jin et al., 2023; Yang et al., 2024b) or selecting parameters (Yadav et al., 2023; Wang et al., 2024; Yu et al., 2024) according to the estimated importance of each parameter with respect to given tasks.

In parallel, if task-specific model parameters have less interference with each other to begin with, the effectiveness of model merging can be amplified. As such, few recent works have focused on designing a finetuning process such that resulting finetuned model parameters will have less interference and result in less performance gap between a merged model and task-specific models. Ortiz-Jimenez et al. (2023) show that linearized finetuning (finetuning in the space tangent to pretrained initialization) leads to less interference (specifically better weight disentanglement with respect to model outputs), aspring other linearized finetuning methods (Jin et al., 2024; Tang et al., 2024).

Sharpness-Aware Minimization (SAM). Foret et al. (2021) introduce a new optimization objective function that minimizes both loss and loss sharpness to seek flat loss minima that may lead to better generalization performance. SAM defines loss sharpness as a maximum loss difference measured at current parameters and nearby parameters (obtained by perturbing current parameters). Several follow-up works have strived to improve SAM via improving perturbation methods (Mi et al., 2022; Kwon et al., 2021), improving gradient (Wang et al., 2023; Zhao et al., 2022), or combining other flatness-aware optimizers (Cha et al., 2021; Kaddour et al., 2022) for better generalization. While previous studies have primarily focused on single-task learning, Phan et al. (2022) incorporates SAM into joint multi-task training as a regularization technique for multi-task learning. We note that our method and Phan et al. (2022) target two different scenarios. Phan et al. (2022) target a traditional multi-task learning scenario, where the training is performed on all tasks jointly. By contrast, our work tackles multi-task model merging, where the goal is to merge different task-specific models, each of which is independently finetuned from a common pretrained model without the knowledge of other tasks. This approach eliminates the need to train all tasks at the same time and avoids retraining from scratch when new tasks are introduced. The lack of the knowledge of other tasks also brings several challenges, such as parameter interference between different task-specific models that cause degradation of single-task performance after merging.

In this work, we introduce a new objective function for single-task finetuning aimed for model merging, from which we present a new insight that draws connections between the objective of multi-task model merging and that of sharpness-aware minimization (SAM). Furthermore, we theoretically (in Appendix D) and empirically show that, SAM can reduce parameter interference, even without the knowledge of other tasks during finetuning.

3 BACKGROUND

Sharpness-aware minimization (SAM). To achieve better generalization, SAM Foret et al. (2021) seeks for wider minima by minimizing both loss value and loss sharpness during optimization, where the loss sharpness is formulated as a difference between a loss at the current parameters and the maximum loss value at nearby parameter values:

$$\min_{\boldsymbol{\theta}} \underbrace{\left[\max_{\boldsymbol{\epsilon}:\|\boldsymbol{\epsilon}\|_2 \le \rho} \mathcal{L}(\boldsymbol{\theta}+\boldsymbol{\epsilon};\mathcal{D}) - \mathcal{L}(\boldsymbol{\theta};\mathcal{D}) \right]}_{\text{loss sharpness}} + \underbrace{\mathcal{L}(\boldsymbol{\theta};\mathcal{D})}_{\text{loss}}, \tag{1}$$

where $\boldsymbol{\epsilon}$ is a perturbation vector which is bounded above by a predefined ρ that controls the radius of the neighborhood; and $\boldsymbol{\theta}$ are network parameters to be optimized for a given loss function $\mathcal{L}$ over a dataset $\mathcal{D}$. For efficiency, Foret et al. (2021) approximates the inner maximization via Taylor approximation. Then, along with canceling identical terms $\mathcal{L}(\boldsymbol{\theta};\mathcal{D})$ with opposite signs, the original optimization is reduced to

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}+\hat{\boldsymbol{\epsilon}};\mathcal{D}) \quad \text{where} \quad \hat{\boldsymbol{\epsilon}} \triangleq \rho \frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta};\mathcal{D})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta};\mathcal{D})\|}. \tag{2}$$

However, the same neighborhood radius for all parameters may impact each parameter differently, especially if their scales differ by several factors. To take such varying scales of parameters into account, Adaptive SAM (ASAM) (Kwon et al., 2021) proposes to scale the perturbation vector $\boldsymbol{\epsilon}$ according to the scale of each parameter as follows:

$$\hat{\boldsymbol{\epsilon}}_{\text{ASAM}} \triangleq \rho \frac{\boldsymbol{\theta}^2 \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta};\mathcal{D})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta};\mathcal{D})\|}. \tag{3}$$

Adjusting the scale of perturbations according that of parameters can be even more effective in the pretraining-finetuning paradigm, since pretrained models likely have parameters of different scales after training on large-scale datasets.

Problem setting. In the pretraining-finetuning paradigm, there exists a large pretrained model $f : \mathcal{X} \times \Theta \to \mathcal{Y}$, parameterized by trained parameters $\boldsymbol{\theta}_0 \in \Theta$, that is in turn finetuned to T downstream tasks. Each downstream task, indexed by t, is accompanied with a dataset $\mathcal{D}^{(t)} = \{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^{N_t}$, where $\boldsymbol{x}_i^{(t)} \in X^{(t)} \subseteq \mathcal{X}$ is an input with a corresponding label $y_i^{(t)} \in Y^{(t)} \subseteq \mathcal{Y}$. Employing a standard loss function (e.g., cross-entropy loss for classification) and an optimizer (e.g., SGD), finetuning a pretrained model $f_{\boldsymbol{\theta}_0}$ to each downstream task t will lead to a task-specific model $f_{\boldsymbol{\theta}_t}$ with its parameters $\boldsymbol{\theta}_t$:

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta};\mathcal{D}^{(t)}). \tag{4}$$

Task arithmetic. To perform an arithmetic on model parameters and merge models into a single model, Ilharco et al. (2023) have introduced the concept of task vector, which is essentially a vector pointing to task-specific parameters $\boldsymbol{\theta}_t$ from pretrained model parameters $\boldsymbol{\theta}_0$, obtained by taking a difference between them: $\boldsymbol{\tau}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$. Ilharco et al. (2023) note that a task vector $\boldsymbol{\tau}_t$ can be considered as the representation of the knowledge learned for a task t. As such, they claim that the knowledge of each task can be manipulated by a simple arithmetic on pretrained model parameters: $\boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\theta}_t$, where $\alpha_t > 0$ will add the knowledge of task t while $\alpha_t < 0$ will result in forgetting the knowledge of task t, while $|\alpha_t|$ controls the extent of learning/forgetting. Using these task vectors $\boldsymbol{\tau}_t$ with corresponding task coefficients α_t, task-specific models can be merged into a merged multi-task model, parameterized by $\boldsymbol{\theta}_{\text{merge}}$ as follows:

$$\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t. \tag{5}$$

4 MITIGATING PARAMETER INTERFERENCE VIA SHARPNESS-AWARE FINETUNING

Since a merged model is formed by simply performing linear arithmetic on task vectors, there is a high chance for interference among tasks (Ilharco et al., 2023). Such interference leads to the performance degradation on downstream tasks after merging. Some works focus on reducing interference during merging process, which is a challenging task as finetuned model parameters are fixed. On the other hand, few recent works propose to modify a finetuning process that results in task-specific models whose parameters have less interference with each other. In particular, they show that finetuning a (partially) linearized model or its linear layers only results in less interference. However, such linearization of finetuning results in the performance degradation of each task-specific model, limiting the overall performance of a merged model.

In this work, we claim that we need to achieve both (1) *less performance gap between a merged model and each finetuned model (i.e., less parameter interference)* and (2) *generalization performance of each finetuned model* on each respective dataset. As such, we aim to design a new objective function for finetuning to achieve these two objectives:

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}} \underbrace{\mathcal{L}(\boldsymbol{\theta}_{\text{merge}}(\boldsymbol{\theta});\, \mathcal{D}^{(t)}) - \mathcal{L}(\boldsymbol{\theta};\, \mathcal{D}^{(t)})}_{\text{Objective (1)}} + \underbrace{\mathcal{L}(\boldsymbol{\theta};\, \mathcal{D}^{(t)})}_{\text{Objective (2)}}, \tag{6}$$

where $\boldsymbol{\theta}_{\text{merge}}(\boldsymbol{\theta})$ is to demonstrate that $\boldsymbol{\theta}_{\text{merge}}$ changes as $\boldsymbol{\theta}$ is optimized, while considering parameters for other tasks to be fixed. While this objective function already looks similar to the SAM objective function in Equation 1, after some simplifications (deriviations are delineated in Appendix B), we get the final objective function as follows:

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} + \sum_{s \neq t} \alpha_s \boldsymbol{\tau}_s + (\alpha_t - 1)\boldsymbol{\tau};\, \mathcal{D}^{(t)}), \tag{7}$$

where $\sum_{s \neq t} \alpha_s \boldsymbol{\tau}_s + (\alpha_t - 1)\boldsymbol{\tau}$ represents the parameter offsets a model merging would introduce to the parameters of a task-specific model θ undergoing optimization on a task t. Hence, $\sum_{s \neq t} \alpha_s \boldsymbol{\tau}_s + (\alpha_t - 1)\boldsymbol{\tau}$ can be considered as perturbations that would cause parameter interference during model merging, from the perspective of each task-specific model. However, we do not assume access to other tasks, as each task-specific model is independently trained. Since other tasks are unknown, we consider $\sum_{s \neq t} \alpha_s \boldsymbol{\tau}_s + (\alpha_t - 1)\boldsymbol{\tau}$ to be random perturbations. Furthermore, because the perturbation $\sum_{s \neq t} \alpha_s \boldsymbol{\tau}_s + (\alpha_t - 1)\boldsymbol{\tau}$ depends on $\boldsymbol{\tau} = \boldsymbol{\theta} - \boldsymbol{\theta}_0$ and is thus varying during training, we use ASAM that models $\hat{\epsilon}$ as parameter-dependent perturbation (Equation 3). In other words, we use $\hat{\boldsymbol{\epsilon}}_{\text{ASAM}}$ as a surrogate of $\sum_{s \neq t} \alpha_s \boldsymbol{\tau}_s + (\alpha_t - 1)\boldsymbol{\tau}$, thereby our final objective function for finetuning aimed for model merging is Equation 2 with $\hat{\boldsymbol{\epsilon}}_{\text{ASAM}}$ from Equation 3.

From our perspective described above, we can consider parameter interference to be caused by parameter perturbations $\sum_{s \neq t} \alpha_s \boldsymbol{\tau}_s + (\alpha_t - 1)\boldsymbol{\tau}$ that would be introduced during model merging, the information of which is however not available during finetuning for each task. The perturbations will bring a model to a new location in the loss landscape, away from the found local minimum.

If the region around the local minimum is not flat enough, the new location (i.e., merged model parameters) brought by perturbations will most likely have a higher loss, resulting in a large performance gap between a merged model and a task-specific model. In other words, to minimize the interference caused by parameter perturbations, it is essential to identify flat minima. Flat minima can effectively prevent the loss from increasing after parameter perturbations (e.g., model merging). Thus, we argue that finding flat minima (or equivalently, minimizing sharpness) via sharpness-aware finetuning can greatly reduce parameter interference.

In the subsequent section, we experimentally validate our argument by showing that sharpness-aware finetuning leads to better weight disentanglement (Figure 1 and Figure 2), better cross-task linearity (Figure 3), and better joint-task loss linearity (Figure 4 and Figure 5), which are the signs of less parameter interference. Better performance by our proposed method, compared to standard SGD and other finetuning schemes specifically designed for model merging, further underlines the effectiveness of sharpness-aware finetuning in reducing parameter interference. Then, at the end of the subsequent section, we also theoretically show that the capability of SAM to reduce the dominant Hessian eigenvalues induces joint-task loss linearity (the linearity of loss on all joint tasks).

5 SAM MITIGATES PARAMETER INTERFERENCE

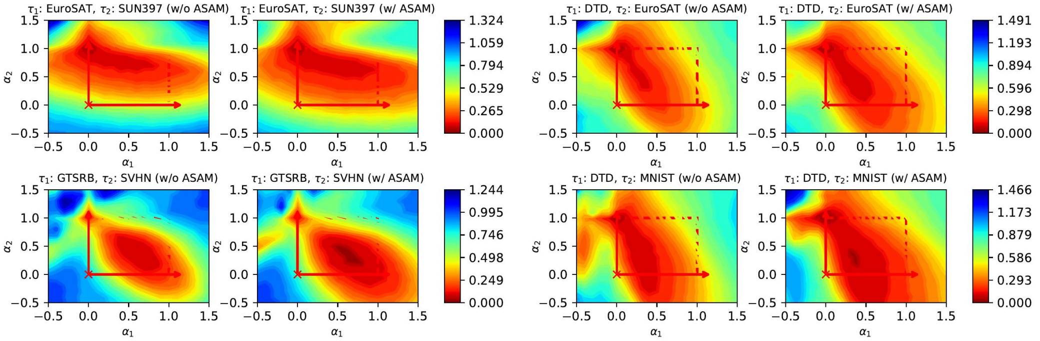

Figure 1: **Weight disentanglement visualization of two fine-tuned models across two tasks**. Each pixel in the heatmap corresponds to the disentanglement error between interpolation $\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \alpha_1\boldsymbol{\tau}_1 + \alpha_2\boldsymbol{\tau}_2$ and two fine-tuned model parameter $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 + \alpha_1\boldsymbol{\tau}_1$ and $\boldsymbol{\theta}_2 = \boldsymbol{\theta}_0 + \alpha_2\boldsymbol{\tau}_2$, evaluated on task 1 and task 2. We use CLIP ViT-B/32 on the EuroSAT-SUN397 and DTD-MNIST task pairs to generate these visualizations. The light regions indicate low-loss areas in the parameter space. The red box highlights the search space used to find the optimal task coefficient α of task arithmetic.

Ortiz-Jimenez et al. (2023) argue that for model merging via task arithmetic to be effective, weight disentanglement (a task vector for one task not affecting the outputs of task-specific model on other tasks) is a necessary condition. In this work, we show that SAM indeed achieves better weight disentanglement, in comparison to a standard objective function.

Weight Disentanglement. Weight disentanglement is met when task-specific parameter update $\boldsymbol{\tau}_t$ does not affect the output of task-specific models on input from other task datasets, imparting influence on a model only on input $\boldsymbol{x}^{(t)}$ from a given task t. Ortiz-Jimenez et al. (2023) formally expresses the localized influence of task vectors on the input space as

$$f\left(\boldsymbol{x}; \boldsymbol{\theta}_{\text{merge}}\right) = f\left(\boldsymbol{x}; \boldsymbol{\theta}_0 + \sum_{s=1}^{T} \alpha_s \boldsymbol{\tau}_s\right) \tag{8}$$

$$= f\left(\boldsymbol{x}; \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t\right) \quad \text{when} \quad \boldsymbol{x} \in X^{(t)}. \tag{9}$$

To evaluate how well weight disentanglement is satisfied, Ortiz-Jimenez et al. (2023) quantify disentanglement error as the discrepancy between the output of a merged model and t-th task-specific model on input data of t-th task. Lower disentanglement errors imply that each task contributes

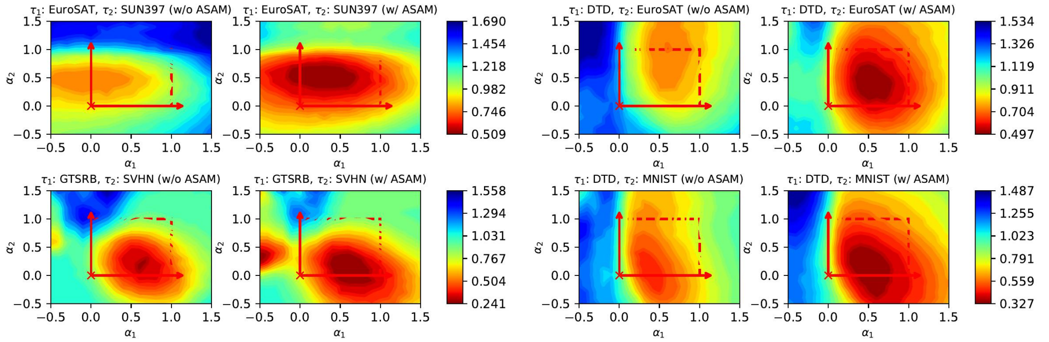

Figure 2: **Weight disentanglement visualization of merged models on the eight tasks across two tasks**. Each pixel in the heatmap corresponds to the disentanglement error between the multi-task model parameter $\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t$, merged from eight fine-tuned model parameters, and each single-task fine-tuned model parameter, evaluated on tasks 1 and 2. To visualize the landscape of the merged multi-task model on a 2D heatmap, we adjust only two task coefficients corresponding to the evaluation tasks. We also use CLIP ViT-B/32 on the four task pair as in Figure 1 to compare the difference between the interpolation model $\boldsymbol{\theta}_{\text{merge}}$ and the merged model $\boldsymbol{\theta}_{\text{merge}}$. The meaning of the light regions and the red box is the same as in Figure 1.

appropriately without adversely affecting others. Ortiz-Jimenez et al. (2023) consider merging of two tasks while computing weight disentanglement error as follows:

$$\xi(\alpha_1, \alpha_2) = \sum_{t=1}^{2} \mathbb{E}_{\boldsymbol{x} \in X^{(t)}} \left[\text{dist} \left(f(\boldsymbol{x}; \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t), f(\boldsymbol{x}; \boldsymbol{\theta}_0 + \alpha_1 \boldsymbol{\tau}_1 + \alpha_2 \boldsymbol{\tau}_2) \right) \right], \tag{10}$$

where $\xi(\alpha_1, \alpha_2)$ is the disentanglement error with respect to two given tasks and visualized in Figure 1. We further stress-test and evaluate the disentanglement error while considering merging of all task-specific models ($T = 8$ in this work), thereby evaluating how well an actual merged model achieves weight disentanglement. However, it is difficult to visualize if all T task coefficients are adjusted. In this work, for ease of visualization, we adjust task-coefficients of two tasks while fixing other task coefficients but still considering all task vectors:

$$\xi_{\text{all}}(\alpha_1, \alpha_2) = \tag{11}$$

$$\sum_{t=1}^{2} \mathbb{E}_{\boldsymbol{x} \in X^{(t)}} \left[\text{dist} \left(f(\boldsymbol{x}; \boldsymbol{\theta}_0 + \alpha_t \boldsymbol{\tau}_t), f(\boldsymbol{x}; \boldsymbol{\theta}_0 + \alpha_1 \boldsymbol{\tau}_1 + \alpha_2 \boldsymbol{\tau}_2 + \sum_{s \notin \{1,2\}} \alpha_s \boldsymbol{\tau}_s) \right) \right], \tag{12}$$

where $\xi_{\text{all}}(\alpha_1, \alpha_2)$ represents the total disentanglement error across all tasks (visualized in Figure 2, and $\text{dist}(\cdot, \cdot)$ is a distance metric measuring the divergence between the outputs of the individually fine-tuned model and the merged model. Small $\xi(\alpha_1, \alpha_2)$ or $\xi_{\text{all}}(\alpha_1, \alpha_2)$ implies that the merged model parameter $\boldsymbol{\theta}_{\text{merge}}$ accurately reflects the individual contributions of each task, signifying reduced parameter interference. Indeed, the visualizations of weight disentanglement when considering two tasks in Figure 1 and all tasks ($T = 8$) in Figure 2 demonstrate the effectiveness of SAM-applied finetuning in achieving better weight disentanglement. In particular, we note that the weight disentanglement error of the model merging with standard finetuning optimization increases significantly when considering all tasks in model merging, compared to considering two tasks. On the other hand, SAM-applied finetuning reduces the weight disentanglement even when considering all tasks, further higlighting the effectiveness of sharpness-aware finetuning in model merging.

Cross-Task Linearity. Cross-Task Linearity (CTL) (Zhou et al., 2024) is a property that ensures the linear separability of task influences on the model outputs across all layers of the network. To satisfy CTL, for every layer ℓ, the model response to a combination of task vectors should be approximately equal to the combination of the individual task responses scaled by their respective coefficients. Formally, CTL condition can be defined as:

$$f^{(\ell)}\left(\boldsymbol{x}; \lambda \boldsymbol{\theta}_s + (1 - \lambda) \boldsymbol{\theta}_t\right) \approx \lambda f^{(\ell)}(\boldsymbol{x}; \boldsymbol{\theta}_s) + (1 - \lambda) f^{(\ell)}(\boldsymbol{x}; \boldsymbol{\theta}_t), \tag{13}$$

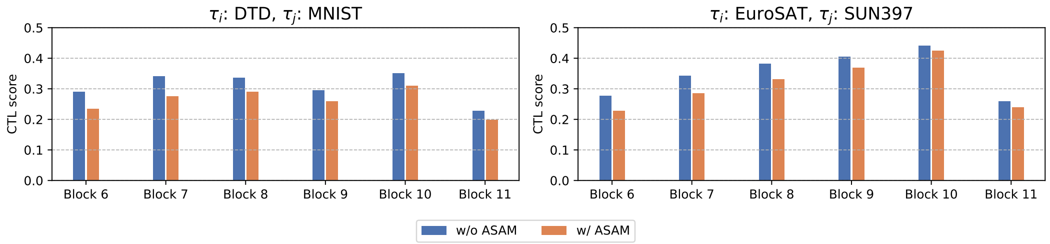

Figure 3: **Verification of CTL between merged model and fine-tuned models**. We compare $\mathbb{E}_{\mathcal{D}^{(s)} \cup \mathcal{D}^{(t)}}[1 - \cos^{(\ell)}(\boldsymbol{x}; 2\lambda\boldsymbol{\tau}_s, 2\lambda\boldsymbol{\tau}_t)]$ of each fine-tuning methods. The values for the last six blocks are evaluated on the two task pairs DTD-MNIST and EuroSAT-SUN397. We set the scaling term λ to 0.3.

where $\lambda \in \mathbb{R}$ is a scaling term; $\boldsymbol{x} \in X^{(s)} \cup X^{(t)}$; $\boldsymbol{\theta}_s, \boldsymbol{\theta}_t$ are parameters of task-specific models finetuned on task s and t respectively; and $f^{(\ell)}(\boldsymbol{x}; \boldsymbol{\theta})$ represents the response (or a feature) of ℓ-th layer of a network f for the given input $\boldsymbol{x}$.

This linearity at each layer implies that the influence of one task on another is minimal, thereby facilitating effective weight disentanglement. Zhou et al. (2024) demonstrate that satisfying CTL condition leads to reducing the disentanglement error $\xi(\alpha)$.

To evaluate whether CTL is satisfied, the following cosine similarity metric is used:

$$\begin{aligned}&\cos^{(\ell)}(\boldsymbol{x}; 2\lambda\boldsymbol{\tau}_s, 2\lambda\boldsymbol{\tau}_t)\\ &= \cos\left[f^{(\ell)}(\boldsymbol{x}; \boldsymbol{\theta}_0 + \lambda(\boldsymbol{\tau}_s + \boldsymbol{\tau}_t)), \frac{1}{2}f^{(\ell)}(\boldsymbol{x}; \boldsymbol{\theta}_0 + 2\lambda\boldsymbol{\tau}_s) + \frac{1}{2}f^{(\ell)}(\boldsymbol{x}; \boldsymbol{\theta}_0 + 2\lambda\boldsymbol{\tau}_t)\right],\end{aligned} \tag{14}$$

where $\boldsymbol{x} \in X^{(s)} \cup X^{(t)}$.

The metric measures the cosine similarity between the model output when trained on the combined task vectors and the averaged outputs of models trained on individual tasks. Following the settings in (Zhou et al., 2024), we use the metric $\mathbb{E}_{\mathcal{D}}[1 - \cos^{(\ell)}(\boldsymbol{x}; 2\lambda\boldsymbol{\tau}_s, 2\lambda\boldsymbol{\tau}_t)]$ to evaluate how well CTL is satisfied, where smaller values of $\mathbb{E}_{\mathcal{D}}[1 - \cos^{(\ell)}(\boldsymbol{x}; 2\lambda\boldsymbol{\tau}_s, 2\lambda\boldsymbol{\tau}_t)]$ indicate stronger CTL. Since satisfying CTL leads to better weight disentanglement, smaller values of $\mathbb{E}_{\mathcal{D}}[1 - \cos^{(\ell)}(\boldsymbol{x}; 2\lambda\boldsymbol{\tau}_s, 2\lambda\boldsymbol{\tau}_t)]$ should result in lower disentanglement error $\xi(\alpha_1, \alpha_2)$, as noted by Zhou et al. (2024).

Figure 3 shows that SAM-applied finetuning reduces $\mathbb{E}_{\mathcal{D}}[1 - \cos^{(\ell)}(\boldsymbol{x}; 2\lambda\boldsymbol{\tau}_s, 2\lambda\boldsymbol{\tau}_t)]$ in comparison to standard optimization, demonstrating that SAM results in not just better weight disentanglement, but also better cross-task linearity.

Joint-Task Loss Landscape. We empirically demonstrate that SAM-applied finetuning reduces parameter interference by finding flatter minima across the joint tasks. Figure 4 shows the joint-task loss landscape visualizations for two fine-tuned models trained on corresponding two tasks. We observe that SAM-applied finetuning allows models to reach flatter minima across the joint tasks compared to SGD, particularly around the boundaries of the task coefficients search space in task arithmetic. SAM-applied finetuning increases the likelihood of finding a merged model connected to each fine-tuned model along a low-loss path, which indicates a smaller performance gap between the merged model and the individual fine-tuned models. Consequently, SAM makes it easier to find a merged model with reduced parameter interference compared to SGD.

Yet, the capability of interpolation between pre-trained weights and two task vectors may not be the same as that of a multi-task model merged from more than two fine-tuned task-specific models due to interference between the parameters of different models (Yadav et al., 2023). Therefore, we also visualize the loss landscape of the multi-task model $\boldsymbol{\theta}_{\text{merge}}$ built by merging all 8 tasks, denoted as $\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \sum_{t=1}^{8} \alpha_t \boldsymbol{\tau}_t$. To represent the loss of the every-task-merged model on a 2D heatmap, we vary only the two task coefficient of $\boldsymbol{\theta}_{\text{merge}}$ corresponding to the tasks being evaluated.

Figure 5 shows the loss landscape of the multi-task model built by the eight fine-tuned models. Compared to Figure 4, the minima in the landscape shrink in every case as the number of tasks to be merged increases. However, while the minima found by SGD shrink significantly, the minima

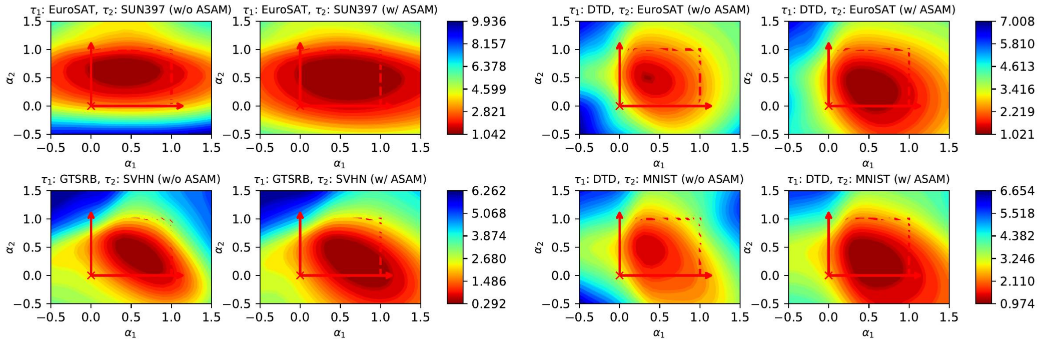

Figure 4: **Joint-task loss landscape visualization of two fine-tuned models across two tasks.** Each pixel in the heatmap corresponds to the loss values $\mathcal{L}(\boldsymbol{\theta}_{\text{merge}}; \mathcal{D}^{(1)}) + \mathcal{L}(\boldsymbol{\theta}_{\text{merge}}; \mathcal{D}^{(2)})$ of the interpolation $\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \alpha_1 \boldsymbol{\tau}_1 + \alpha_2 \boldsymbol{\tau}_2$ between pre-trained model $\boldsymbol{\theta}_0$ and two task vectors $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$, evaluated on task 1 and task 2. The setting of the model, task pair, light regions, and red box is the same as in Figure 1. We use CLIP ViT-B/32 on the EuroSAT-SUN397 and DTD-MNIST task pairs to generate these visualizations. The light regions indicate low-loss areas in the parameter space. The red box highlights the search space used to find the optimal task coefficient α of task arithmetic.

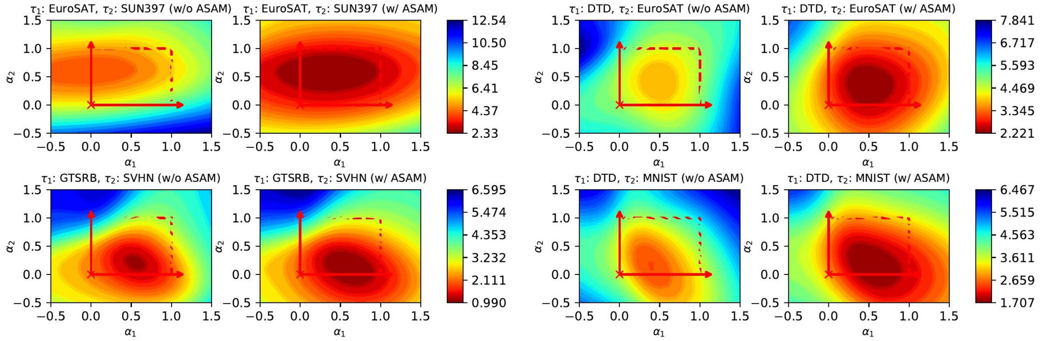

Figure 5: **Joint-task loss landscape visualization of merged models on the eight tasks across two tasks.** Each pixel in the heatmap corresponds to the loss values $\mathcal{L}(\boldsymbol{\theta}_{\text{merge}}; \mathcal{D}^{(1)}) + \mathcal{L}(\boldsymbol{\theta}_{\text{merge}}; \mathcal{D}^{(2)})$ of the multi-task model $\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \sum_{t=1}^{8} \alpha_t \boldsymbol{\tau}_t$ merged by eight fine-tuned models, evaluated on tasks 1 and 2. We adjust only the two task coefficients corresponding to the evaluation tasks to visualize the weight disentanglement on a 2D map, as in Figure 2. The setting of the model, task pair, light regions, and red box is the same as in Figure 4. We use CLIP ViT-B/32 on the four task pair as in Figure 4 to compare the difference between the interpolation model $\boldsymbol{\theta}_{\text{merge}}$ and the merged model $\boldsymbol{\theta}_{\text{merge}}$.

found by our method are less affected by this shrinkage in all cases. This suggests that our method maintains the ability to reduce parameter interference and preserve the performance of the merged model, even as more tasks are merged.

Theoretical Results. Here, we theoretically demonstrate that SAM leads to joint-task loss linearity:

Theorem 1 (SAM Induces Joint-Task Loss Linearity (proof in Appendix D.2)**.** Given parameters $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_t$, let δ be defined as the difference between the interpolated Joint-Task Loss and the convex combination of individual losses:

$$\delta = \mathcal{L}_{JTL}(\alpha \boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t; \mathcal{D}) - \alpha \mathcal{L}_{JTL}(\boldsymbol{\theta}_s; \mathcal{D}) - (1-\alpha)\mathcal{L}_{JTL}(\boldsymbol{\theta}_t; \mathcal{D}). \tag{15}$$

Then, it holds that:

$$|\delta| \leq \frac{1}{2}\alpha(1-\alpha)(\lambda_{\max}(\boldsymbol{\theta}_s; \mathcal{D}_s) + \lambda_{\max}(\boldsymbol{\theta}_t; \mathcal{D}_t))\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|^2 + \epsilon. \tag{16}$$

By Property 1 in Appendix D, SAM reduces the dominant Hessian eigenvalues $\lambda_{\max}(\boldsymbol{\theta}_s; \mathcal{D}_s)$ and $\lambda_{\max}(\boldsymbol{\theta}_t; \mathcal{D}_t)$, thereby decreasing the deviation δ. This reduction makes the approximation in Equation 18 closer, thereby inducing Joint-Task Loss Linearity.

Table 1: **Multi-task performance across different finetuning methods**. We report the average absolute and normalized accuracies for three finetuning baselines: SGD, FTTS, and FTLO. Results are shown for three finetuning methods, grouped by whether ASAM is applied. ViT-B/32 is used as the image encoder of CLIP, with task arithmetic as the model merging method in every case across eight tasks.

Finetuning method (→)	SGD		FTTS		FTLO	
	Abs.	Norm.	Abs.	Norm.	Abs.	Norm.
w/o ASAM	68.23	75.47	78.35	86.83	75.93	85.74
w/ ASAM (Ours)	**69.45**	**76.32**	**79.38**	**87.72**	**77.49**	**88.77**

Theorem 1 establishes a direct connection between the ability of SAM to reduce the dominant Hessian eigenvalues and the induction of Joint-Task Loss Linearity. Through Property 1 and Theorem 1, we can observe that the lower dominant Hessian eigenvalues for parameters $\boldsymbol{\theta}$ along the line segment $\overline{\boldsymbol{\theta}_0\boldsymbol{\theta}_t}$ between the pretrained parameter $\boldsymbol{\theta}_0$ and the finetuned parameter $\boldsymbol{\theta}_t$ ensure the linearity of the interpolated Joint-Task Loss. The merged model parameter $\boldsymbol{\theta}_{\text{merge}}$ via such interpolation maintains a stable and low Joint-Task Loss due to this linearity. Consequently, since parameter interference leads to an increase in Joint-Task Loss, the linearity induced by SAM effectively suppresses parameter interference. Thus, SAM induces Joint-Task Loss Linearity, which is intrinsically connected to the reduction of parameter interference, facilitating more stable and effective model merging.

6 EXPERIMENTS

In this section, following the settings of previous works Ortiz-Jimenez et al. (2023), we conduct experiments on diverse vision tasks to demonstrate the effectiveness of SAM-applied finetuning in improving the overall performance of a merged model. We compare against three finetuning baselines: SGD, linearized finetuning in the tangent space (FTTS) (Ortiz-Jimenez et al., 2023), and finetuning linear layers only (FTLO) (Jin et al., 2024). We also validate the effectiveness, applicability, and generalizability of SAM-applied finetuning with three different model merging methods: weighted average, task arithmetic (Ilharco et al., 2023), and TIES (Yadav et al., 2023) across two backbones: ViT-B/32 and ViT-B/16 (Dosovitskiy et al., 2021). Moreover, we empirically show that SAM reduces the parameter interference through the lens of joint-task loss landscape and weight disentanglement.

6.1 TRAINING SETUP

Following the same training protocol outlined in Ilharco et al. (2022), we finetune three CLIP (Radford et al., 2021) models: (a) ViT-B/32, (b) ViT-B/16, (c) ViT-L/14. Our experiments are conducted across eight diverse datasets: (1) Cars (Krause et al., 2013), (2) DTD (Cimpoi et al., 2014), (3) EuroSAT (Helber et al., 2019), (4) GTSRB (Stallkamp et al., 2011), (5) MNIST (Deng, 2012), (6) RESISC45 (Cheng et al., 2017), (7) SUN397 (Xiao et al., 2016), (8) SVHN (Netzer et al., 2011). All finetuning processes begin from the same CLIP pretrained checkpoint obtained from the `open_clip` (Radford et al., 2021) repository. We finetune each model for 8000 iterations with a batch size of 128 and a learning rate of 10^{-5} for all backbones and all finetuning methods. The learning rate schedule follows a cosine annealing approach with 500 warm-up steps, and optimization is performed using the AdamW (Loshchilov & Hutter, 2019). Consistent with Ilharco et al. (2022), we freeze the weights of the classification layer derived from encoding a standard set of zero-shot template prompts for each dataset. This strategy ensures that no additional learnable parameters are introduced during finetuning and does not compromise model accuracy. For more experimental details, please refer to Appendix A.

6.2 MAIN RESULTS

We evaluate the effectiveness of SAM-applied finetuning in closing the performance gap between a merged model and each task-specific models, in comparison to other three finetuning baselines.

Table 1 shows that SAM-applied finetuning achieves the higher absolute and normalized accuracies in every case, compared to other finetuning methods. Normalized accuracy is defined as the abso-

Table 2: **Multi-task performance across different model merging methods and image encoder models**. We report the average absolute and normalized accuracies for three model merging methods: weighted average, task arithmetic, and TIES merging. We also compare the performance of two different models used as the image encoder of CLIP: ViT-B/32 and ViT-B/16. All cases are finetuned using SGD and evaluated across eight tasks.

Merging method (→)	Weighted average		Task arithmetic		TIES merging	
	Abs.	Norm.	Abs.	Norm.	Abs.	Norm.
	ViT-B/32					
w/o ASAM	65.72	72.91	68.23	75.47	74.57	82.29
w/ ASAM (Ours)	**66.76**	**73.62**	**69.45**	**76.32**	**75.45**	**82.86**
	ViT-B/16					
w/o ASAM	71.58	77.37	73.40	79.31	77.94	84.04
w/ ASAM (Ours)	**71.84**	**77.53**	**76.77**	**82.50**	**80.14**	**86.23**

lute accuracy divided by the corresponding accuracy of the finetuned task-specific model, evaluating the performance gap between a merged model and task-specific models. These results suggest that SAM-applied finetuning not only improves performance in downstream tasks but also narrows the performance gap between the merged model and finetuned models, improving the overall performance. Moreover, SAM-applied finetuning synergizes not only standard SGD but also with other finetuning methods (FTTS (Ortiz-Jimenez et al., 2023) and FTLO (Jin et al., 2024)), enhancing performance in multi-task settings during model merging, demonstrating its generalizability and applicability. In particular, FTTS (Ortiz-Jimenez et al., 2023) and FTLO (Jin et al., 2024) demonstrate better multi-task performance compared to SGD, as these finetuning methods reduce interference between tasks by encouraging weight disentanglement (Malladi et al., 2023; Ortiz-Jimenez et al., 2023). Thus, the performance improvement brought by SAM-applied finetuning on top of these finetuning methods demonstrates the orthogonality of our proposal.

In Table 2, we display the multi-task performance across different model merging methods and image encoder models used in CLIP. Our method outperforms in every combination of model merging methods and image encoder models. Notably, our method achieves better performance in both weighted average and task arithmetic. Weighted average is a specific case of task arithmetic, where $\alpha_t = \frac{1}{T}$, while task arithmetic searches for the optimal task coefficient within a given search space. This suggests that our method finds flatter minima that covers the task coefficients search space in task arithmetic compared to SGD. Moreover, SAM-applied finetuning also performs better in the case of TIES merging, indicating that interference mitigation by our method complements the mitigation achieved by TIES merging. As a result, applying our method to TIES merging yields the best performance among all combinations.

7 CONCLUSION

In this work, we draw connections between two research fields of machine learning: sharpness-aware minimization and multi-task model merging. Particularly, the connections are drawn from the formulation of two objectives of model merging: (1) reducing parameter interference between task-specific models and (2) achieving better generalization of each task-specific model. Upon observation, we propose to apply SAM to finetuning process to improve the overall performance of a merged model. Experimental results demonstrate that SAM-applied finetuning indeed results in less performance interference and better performance of a merged model, even when applied together with other merging and finetuning methods designed for model merging. Motivated by the effectiveness and applicability of our proposal, we hope that this work encourages further research on investigating the relationship between sharpness-aware optimization and model merging, opening a new research avenue.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report, 2023. `https://arxiv.org/pdf/2303.08774`.

Atish Agarwala and Yann Dauphin. Sam operates far from home: eigenvalue regularization as a dynamical phenomenon. In *ICML*, 2023.

Rich Caruana. Multitask learning. *Machine Learning*, 1997.

Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. In *NeurIPS*, 2021.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 2017.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

Michael Crawshaw. Multi-task learning with deep neural networks: A survey, 2020. `https://arxiv.org/abs/2009.09796`.

Nico Daheim, Thomas Möllenhoff, Edoardo Maria Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *ICLR*, 2024.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 2012.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 2023.

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 *ICLR*, 2021.

Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. In *NeurIPS*, 2022.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *ICLR*, 2021.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

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 *NeruIPS*, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon W. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018.

Ruochen Jin, Bojian Hou, Jiancong Xiao, Weijie Su, and Li Shen. Fine-tuning linear layers only is a simple yet effective way for task arithmetic, 2024. `https://arxiv.org/abs/2407.07089`.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *ICLR*, 2023.

Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J. Kusner. When do flat minima optimizers work? In *NeurIPS*, 2022.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop*, 2013.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *ICLR*, 2021.

Tao Li, Weihao Yan, Zehao Lei, Yingwen Wu, Kun Fang, Ming Yang, and Xiaolin Huang. Efficient generalization improvement guided by random weight perturbation, 2022. `https://arxiv.org/pdf/2211.11489`.

Tao Li, Pan Zhou, Zhengbao He, Xinwen Cheng, and Xiaolin Huang. Friendly sharpness-aware minimization. In *CVPR*, 2024.

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. Deep model fusion: A survey, 2023. `https://arxiv.org/abs/2309.15698`.

Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *CVPR*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *ICML*, 2023.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *NeurIPS*, 2022.

Peng Mi, Li Shen, Tianhe Ren, Yiyi Zhou, Xiaoshuai Sun, Rongrong Ji, and Dacheng Tao. Make sharpness-aware minimization stronger: A sparsified perturbation approach. In *NeurIPS*, 2022.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y. Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop*, 2011.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *NeurIPS*, 2023.

Hoang Phan, Lam Tran, Quyen Tran, Ngoc N. Tran, Tuan Truong, Nhat Ho, Dinh Phung, and Trung Le. Improving multi-task learning via seeking task-based flat regions, 2022. `https://arxiv.org/pdf/2211.13723`.

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 *ICML*, 2021.

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, 2019. `https://arxiv.org/pdf/1910.10683`.

Khaled Saab, Tao Tu, Wei-Hung Weng, Ryutaro Tanno, David Stutz, Ellery Wulczyn, Fan Zhang, Tim Strother, Chunjong Park, Elahe Vedadi, et al. Capabilities of gemini models in medicine, 2024. `https://arxiv.org/pdf/2404.18416`.

Ken Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH*, 1985.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *IJCNN*, 2011.

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. Parameter efficient multi-task model fusion with partial linearization. In *ICLR*, 2024.

Joachim Utans. Weight averaging for neural networks and local resampling schemes. In *AAAI Workshop*, 1996.

Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Denxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeruIPS*, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.

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

Pengfei Wang, Zhaoxiang Zhang, Zhen Lei, and Lei Zhang. Sharpness-aware gradient matching for domain generalization. In *CVPR*, 2023.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *ICLR*, 2022.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya 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 *ICML*, 2022a.

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 *CVPR*, 2022b.

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*, 2016.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In *NeurIPS*, 2023.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities, 2024a. `https://arxiv.org/abs/2408.07666`.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *ICLR*, 2024b.

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 *ICML*, 2024.

Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning. In *ICML*, 2022.

Zhanpeng Zhou, Zijun Chen, Yilan Chen, Bo Zhang, and Junchi Yan. On the emergence of cross-task linearity in the pretraining-finetuning paradigm. In *ICML*, 2024.

A EXPERIMENTAL DETAILS

A.1 FINETUNING BASELINES

We compare the following three finetuning baselines with and without SAM:

(1) **SGD**: This refers to standard finetuning that uses only an optimizer such as AdamW (Loshchilov & Hutter, 2019).

(2) **F**ine**T**uning in the **T**angent **S**pace (**FTTS**) (Ortiz-Jimenez et al., 2023): This finetunes the model in the tangent space at its pretrained initialization. It achieves this by linearizing the model using a first-order Taylor expansion $f_{\text{lin}}(\boldsymbol{\theta}; \mathcal{D}) = f(\boldsymbol{\theta}_0; \mathcal{D}) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla f(\boldsymbol{\theta}_0; \mathcal{D})$, where $\boldsymbol{\theta}_0$ represents the parameters of the pretrained model and $\mathcal{D}$ is the training dataset. The method freezes $\boldsymbol{\theta}_0$ and updates only $\boldsymbol{\theta}$.

(3) **F**ine**T**uning **L**inear Layers **O**nly (**FTLO**) (Jin et al., 2024): This exclusively finetunes the linear layers within the attention module. Therefore, this method can only be applied to model architectures that include attention modules such as Transformer (Vaswani et al., 2017).

We utilize ASAM (Kwon et al., 2021) as a default SAM method in every experiments, since it finds minima adaptively by considering correlation between generalization gap and sharpness. We set the ρ value of ASAM to 0.5, following the default setup outlined in ASAM, along with all other ASAM hyperparameters.

A.2 MERGING METHODS

We merge the models that achieve the best performance for each corresponding task. These best models are selected based on their performance on a validation set split, which is split from the training set at a 0.1 ratio, as specified in Ilharco et al. (2023).

We use the following model merging methods as baselines:

(1) **Weighted Average**: This merges finetuned models by averaging their parameters element-wise, denoted as $\boldsymbol{\theta}_{\text{merge}} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\theta}_t$, where $\boldsymbol{\theta}_t$ represents the finetuned parameters for each corresponding downstream task, T is the number of downstream tasks being merged.

(2) **Task arithmetic** (Ilharco et al., 2023): This method calculates task vectors $\boldsymbol{\tau}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_0$ for each downstream task t, where $\boldsymbol{\theta}_t$ represents the finetuned parameters for task t and $\boldsymbol{\theta}_0$ represents the pretrained parameters. A linear combination of these task vectors is then added to the pretrained parameters, denoted as $\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \sum_{t=1}^{T} \alpha_t \boldsymbol{\tau}_t$, where α_t is a task coefficient that scales the corresponding task vector. This method generalizes the weighted average when $\alpha_t = \frac{1}{T}$ for $t = 1, 2, \ldots, T$.

Since the search space for α_t becomes too large as the number of tasks increases, we set the task coefficients to be the same for all tasks and search for the optimal coefficient within the range $[0.1, 0.3, 0.5, 0.7, 0.9, 1.0]$ using the validation set of each task.

(3) **TIES merging** (Yadav et al., 2023): This method mitigates parameter interference before merging models. First, it trims parameters changed that change minimally during fine-tuning, as these small changes in each model can become more pronounced after element-wise parameter merging. Second, it resolves parameter interference due to sign conflicts by determining the sign of each parameter through a majority election before merging the models.

We apply TIES merging to task arithmetic. To find the optimal merged model, we search for the task coefficients in task arithmetic within the range $[0.1, 0.3, 0.5, 0.7, 0.9, 1.0]$ and the percentile of parameters to be pruned to zero within $[0.7, 0.8, 0.9]$, using the validation set for each task.

A.3 VISUALIZATION SETUP

We produce the joint-task loss landscape and disentanglement error under two distinct settings: (1) merging two finetuned models across two tasks, and (2) merging models finetuned on eight tasks across two tasks.

1. Two Finetuned Models Across Two Tasks: In the first setting, we define the merged model as:
$$\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \alpha_1\boldsymbol{\tau}_1 + \alpha_2\boldsymbol{\tau}_2,$$
where $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ represent the parameters of models finetuned on tasks 1 and 2, respectively.
2. Merged Models on Eight Tasks Across Two Tasks: In the second setting, we define the merged model as:
$$\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \alpha_1\boldsymbol{\tau}_1 + \alpha_2\boldsymbol{\tau}_2 + \sum_{k\notin\{1,2\}} \alpha\boldsymbol{\tau}_k,$$
with $\alpha = 0.3$, where $\boldsymbol{\tau}_k$ denotes the parameters of the additional six tasks.

For both settings, we use (α_1, α_2) pairs spanning from -0.5 to 1.5 with 21 evenly spaced points along each axis, resulting in a 21×21 grid.

Joint-task Loss Landscape (Figure 4) We produce the joint-task loss landscape by computing the combined loss:
$$\mathcal{L}(\boldsymbol{\theta}_{\text{merge}}; \mathcal{D}^{(1)}) + \mathcal{L}(\boldsymbol{\theta}_{\text{merge}}; \mathcal{D}^{(2)}),$$
for each (α_1, α_2) pair on the defined grid.

Disentanglement Error We evaluate the disentanglement error $\xi(\alpha_1, \alpha_2)$ for both settings to quantify the discrepancy between the desired and actual influences of individual tasks on the merged model's outputs. The disentanglement error is defined as:
$$\xi(\alpha_1, \alpha_2) = \sum_{t=1}^{T} \mathbb{E}_{x\sim\mu_i}\left[\text{dist}\left(f(x; \boldsymbol{\theta}_0 + \alpha_t\boldsymbol{\tau}_t), f(x; \boldsymbol{\theta}_{\text{merge}})\right)\right].$$

For each (α_1, α_2) task coefficient pair on the grid, we compute $\xi(\alpha_1, \alpha_2)$ and visualize the error values using contour plots to identify regions where disentanglement is effective.

Since task coefficients are real numbers, we utilize contour plots to effectively visualize the variations in loss landscape and disentanglement error across the continuous (α_1, α_2) parameter space.

B DERIVATION OF EQUATION 7

We start with simplifying Equation 6, which is the objective function that incorporates the goals of model merging:
$$\begin{aligned}\boldsymbol{\theta}_t &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\text{merge}}(\boldsymbol{\theta});\, \mathcal{D}^{(t)}) - \mathcal{L}(\boldsymbol{\theta};\, \mathcal{D}^{(t)}) + \mathcal{L}(\boldsymbol{\theta};\, \mathcal{D}^{(t)}) \\ &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{\text{merge}}(\boldsymbol{\theta});\, \mathcal{D}^{(t)}).\end{aligned}$$

Here, we consider task coefficients $\{\alpha_s\}$ and other task vectors $\{\boldsymbol{\tau}_s\}_{s\neq t}$ to be fixed. Since Then, instead of $\boldsymbol{\theta}_{\text{merge}} = \boldsymbol{\theta}_0 + \sum_{s=1}^{T} \alpha_s\boldsymbol{\tau}_s$ in Equation 5, we express $\boldsymbol{\theta}_{\text{merge}}(\boldsymbol{\theta})$ as $\boldsymbol{\theta}_0 + \sum_{s\neq t} \alpha_s\boldsymbol{\tau}_s + \alpha_t\boldsymbol{\tau}$, where $\boldsymbol{\tau} = \boldsymbol{\theta} - \boldsymbol{\theta}_0$, since $\boldsymbol{\theta}_t$ has not been found yet during the process of optimizing $\boldsymbol{\theta}$ for task t. We now have
$$\begin{aligned}\boldsymbol{\theta}_t &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_0 + \sum_{s\neq t} \alpha_s\boldsymbol{\tau}_s + \alpha_t\boldsymbol{\tau};\, \mathcal{D}^{(t)}) \\ &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_0 + \boldsymbol{\tau} - \boldsymbol{\tau} + \sum_{s\neq t} \alpha_s\boldsymbol{\tau}_s + \alpha_t\boldsymbol{\tau};\, \mathcal{D}^{(t)}) \\ &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} - \boldsymbol{\tau} + \sum_{s\neq t} \alpha_s\boldsymbol{\tau}_s + \alpha_t\boldsymbol{\tau};\, \mathcal{D}^{(t)}) \quad \because \boldsymbol{\theta} = \boldsymbol{\theta}_0 + \boldsymbol{\tau} \\ &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} + \sum_{s\neq t} \alpha_s\boldsymbol{\tau}_s + (\alpha_t - 1)\boldsymbol{\tau};\, \mathcal{D}^{(t)}).\end{aligned}$$

C ADDITIONAL RESULTS

C.1 EFFECT OF THE NUMBER OF STEPS DURING FINETUNING

SAM has the capability to increase training epochs while still improving accuracy (Foret et al., 2021). Therefore, we conduct an ablation study to evaluate whether SAM can enhance the performance of a single downstream task by doubling the number of training steps. In this study, we employ ASAM. As shown in Table 3, it appears that SGD converges, as performance plateaus after 2000 steps. In contrast, SAM continues to improve performance consistently up to 8000 steps.

Table 3: Average accuracies of finetuned ViT-B/32 over steps across the eight tasks.

Finetuning Steps	2000	4000	8000
SGD	90.37	90.21	90.48
ASAM	**90.50**	**90.84**	**91.03**

C.2 FINETUNING PERFORMANCE OF SAM VARIANTS

To empirically justify our choice of SAM variant for our main experiments, we evaluate various SAM variants on the same datasets (i.e., eight vision tasks) as our main experiments. In particular, we investigate how the performance of a merged model changes when applying SAM (Foret et al., 2021), ASAM (Kwon et al., 2021), Friendly SAM (Li et al., 2024), WA-SAM (Kaddour et al., 2022), SAGM (Wang et al., 2023), PGN (Zhao et al., 2022), SSAM-F (Mi et al., 2022), and SSAM-D (Mi et al., 2022), as shown in Table 4. The results demonstrate that ASAM brings better performance improvement, compared to SAM and other SAM variants. As a result, ASAM shows the best single-task performance among other variants. Therefore, we use ASAM as a default SAM variant in all experiments.

Table 4: Average accuracy of finetuned ViT-B/32 over steps across various SAM variants and finetuning methods.

SAM variants	Accuracy
SGD	90.45
SAM	90.16
ASAM	**91.29**
Friendly SAM	90.29
WA-SAM	91.06
SAGM	90.96
PGN	90.90
SSAM-F	90.80
SSAM-D	90.60

C.3 CROSS-TASK LINEARITY

We provide additional results that demonstrate that SAM-applied finetuning satisfies cross-task linearity on other pairs of datasets in Figure 6. We utilize ViT-B/32 as the image encoder to visualize this figure, just the same as Figure 3. The results show that our method achieves lower CTL scores across all layers for various task combinations. This suggests that our approach better satisfies CTL for a broader range of data, implying improved weight disentanglement and task arithmetic properties. Consequently, it can be concluded that our method reduces parameter interference and minimizes the performance gap.

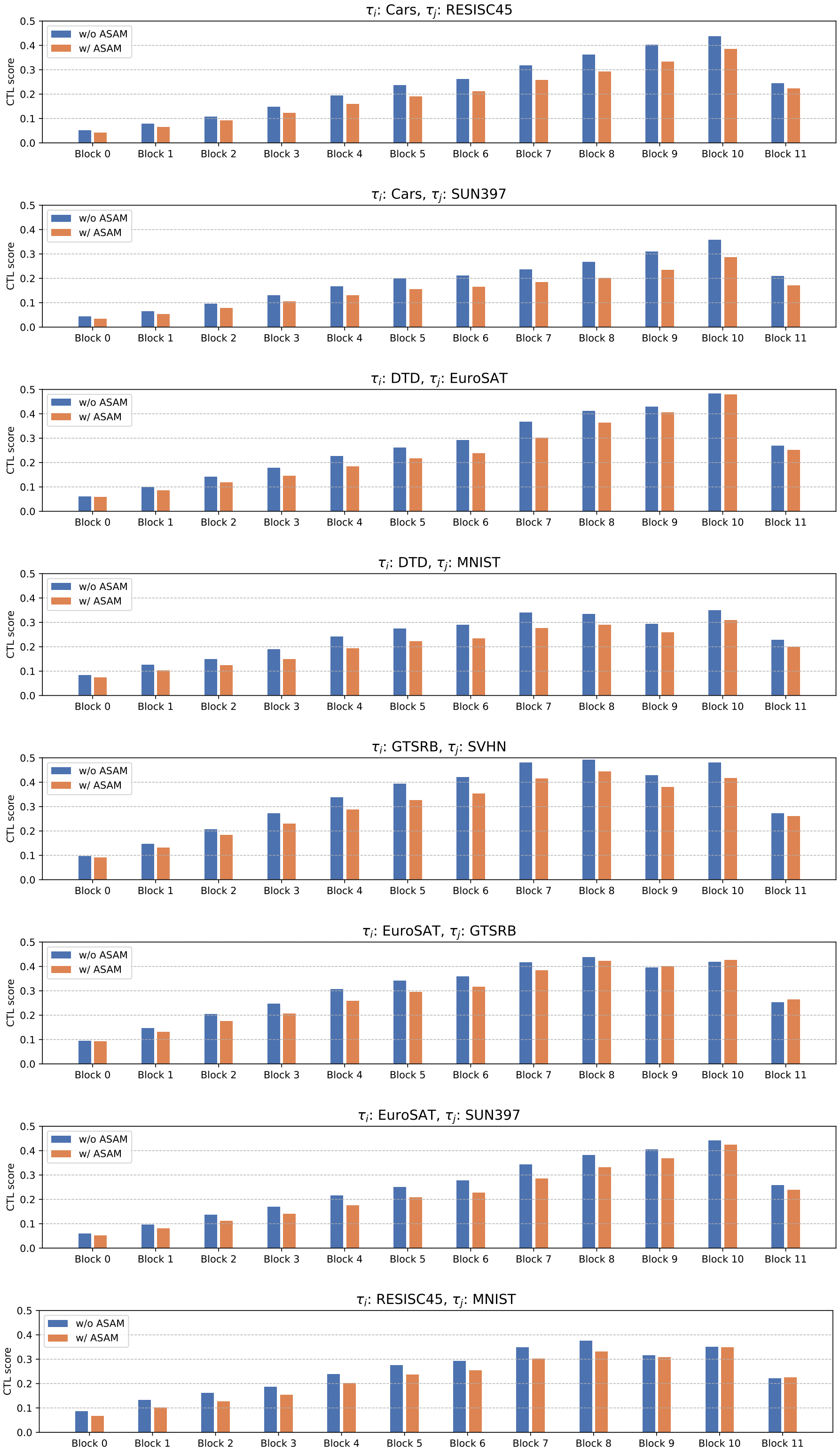

Figure 6: **Verification of all blocks CTL between merged model and fine-tuned models**. We compare $\mathbb{E}_{\mathcal{D}^{(s)}\cup\mathcal{D}^{(t)}}[1-\cos^{(\ell)}(\boldsymbol{x};2\lambda\boldsymbol{\tau}_s,2\lambda\boldsymbol{\tau}_t)]$ We set the scaling term λ to 0.3.

Table 5: **Multi-task performance when merging a CLIP image encoder on every 8 tasks**. We report the average absolute and normalized accuracies for different 5 model merging methods. Results are shown for the 6 finetuning methods, categorized by whether ASAM is applied.

Merging methods (→)	Weighted average		Task arithmetic		TIES merging	
Finetuning baselines (↓)	Abs.	Norm.	Abs.	Norm.	Abs.	Norm.
	ViT-B/32					
SGD w/o ASAM	65.72	72.91	68.23	75.47	74.57	82.29
SGD w/ ASAM (Ours)	**66.76**	**73.62**	**69.45**	**76.32**	**75.45**	**82.86**
FTTS w/o ASAM	72.47	82.04	78.35	86.83	**76.89**	**86.84**
FTTS w/ ASAM (Ours)	**75.10**	**86.10**	**79.38**	**87.72**	73.77	84.46
FTLO w/o ASAM	**65.96**	**73.83**	75.93	85.74	**77.39**	**85.89**
FTLO w/ ASAM (Ours)	65.34	72.78	**77.49**	**88.77**	76.30	84.62
	ViT-B/16					
SGD w/o ASAM	71.58	77.37	73.40	79.31	77.94	84.04
SGD w/ ASAM (Ours)	**71.84**	**77.53**	**76.77**	**82.50**	**80.14**	**86.23**
FTTS w/o ASAM	77.20	84.87	79.37	87.33	**81.09**	**89.05**
FTTS w/ ASAM (Ours)	**78.09**	**86.45**	**79.78**	**88.26**	78.41	86.72
FTLO w/o ASAM	70.97	**77.11**	80.00	86.55	78.25	84.91
FTLO w/ ASAM (Ours)	**71.03**	76.78	**82.59**	**89.11**	**79.49**	**85.92**

C.4 LOSS BETWEEN A MERGED MODEL AND FINETUNED MODELS

To demonstrate that SAM-applied finetuning indeed reduces the loss sharpness and the performance gap between a merged model and finetuned models, we visualize loss changes as we traverse along a linear path between a merged model and a finetuned model on a given task in Figure 7. SAM-applied finetuning indeed results in reduced loss barrier, leading to less performance gap as exhibited in less weight disentanglement error, better cross-task linearity, and better overall performance in our main paper.

C.5 ADDITIONAL RESULTS OF FINTUNING BASELINES AND MODEL MERGING METHODS

Following Section 6.2, we conduct experiments on all combinations of finetuning baselines (SGD, FTTS, FTLO) and model merging methods (weighted average, task arithmetic, TIES), as summarized in Table 5. In the case of weighted average, our method leads to performance improvements in most cases, and for task arithmetic, it achieves performance improvements in all cases. For weighted average, our method improves performance in most cases, while task arithmetic consistently yields performance improvements across all cases. In contrast, TIES shows performance improvements in only half of the cases. Upon closer examination, when linear finetuning methods such as FTTS and FTLO — which regularize the model output to satisfy linearity — are used without ASAM, TIES generally outperforms task arithmetic. However, with ASAM applied, TIES consistently performs worse than task arithmetic.

This seems that since the combination of linear finetuning and ASAM has already enhanced weight disentanglement and reduced parameter interference, parameter trimming via TIES may rather remove critical parameters not noisy parameters, leading to performance degradation. Specifically, with SGD, the combination of TIES and ASAM delivers the best performance. Conversely, with FTTS and FTLO, task arithmetic paired with ASAM achieves superior results. In some instances, TIES combined with ASAM performs similarly to weighted average. Thus, for linear finetuning methods like FTTS and FTLO, combining ASAM with TIES can negatively impact performance. Additional analysis of this behavior is reserved for future work.

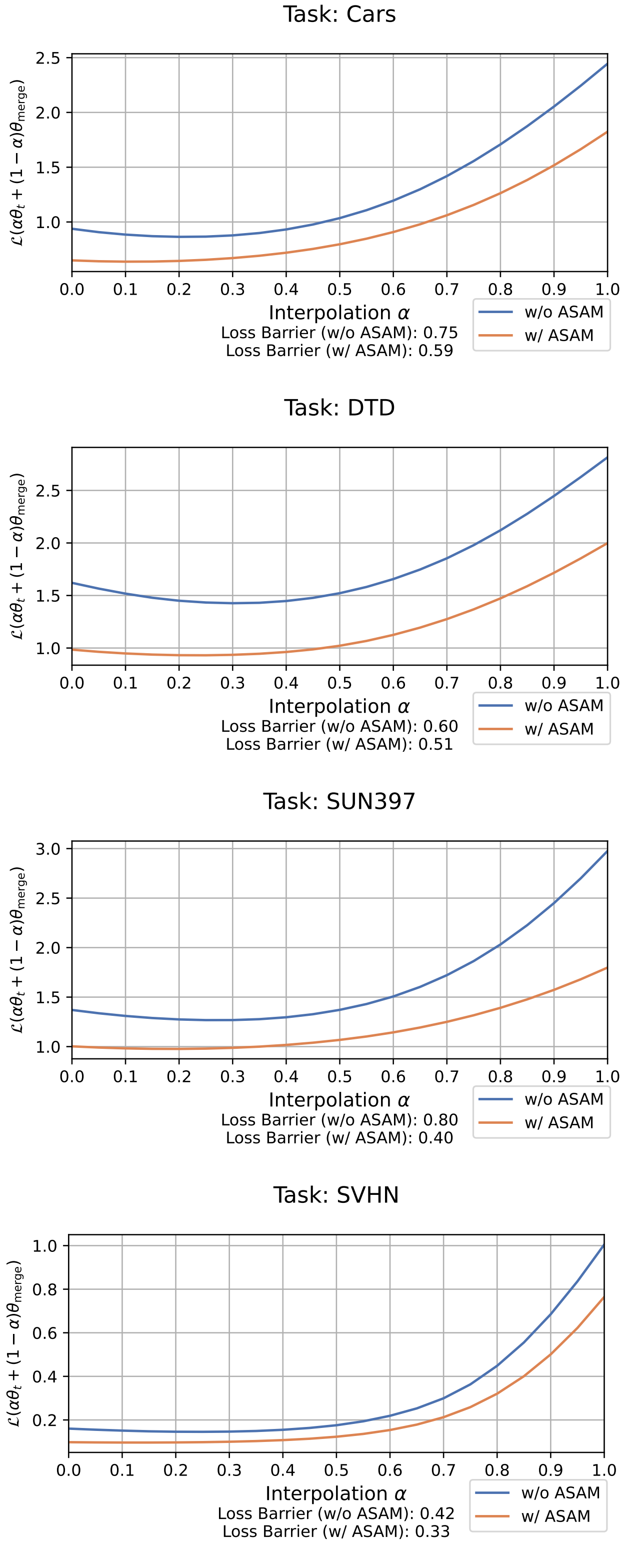

Figure 7: Test loss barrier between the merged model and each finetuned model.

Table 6: **Multi-task performance of task arithmetic with and without hyperparameter tuning**. We compare the average absolute and normalized accuracies of task arithmetic whether the hyperparameter is tuned. The fixed hyperparameter α is 0.4 for every 8 vision tasks.

Finetuning baselines (↓)	w/o tuning		w/ tuning		w/o tuning		w/ tuning	
	Abs.	Norm.	Abs.	Norm.	Abs.	Norm.	Abs.	Norm.
	ViT-B/32				ViT-B/16			
SGD w/o ASAM	46.34	48.72	68.23	75.47	46.91	49.98	73.40	79.31
SGD w/ ASAM (Ours)	**57.27**	**62.06**	**69.45**	**76.32**	**71.37**	**76.34**	**76.77**	**82.50**
FTTS w/o ASAM	72.89	81.97	78.35	86.83	76.69	84.05	79.37	87.33
FTTS w/ ASAM (Ours)	**75.21**	**85.78**	**79.38**	**87.72**	**78.87**	**87.17**	**79.78**	**88.26**
FTLO w/o ASAM	48.20	55.29	75.93	85.74	77.36	83.60	80.00	86.55
FTLO w/ ASAM (Ours)	**79.68**	**88.77**	**77.49**	**87.92**	**82.50**	**88.95**	**82.59**	**89.11**

C.6 MERGING WITH FIXED α_t

Previous research (Ilharco et al., 2023; Jin et al., 2023; Matena & Raffel, 2022; Yadav et al., 2023) on model merging has focused on finding better merged models through hyperparameter tuning. However, such methods become increasingly costly as the number of hyperparameters grows, and they need to be re-applied whenever tasks are added or changed. Therefore, it is essential to create a robust merged model that performs well regardless of the selected hyperparameters.

Our method achieves robustness by identifying flatter minima for joint loss and weight disentanglement compared to SGD, enabling the discovery of optimal hyperparameters across a wider range of conditions. To support this claim, we evaluate task arithmetic by fixing the task coefficients α to 0.4 for all merging tasks, following the recommendation of Ilharco et al. (2023). As shown in Table 6, our method outperforms other fine-tuning baselines in all cases, achieving improvements of up to 30% in both absolute accuracy and normalized accuracy. Additionally, there are several cases where the performances are nearly identical to those of hyperparameter tuning. Therefore, our method ensures that a merged model with reliable performance can be obtained, even when arbitrary hyperparameters are chosen.

C.7 MULTI-TASK PERFORMANCE OF OTHER FLAT-MINIMA TECHNIQUES

Table 7: **Multi-task performance across different flat-minima techniques**. We compare the average absolute and normalized accuracies of ViT-B/32 for five finetuning methods including three flat-minima techniques: SWA, RWP, and SAGM. We also compare the performance of merged model with and without hyperparameter tuning. All cases are merged with eight vision tasks by task arithmetic.

Finetuning baselines (↓)	w/o tuning		w/ tuning	
	Abs.	Norm.	Abs.	Norm.
SGD	46.34	48.72	68.23	75.47
SWA	48.94	52.46	68.58	76.11
RWP	34.97	36.58	62.48	73.02
SAGM	40.18	42.93	64.36	71.16
ASAM (Ours)	**57.27**	**62.06**	**69.45**	**76.32**

We also evaluate the performance of other flat-minima techniques, in addition to SAM variants like ASAM. Flat-minima techniques, including SWA (Izmailov et al., 2018), RWP (Li et al., 2022), and SAGM (Wang et al., 2023), are finetuning methods designed to minimize the loss while finding flat-minima during model training. As shown in Table 7, both our method and SWA improve multi-task performance compared to SGD. However, RWP and SAGM show worse performance than SGD. In addition, as discussed in Appendix C.6, we also present the results of performance evaluation without hyperparameter tuning in Table 7. Our method not only achieves the best performance when the hyperparameters are tuned but also outperforms in cases without tuning. Furthermore, the performance gap between our method and SGD, as well as other flat-minima techniques, widens significantly in these scenarios. This demonstrates that our method is superior to other flat-minima techniques and exhibits less performance degradation due to variations in parameter settings.

The primary difference among flat-minima techniques lies in the strategies used to derive perturbations. This seems to influence how effectively these techniques can reduce the performance gap between each finetuned model and merged model. Our method introduces perturbations by minimizing the loss difference between the current point in the parameter space and the point with the highest loss in its neighborhood during finetuning. This approach aligns closely with the objective of model merging, which aims to minimize the loss difference between the merged model and the individual finetuned models. Therefore, perturbation strategies derived from finetuning objectives similar to the model merging objective could result in greater performance improvements compared to other flat-minima techniques.

C.8 RESULTS IN NATURAL LANGUAGE PROCESSING

Table 8: **Multi-task performance of the merged model across four natural language understanding tasks**. We report the average absolute and normalized accuracies on four GLUE benchmark tasks: CoLA, MPRC, RTE, and SST-2. We finetune Flan-T5-base using either SGD or SGD with ASAM and merged the four GLUE tasks.

Finetuning	CoLA		MRPC		RTE		SST-2		Average	
baselines (↓)	Abs.	Norm.	Abs.	Norm.	Abs.	Norm.	Abs.	Norm.	Abs.	Norm.
SGD	58.77	75.58	25.74	29.75	37.55	43.52	64.11	68.68	46.54	54.38
FTTS	66.06	92.61	28.19	34.96	1.81	2.35	**87.39**	**94.90**	45.86	56.21
FTLO	66.83	96.67	**67.40**	**79.71**	0	0	13.30	14.50	36.88	47.72
ASAM (Ours)	**68.65**	**99.31**	42.16	52.92	**49.82**	**60.00**	45.30	49.38	**51.48**	**75.47**

To demonstrate the effectiveness of our method in other domains, we evaluate our method on NLP tasks. Following the evaluation settings of Ilharco et al. (2023), we finetune the Flan-T5-Base (Raffel et al., 2019; Wei et al., 2022) on four NLU tasks: CoLA, MRPC, RTE, and SST-2 in GLUE benchmark (Wang et al., 2019). All finetuning processes start from the Flan-T5 pretrained checkpoint available on HuggingFace. We finetune each model for 8000 iterations with a batch size of 16 and a learning rate of 10^{-5}. AdamW is used as the optimizer, and a linear annealing approach without warmup is applied as the learning rate scheduler. For efficient finetuning, we convert all downstream NLP tasks into a text-to-text format, following the approach in Jin et al. (2024). We measure the multi-task performance of the multi-task model merged by all four tasks using task arithmetic.

As shown in Table 8, our method outperforms SGD on all tasks except RTE. These results indicate that our method can enhance performance not only in vision tasks but also in NLP tasks. As shown in Table 8, our method outperforms SGD on all tasks except RTE, and on average, it achieves better multi-task performance compared to SGD.

C.9 TRAINING COSTS OF FINETUNING.

Table 9 presents a comparison of training costs between SGD and our method across various models and finetuning methods. We use AdamW as the optimizer for all training, setting the batch size to 64 only for finetuning ViT-B/16 using FTTS, while using a batch size of 128 for all other cases. Additionally, all training is conducted using Nvidia GeForce RTX 3090 GPUs. Training time approximately doubles after applying ASAM, while VRAM usage increases slightly.

Recently, there has been active research aimed at reducing the computational cost of SAM (Du et al., 2022; Liu et al., 2022). Model merging is an approach designed to efficiently build multi-task

Table 9: Training cost of SAM finetuning.

Fintuning baselines (↓)	ViT-B/32		ViT-B/16	
	Time (it/s)	VRAM (GB)	Time (it/s)	VRAM (GB)
SGD	3.75	7.3	1.01	21.5
FTTS	1.93	12.6	1.07	20.9
FTLO	4.61	5.8	1.25	19.0
ASAM (Ours)	1.98	7.8	0.50	21.6

models, and since our work seeks to establish a connection between model merging and SAM, we believe our research can significantly contribute to works focused on improving the efficiency of SAM.

D THEORETICAL DETAILS

D.1 CONNECTION BETWEEN SAM OBJECTIVE AND PARAMETER INTERFERENCE

Definition 1 (Joint-Task Loss)**. Joint-Task Loss**, denoted as $\mathcal{L}_{JTL}(\boldsymbol{\theta};\mathcal{D})$, represents the aggregate loss over multiple tasks. Here, $\boldsymbol{\theta}$ denotes the model parameters, and $\mathcal{D}$ is the combined dataset formed by the union of individual task datasets $\mathcal{D}_s$ and $\mathcal{D}_t$. Formally, it is defined as:

$$\mathcal{L}_{JTL}(\boldsymbol{\theta};\mathcal{D}) = \mathcal{L}(\boldsymbol{\theta};\mathcal{D}_s) + \mathcal{L}(\boldsymbol{\theta};\mathcal{D}_t), \quad \text{where} \quad \mathcal{D} = \mathcal{D}_s \cup \mathcal{D}_t, \tag{17}$$

where $\mathcal{L}(\boldsymbol{\theta};\mathcal{D}_s)$ and $\mathcal{L}(\boldsymbol{\theta};\mathcal{D}_t)$ denote the loss functions for datasets $\mathcal{D}_s$ and $\mathcal{D}_t$, respectively.

Definition 2 (Joint-Task Loss Linearity)**. Joint-Task Loss Linearity (JTL Linearity)** describes the linear relationship between the Joint-Task Loss of an interpolated model and the weighted sum of the individual Joint-Task Losses of task-specific models. Specifically, for datasets $\mathcal{D}_s$ and $\mathcal{D}_t$, with their respective fine-tuned parameters $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_t$, JTL Linearity holds if:

$$\mathcal{L}_{JTL}(\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t;\mathcal{D}) \approx \alpha\mathcal{L}_{JTL}(\boldsymbol{\theta}_s;\mathcal{D}) + (1-\alpha)\mathcal{L}_{JTL}(\boldsymbol{\theta}_t;\mathcal{D}), \tag{18}$$

where $\alpha \in [0,1]$ is a scalar coefficient. This approximation implies that the Joint-Task Loss of the parameter combination $\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t$ is approximately equal to the weighted sum of the individual Joint-Task Losses.

Property 1 (SAM Reduces the Dominant Hessian Eigenvalue $\lambda_{\max}$)**.** Let $\mathbf{H}(\boldsymbol{\theta};\mathcal{D}) = \nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta};\mathcal{D})$ be the Hessian matrix respect to the model parameter $\boldsymbol{\theta}$. The dominant Hessian eigenvalue $\lambda_{\max}(\boldsymbol{\theta};\mathcal{D})$ is the largest eigenvalue of $\mathbf{H}(D;\boldsymbol{\theta})$:

$$\lambda_{\max}(\boldsymbol{\theta};\mathcal{D}) = \max_{\|v\|_2=1} v^\top \mathbf{H}(\boldsymbol{\theta};\mathcal{D})v. \tag{19}$$

Agarwala & Dauphin (2023) demonstrated that SAM provides strong regularization of the eigenvalues throughout the learning trajectory. As illustrated in Figure 8, we further discover that SAM not only regularizes the learning trajectory but also reduces the dominant Hessian eigenvalue for parameters θ along the line segment between the pretrained parameter θ_0 and the finetuned parameter θ_t.

D.2 PROOF OF THEOREM 1

Proof. We aim to show that:

$$|\delta| \leq \frac{1}{2}\alpha(1-\alpha)(\lambda_s+\lambda_t)\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|^2 + \epsilon, \tag{20}$$

$$where \quad \lambda_s = \lambda_{\max}(\boldsymbol{\theta}_s;\mathcal{D}_s), \lambda_t = \lambda_{\max}(\boldsymbol{\theta}_t;\mathcal{D}_t) \tag{21}$$

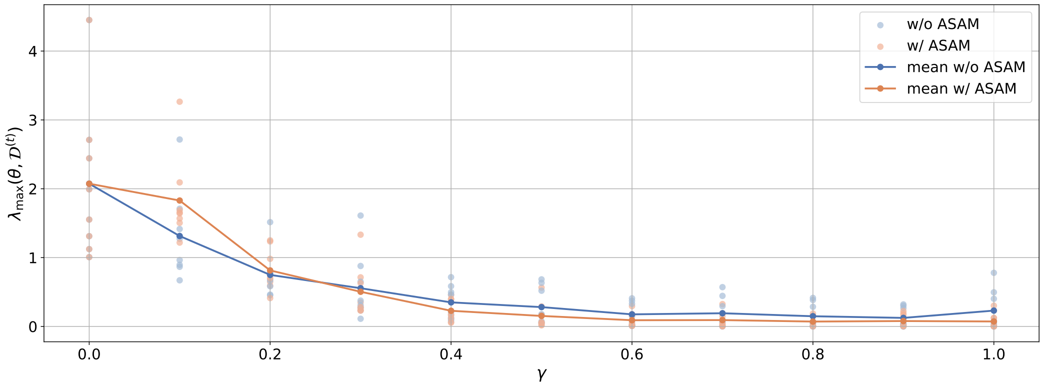

Figure 8: **Comparison of the dominant Hessian eigenvalue for parameters on the line segment between the pretrained parameter and finetuned parameter.** We compare the dominant Hessian eigenvalue $\lambda_{\max}(\boldsymbol{\theta}; \mathcal{D}^{(t)})$ of parameter $\boldsymbol{\theta}$ along the line segment $\overline{\boldsymbol{\theta}_0 \boldsymbol{\theta}_t}$, where $\boldsymbol{\theta} = \boldsymbol{\theta}_0 + \gamma(\boldsymbol{\theta}_t - \boldsymbol{\theta}_0)$ for $\gamma \in [0, 1]$, $\mathcal{D}^{(t)}$ denotes the dataset for task k. The line represents the mean of the dominant Hessian eigenvalues for all tasks.

Recall that the Joint-Task Loss is defined as $\mathcal{L}_{JTL}(\boldsymbol{\theta}; \mathcal{D}) = \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_s) + \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_t)$, define δ as:

$$\begin{aligned}
\delta &= \mathcal{L}_{JTL}(\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t; \mathcal{D}) - \alpha\mathcal{L}_{JTL}(\boldsymbol{\theta}_s; \mathcal{D}) - (1-\alpha)\mathcal{L}_{JTL}(\boldsymbol{\theta}_t; \mathcal{D}) \\
&= [\mathcal{L}(\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t; \mathcal{D}_s) - \alpha\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) - (1-\alpha)\mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_s)] \\
&\quad + [\mathcal{L}(\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t; \mathcal{D}_t) - \alpha\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_t) - (1-\alpha)\mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_t)] \\
&= \delta_s + \delta_t, \qquad (22) \\
\textit{where} \quad \delta_s &= \mathcal{L}(\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t; \mathcal{D}_s) - \alpha\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) - (1-\alpha)\mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_s), \\
\delta_t &= \mathcal{L}(\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t; \mathcal{D}_t) - \alpha\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_t) - (1-\alpha)\mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_t).
\end{aligned}$$

Performing a third-order Taylor expansion of $\mathcal{L}(\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t; \mathcal{D}_s)$ around $\boldsymbol{\theta}_s$:

$$\begin{aligned}
\mathcal{L}(\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t; \mathcal{D}_s) &= \mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) + (1-\alpha)\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s)^\top(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) \\
&\quad + \frac{1}{2}(1-\alpha)^2(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_s(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + R_s, \qquad (23)
\end{aligned}$$

where $\mathbf{H}_s = \nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s)$ and R_s is the remainder term.

Similarly, expand $\mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_s)$ around $\boldsymbol{\theta}_s$:

$$\mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_s) = \mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) + \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s)^\top(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + \frac{1}{2}(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_s(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + R'_s. \qquad (24)$$

Multiply both sides by $(1-\alpha)$:

$$\begin{aligned}
(1-\alpha)\mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_s) &= (1-\alpha)\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) + (1-\alpha)\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s)^\top(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) \\
&\quad + \frac{1}{2}(1-\alpha)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_s(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + (1-\alpha)R'_s. \qquad (25)
\end{aligned}$$

Compute δ_s:

$$\begin{aligned}
\delta_s &= \mathcal{L}(\alpha\boldsymbol{\theta}_s + (1-\alpha)\boldsymbol{\theta}_t; \mathcal{D}_s) - \alpha\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) - (1-\alpha)\mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_s) \\
&= \left[\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) + (1-\alpha)\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s)^\top(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + \frac{1}{2}(1-\alpha)^2(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_s(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + R_s\right] \\
&\quad - \alpha\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) - \Big[(1-\alpha)\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) + (1-\alpha)\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s)^\top(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) \\
&\quad + \frac{1}{2}(1-\alpha)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_s(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + (1-\alpha)R'_s\Big]. \qquad (26)
\end{aligned}$$

Simplify the expression:

$$\begin{aligned}
\delta_s &= \mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) - \alpha \mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) - (1-\alpha)\mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s) \\
&\quad + (1-\alpha)\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s)^\top (\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) - (1-\alpha)\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_s; \mathcal{D}_s)^\top (\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) \\
&\quad + \frac{1}{2}(1-\alpha)^2 (\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_s (\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) - \frac{1}{2}(1-\alpha)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_s (\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) \\
&\quad + R_s - (1-\alpha)R'_s. \qquad (27) \\
&= -\frac{1}{2}\alpha(1-\alpha)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_s (\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + \left(R_s - (1-\alpha)R'_s\right). \qquad (28)
\end{aligned}$$

Similarly, compute δ_t by expanding around $\boldsymbol{\theta}_t$:

$$\delta_t = -\frac{1}{2}\alpha(1-\alpha)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_t (\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + \left(R_t - \alpha R'_t\right), \tag{29}$$

$$\text{where} \quad \mathbf{H}_t = \nabla^2_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t; \mathcal{D}_t). \tag{30}$$

Combining δ_s and δ_t:

$$\delta = \delta_s + \delta_t = -\frac{1}{2}\alpha(1-\alpha)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top (\mathbf{H}_s + \mathbf{H}_t)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) + \epsilon, \tag{31}$$

$$\text{where} \quad \epsilon = \left(R_s - (1-\alpha)R'_s\right) + \left(R_t - \alpha R'_t\right). \tag{32}$$

Since $(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_s (\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) \leq \lambda_s \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|^2$ and $(\boldsymbol{\theta}_t - \boldsymbol{\theta}_s)^\top \mathbf{H}_t (\boldsymbol{\theta}_t - \boldsymbol{\theta}_s) \leq \lambda_t \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|^2$, $|\delta|$ is bounded as:

$$|\delta| \leq \frac{1}{2}\alpha(1-\alpha)(\lambda_s + \lambda_t)\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|^2 + \epsilon, \tag{33}$$

where $\epsilon = (R_s - (1-\alpha)R'_s) + (R_t - \alpha R'_t)$ is the remainder term. $\square$