Mastering Task Arithmetic: τJp as a Key Indicator for Weight Disentanglement

Kotaro Yoshida*
Institute of Science Tokyo
yoshida.k.0253@m.isct.ac.jp

Yuji Naraki
Independent Researcher
yuji.1277@gmail.com

Takafumi Horie*
Ritsumeikan University
horie.takafumi@em.ci.ritsumei.ac.jp

Ryosuke Yamaki
Ritsumeikan University, ProPlace Inc
yamaki.ryosuke@em.ci.ritsumei.ac.jp

Ryotaro Shimizu
University of California San Diego, ZOZO Research
r2shimizu@ucsd.edu

Yuki Saito
ZOZO Research
yuki.saito@zozo.com

Julian McAuley
University of California San Diego
jmcauley@ucsd.edu

Hiroki Naganuma
Université de Montréal, Mila, ProPlace Inc
naganuma.hiroki@mila.quebec

Abstract

Model-editing techniques using task arithmetic have rapidly gained attention. Through task arithmetic, simply through arithmetic operations on the weights of pre-trained and fine-tuned models create desired models, such as multi-task models, models with specific tasks unsolvable, or domain-transferred models. However, task arithmetic faces challenges, such as low reproducibility and the high cost associated with adjusting coefficients in the arithmetic operations on model parameters, which have limited its practical success. In this paper, we present three key contributions in the context of task addition and task negation within task arithmetic. First, we propose a new metric called τJp which is based on the product of the task vector (τ) and the Jacobian of the pre-trained model with respect to its weights. We show that τJp has a causal relationship with the interference that occurs from arithmetic operations. Second, we show that introducing regularization to minimize τJp significantly mitigates interference between task inferences, which leads to eliminating coefficient tuning and better accuracy on each task. Third, in the context of incremental learning, we confirmed that our τJp regularization demonstrates more robust performance in environments where future tasks to be learned are not accessible, validating the scalability of the approach. Finally, we demonstrate that the τJp regularizer further reinforces the performance of task arithmetic by leveraging publicly available fine-tuned models, offering practical benefits for real-world applications. Our code is available at https://github.com/katoro8989/tau-Jp_Task_Arithmetic

*Work was performed when K.Yoshida and T.Horie were ProPlace interns

38th Workshop on Fine-Tuning in Machine Learning (NeurIPS 2024).

1 Introduction

While there is a growing demand for foundational models in recent machine learning trends, the high computational costs associated with their training (Zhou et al., 2023; Kaplan et al., 2020; Villalobos et al., 2022) remain a significant barrier to broader practical use. To address this, model-editing techniques using task arithmetic (Ilharco et al., 2023) have rapidly gained attention in the fields of deep learning (Yadav et al., 2023; Davari and Belilovsky, 2023; Yu et al., 2024; Tang et al., 2023b; Ortiz-Jimenez et al., 2023). Task arithmetic offers a significant advantage over traditional approaches by enabling the efficient creation of edited models without the need for additional training, simply through arithmetic operations on the weights of pre-trained and fine-tuned models. Specifically, task arithmetic enables three operations; the creation of a single model capable of handling multiple tasks (task addition), a model that selectively reduces the performance for a specific task (task negation), and a model capable of handling tasks not explicitly included in the training data (task analogies). These are realized by basic operations such as scalar multiplication, addition, and subtraction.

However, task arithmetic faces challenges, such as low reproducibility and the high cost associated with adjusting coefficients in the arithmetic operations on model parameters, which have limited its practical success (see Table 1 and Table 2). In addition, there is still limited theoretical understanding of why and how these techniques work (Ortiz-Jimenez et al., 2023). Ortiz-Jimenez et al. (2023) demonstrated in their experimental setup for task addition and task negation that the degree of interference between task inferences can be quantified using a metric called the weight disentanglement error. They also observed that linearizing the model by the neural tangent kernel (NTK) (Jacot et al., 2018) approximation reduced the weight disentanglement error. However, while their study provides important insight into the condition of successful task arithmetic, its scope is limited to indirect explanations and approaches to improvement.

Ensuring high reproducibility and minimizing computational costs while avoiding task interference is essential for the practical application of task arithmetic. To address these challenges, we shed light on the product of task vectors τ and the Jacobian matrix of the model function with respect to its parameters. In particular, we investigate the relationship between this product and weight disentanglement, drawing insights from the NTK regime and model linearization (Jacot et al., 2018; Ortiz-Jimenez et al., 2023). We introduce a novel metric, τJp, and theoretically demonstrate that it has a causal link to weight disentanglement. Based on this insight, we introduce the regularization to minimize τJp and acquire task vectors with small interference between tasks. Moreover, we demonstrate the effectiveness of the τJp regularizer in scenarios where future tasks to be learned remain unknown or inaccessible. This is a critical requirement for scaling task arithmetic to more complex and realistic environments. We further explore improving task arithmetic performance by applying τJp regularization to the continual training of existing fine-tuned models. Our results show that this approach is effective even with publicly available fine-tuned models, providing practical advantages for real-world applications.

In this paper, we present three key contributions in the context of task addition and task negation within task arithmetic.

- We propose a new metric, τJp (τ-Jacobian product), which can be shown to have a causal relationship with weight disentanglement. We show that τJp tends to be inversely correlated with normalized accuracy, i.e., the metric of performance variation from accuracy before task arithmetic (Section 4).
- By introducing regularization during fine-tuning to minimize τJp, we significantly reduce the interference between task predictions, thus greatly reducing the need for coefficient adjustments (Section 5.1 and Section 5.2).
- We demonstrate that the regularization of τJp is effective in two practical scenarios: i) when future tasks to be learned are unknown, or ii) when using publicly available fine-tuned models. Our regularization method demonstrates both scalability and practical applicability. (Section 5.3).

We believe that these contributions will lead to the practical application of model-editing techniques using task arithmetic.

2 Related Work

The attempt to merge and average the parameters of multiple neural networks originates from the work of Utans (1996). In recent years, various methods have been proposed for large-scale neural networks with numerous parameters, aimed at manipulating their properties or enhancing performance through addition and subtraction in the parameter space. Among these are many techniques related to task arithmetic, including task analogies, as well as approaches associated with model merging. Among the simpler methods of model merging are techniques that utilize either the simple or weighted averaging of multiple models (Wortsman et al., 2022a; Choshen et al., 2022; Don-Yehiya et al., 2023; Ramé et al., 2023; Muqeeth et al., 2023; Jolicoeur-Martineau et al., 2024), or the linear interpolation between pre-trained and fine-tuned models (Ilharco et al., 2022; Wortsman et al., 2022b).

In the integration of models via model merging or task arithmetic, interference between the parameters of multiple models or task vectors can arise, and various methods have been proposed to mitigate such conflicts. For instance, some approaches apply masking operations to task vectors (Tang et al., 2023a; Wang et al., 2024a; Huang et al., 2024), while others involve trimming or scaling techniques (Yadav et al., 2023; Davari and Belilovsky, 2023; Yu et al., 2024), or leverage model linearization (Tang et al., 2023b; Ortiz-Jimenez et al., 2023).

Theoretical and analytical studies on the effectiveness of model merging and task arithmetic include research based on analyses of the loss landscape (Entezari et al., 2022; Qin et al., 2022; Gueta et al., 2023), as exemplified by linear mode connectivity (Frankle et al., 2020), as well as approaches that leverage model linearization within the NTK regime (Jacot et al., 2018). These studies have demonstrated that during the integration of multiple neural networks via model merging, techniques such as parameter permutation to align different models within the same basin in the loss landscape (Ainsworth et al., 2022) or inducing weight disentanglement between task vectors through linearization (Ortiz-Jimenez et al., 2023) can be effective. Further detailed discussion is given in Appendix A.

3 Background

Notation. Let $\theta \in \mathbb{R}^p$ represent the weights of a neural network $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^c$ are the input and output spaces with dimensionalities d and c, respectively. The parameter θ has dimensionality p, representing the total number of model parameters. Additionally, let θ_0 represent the pre-trained weights and $\theta^\star$ represent the fine-tuned weights. Let $\mathcal{T}$ denote the index set of all possible tasks. Define the index set $T \subseteq \mathcal{T}$ as the set containing the indices of all tasks used. For each task $t \in T$, the corresponding dataset $D_t = \{(x_{t_i}, y_{t_i})\}_{i=1}^{|D_t|}$ is defined, where $x_{t_i} \in \mathcal{X}$ and $y_{t_i} \in \mathcal{Y}$. For a task t, fine-tuning is conducted by minimizing the loss function $\frac{1}{|D_t|}\sum_{i=1}^{|D_t|} L(f(x_{t_i};\theta), y_{t_i})$, starting from θ_0, and yielding the fine-tuned weights $\theta_t^\star$.

3.1 Model Editing via Task Arithmetic

Task arithmetic (Ilharco et al., 2023) represents the difference between the weights of a fine-tuned model and those of a pre-trained model — specifically, $\tau = \theta^\star - \theta_0$ — as a task vector. By performing arithmetic operations, such as the addition or subtraction of multiple task vectors, and then adding the result to the pre-trained weights θ_0, the model can be effectively edited. Three key methods leveraging task vectors for model editing have gained recognition in the field. **Task addition** creates a multi-task model by summing task vectors obtained from various tasks and then adding this sum to the weights of a pre-trained model. **Task negation** suppresses or erases the abilities and properties only learned from a specific task by subtracting the corresponding task vector from the pre-trained model's weights. For instance, it can be used to forget harmful behaviors or biases learned during training. **Task analogies** enable transfer learning to unseen tasks by using task vector addition and subtraction based on analogical relationships between tasks, such as "A is to B as C is to D." For example, by leveraging three known tasks, task analogies can be used to infer the properties or performance of a fourth, previously unseen task.

3.2 Weight Disentanglement

Ortiz-Jimenez et al. (2023) introduced the concept of **weight disentanglement** to measure the degree of interference between task vectors in task arithmetic. Satisfying weight disentanglement is represented by the following condition:

$$f\left(x;\theta_0+\sum_{t\in T}\alpha_t\tau_t\right)=\sum_{t\in T}f(x;\theta_0+\alpha_t\tau_t)\mathbb{1}(x\in D_t)+f(x;\theta_0)\mathbb{1}\left(x\notin\bigcup_{t\in T}D_t\right) \tag{1}$$

The above equation implies that when performing task arithmetic using all task vectors within T , for a given task t , the model will produce the same output as when using only the task vector τ_t , and for tasks outside of T , the model will produce the same output as the pre-trained model. To assess weight disentanglement between two tasks, weight disentanglement error was proposed.

$$\xi\left(\alpha_1,\alpha_2\right)=\sum_{t=1}^{2}\mathbb{E}_{x\sim\mu_t}\left[\operatorname{dist}\left(f\left(x;\theta_0+\alpha_t\tau_t\right),f\left(x;\theta_0+\alpha_1\tau_1+\alpha_2\tau_2\right)\right)\right] \tag{2}$$

where $\operatorname{dist}(\cdot,\cdot)$ measures the distance between two models' vector outputs. For classification tasks, it checks whether the predicted labels from the two models, $\hat{y}_1$ and $\hat{y}_2$, match, i.e., $\operatorname{dist}(\hat{y}_1,\hat{y}_2)=\mathbb{1}(\hat{y}_1\neq\hat{y}_2)$. This error captures the difference in output distributions when task vectors are applied individually or jointly to a pre-trained model, reflecting the interference between task vectors in function space. Ideally, in task arithmetic, each task vector would independently influence the model's output, resulting in the error being small.

3.3 Neural Tangent Kernel

The Neural Tangent Kernel (NTK) (Jacot et al., 2018) is a kernel that linearizes the learning dynamics of infinite-width neural networks. In infinite-width networks, parameter updates during training become infinitesimally small, allowing the following first-order Taylor approximation to hold:

$$f(x;\theta)\approx f(x;\theta_0)+(\theta-\theta_0)^\top\nabla_\theta f(x;\theta_0). \tag{3}$$

This approximation is valid in a regime commonly referred to as the NTK regime, or tangent space (hereafter referred to as NTK regime), where the relationship between the parameter space and function space becomes linearized. Recent studies have observed that fine-tuning large pre-trained neural networks often operate within the NTK regime, as the parameter changes during fine-tuning remain sufficiently small (Malladi et al., 2023; Ren et al., 2023). In contrast, it has also been reported that in practice, fine-tuning finite-width models does not always result in perfectly linear behavior, and fine-tuning can exhibit non-linear characteristics (Ortiz-Jimenez et al., 2023).

3.4 Task Arithmetic in the NTK regime

In task arithmetic, the reason why linear operations in the weight space of neural networks translate directly to changes in the function space can be explained by the following NTK approximation:

$$f(x;\theta_0+\sum_{t\in T}\alpha_t\tau_t)\approx f(x;\theta_0)+\sum_{t\in T}(\alpha_t\tau_t)^\top\nabla_\theta f(x;\theta_0) \tag{4}$$

where for all $t\in T$, τ_t denotes the task vector for task t, defined as $\tau_t=\theta_t^\star-\theta_0$, and $\alpha_t\in\mathbb{R}$. In simple terms, in the NTK regime, the linearity of operations on task vectors is preserved in the model's output, resulting in corresponding linear effects on performance.

In practice, it has been reported that explicitly enforcing fine-tuning within the NTK regime improves task arithmetic (Ortiz-Jimenez et al., 2023; Tang et al., 2023b). Ortiz-Jimenez et al. (2023); Tang et al. (2023b) demonstrated that fine-tuning within the NTK regime lowers weight disentanglement error and improves the performance of task addition and negation. One linearization method proposed by Ortiz-Jimenez et al. (2023) is to fine-tune linearized models $f_{\text{lin}}(x,\theta)$ within their NTK regime when creating task vectors and the formulation follows:

$$f_{\text{lin}}(x,\theta)=f(x,\theta_0)+\tau^\top\nabla_\theta f(x,\theta_0) \tag{5}$$

However, it remains unclear why linearizing the model suppresses weight disentanglement error and how this, in turn, enhances task arithmetic. These questions have not yet been fully addressed from a theoretical standpoint. We focus on the term $\tau^\top \nabla_\theta f(x;\theta_0)$ in the NTK approximation and aim to provide a theoretical explanation. Building on this theoretical foundation, we propose a novel method to enhance task arithmetic.

4 Causal Impact of the τ-Jacobian Product on Weight Disentanglement

We theoretically explain weight disentanglement in the NTK regime and propose the τ-Jacobian product as the underlying mechanism that drives weight disentanglement. We also experimentally demonstrate the relationship between the τ-Jacobian product and model interference.

4.1 Weight disentanglement in the NTK regime

In this section, we attempt to provide a theoretical explanation of the relationship between weight disentanglement and the task vector Jacobian product in the NTK regime. For simplicity, we consider task arithmetic involving two tasks, A and B. In the NTK regime, the model's output can be approximated as follows:

$$f(x,\theta_0+\alpha_A\tau_A+\alpha_B\tau_B) \approx f(x,\theta_0)+\alpha_A\tau_A^\top\nabla_\theta f(x,\theta_0)+\alpha_B\tau_B^\top\nabla_\theta f(x,\theta_0) \tag{6}$$

with $\alpha_A, \alpha_B \in \mathbb{R}$. In this case, for inputs x_A and x_B from tasks A and B, it is evident that satisfying the following two conditions is equivalent to achieving the weight disentanglement error of 0.

$$\begin{aligned} f(x_A,\theta_0+\alpha_A\tau_A+\alpha_B\tau_B) &\approx f(x_A,\theta_0)+\alpha_A\tau_A^\top\nabla_\theta f(x_A,\theta_0)+\mathbf{0} \approx f(x_A,\theta_0+\alpha_A\tau_A),\\ f(x_B,\theta_0+\alpha_A\tau_A+\alpha_B\tau_B) &\approx f(x_B,\theta_0)+\mathbf{0}+\alpha_B\tau_B^\top\nabla_\theta f(x_B,\theta_0) \approx f(x_B,\theta_0+\alpha_B\tau_B). \end{aligned} \tag{7}$$

The above equations imply that the weight disentanglement error is 0 when the task vectors satisfy the following conditions:

$$\begin{aligned} \tau_A^\top\nabla_\theta f(x_B,\theta_0)&=\mathbf{0},\\ \tau_B^\top\nabla_\theta f(x_A,\theta_0)&=\mathbf{0}. \end{aligned} \tag{8}$$

These conditions imply that the task vector for a given task is orthogonal to the Jacobian of the pre-trained model, with respect to its parameters θ_0, on the other task. In other words, linearizing the model alone does not guarantee weight disentanglement; it is also necessary to satisfy the aforementioned conditions (Eq. 8), as demonstrated theoretically.

We propose the following τ-Jacobian product (τJp) as a measure of how well the condition in Eq. (8) is satisfied between two tasks:

$$\tau\text{Jp} = \frac{1}{2}\left(||\tau_A^\top\nabla_\theta f(x_B,\theta_0)||^2+||\tau_B^\top\nabla_\theta f(x_A,\theta_0)||^2\right). \tag{9}$$

The τJp is the average of the product of one task vector and the gradient of the pre-trained model with respect to its weights on the other dataset, taken across both datasets. According to the condition Eq. (8), a smaller τJp is desirable.

4.2 Relationship between τ-Jacobian product and interference

As demonstrated in Section 4.1, a smaller τJp improves weight disentanglement and reduces interference between task vectors. In this section, we experimentally show that minimizing τJp effectively mitigates task vector interference.

In the experiments, linearized fine-tuning (FT) (Ortiz-Jimenez et al., 2023) of different pre-trained Vision Transformers (ViTs) (Dosovitskiy et al., 2021) under the same conditions as in Ilharco et al. (2022); Ortiz-Jimenez et al. (2023) was conducted using CLIP (Radford et al., 2021) on eight image tasks. Specifically, the eight tasks are 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).

First, we investigated the relationship between τJp and weight disentanglement. Figure 1 visualizes weight disentanglement alongside τJp. In the top row showing Linear FT, we can see that when τJp

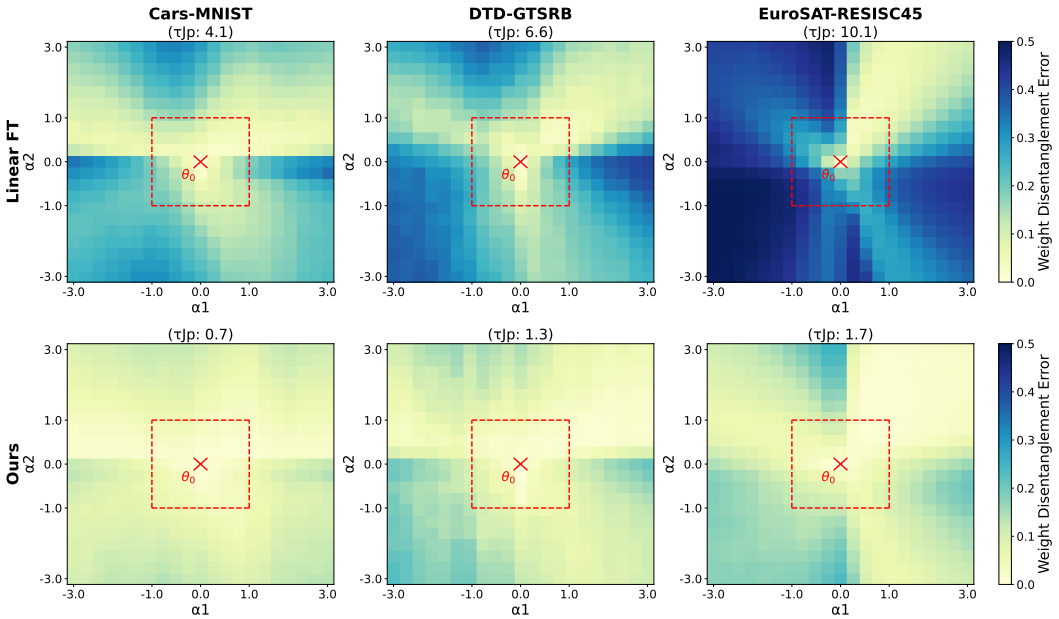

Figure 1: Visualization of weight disentanglement in ViT-B-32 with respect to τJp. The upper row illustrates the linearized model without regularization, while the lower row presents the model with our proposed regularization. Overall, it is observed that when τJp is large, weight disentanglement becomes sensitive to the coefficients. As τJp increases, weight disentanglement shows greater robustness to variations in the coefficients. Furthermore, our proposed regularization enhances this robustness with respect to the coefficients. The red cross at the center represents the pre-trained model, and the red box indicates the typical coefficient search range in task arithmetic.

is large, the blue area becomes more prominent, indicating that the weight disentanglement error is more sensitive to each coefficient and interference is not being prevented. As τJp decreases, the error tends to become more robust to changes in the coefficients.

Next, we focus on the actual performance of task arithmetic. We analyzed the correlation between normalized accuracy and τJp, presenting the resulting scatter plot in Figure 2. Each data point represents a model trained by performing task addition on two out of the eight image tasks. Normalized accuracy is defined as the accuracy of each task after applying task arithmetic relative to its accuracy before task arithmetic, which is set to 1.0. Across all model scales, we observed a consistent trend where task pairs with smaller τJp values tend to exhibit higher normalized accuracy.

5 Enhancing Task Arithmetic by Mitigating Interference Between Tasks

5.1 τ-Jacobian Product for Regularization

As demonstrated in Section 4, to prevent interference between task vectors in task arithmetic and to improve performance, it is necessary not only to linearize the model but also to keep τJp small simultaneously. Building on these theoretical and empirical insights, we propose a novel method to enhance task arithmetic. Specifically, we introduce a regularization during fine-tuning that encourages τJp to be small — that is, we promote learning to occur in a subspace where τ is orthogonal to the Jacobian of the pre-trained model, with respect to θ_0, for different tasks.

Inspired by this requirement, we propose the following τJp-based regularized loss function:

$$L_{\tau\text{Jp}}(f_{\text{lin}}(x;\theta),y) = L(f_{\text{lin}}(x;\theta),y) + \lambda \sum_{t\in T_{\text{orth}}} ||(\theta-\theta_0)^\top \nabla_\theta f_{\text{lin}}(x_t,\theta_0)||^2 \tag{10}$$

where T_{orth} denotes the set containing indices of other tasks for which we aim to suppress interference. Our objective is to ensure that $\theta^\star - \theta_0$ is orthogonal to all $\nabla_\theta f(x_t,\theta_0)$ for $t \in T_{\text{orth}}$; that is, we add the L2 norm of their product as a regularization term. The hyperparameter λ adjusts the strength of

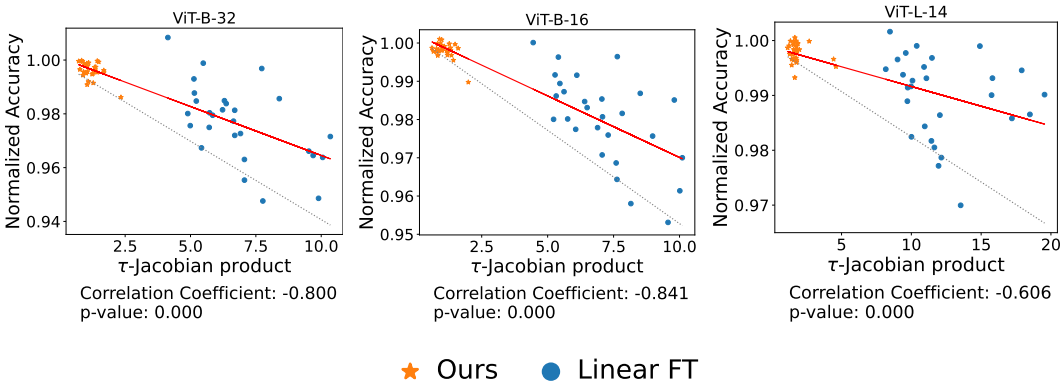

Figure 2: Visualization of the relationship between τJp and normalized accuracy. Each point represents a pair of tasks from the set of eight tasks, yielding $\binom{8}{2}$ combinations, i.e., 28 in total. We observed a correlation, where smaller τJp values are associated with higher normalized accuracy. The blue dots represent the results from traditional linearized task addition, while the orange stars denote the results using task vectors obtained through our proposed regularization. A significant difference in τJp values between the two approaches is evident, indicating that our proposed regularization reduces τJp and improves task addition performance.

the regularization. It is important to note that, for the computation of the regularization terms, only the input data for each sample in T_{orth} is required, and labels are not necessary.

However, in practical applications, when there are numerous tasks in T_{orth} where interference needs to be reduced, calculating penalties for all tasks at each iteration results in significant memory and computational overhead. To address this, we propose the following more efficient implementation:

$$\hat{L}^{(i)}_{\tau\text{Jp}}(f_{\text{lin}}(x;\theta),y) = L(f_{\text{lin}}(x;\theta),y) + \lambda||(\theta-\theta_0)^\top \nabla_\theta f_{\text{lin}}(x_{(i \bmod |T_{\text{orth}}|)},\theta_0)||^2 \tag{11}$$

where i denotes the iteration number, and at each iteration, the task for which the penalty is calculated is rotated within T_{orth} (specifically, $(i \bmod |T_{\text{orth}}|)$). With this approach, it is sufficient to calculate the penalty for one task per iteration, ensuring scalability with respect to the size of T_{orth}. In Appendix D, we conducted a comparison between the loss functions in Eq. (10) and Eq. (11) using ViT-B-32. Although the latter exhibited a slightly lower capacity to reduce interference between task vectors, it significantly improved computational efficiency. Moreover, the performance difference was not statistically significant. Therefore, for the remainder of the experiments, we will adopt $\hat{L}_{\tau\text{Jp}}$ in Eq. (11).

5.2 Enhancement through τ-Jacobian Product Regularization

We compared linearized fine-tuning using the regularization in $\hat{L}_{\tau\text{Jp}}$, standard fine-tuning (Non-lin. FT), and fine-tuning with only linearization (Linear FT) in both task addition and negation scenarios. The experimental setup for task addition follows that described in Section 4.2. For task negation, we also introduced a control task — ImageNet (Deng et al., 2009) — to maintain performance during negation. In our proposed method, during training for each task within the eight tasks, T_{orth} consisted of all tasks except the target task, as well as ImageNet. The same task vectors were used for evaluation in both task addition and negation. Further details on the fine-tuning settings can be found in Appendix C.

First, the results of task addition presented in Table 1 show that both the "One-shot" and "Tuned" of our method consistently outperform existing tuned approaches (see table caption for definitions). These methods achieve notable improvements in both average absolute (Abs.) and normalized (Norm.) performance. The one-shot method performs better than prior methods while reducing the cost of tuning the inference-time hyperparameter. Notably, for ViT-L-14, the both one-shot and tuned methods yield the same results, indicating that $\alpha_t = 1.0$ is optimal, and achieve a normalized accuracy of 99%. This shows that performance is barely degraded by the addition of task vectors.

Next, examining the task negation results presented in Table 2, we observe that although our method's one-shot approach does not achieve sufficient forgetting of the target task (Targ.), it significantly

outperforms existing methods in preserving the performance on the control tasks (Cont.). In contrast, in the tuned case, our method greatly enhances the forgetting of the target task, while still surpassing existing methods in preserving control task performance across all cases.

We clarify why the one-shot approach (with $\alpha_t = 1.0$) was effective for task addition but not for task negation. As detailed in Appendix B, in the ideal case where τJp is zero (no interference), the optimal coefficient α_t for task addition is 1, making the one-shot method effective. Conversely, for task negation, the optimal α_t should be infinitely large in this ideal scenario. However, in realistic situations where τJp is not zero and interference exists, there is no well-defined theoretical optimal coefficient for task negation. This makes the one-shot approach with $\alpha_t = 1.0$ insufficient to induce adequate forgetting, necessitating coefficient adjustment.

Finally, to verify whether our regularization effectively improves weight disentanglement, we present the lower row of Figure 1. Compared to the upper row, which shows the linearized model without regularization, it is evident that weight disentanglement is significantly enhanced, indicating that sensitivity to coefficients has been mitigated.

Method	Coef. Tuning	ViT-B-32		ViT-B-16		ViT-L-14	
		Abs. (↑)	Norm. (↑)	Abs. (↓)	Norm. (↑)	Abs. (↓)	Norm. (↑)
Pre-trained	-	47.3	-	54.5	-	65.1	-
Non-lin. FT	One-shot	19.9	20.5	19.1	19.7	37.6	39.0
	Tuned	70.4	78.0	75.5	81.5	84.0	89.3
Linear FT	One-shot	55.4	61.7	58.2	63.6	80.5	86.7
	Tuned	74.3	85.0	78.7	87.6	85.8	92.8
Ours	One-shot	84.2	97.2	87.5	98.4	**90.8**	**99.0**
	Tuned	**84.5**	**97.6**	**87.6**	**98.5**	**90.8**	**99.0**

Table 1: Results of task addition using the eight tasks presented in Section 4.2. "Ours" refers to our proposed linearized fine-tuning with τJp regularization. "One-shot" indicates task addition where task vectors are used without scaling, i.e., with $\alpha = 1.0$, while "Tuned" refers to task vectors that are further optimized with coefficient adjustments. Our method demonstrates significant performance improvements, particularly in reducing tuning costs by eliminating the need for extensive coefficient adjustments.

Method	Coef. Tuning	ViT-B-32		ViT-B-16		ViT-L-14	
		Targ. (↓)	Cont. (↑)	Targ. (↓)	Cont. (↑)	Targ. (↓)	Cont. (↑)
Pre-trained	-	47.3	66.7	54.5	69.3	65.1	77.3
Non-lin. FT	One-shot	(10.9)	(44.7)	(10.8)	(51.6)	(15.2)	(68.6)
	Tuned	24.0	60.7	20.3	64.7	18.4	72.4
Linear FT	One-shot	(6.3)	(57.2)	(5.4)	(62.2)	(3.0)	(67.9)
	Tuned	11.8	60.6	8.8	65.0	8.3	72.2
Ours	One-shot	11.8	**62.5**	11.8	**67.8**	15.1	**75.1**
	Tuned	**6.7**	60.8	**4.7**	66.0	**3.7**	73.0

Table 2: Results of task negation using the eight tasks presented in Section 4.2. We report the minimum accuracy on the target tasks while maintaining 95% of the pretrained model's accuracy on control tasks (note: results in $(\cdot)$ are reference values where control task performance did not exceed 95% of the pretrained model's accuracy). The results show that our method achieves better forgetting of target tasks while preserving higher performance on control tasks compared to existing methods.

5.3 Scalable Regularization in Practical Applications

First, in situations where tasks are introduced incrementally, similar to incremental learning, we demonstrate that comparable performance can be achieved by applying regularization exclusively to previously learned tasks(Section 5.3.1). Then, we demonstrate that simply adding a few additional steps of regularization-based training to existing linearized task vectors yields significant improvements (Section 5.3.2).

5.3.1 Incremental Addition

In practical applications, scalability to new tasks is critical. Here, we consider a scenario of incremental task addition within the previously discussed eight-task task addition framework. Specifically, when training on a task $t \in T$, future tasks are not taken into account, and regularization is applied only with respect to past tasks, i.e., ($T_{\text{orth}} = \{1, 2, \ldots, t-1\}$).

Method	Abs. (↑)	Norm. (↑)
No reg. (Linear FT)	74.3	85.0
Incremental reg. (Ours)	83.6	96.5
Full reg. (Ours)	84.5	97.6

Table 3: Comparison of original regularization and the incremental regularization in task addition on ViT-B-32

Table 3 presents a comparison of task addition on ViT-B-32 using three approaches: applying regularization to all tasks (Full reg.), applying regularization incrementally (Incremental reg.), and Linear FT (No reg.). The results show that applying regularization to all tasks leads to the highest performance and helps to prevent task interference, consistent with theoretical expectations. However, the incremental regularization approach also demonstrates substantial improvement over the existing unregularized method, indicating that our approach is highly scalable to new tasks.

5.3.2 Penalization on a Existing Task Vector

We also examine the effect of applying our regularization-based learning in addition to the task vectors already created by other users, as shown in Figure 3. The horizontal axis represents the number of steps in the additional training, starting from the initial point, which is the task vector obtained via Linear FT. The left vertical axis (blue) shows the normalized accuracy during task addition, while the right vertical axis (red) represents τJp. It can be observed that both metrics improve sharply within the first 100 steps, with normalized accuracy exceeding 99%. Afterward, the improvement is more gradual. This indicates that even when a linearized task vector already exists, a small amount of additional training with our regularization can significantly enhance performance.

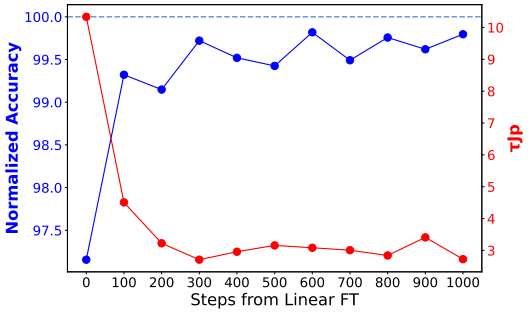

Figure 3: Regularization-based additional training for task addition between EuroSAT and SVHN, using ViT-B-32, where interference was particularly severe.

6 Limitations

Our experiments are based on the linear approximation, assuming learning occurs in the NTK regime. As noted by Ortiz-Jimenez et al. (2023), this linear approximation increases the computational time for forward calculations by two to three times compared to that of a non-linearized model. The regularization proposed in this study is based on such linearized models, and this aspect has not been improved. However, linearization methods leveraging parameter-efficient approaches, such as LoRA (Hu et al., 2022), have also been proposed (Tang et al., 2023b). Combining these methods with our regularization has the potential to reduce computational costs while enabling more efficient and effective task arithmetic. Our contribution lies in elucidating the internal structure of task arithmetic using τJp and confirming the sufficient effectiveness of our regularization under precise linearization. Validation on larger models (e.g., LLMs) using approximate linearization methods, such as those mentioned above, is left for future work.

7 Conclusion

In this paper, we proposed a novel metric, τJp, to better understand weight disentanglement in task arithmetic and demonstrated its inverse correlation with normalized accuracy. By incorporating regularization to minimize τJp during fine-tuning, we significantly reduced task interference, minimizing the need for coefficient adjustments in task addition and negation. In incremental learning, we found that our τJp regularization method shows strong performance in situations where future tasks to be

learned are unknown or accessible, confirming the scalability of the approach. Furthermore, the τJp regularizer improves the performance of task arithmetic by utilizing publicly available fine-tuned models, which makes it beneficial for practical use in real-world scenarios. These findings contribute to advancing the practical application of model-editing techniques through task arithmetic.

Acknowledgments and Disclosure of Funding

Our deepest gratitude goes out to the anonymous reviewers whose invaluable insights substantially enhanced the quality of this manuscript. We sincerely thank Yoshikazu Ikeda (ProPlace Inc, Osaka University) for his invaluable assistance in setting up the infrastructure, which greatly contributed to the success of this research. This research was supported by the GCP Startups Booster Program and Microsoft for Startups.

References

Ainsworth, S. K., Hayase, J., and Srinivasa, S. (2022). Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*. 3, 13

Caruana, R. (1997). Multitask learning. *Machine learning*, 28:41–75. 15

Cheng, G., Han, J., and Lu, X. (2017). Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883. 5

Choshen, L., Venezian, E., Slonim, N., and Katz, Y. (2022). Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*. 3, 13

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. (2014). Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613. 5

Davari, M. and Belilovsky, E. (2023). Model breadcrumbs: Scaling multi-task model merging with sparse masks. *arXiv preprint arXiv:2312.06795*. 2, 3, 13

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. 7

Don-Yehiya, S., Venezian, E., Raffel, C., Slonim, N., and Choshen, L. (2023). ColD fusion: Collaborative descent for distributed multitask finetuning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 788–806. 3, 13

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*. 5

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. (2022). The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*. 3, 13

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. (2020). Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the International Conference on Machine Learning*, pages 3259–3269. 3, 13

Gueta, A., Venezian, E., Raffel, C., Slonim, N., Katz, Y., and Choshen, L. (2023). Knowledge is a region in weight space for fine-tuned language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 3, 13

Helber, P., Bischke, B., Dengel, A., and Borth, D. (2019). Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226. 5, 17

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*. 9

Huang, C., Ye, P., Chen, T., He, T., Yue, X., and Ouyang, W. (2024). Emr-merging: Tuning-free high-performance model merging. *arXiv preprint arXiv:2405.17461*. 3, 13

Huang, Y., Zhang, Y., Chen, J., Wang, X., and Yang, D. (2021). Continual learning for text classification with information disentanglement based regularization. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y., editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2736–2746, Online. Association for Computational Linguistics. 13

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. (2023). Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*. 2, 3, 17

Ilharco, G., Wortsman, M., Gadre, S. Y., Song, S., Hajishirzi, H., Kornblith, S., Farhadi, A., and Schmidt, L. (2022). Patching open-vocabulary models by interpolating weights. *Advances in Neural Information Processing Systems*. 3, 5, 13, 14, 15

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 31. 2, 3, 4, 13

Jolicoeur-Martineau, A., Gervais, E., FATRAS, K., Zhang, Y., and Lacoste-Julien, S. (2024). Population parameter averaging (PAPA). *Transactions on Machine Learning Research*. 3, 13

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. 2

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526. 16

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561. 5

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*. 5, 18

Malladi, S., Wettig, A., Yu, D., Chen, D., and Arora, S. (2023). A kernel-based view of language model fine-tuning. In *Proceedings of the International Conference on Machine Learning*, pages 23610–23641. 4

Muqeeth, M., Liu, H., and Raffel, C. (2023). Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*. 3, 13

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., et al. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada. 5, 18

Ortiz-Jimenez, G., Favero, A., and Frossard, P. (2023). Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*. 2, 3, 4, 5, 9, 13, 14, 15, 16

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, New York, NY, USA. Association for Computing Machinery. 16

Qin, Y., Qian, C., Yi, J., Chen, W., Lin, Y., Han, X., Liu, Z., Sun, M., and Zhou, J. (2022). Exploring mode connectivity for pre-trained language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 6726–6746. 3, 13

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR. 5

Ramé, A., Ahuja, K., Zhang, J., Cord, M., Bottou, L., and Lopez-Paz, D. (2023). Model ratatouille: Recycling diverse models for out-of-distribution generalization. In *Proceedings of the International Conference on Machine Learning*, pages 28656–28679. 3, 13

Ren, Y., Guo, S., Bae, W., and Sutherland, D. J. (2023). How to prepare your task head for finetuning. *arXiv preprint arXiv:2302.05779*. 4

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2011). The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE. 5

Tang, A., Shen, L., Luo, Y., Ding, L., Hu, H., Du, B., and Tao, D. (2023a). Concrete subspace learning based interference elimination for multi-task model fusion. *arXiv preprint arXiv:2312.06173*. 3, 13

Tang, A., Shen, L., Luo, Y., Zhan, Y., Hu, H., Du, B., Chen, Y., and Tao, D. (2023b). Parameter efficient multi-task model fusion with partial linearization. *arXiv preprint arXiv:2310.04742*. 2, 3, 4, 9, 13

Utans, J. (1996). Weight averaging for neural networks and local resampling schemes. In *Proceedings of the AAAI Workshop on Integrating Multiple Learned Models*, pages 133–138. 3, 13

Villalobos, P., Sevilla, J., Besiroglu, T., Heim, L., Ho, A., and Hobbhahn, M. (2022). Machine learning model sizes and the parameter gap. 2

Wang, K., Dimitriadis, N., Ortiz-Jiménez, G., Fleuret, F., and Frossard, P. (2024a). Localizing task information for improved model merging and compression. In *Proceedings of the International Conference on Machine Learning*. 3, 13

Wang, L., Zhang, X., Su, H., and Zhu, J. (2024b). A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383. 13

Wang, X., Chen, T., Ge, Q., Xia, H., Bao, R., Zheng, R., Zhang, Q., Gui, T., and Huang, X. (2023). Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*. 13, 17

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., and Schmidt, L. (2022a). Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the International Conference on Machine Learning*, volume 162, pages 23965–23998. PMLR. 3, 13

Wortsman, M., Ilharco, G., Kim, J. W., Li, M., Kornblith, S., Roelofs, R., Lopes, R. G., Hajishirzi, H., Farhadi, A., Namkoong, H., et al. (2022b). Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971. 3, 13

Xiao, J., Ehinger, K. A., Hays, J., Torralba, A., and Oliva, A. (2016). Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119:3–22. 5

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. (2023). TIES-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*. 2, 3, 13

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. (2024). Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the International Conference on Machine Learning*. 2, 3, 13

Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., Zhang, K., Ji, C., Yan, Q., He, L., Peng, H., Li, J., Wu, J., Liu, Z., Xie, P., Xiong, C., Pei, J., Yu, P. S., and Sun, L. (2023). A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. 2

A Related Work

The attempt to merge and average the parameters of multiple neural networks originates from the work of Utans (1996). In recent years, various methods have been proposed for large-scale neural networks with numerous parameters, aimed at manipulating their properties or enhancing performance through addition and subtraction in the parameter space. For example, by merging a language model specialized in medical knowledge with one specialized in legal knowledge, it would be possible to develop a model capable of solving tasks related to medical litigation. Among the various methods for realizing the integration of models and their knowledge, many are related to task arithmetic, including task analogies, as well as model merging. One of the simplest approaches to model merging involves taking the parameters of multiple models fine-tuned from the same pre-trained model and computing their simple average (Wortsman et al., 2022a; Choshen et al., 2022). Building on this, various extensions have been proposed. For instance, Don-Yehiya et al. (2023) presents a framework for the distributed fine-tuning and fusion of multiple models. Ramé et al. (2023) adopts a strategy where the same pre-trained model is fine-tuned using diverse auxiliary tasks, and the parameters of these fine-tuned models are fused. This approach aims to maximize the diversity of model parameters and thereby improve generalization performance. Jolicoeur-Martineau et al. (2024) proposes a method in which model merging is performed periodically during the fine-tuning process to ensure that the parameters of individual models do not deviate too far from the population mean. Muqeeth et al. (2023) introduces a technique in the context of Mixture-of-Experts (MoE), where a merged expert is created by computing the weighted average of parameters across multiple expert networks. Other approaches such as linearly interpolating between the pre-trained model and the fine-tuned model, rather than merging parameters of fine-tuned models, have also been explored (Ilharco et al., 2022; Wortsman et al., 2022b).

On the other hand, in the integration of models via model merging or task arithmetic, interference between the parameters of multiple models or task vectors can arise, and various methods have been proposed to mitigate such conflicts. For instance, several methodologies utilize masking operations on task vectors (Tang et al., 2023a; Wang et al., 2024a; Huang et al., 2024), while others involve trimming or scaling techniques (Yadav et al., 2023; Davari and Belilovsky, 2023; Yu et al., 2024), or leverage model linearization (Tang et al., 2023b; Ortiz-Jimenez et al., 2023). Additionally, in incremental learning (Wang et al., 2024b), Huang et al. (2021) and Wang et al. (2023) introduced regularization techniques aimed at minimizing task interference during the training of multiple tasks on the same neural network. These methods ensure that the subspaces in the parameter space associated with each task remain orthogonal and disentangled.

Theoretical and analytical studies on the effectiveness of model merging and task arithmetic include research based on analyses of the loss landscape (Entezari et al., 2022; Qin et al., 2022; Gueta et al., 2023), as exemplified by linear mode connectivity (Frankle et al., 2020), as well as approaches that leverage model linearization within the Neural Tangent Kernel (NTK) regime (Jacot et al., 2018). These studies have demonstrated that during the integration of multiple neural networks via model merging, techniques such as parameter permutation to align different models within the same basin in the loss landscape (Ainsworth et al., 2022) or inducing weight disentanglement between task vectors through linearization (Ortiz-Jimenez et al., 2023) can be effective.

B Scaling Coefficients for Task Vectors in the NTK Regime

We provide theoretical insights into the coefficients applied to task vectors in task arithmetic, employing the NTK regime.

B.1 Theoretical insights

First, as a preliminary step, we present the following two important theorems.

Theorem 1. *In the weight space $\mathbb{R}^p$, let $\theta_0 \in \mathbb{R}^p$ denote the initial point and $\theta^\star \in \mathbb{R}^p$ the fine-tuned point. For any scalar $\alpha \in \mathbb{R}$, define a point on the straight line passing through θ_0 and $\theta^\star$ as:*

$$\theta(\alpha) = (1-\alpha)\theta_0 + \alpha\theta^\star.$$

Under the NTK regime, the model's output can be approximated by:

$$f(x;\theta(\alpha)) \approx (1-\alpha)f(x;\theta_0) + \alpha f(x;\theta^\star).$$

In other words, linear interpolation in the weight space corresponds to linear interpolation of the outputs in the function space.

Proof. Noting Eq.(3) and that $f(x;\theta^\star) \approx f(x;\theta_0) + \tau^\top \nabla_\theta f(x;\theta_0)$ based on it, we obtain the following:

$$\begin{aligned} f(x;\theta(\alpha)) &= f\big(x;(1-\alpha)\theta_0 + \alpha\theta^\star\big) \\ &= f\big(x;\theta_0 + \alpha(\theta^\star - \theta_0)\big) \\ &= f(x;\theta_0 + \alpha\tau) \\ &\approx f(x;\theta_0) + \alpha\tau^\top \nabla_\theta f(x;\theta_0) \\ &\approx f(x;\theta_0) + \alpha\big(f(x;\theta^\star) - f(x;\theta_0)\big) \\ &= (1-\alpha)f(x;\theta_0) + \alpha f(x;\theta^\star) \end{aligned}$$

□

Theorem 2. *Consider a convex loss function $L(f(x;\theta))$ with respect to the model output $f(x;\theta)$. Then, in the NTK regime, the loss function $L(f(x;\theta(\alpha)))$ is convex with respect to α.*

Proof. According to Theorem 1, in the NTK regime, $f(x;\theta(\alpha))$ is a linear interpolation between $f(x;\theta_0)$ and $f(x;\theta^\star)$, such that

$$f(x;\theta(\alpha)) = (1-\alpha)f(x;\theta_0) + \alpha f(x;\theta^\star).$$

Since the loss function L is convex with respect to the model output, the composite function $L(f(x;\theta(\alpha)))$ is convex with respect to $\alpha \in \mathbb{R}$. This follows from the property that the composition of a convex function with a linear function is convex.

Specifically, because L is convex and $f(x;\theta(\alpha))$ is a linear function of α, the function $L(f(x;\theta(\alpha)))$ is convex with respect to α. □

Finally, based on Theorem 2, we can provide the following explanations for the coefficients in task addition and negation of the linearized model.

Task addition. If the $\theta^\star$ obtained through finetuning for each task is optimal (i.e., minimizes the loss), then, according to Theorem 2, the loss is minimized at $\alpha = 1.0.0$. Furthermore, if all τJp are zero, setting the task-specific coefficients $\alpha_1 = \alpha_2 = \cdots = \alpha_T = 1.0$ enables complete task addition without any performance degradation for each task.

Task negation. If the $\theta^\star$ obtained through finetuning for a particular task is optimal (i.e., minimizes the loss), then the loss decreases monotonically in the direction from θ_0 towards $\theta^\star$ along τ. Conversely, moving in the direction of $-\tau$ leads to an increase in loss (i.e., forgetting occurs). This is because the loss is convex with respect to α. Therefore, in this case, optimal coefficients cannot theoretically be obtained, and as long as the NTK regime holds, increasing α indefinitely in the negative direction results in greater forgetting.

C Implementation Details

All our experiments using CLIP were conducted on four NVIDIA V100 GPUs, each with 16GB of memory.

C.1 Finetuning Details

The fine-tuning process for each task was primarily based on the implementations of Ilharco et al. (2022); Ortiz-Jimenez et al. (2023). Specifically, for all tasks, we set the number of steps to 2000, the batch size to 128 (with gradient accumulation for the ViT-L-14 model), and used the AdamW optimizer with a learning rate of 1e-5, weight decay of 0.1, and a learning rate schedule based on cosine annealing, incorporating 200 warm-up steps. As noted by Ilharco et al. (2022); Ortiz-Jimenez et al. (2023), freezing the text encoder during the fine-tuning of CLIP does not significantly impact final performance, so we adopted a fixed classification head by using the output of the pre-trained

text encoder on class-specific text prompts (e.g., "$a\ photo\ of\ \{\text{classname}\}$"), while fine-tuning only the image encoder. For the fine-tuning of the linearized model, we followed the exact implementation outlined in Ilharco et al. (2022).

The computation of the τ-Jacobian product for the proposed regularization was performed efficiently using Jacobian-vector products, as in Ortiz-Jimenez et al. (2023). Due to computational constraints, hyperparameter tuning for the penalty coefficient λ was performed on a single dataset (Cars) using the values [1e-0, 1e-1, 1e-2]. The optimal value selected from this tuning process, $\lambda = 0.1$, was then applied consistently across all other datasets.

C.2 Task Vector Coefficients

In the CLIP experiments, the coefficients for the models with Non-lin. FT, Linear FT, and Ours (Tuned), which required coefficient adjustment, were standardized across all task vectors. Specifically, in Eq. (4), we set $\alpha_1 = \alpha_2 = \cdots = \alpha_T$. For task addition, the coefficient search range was set to $\alpha \in \{0.0, 0.05, \ldots, 1.0\}$, and for task negation, the range was $\alpha \in \{0.0, 0.1, \ldots, 3.0\}$.

As demonstrated in Appendix B, under the NTK regime, theoretically, if there is no interference between task vectors, an optimal coefficient of $\alpha = 1.0$ should be achieved for task addition. However, for task negation, the NTK approximation theoretically allows α to grow arbitrarily large within the valid approximation range. Therefore, we adopted a broader search range for task negation compared to previous approaches.

For coefficient selection, in task addition, we chose the coefficient that yielded the highest normalized accuracy on the validation split. For task negation, we selected the coefficient that achieved the lowest accuracy while still maintaining at least 95% of the pre-trained model's accuracy on the control task (ImageNet) validation split.

D Comparison of Strict Regularization and Cyclical Regularization

Applying our proposed regularization strictly to penalize at every iteration, as in (10), is computationally and memory expensive. Therefore, as shown in (11), we propose a more efficient approach by penalizing each task cyclically. Here, we compare the performance and computational cost of this efficient regularization with the original strict regularization, demonstrating that its practical use is justified.

Table 4 presents the results of comparing the two approaches in task addition using ViT-B-32. The comparison metrics include absolute accuracy, normalized accuracy, and the actual time taken per iteration. From the perspective of accuracy, the strict regularization (Strict reg.) slightly outperforms the efficient implementation (Cyclical reg.), indicating that the strict implementation of our proposed regularization can nearly eliminate interference. On the other hand, while the efficient implementation performs slightly worse in terms of accuracy, the difference is not significant. Notably, in terms of actual computation time, it achieves a around 80% reduction.

Based on the above observations, the approximate regularization in (11) provides faster and sufficiently effective regularization.

Method	Abs. (↑)	Norm. (↑)	Sec. / Iter.(↓)
Cyclical reg. (11)	84.5	97.6	0.374
Strict reg. (10)	86.4	99.3	2.027

Table 4: Comparison of the strict regularization and the efficient regularization in task addition on ViT-B-32

E Task Arithmetic and Multi Task Learning

Multi-Task Learning (MTL) (Caruana, 1997) involves training a single model simultaneously on data from multiple tasks. When sufficient input data and labels are available for each target task,

leveraging them concurrently enables the construction of a unified model capable of handling multiple tasks effectively.

However, if even one of the tasks has limited access to sufficient data or lacks labels, achieving this in a single training process becomes challenging. Additionally, adding new capabilities to a pre-trained model while maintaining its performance on other tasks (Kirkpatrick et al., 2017), or forgetting harmful abilities, is not a straightforward task.

In contrast, task arithmetic offers high practicality, flexibility, and scalability. Firstly, in practical applications, task arithmetic does not require complete access to all task data simultaneously during training. Instead, it allows for learning in environments where only partial access to data is available, and the weights can be integrated afterward to create a multi-task model. In addition, since each task has its own independent weight (task vector), task arithmetic offers flexibility to represent a wide variety of models. For example, with task vectors for N tasks, it is possible to represent 2^N different models through task vector addition or negation. In the context of recent advancements such as Large Language Model (LLM)-based chatbots and multi-agent systems (Park et al., 2023), where diverse models are needed to adapt to various situations, the flexibility of task arithmetic is highly significant. Furthermore, as discussed in Section 5.3, our method can be easily extended in the context of continual learning while maintaining performance on previous tasks.

Method	ViT-B-32	ViT-B-16	ViT-L-14	x_{self}	y_{self}	x_{other}	y_{other}	Flexibility
Non-lin. FT	70.4	75.5	84.0	✓	✓	✗	✗	✓
Linear FT	74.3	78.7	85.5	✓	✓	✗	✗	✓
Ours	84.5	87.6	90.8	✓	✓	✓	✗	✓
MTL	87.8	90.8	92.6	✓	✓	✓	✓	✗

Table 5: Comparison of task addition and MTL. On the right side, the table shows the types of data required for training on task $t \in T$, as well as the flexibility of the model. Here, x_{self} represents the input data of the current task, y_{self} represents the labels of the current task, and x_{other} and y_{other} refer to data from tasks other than t. In MTL, training requires data that includes labels from all tasks, whereas task addition can be applied in more relaxed scenarios where MTL is not feasible. Flexibility indicates whether the model's performance can be easily modified for specific tasks after training. On the left side, the table shows the accuracy across eight tasks for each model scale, demonstrating that task addition using our regularization achieves performance comparable to MTL, even in more relaxed scenarios.

F Additional Results

Here, we present more detailed experimental results related to the discussions in the main text.

F.1 Single Task Accuracy on Each Task

Figure 4 presents the accuracy for each task using the three FT methods described in Section 5.2, along with the pre-trained model.

As noted by Ortiz-Jimenez et al. (2023), Non-linear FT outperforms the linearized FT methods (Linear FT and Ours) due to the non-linear advantage.

Notably, despite the regularization in our proposed method, which constrains learning to a subspace orthogonal to $\nabla_\theta f(x_t; \theta_0),\ t \in T_{\text{orth}}$, there is no degradation in performance compared to the original Linear FT. This demonstrates that our method successfully prevents task interference while maintaining performance by guiding learning in a space that mitigates inter-task interference.

F.2 Effect of Task Addition on Each Task

In Figure 5, we present the absolute and normalized accuracies for each task after task addition, comparing different methods. The right-hand plots of normalized accuracy demonstrate that our method not only achieves the highest accuracy across most tasks but also maintains consistent

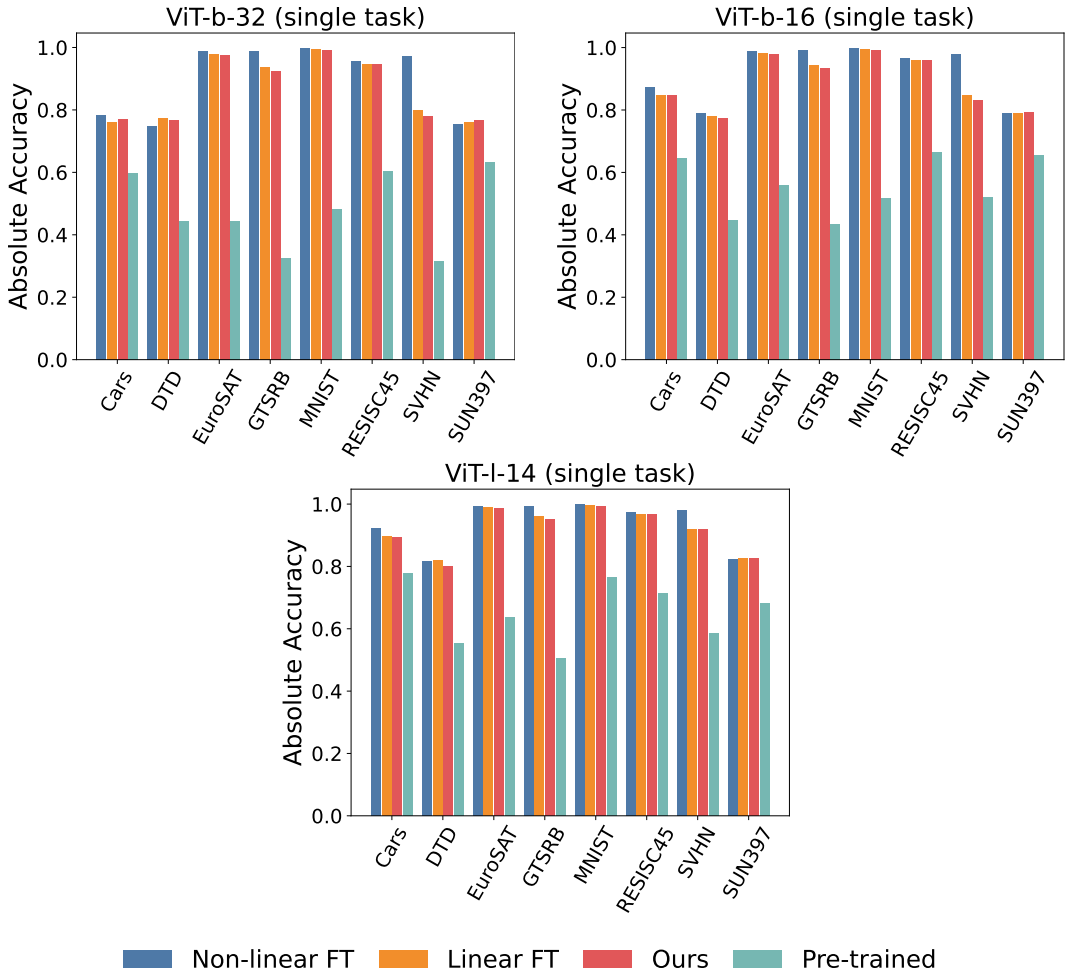

Figure 4: The absolute accuracy after fine-tuning for each of the eight tasks, comparing Non-linear FT (blue), Linear FT (orange), Ours (red), and the pre-trained model (green).

performance across all tasks, indicating that task-independent regularization is effectively achieved. Moreover, in the left-hand plots of absolute accuracy, our method outperforms existing methods on all tasks except for EuroSAT (Helber et al., 2019). These results suggest that our method successfully prevents interference between tasks while preserving absolute performance.

F.3 τJp on Each Task Pair

Figure 6 illustrates the τJp between task pairs for both Linear FT and Ours. These results demonstrate that our proposed regularization reduces the τJp between tasks. Compared to Linear FT, Ours shows a notably lower τJp in the off-diagonal components of the heatmap, i.e., between different datasets. These results suggest that our proposed method effectively decreases the τJp values.

F.4 Relationship between τJp and cosine similarity of task vectors

Interpreting similarity or interference between tasks in terms of cosine similarity of their task vectors has been a common practice (Ilharco et al., 2023; Wang et al., 2023). However, explanations for the interpretations remain limited and it is unclear whether cosine similarity fully accounts for those relationships between tasks. In this section, we attempt to analyze the relationship between task interference and task vector similarity through the lens of τJp.

We consider two tasks, A and B. Assuming that the fine-tuning of task B is conducted with a single update using the entire dataset, τ_B can be expressed as follows:

$$\begin{aligned}\tau_B &= \nabla_\theta L_B(f(x_B, \theta_0)) \\ &= \nabla_\theta f(x_B, \theta_0) \frac{\partial L_B}{\partial f(x_B, \theta_0)}\end{aligned} \tag{12}$$

The first equation above is derived from the fact that the loss function becomes convex with respect to the weights in the NTK regime (Theorem 2 in Appendix B). Using this expression, the cosine similarity can be rewritten as:

$$\begin{aligned}cos(\tau_A, \tau_B) &= \frac{\tau_A^\top \tau_B}{|\tau_A| \cdot |\tau_B|} \\ &= \frac{1}{|\tau_A| \cdot |\tau_B|} \tau_A^\top \nabla_\theta f(x_B, \theta_0) \frac{\partial L_B}{\partial f(x_B, \theta_0)}\end{aligned} \tag{13}$$

In the above, $\tau_A^\top \nabla_\theta f(x_B, \theta_0)$ is part of τJp, which, as we have demonstrated, explicitly affects task interference (or weight disentanglement) in the model. Although $\tau_A^\top \nabla_\theta f(x_B, \theta_0)$ is included in the cosine similarity, based on the equation, the presence of other components also affects their relationship, making it difficult to claim a theoretical correlation between them.

Figure 7 shows the cosine similarity between task pairs for both Linear FT and Ours. In Linear FT, the cosine similarity between MNIST (LeCun, 1998) and SVHN (Netzer et al., 2011) is particularly high, whereas in Ours, the values are much smaller and comparable to those of other task pairs. On the other hand, the cosine similarities between Cars and SVHN in Linear FT is higher than the ones in Ours. Therefore, no consistent trend was observed between cosine similarity and τJp.

In Figure 8, we present a scatter plot with τJp on the horizontal axis and the cosine similarity between the two task vectors on the vertical axis. Weak positive correlations were observed between these values in three model sizes. In particular, since cosine similarity tends to be small value when the number of dimension is large, the correlation is considered weak in the setting of ViT-L-14.

Based on these analysis, cosine similarity appears to be less effective in representing weight disentanglement compared to τJp. This implies that τJp regularization performs better than cosine similarity for reducing task interference.

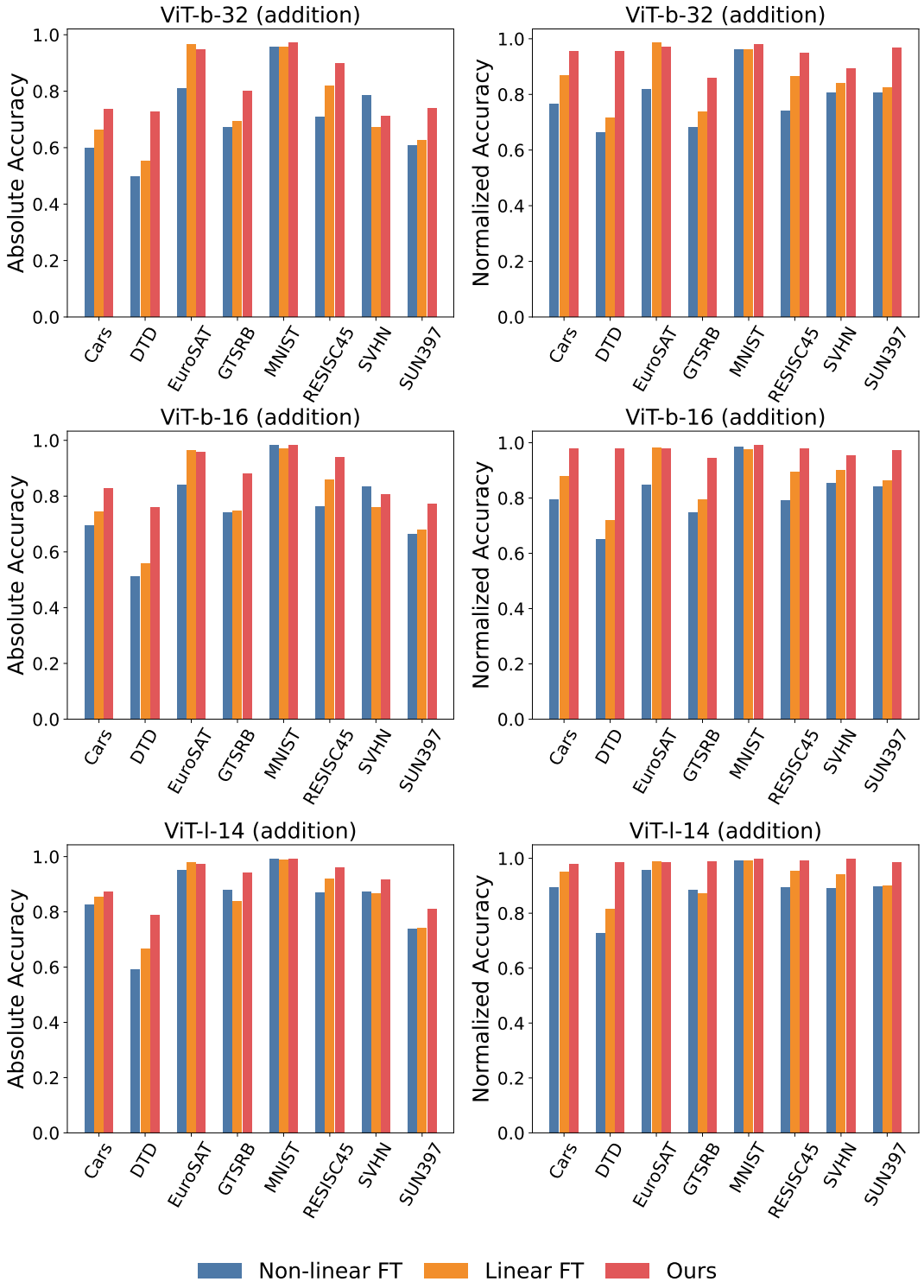

Figure 5: The absolute accuracy (left column) and normalized accuracy (right column) for each of the eight tasks after task addition, comparing Non-linear FT (blue), Linear FT (orange), and Ours (red).

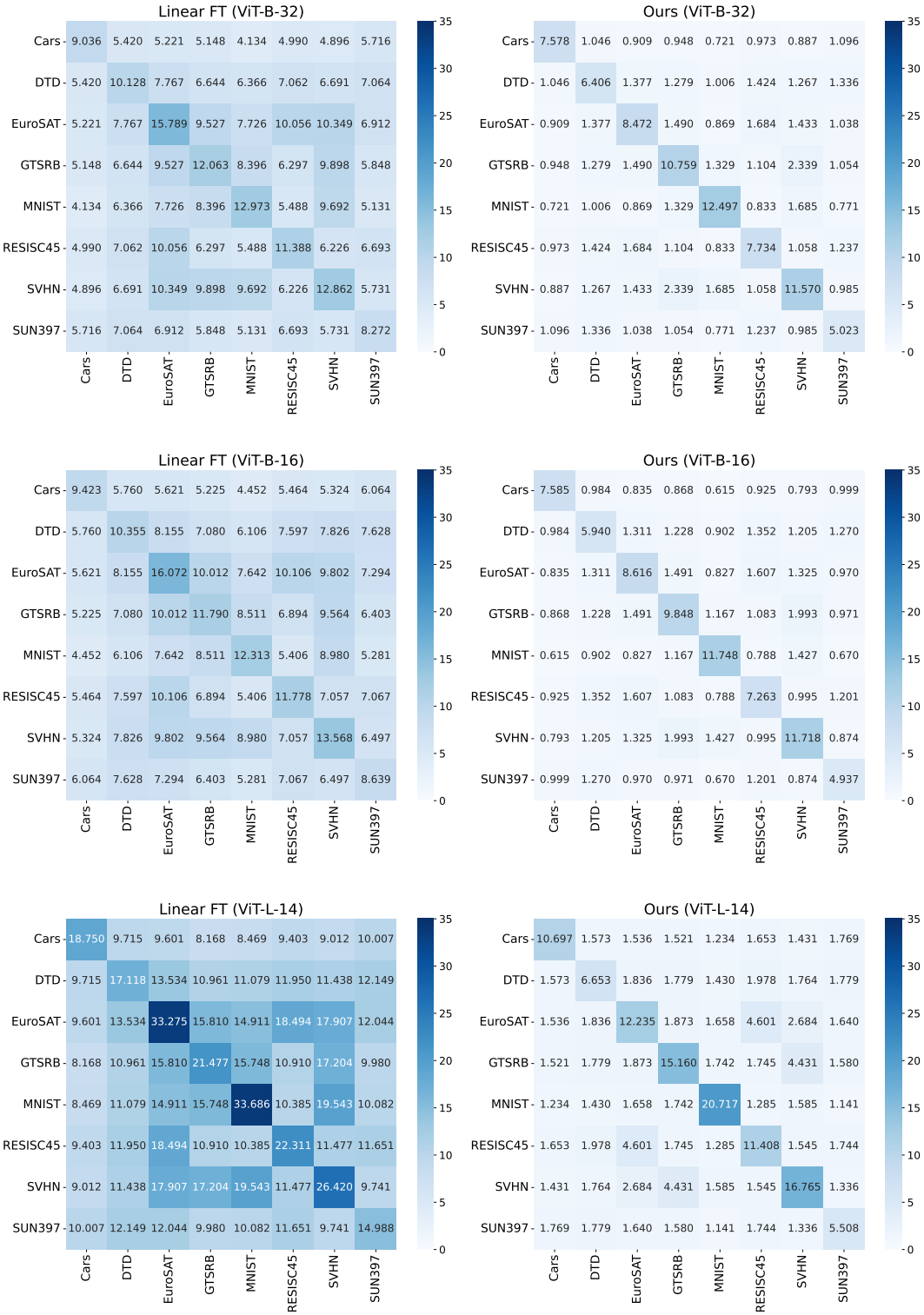

Figure 6: Heatmaps visualizing τJp on each task pair. The darker the color of the cell, the higher the value it represents. The values within cells indicates τJp. The figures in the left columns show the model with our proposed regularization, while the figures in the right columns show the existing linearized model without regularization. Our proposed regularization results lower τJp between different tasks.

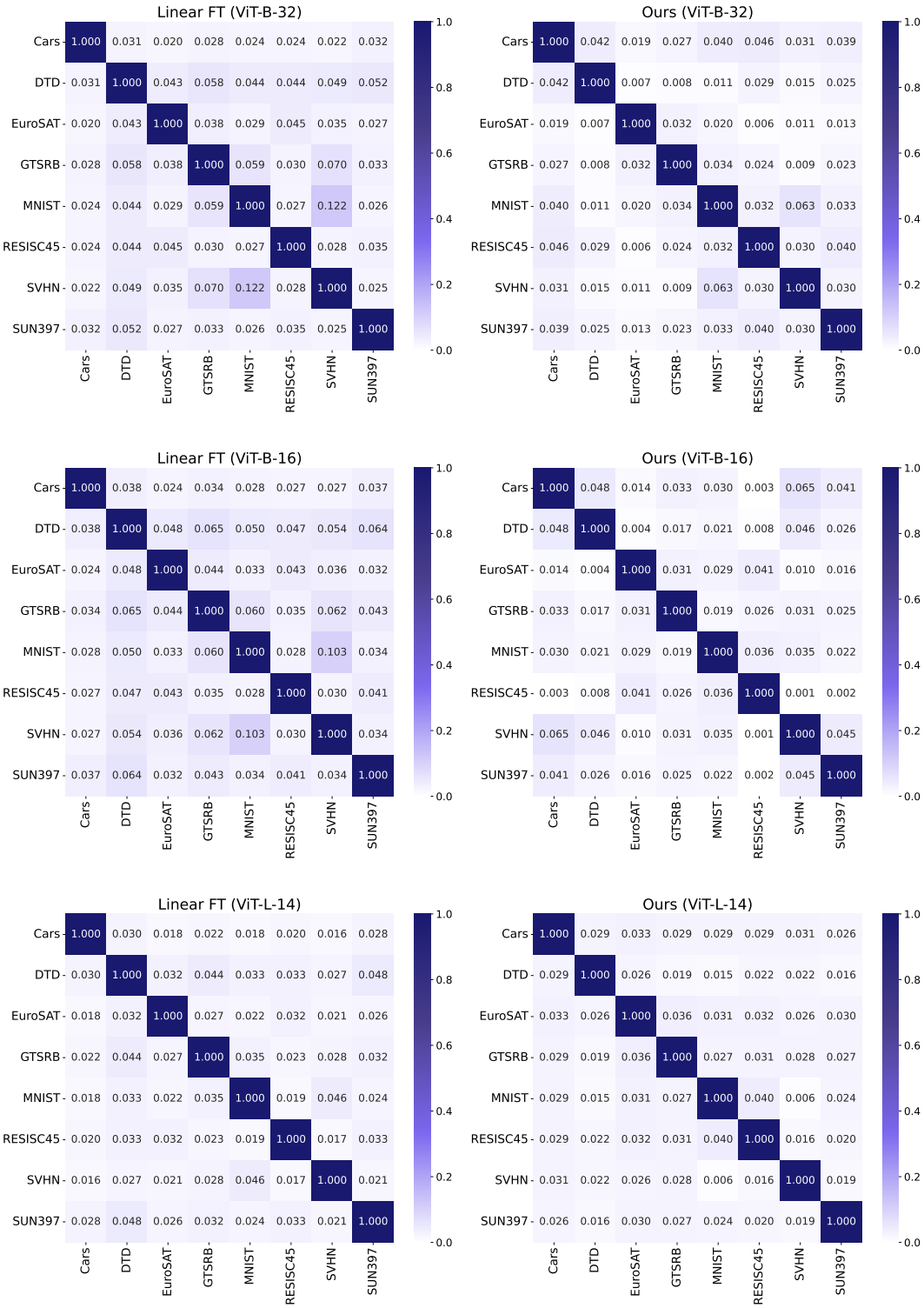

Figure 7: Heatmaps visualizing cosine similarity of taskvectors on each task pair. The darker the color of the cell, the higher the value it represents. The values within cells indicates cosine similarity. The figures in the left columns show the model with our proposed regularization, while the figures in the right columns show the existing linearized model without regularization.

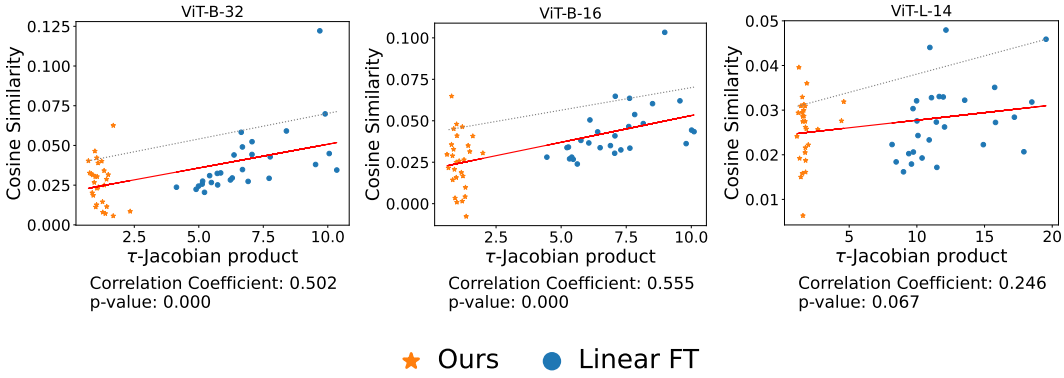

Figure 8: Visualization of the relationship between τJp and cosine similarity. Each point represents a pair of tasks from the set of eight tasks, yielding $\binom{8}{2}$ combinations, i.e., 28 in total. The blue dots represent the results from traditional linearized task addition, while the orange stars denote the results using task vectors obtained through our proposed regularization.