# Fine-Tuning Linear Layers Only Is a Simple yet Effective Way for Task Arithmetic

Ruochen Jin<sup>1,2</sup>, Bojian Hou<sup>2</sup>, Jiancong Xiao<sup>2</sup>, Weijie Su<sup>2</sup>, and Li Shen<sup>2</sup>

<sup>1</sup> East China Normal University, Shanghai, China,

<sup>2</sup> University of Pennsylvania, Philadelphia, PA, USA
{kyrie.jin, li.shen}@pennmedicine.upenn.edu

#### **Abstract**

Task arithmetic has recently emerged as a cost-effective and scalable approach to edit pre-trained models directly in weight space, by adding the fine-tuned weights of different tasks. The performance has been further improved by a linear property which is illustrated by weight disentanglement. Yet, conventional linearization methods (e.g., NTK linearization) not only double the time and training cost but also have a disadvantage on single-task performance. We propose a simple yet effective and efficient method that only fine-tunes linear layers, which improves weight disentanglement and efficiency simultaneously. Specifically, our study reveals that only fine-tuning the linear layers in the attention modules makes the whole model occur in a linear regime, significantly improving weight disentanglement. To further understand how our method improves the disentanglement of task arithmetic, we present a comprehensive study of task arithmetic by differentiating the role of representation model and task-specific model. In particular, we find that the representation model plays an important role in improving weight disentanglement whereas the task-specific models such as the classification heads can degenerate the weight disentanglement performance. Overall, our work uncovers novel insights into the fundamental mechanisms of task arithmetic and offers a more reliable and effective approach to editing pre-trained models <sup>1</sup>.

# 1 Introduction

The emergence of large pre-trained models in the open-source community maximizes the potential to boost performance on downstream tasks [1, 2, 3], align with human preferences [4, 5, 6, 7, 8], and enhance robustness [9, 10, 11, 12]. Traditional methods involve expensive joint fine-tuning across various tasks [3] and reliance on human feedback [5], limiting their scalability and widespread adoption. Moreover, optimizing performance for specific downstream tasks usually compromises the model's initial pre-training performance or zero-shot accuracy [13, 14].

Based on the extensive resources of open-source models, we often aim to edit the pre-trained models without requiring access to additional training data, in order to improve performance on multiple downstream tasks. *Task arithmetic* [1] is a method proposed for this goal. The central to task arithmetic is a concept called the *task vector* that enables models to adapt to new tasks [1]. A task vector can be viewed as a set of weight adjustments specifically calibrated for a given task through fine-tuning, obtained by subtracting the task-specific weights from the original pre-trained weights. Essentially, each task vector encodes a unique representational signature tailored to a particular task. The illustration of task arithmetic is shown in Figure 1. Recent research in this domain, regarding task vector-centric approaches [1, 15, 16, 17, 18], has demonstrated that by aggregating multiple task

<sup>&</sup>lt;sup>1</sup>The code is available at https://github.com/kyrie-23/linear\_task\_arithmetic.

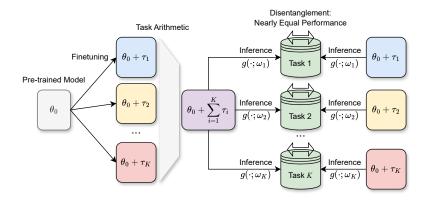


Figure 1: Illustration of task arithmetic, that is adding  $\tau_t$  does not modify the output of model outside  $D_t$ . This allows a model to manipulate parameters independently by adding linear combinations of  $\tau_t$  to a pre-trained checkpoint  $\theta_0$ .  $g(\cdot; \omega_k)$  refers to kth task-specific model parameterized by  $\omega_k$ .

vectors and integrating them into a pre-trained model, it is possible to create a new model, which we refer to as a unified model, and adapt it to multi-task learning.

However, task arithmetic from non-linear fine-tuning is not without limitations. For each individual task, although the unified model shows some improvement over the pre-trained model, its performance is usually not comparable to that of a model specifically trained for that task. This is because a task vector for one particular task usually has a negative effect on the performance of the other tasks. Therefore, the first question of this paper arises:

#### Question 1: How can we improve the performance of task arithmetic?

Ideally, Question 1 is supposed to be resolved by weight disentanglement, where the unifined model's performance should not be affected by other tasks when it is applied to one specific task (illustrated in the right handside of Figure 1). However, weight disentanglement is the most challenging problem in task arithmetic. It has been found that enforcing models to be fine-tuned in the tangent space significantly improves weight disentanglement, thanks to the fact that models inherently operate in a *linear regime* [17]. Here the linear regime refers to the behavior of neural networks during the beginning of the fine-tuning phase, where the network updates occur primarily around the pretrained parameter initialization. This phenomenon is formalized by the neural tangent kernel (NTK) theory [19]. While NTK linearization is effective, it requires two to three times more computational resources and doubles the memory footprint compared to its nonlinear counterpart. The additional cost of NTK linearization violates the original goal of task arithmetic we described above. As a result, Question 1 boils down to the following:

#### Question 2: How to improve the disentanglement and efficiency of task arithmetic simultaneously?

To answer Question 2, we aim to propose a novel fine-tuning method with the following two properties. For disentanglement, the fine-tuning method is supposed to operate better in the linear regime. For efficiency, the fine-tuning method should only be applied to a part of the network to reduce computational cost. Hence, we propose a simple yet efficient and effective method that is to fine-tune only linear layers to achieve both of the two goals simultaneously. Specifically, our study reveals that only fine-tuning the linear layers can make the attention modules occur in a linear regime, significantly improving both weight disentanglement and accuracy compared to both nonlinear counterparts and NTK linearization [17] (see Table 1). Meanwhile, our method is much more efficient, as it fine-tunes only a small portion of layers, thereby reducing the computational burden and memory usage. Additionally, our approach mitigates the non-linear advantage where non-linear fine-tuning consistently achieves the highest accuracy, offering a balanced and efficient alternative.

To further understand how our method improves the disentanglement of task arithmetic, we present a study by differentiating the role of representation model and task-specific model, while existing literature [17] formulated task arithmetic using a single model without clearly differentiating them. We

conduct a comprehensive study of task arithmetic in contrastively pre-trained vision-language (ViT) models like CLIP [20], providing new insights into its fundamental mechanisms and proposing novel methods to improve the performance of pre-trained models through task arithmetic. Specifically, we illustrate that the representation model plays an important role in improving weight disentanglement whereas this has been constrained by task-specific models, such as classification heads.

Notably, we demonstrate that the attention module lies in a strong linear regime within its linear layers. Without any prior knowledge, we can achieve superior performance to the current state-of-the-art methods by simply fine-tuning all linear layers in the attention module. However, the performance can either improve or degrade depending on whether the bias parameters are fine-tuned. The best results are obtained when the settings align closely with those used in LoRA [21], which fine-tunes only Q, K, V, and output projection weights, indicating that this aspect warrants further exploration.

In particular, our main contributions are as follows:

- We propose a simple yet effective and efficient method that only fine-tunes linear layers, which improves weight disentanglement and multi-task performance up to 2.38% improvement compared to the state-of-the-art methods and 8.37% over the nonlinear baseline on several vision-language benchmarks.
- We demonstrate that fine-tuning all linear layers within the attention module occurring in a linear regime, without selectively freezing or omitting certain layers achieves superior performance compared to current state-of-the-art methods on this benchmark task.
- We reformulate the architecture of task arithmetic in [17] into a representation model and several task-specific models, aligning our notation with previous work. This allows us to more clearly illustrate their individual contributions.
- We illustrate that the representation model plays an important role in improving weight disentanglement, whereas the effectiveness of task arithmetic has been constrained by the task-specific models, such as classification heads.

Overall, our work provides new insights into the fundamental mechanisms of task arithmetic, enhancing the reliability and scalability of model editing. Our findings indicate that fine-tuning pre-trained models, particularly focusing on the orthogonality of task vectors, deserves further investigation due to its significant potential for improving model editing effectiveness. These insights can lead to the development of more efficient and accurate model editing techniques, enabling practitioners to adapt pre-trained models to a wider array of tasks.

#### 2 Task arithmetic is a reflection of weight disentanglement

Existing literature [17] formulated task arithmetic using a single model, which actually combines a representation model (e.g., CLIP) with several task-specific models (e.g., classification heads). This combination leads to different pre-trained parameters  $\theta_0$  for each task due to the use of different task-specific models (e.g., different classification heads for different datasets). The implementation of task arithmetic focuses on the representation model, extracting task vectors, fine-tuning, and replacing different task-specific models.

To better understand the relationship between the representation model and the task-specific models, we separate their definitions. This allows us to more clearly illustrate their individual contributions to task arithmetic. We formulate the task arithmetic property along with weight disentanglement, providing distinct definitions for the representation and task-specific models while aligning our notation with previous work.

Let  $F: \mathcal{X} \times \Theta \to \mathcal{Y}$  be a neural network taking inputs  $x \in \mathcal{X}$  and parameterized by a set of weights  $\vartheta \in \Theta$ , which consists of a representation model  $f(\cdot;\theta)$  and a task-specific model  $g(\cdot;\omega)$  where  $\vartheta = \{\theta,\omega\}$ . We will assume  $\mathcal{X} \subseteq \mathbb{R}^d$ ,  $\Theta \subseteq \mathbb{R}^m$  and  $\mathcal{Y} \subseteq \mathbb{R}^c$ . Consider T tasks, with every task t consisting of a triplet  $(D_t,\mu_t,F_t^*)$ , where  $D_t \subseteq \mathcal{X}$  is a data support (e.g., ImageNet [22] images),  $\mu_t$  an input distribution such that  $\operatorname{supp}(\mu_t) = D_t$ , and  $F_t^*: D_t \to \mathcal{Y}$  a target function (e.g., labels). In practice, each task is identified with a training set  $\{(x_v,F_t^*(x_v))\}_{v\in[n]}$  where  $F_t^*(x_v) = g(f(x_v;\theta_t^*);\omega_t)$  with  $x \sim \mu_t$ , that is used to fine-tune the representation models starting from the pre-trained weights  $\theta_0$  and obtain the fine-tuned weights  $\theta_t^*$ , while the task specified models are fixed at  $\omega_t$ .

**Task arithmetic.** Let the task vector of task t be the difference between the fine-tuned and the pretrained weights, i.e.,  $\tau_t = \theta_t^* - \theta_0$ . The following property formalizes the notion of task arithmetic introduced in Ortiz-Jimenez et al. [17], where the authors observed that the accuracies of pre-trained models on different datasets can be modified independently through the addition or removal of task vectors.

**Property 1 (Task arithmetic)** Consider a set of task vectors  $T = \{\tau_t\}_{t \in [T]}$  with associated non-intersecting task supports  $D = \{D_t \subset \mathcal{X}\}_{t \in [T]}$ , i.e.,  $\forall t, t' \text{ if } t \neq t' \text{ then } D_t \cap D_{t'} = \emptyset$ . We say a network F satisfies the task arithmetic property around  $\vartheta_0$  with respect to T and D if

$$F\left(x;\vartheta_0 + \sum_{t=1}^T \alpha_t \tau_t\right) = \begin{cases} F(x;\vartheta_0 + \alpha_t \tau_t) & \text{if } x \in D_t, \\ F(x;\vartheta_0) & \text{if } x \notin \bigcup_{t=1}^T D_t, \end{cases} \tag{1}$$

where  $\vartheta_0 = \{\theta_0, \omega_t\}$  and  $\vartheta_0 + \alpha_t \tau_t = \{\theta_0 + \alpha_t \tau_t, \omega_t\}$  in certain task t, with  $(\alpha_1, \dots, \alpha_T) \in \mathcal{A} \subseteq \mathbb{R}^T$ .

In short, a model satisfies Property 1 if adding  $\tau_t$  does not modify the output of the model outside  $D_t$ .

**Property 2 (Weight disentanglement)** A parametric function  $F: \mathcal{X} \times \Theta \to \mathcal{Y}$  is weight disentangled with respect to a set of task vectors  $T = \{\tau_t\}_{t \in [T]}$  and the corresponding supports  $D = \{D_t\}_{t \in [T]}$  if

$$F\left(x;\vartheta_0 + \sum_{t=1}^T \alpha_t \tau_t\right) = F(x;\vartheta_0 + \alpha_t \tau_t) \mathbb{1}(x \in D_t) + F(x;\vartheta_0) \mathbb{1}\left(x \notin \bigcup_{t \in [T]} D_t\right).$$

From Property 1 and Property 2 we can see that weight disentanglement is not necessarily linked to performance on various tasks. In other words, a model can achieve weight disentanglement relative to a group of task vectors but still underperforms on a given task. For example, if  $f(\cdot; \theta_0 + \alpha \tau)$  does not generalize well for some  $\alpha$ , the model may still fail to perform effectively despite being weight disentangled [17].

# 3 Fine-tuning linear layers only is much more efficient than NTK linearization

The objective of this work is to illustrate the relationship between the linear regime while fine-tuning ViT models (attention module) and task arithmetic performance. Existing work [17] demonstrated that the linear regime is not necessary for task arithmetic, but models within the linear regime exhibit superior disentanglement performance.

We have seen that linearized models are more weight-disentangled than non-linear ones[17]. However, NTK linearization often degrades single-task performance, which is demonstrated as a non-linear advantage. In this study, we show that fine-tuning only the linear layers significantly improves task arithmetic performance by reducing the single-task accuracy gap. This approach maintains the benefits of weight disentanglement while enhancing the overall effectiveness of task arithmetic across various settings.

Neural tangent kernel. Around the initialization weights  $\theta_0$ , a representation model  $f(x; \theta)$  can be approximated with a first-order Taylor expansion:

$$f(x;\theta) = f(x;\theta_0) + (\theta - \theta_0)^{\top} \nabla_{\theta} f(x;\theta_0) + \text{higher order terms.}$$
 (2)

This approximation is equivalent to the *neural tangent kernel (NTK)* [19],  $k_{\text{NTK}}(x, x') = \nabla_{\theta} f(x; \theta_0)^{\top} \nabla_{\theta} f(x'; \theta_0)$ , and defines a neural tangent space in which the relationship between weights and functions is linear. However, this linear approximation is often invalid at finite widths, as the evolution of parameters during training is inadequately captured by Eq. (2). In such cases, training occurs in a non-linear regime. Conversely, often during fine-tuning, parameter evolution in many pre-trained models is frequently minimal, meaning that training does not exit the tangent space and Eq. 2 closely approximates the network behavior [17]. In such cases, training occurs in a *linear regime*.

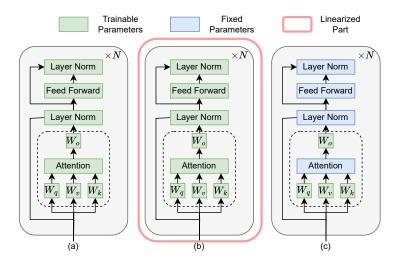


Figure 2: **Three types of fine-tuning paradigms.** (a) Full parameter fine-tuning where all the parameters will be updated. (b) Full-model linearization. (c) Linear layers only fine-tuning where only  $W_q$ ,  $W_v$ ,  $W_k$  and  $W_o$  will be updated. In this paper, we explore linear layers only fine-tuning.

Specifically, rather than constraining models to fine-tune within the tangent space as proposed by [17], we enhance the linear regime within the attention module, which naturally arises from the linear architecture. This enhancement leverages the inherent properties of linear layers to simplify the model's adaptation process. By fine-tuning only a select number of linear layers, the model not only lies in a pronounced linear regime but also maintains computational efficiency. This approach achieves performance levels comparable to full fine-tuning while significantly reducing the complexity and resources required.

Figure 2 illustrates three different types of fine-tuning paradigms. The first two are existing methods, while the third is our proposed partial linearization fine-tuning method. (a) The full fine-tuning paradigm where all parameters  $\theta$  are updated during fine-tuning. (b) The full-model linearization paradigm where we fine-tune the model in the tangent space. It is worth noting that although the Jacobian-vector products can be computed in a single forward pass [23], training and inference in this paradigm are usually twice or three times as expensive as full fine-tuning [18]. (c) The linear layers only fine-tuning paradigm in which only a small number of linear layers are updated, which exhibits a linear regime, which is similar to linear fine-tuning. This approach incurs only a fraction of the training and inference costs compared to NTK linearization.

The results in Figure 3 indicate that our method reduces the gap between linear and non-linear models and has a better performance than NTK linearization respectively<sup>2</sup>. Specifically, we fine-tune (FT) several CLIP pre-trained ViTs [24] of different sizes following the same setup as Ilharco et al. [1] on 8 tasks: Cars [25], DTD [26], SUN397 [27], EuroSAT [28], GTSRB [29], MNIST [30], SVHN [31] and RESISC45 [32]. The round dots represent the comparison between our method and non-linear models, while the triangle dots show the comparison between NTK linearization and non-linear models. The proximity of dots to the diagonal dashed line indicates the accuracy of the linearization method. Our method, represented by round dots, consistently appears closer to the diagonal dashed line than the NTK linearization (triangle dots), suggesting superior performance.

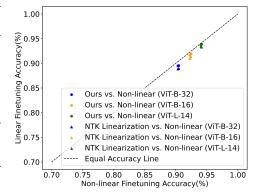


Figure 3: Single-task averaged accuracies of non-linear and linear models. The diagonal dashed line indicates linear fine-tuning performance meets non-linear.

<sup>&</sup>lt;sup>2</sup>Please see Appendix B.1 for performance on each task

Table 1: **Task addition.** Average absolute (%) and normalized accuracies (%) of different ViTs edited by adding the sum of the task vectors of 8 tasks. We report results for the non-linear and linearized models normalizing performance by their single-task accuracies.

Method	ViT-B-32		ViT-B-16		ViT-L-14	
	Abs.(†)	Norm.(†)	Abs.(†)	Norm.(†)	Abs.(†)	Norm.(†)
Pre-trained	48.40	-	55.25	-	66.40	-
Non-linear	70.00	77.04	74.75	80.59	84.40	89.47
Linear	76.26	85.82	79.01	86.32	85.53	91.44
Ours	78.37	87.42	80.44	87.25	87.91	93.66

Even though NTK linearization models exhibit a linear regime and achieve impressive disentanglement performance, they omit higher-order terms, leading to accumulated errors as the number of tasks increases. To address this, we propose leveraging linear layers to enhance disentanglement performance. Our method retains the benefits of NTK linearization while mitigating the drawbacks associated with neglected higher-order terms. By fine-tuning the linear layers within the attention module, we improve computational efficiency and robustness across multiple tasks.

Considering a completely linear representation model f, we can easily determine that  $\nabla_{\theta} f(x;\theta_0) = x$ . This foundational insight supports our approach, as the gradient with respect to the parameters simplifies to the input data itself. Thus, by fine-tuning only the linear layers, we enhance weight disentanglement and maintain model efficiency without extensive retraining, aligning with the concept of weight disentanglement to improve performance across various tasks.

**Proposition 1 (Orthogonality between Vectors and Data Points)** For a linear representation model f, the gradient with respect to the parameters is given by  $\nabla_{\theta} f(x; \theta_0) = x$ . Therefore, the model can be expressed as:

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

For a specific task t = i, the function f can be further expressed as:

$$f\left(x; \theta_0 + \sum_{t=1}^{T} \tau_t\right) = f(x; \theta_0 + \tau_i) + \sum_{t \neq i} \tau_t^{\top} x.$$

When  $\sum_{t \neq i} \tau_t^\top x \to 0$ , weight disentanglement is achieved.

Proposition 1 implies that the orthogonality of data points x and task vectors  $\tau_t$  leads to perfect disentanglement. This allows for the evaluation of cosine similarity between data points and task vectors, providing a clear advantage over NTK linearization, where such perfect disentanglement is not guaranteed.

Inspired by LoRA [21] which should be considered as a special form of linearization, we choose the matrix of Q, K, V, and the output projection. Concentrating on attention modules, give an example of a single attention layer as applied in Vaswani et al. [33], due to the linear property of the weight matrix, we can easily get a disentangle output from each attention head for a certain task i:

$$\begin{split} \text{head} &= \text{Attention}(qW^Q, kW^K, vW^V) \\ &= \text{Attention}(Q_i + \sum_{t \neq i} q_i W_t^Q, K_i + \sum_{t \neq i} k_i W_t^K, V_i + \sum_{t \neq i} v_i W_t^V) \\ &= \text{Attention}(Q_i, K_i, V_i), \end{split}$$

where q,k,v represent the query, key, value input,  $W^Q,W^K,W^V$  represent each weight, e.g.  $Q_i=q(W_0+W_i^Q),W^Q=W_0^Q+\sum_tW_t^Q$ .

Remarkably, as we show in Section 4, this increase in single-task performance does not compromise weight disentanglement, instead, it's much higher than NTK linearization. As a result, linear fine-tuning allows for improved task arithmetic compared to standard non-linear fine-tuning. To better

illustrate the multi-task performance, we employ the benchmark proposed by Ilharco et al. [1] to evaluate the task arithmetic ability of a pre-trained model, which consists of the 8 tasks described before: The sum of the task vectors  $\tau = \sum_t \tau_t$  is added to a pre-trained checkpoint to produce a multi-task model. The success of this benchmark is measured in terms of the maximum average accuracy over the different tasks. Results are shown in Table 1. In contrast to recent work [16, 18], our method focuses on the original task arithmetic setting, evaluating multi-task performance and further examining weight disentanglement, which does not require access to all test data and is not in a parameter-efficient setting either.

To obtain the task vectors, we use the fine-tuned weights of different ViTs from before, and use a single mixing coefficient  $\alpha = \alpha_1 = \cdots = \alpha_T$  optimized separately for each fine-tuning paradigm to ensure a fair comparison. We provide all the details of this experiment in Appendix A.

In particular, Table 1 in its last rows shows that our method significantly outperforms its non-linear counterparts [1] and achieves state-of-the-art results on the task addition benchmarks. Our model achieves higher multi-task accuracies through task addition (up to 2.38% more). Additionally, our method not only outperforms on averaged accuracy but also on normalization accuracy.

In addition to input and output projection layers, can we improve disentanglement performance by fine-tuning all linear layers in the attention module, and how would performance go when bias parameters are also fine-tuned? To dig into this question, we conduct an ablation experiment with four paradigms: (1) only fine-tuning Q, K, V, and output layer weights (ours), (2) fine-tuning Q, K, V, and output layer weights and bias, (3) fine-tuning all linear layers weights in the attention module, (4) fine-tuning all linear layers weights and bias in the attention module.

Surprisingly, all four paradigms outperform NTK linearization in both performance and disentanglement, demonstrating that ViT models exhibit a strong linear regime within the linear layers of the attention module (see Table 2). However, the performance can either improve or degrade depending on whether the bias parameters are fine-tuned. The best results are achieved when the settings align closely with those used in LoRA, suggesting that this is an area deserving further exploration.

Table 2: Single-task and multi-task performance (%) on four different settings.

Paradigm	Single-task Accuracy(†)	Multi-task Abs.(↑) Norm.(↑)	
(1) (2)	<b>89.55</b> 89.48	<b>78.37</b> 77.71	<b>87.42</b> 86.79
(3) (4)	88.95 89.43	76.52 77.80	86.11 86.93

#### 4 Weight disentanglement emerges from representation model

Existing works demonstrate the effectiveness of task arithmetic primarily in the context of downstream tasks. However, the reliance on downstream tasks and task-specific models may limit the generalizability and scalability of these methods. To further illustrate the power of task arithmetic and broaden its applicability, we aim to investigate whether the representation model itself can satisfy Property 3 without the need for a task-specific model.

Unlike the approach outlined in Section 2, we redefine task arithmetic and weight disentanglement within the representation model itself. Our hypothesis is that pre-trained models can exhibit task arithmetic properties independently of downstream tasks, maintaining a similar representation through task arithmetic. By focusing on the representation model alone, we seek to show that the inherent properties of the pre-trained models are sufficient to support task arithmetic, potentially simplifying the process and broadening its applicability.

**Property 3 (Task arithmetic\*)** Consider a set of task vectors  $T = \{\tau_t\}_{t \in [T]}$  with associated non-intersecting task supports  $D = \{D_t \subset \mathcal{X}\}_{t \in [T]}$ , i.e.,  $\forall t, t' \text{ if } t \neq t' \text{ then } D_t \cap D_{t'} = \emptyset$ . We say a representation model f satisfies the task arithmetic property around  $\theta_0$  with respect to T and D if

$$f\left(x;\theta_0 + \sum_{t=1}^T \alpha_t \tau_t\right) = \begin{cases} f(x;\theta_0 + \alpha_t \tau_t) & \text{if } x \in D_t; \\ f(x;\theta_0) & \text{if } x \notin \bigcup_{t=1}^T D_t. \end{cases}$$

**Property 4 (Weight disentanglement\*)** A parametric function  $f: \mathcal{X} \times \Theta \to \mathcal{Y}$  is weight disentangled with respect to a set of task vectors  $T = \{\tau_t\}_{t \in [T]}$  and the corresponding supports

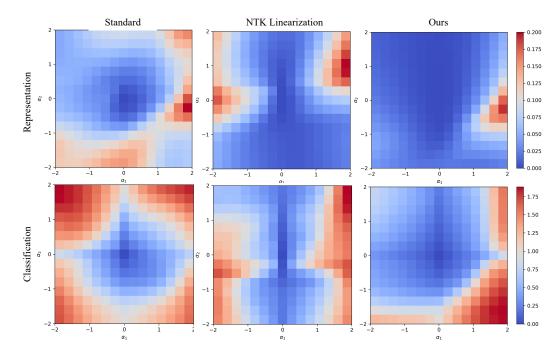


Figure 4: **Visualization of weight disentanglement.** The heatmaps show the disentanglement error  $\xi(\alpha_1,\alpha_2)$  of a single representation model CLIP ViT-B/32 (top) and a combination of representation model and classification model (bottom) on DTD - SUN397 task pair. Three fine-tuning paradigms are used from left to right: standard, NTK linearization, and ours. The light regions denote areas of the weight space where weight disentanglement is stronger.

$$D = \{D_t\}_{t \in [T]} \text{ if}$$

$$f\left(x; \theta_0 + \sum_{t=1}^T \alpha_t \tau_t\right) = \sum_{t=1}^T f(x; \theta_0 + \alpha_t \tau_t) \mathbb{1}(x \in D_t) + f(x; \theta_0) \mathbb{1}\left(x \notin \bigcup_{t \in [T]} D_t\right). \tag{4}$$

Instead of evaluating weight disentanglement on predictors, weight disentanglement is a property of the representation models and is not related to the performance on different tasks. That is, a model could be weight disentangled with respect to a set of task vectors and still perform poorly on a task, e.g., if  $f(\cdot; \theta_0 + \alpha \tau)$  does not generalize for some  $\alpha$ . More generally, we can visualize the level of weight disentanglement of a model by measuring its discrepancy with Eq. (4). To do so, given two tasks, one can check the disentanglement error of a model,

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

where dist denotes any distance metric between output vectors. As we are dealing with representation distributions, in what follows we use the KL divergence as the distance metric <sup>3</sup>. In general, the smaller the value of  $\xi(\alpha_1, \alpha_2)$  the more weight disentangled a model is at  $(\alpha_1, \alpha_2)$ .

Figure 4 displays the disentanglement error of a CLIP ViT-B/32 model concerning several task vector pairs. We observe that the ViT model exhibits a minimal disentanglement error within a small region surrounding  $\theta_0$ , which enables task arithmetic. Different from disentanglement error at downstream tasks, it remains relatively small even for  $\alpha_1, \alpha_2 > 1$ , which indicates the power of task arithmetic has been limited by the performance of task-specific models (classification head).

Comparing the disentanglement error of our models and NTK linearization reveals an interesting finding: linearized models exhibit greater disentanglement than their non-linear counterparts. This is evident from the more extensive regions with low disentanglement errors in Figure 4 (right). This

<sup>&</sup>lt;sup>3</sup>We use prediction error for the combined model for classification task as Ortiz-Jimenez did [17].

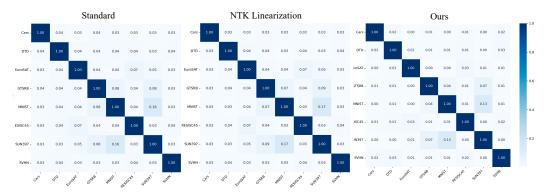


Figure 5: **Similarity heatmaps.** These figures show heatmaps of the cosine similarity between task vectors from task-specific CLIP models [20] fine-tuned on different tasks. Three fine-tuning paradigms from left to right: Full parameter fine-tuning (Standard), Full-model linearization (NTK linearization), and Linear layers only fine-tuning (Ours).

explains why the linear layers fine-tuning only models achieve higher normalized accuracies via task addition (cf. Table 1). The combination of greater disentanglement and better single-task performance comes with higher multi-task performance.

The results in Figures 5 show the cosine similarity between task vectors from ViT, which are three types of fine-tuning (cf. Figure 2) on image classification tasks. We observe that vectors from linear layers only fine-tuning are closer to orthogonal than those from both standard and NTK linearization, which indicates that models fine-tuned with full fine-tuning are more independent than others. This finding is consistent with the discussion about task addition in [1, 18], the experimental results from Table 1 also support our statement. The experimental details are described in Appendix.

#### 5 Related work

Weight Interpolation and Task Arithmetic. Recent research has explored the use of interpolations between model weights and task arithmetic to manipulate and enhance the capabilities of pre-trained models. Studies have shown that interpolating between a model's fine-tuned weights and its pre-trained initialization can improve performance on single tasks, often surpassing the accuracies achieved through fine-tuning alone [34, 35, 36, 37, 38, 39]. In multi-task settings, averaging the parameters of multiple fine-tuned models has been proposed to create superior multi-task models [40, 15, 41, 2, 1], which help avoid catastrophic forgetting [13, 14, 42, 43] and provide better starting points for subsequent fine-tuning [44, 45]. These benefits also extend to models trained from scratch, provided they are properly aligned before merging [46, 47]. This technique has been observed to enhance downstream performance, highlighting the potential of weight interpolation and task arithmetic.

**Model Fusion Techniques.** Model fusion integrates knowledge from multiple models into a single unified model. One approach focuses on fusing entire models through weight interpolation. By averaging or combining the weights of multiple models, effective model fusion can be achieved, as demonstrated in prior works [48, 2, 1, 36, 40, 49]. When models are not well-aligned or lie in different loss basins, feature alignment techniques are employed before fusion to match the models' behaviors or transform them to a similar loss basin [46, 50]. Although fusing entire models leverages knowledge from all layers, it can be computationally expensive.

Advances in Model Merging for MTL. Model merging has emerged as a promising solution to enhance model generalization and facilitate multi-task learning (MTL). Research in this area includes merging models trained on the same task to improve overall generalization [51, 52] or to perform federated learning [53, 54]. Another approach focuses on merging models for different tasks to enable MTL [36, 46, 55, 1, 15, 56, 57]. However, simple model averaging can significantly deteriorate performance across multiple tasks. To address this, advanced techniques have been developed. For example, Fisher Merging uses the Fisher information matrix to measure the importance of individual model parameters and guide model merging [36]. Although effective, computing the Fisher information matrix is computationally and memory-intensive.

#### 6 Conclusion

In this work, we conducted a thorough analysis of task arithmetic in deep neural networks, delving into its fundamental mechanisms and enhancing its performance. Our findings demonstrate that attention module lies in a strong linear regime within its linear layers, which improve both disentanglement and efficiency.

Understanding the nuanced impact of fine-tuning bias on model performance and disentanglement remains an open question. Future research could provide valuable insights into optimizing these settings, potentially leading to more robust and efficient methods for adapting pre-trained models to various tasks. This could significantly enhance their applicability and effectiveness in real-world scenarios.

#### References

- [1] Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.
- [2] Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [3] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 2020.
- [4] Ximing Lu, Sean Welleck, Liwei Jiang, Jack Hessel, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. Quark: Controllable text generation with reinforced unlearning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [5] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback, 2022.
- [6] Marco Tulio Ribeiro and Scott Lundberg. Adaptive testing and debugging of nlp models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.
- [7] Amelia Glaese, Nat McAleese, Maja Trebacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Sona Mokra, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. Improving alignment of dialogue agents via targeted human judgements. https://www.deepmind.com/blog/building-safer-dialogue-agents, 2022.
- [8] Jiancong Xiao, Ziniu Li, Xingyu Xie, Emily Getzen, Cong Fang, Qi Long, and Weijie J Su. On the algorithmic bias of aligning large language models with rlhf: Preference collapse and matching regularization. *arXiv* preprint arXiv:2405.16455, 2024.
- [9] Bo-Jian Hou, Lijun Zhang, and Zhi-Hua Zhou. Learning with feature evolvable streams. *Advances in Neural Information Processing Systems*, 30, 2017.
- [10] Guillermo Ortiz-Jiménez, Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Optimism in the face of adversity: Understanding and improving deep learning through adversarial robustness. *Proceedings of the IEEE*, 2021.
- [11] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [12] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

- [13] Robert M French. Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences, 1999.
- [14] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*. Elsevier, 1989.
- [15] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Resolving interference when merging models. In *Advances in Neural Information Processing Systems* (NeurIPS), 2023.
- [16] E. Yang, Z. Wang, L. Shen, S. Liu, G. Guo, X. Wang, and D. Tao. Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*, October 2023.
- [17] G. Ortiz-Jimenez, A. Favero, and P. Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *Advances in Neural Information Processing Systems*, volume 36, 2024.
- [18] A. Tang, L. Shen, Y. Luo, Y. Zhan, H. Hu, B. Du, and D. Tao. Parameter-efficient multi-task model fusion with partial linearizeation. In *The Twelfth International Conference on Learning Representations*, October 2023.
- [19] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In Advances in Neural Information Processing Systems (NeurIPS), 2018.
- [20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.
- [21] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Seyeon Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, October 2021.
- [22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [23] Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, January 1994.
- [24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.
- [25] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *International Conference on Computer Vision Workshops (ICCVw)*, 2013.
- [26] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [27] 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 (IJCV)*, 2016.
- [28] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.
- [29] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *International Joint Conference on Neural Networks (IJCNN)*, 2011.
- [30] Yann LeCun. The mnist database of handwritten digits, 1998.
- [31] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2011.

- [32] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 2017.
- [33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [34] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020.
- [35] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.
- [36] Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [37] Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization. In *International Conference on Machine Learning (ICML)*, 2022.
- [38] Alexandre Ramé, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. In Advances in Neural Information Processing Systems (NeurIPS), 2022.
- [39] Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [40] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022.
- [41] Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models, 2022.
- [42] Bo-Jian Hou, Yu-Hu Yan, Peng Zhao, and Zhi-Hua Zhou. Storage fit learning with feature evolvable streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7729–7736, 2021.
- [43] Bo-Jian Hou, Lijun Zhang, and Zhi-Hua Zhou. Prediction with unpredictable feature evolution. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5706–5715, 2021.
- [44] Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Cold fusion: Collaborative descent for distributed multitask finetuning, 2022.
- [45] Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing finetuned models for better pretraining, 2022.
- [46] Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations* (*ICLR*), 2023.
- [47] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [48] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.
- [49] Tian Yu Liu and Stefano Soatto. Tangent model composition for ensembling and continual fine-tuning, July 2023. arXiv:2307.08114 [cs].
- [50] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *International Conference on Learning Representations (ICLR)*, 2023.

- [51] Vipul Gupta, Santiago Akle Serrano, and Dennis DeCoste. Stochastic weight averaging in parallel: Large-batch training that generalizes well. In 8th International Conference on Learning Representations (ICLR). OpenReview.net, 2020.
- [52] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. In *Advances in Neural Information Processing Systems*, volume 34, pages 22405–22418, 2021.
- [53] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations (ICLR)*, 2019.
- [54] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations (ICLR)*, 2020.
- [55] George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *International Conference on Learning Representations (ICLR)*, 2023.
- [56] Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. Deep model fusion: A survey, 2023. arXiv preprint arXiv:2309.15698.
- [57] Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. Composing parameter-efficient modules with arithmetic operations. In Advances in Neural Information Processing Systems (NeurIPS), 2023.
- [58] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2829, June 2023.
- [59] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

# A Experimental details

All our experiments were performed using the same hardware consisting of four 3090 NVIDIA GPUs with 24GB of memory each which can be reproduced in less than 150 GPU hours. The details of each experiment are the following.

**Datasets.** We evaluate task arithmetic on a set of popular benchmark datasets from various domains. The dataset collection includes:

- **SVHN** [31]: The Street View House Numbers dataset is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting.
- MNIST [30]: A database of handwritten digits, with 60,000 training images and 10,000 testing images.
- EuroSAT [28]: A dataset based on Sentinel-2 satellite images covering 13 spectral bands, with 10 classes and a total of 27,000 labeled and geo-referenced images.
- **RESISC45** [32]: The remote sensing image scene classification dataset, consisting of 31,500 images in 45 scene classes.
- Cars [25]: This dataset contains images of cars categorized into various fine-grained classes. It is widely used for fine-grained image classification tasks, providing a rich set of vehicle images for training and evaluation.
- **DTD** (**Describable Textures Dataset**) [26]: This dataset is designed for texture recognition and categorization. It consists of texture images organized into 47 categories, each labeled with attributes describing the texture patterns. It is commonly used to evaluate texture recognition algorithms.
- SUN397 [27]: The Scene UNderstanding (SUN) dataset is a large-scale dataset for scene recognition, containing 397 categories with a total of over 100,000 images. It is used to evaluate scene understanding models and to benchmark scene classification algorithms.
- GTSRB (German Traffic Sign Recognition Benchmark) [29]: This dataset comprises images of German traffic signs, classified into over 40 categories. It is used to develop and evaluate traffic sign recognition systems, particularly in the context of autonomous driving and intelligent transportation systems.

**Fine-tuning.** All the fine-tuning experiments follow the same training protocol specified in Ilharco et al. [2] with minor modifications to the training code to use linearized models when needed. In particular, we fine-tune all datasets starting from the same CLIP pre-trained checkpoint downloaded from the open\_clip repository [58]. We fine-tune for 2,000 iterations with a batch size of 128, a learning rate of  $10^{-5}$  and a cosine annealing learning rate schedule with 200 warm-up steps and the AdamW optimizer [59]. As introduced in Ilharco et al. [2], during fine-tuning, we freeze the weights of the classification layer obtained by encoding a standard set of zero-shot template prompts for each dataset. Freezing this layer does not harm accuracy and ensures that no additional learnable parameters are introduced during fine-tuning [2]. We use this exact same protocol to fine-tune the non-linear and linearized models and do not perform any form of hyperparameter search in our experiments.

**Tuning of**  $\alpha$  **in task arithmetic benchmarks.** As in Ilharco et al. [2], we use a single coefficient  $\alpha$  to tune the size of the task vectors used to modify the pre-trained models. This is equivalent to setting  $\alpha = \alpha_1 = \ldots = \alpha_T$  in Eq. 1. In the task addition benchmarks, after fine-tuning, we evaluate different scaling coefficients  $\alpha \in \{0.0, 0.05, 0.1, \ldots, 1.0\}$  and choose the value that achieves the highest target metric on a small held-out proportion of the training set as specified in Ilharco et al. [2]. Namely, maximum normalized average accuracy, and minimum target accuracy on each dataset that still retains at least 95% of the accuracy of the pre-trained model on the control task. The tuning of  $\alpha$  is done independently for non-linear FT, linearized FT, and post-hoc linearization.

**Normalized accuracies in task addition.** Table 1 shows the normalized accuracies after editing different models by adding the sum of the task vectors on 8 tasks  $\tau = \sum_t \tau_t$ . Here, the normalization is performed with respect to the single-task accuracies achieved by the model fine-tuned on each task. Mathematically,

Normalized accuracy = 
$$\frac{1}{T} \sum_{t=1}^{T} \frac{\left[ \operatorname{acc} \left( f(x; \theta_0 + \sum_{t} \tau_t) \right) \right]}{\left[ \operatorname{acc} \left( f(x; \theta_0 + \tau_t) \right) \right]}.$$
 (5)

**Disentanglement error.** To produce the weight disentanglement visualizations of Figure 4, we compute the value of  $\xi(\alpha_1, \alpha_2)$  on a  $15 \times 15$  grid of equispaced values in  $[-2, 2] \times [-2, 2]$ . To estimate the disentanglement error, we use a random subset of 2,048 test points for each dataset.

# **B** Further experimental results

We now present additional experiments that expand the findings discussed in the main text.

## **B.1** Fine-tuning accuracies

In Figure 6, we report the single-task accuracies achieved by different CLIP models after fine-tuning with different dynamics (referred to as non-linear, NTK linearization, and our method).

# B.2 Weight disentanglement on different task pairs

In Figure 7, we illustrate weight disentanglement on different task pairs.

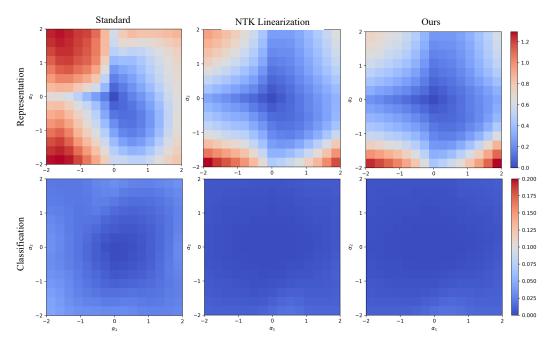


Figure 7: **Visualization of weight disentanglement.** The heatmaps show the disentanglement error  $\xi(\alpha_1,\alpha_2)$  of a single representation model CLIP ViT-B/32 (top) and a combination of representation model and classification model (bottom) on Cars - RESISC45 task pair. Three fine-tuning paradigms are used from left to right: standard, NTK linearization, and ours. The light regions denote areas of the weight space where weight disentanglement is stronger. The red box delimits the search space used to compute the best  $\alpha$  in all our experiments.

# C Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

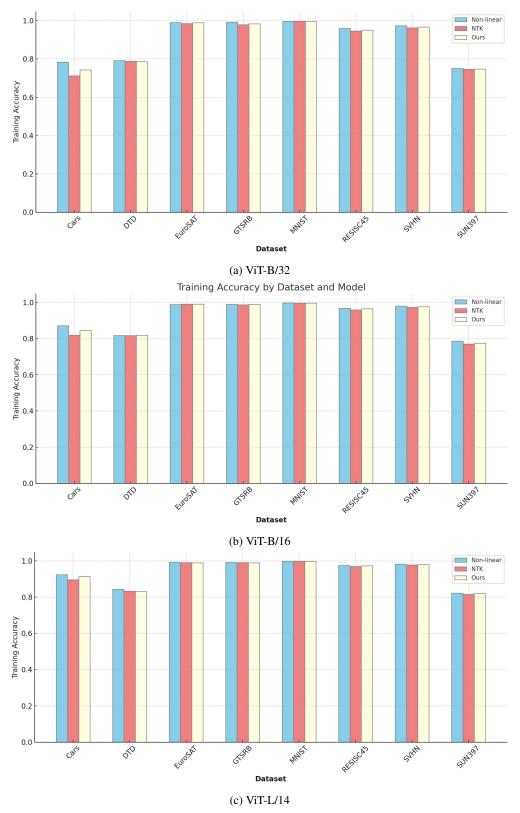


Figure 6: Single-task accuracies (CLIP). Accuracy of different models obtained using different strategies on each of the tasks.