UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

UNIVERSITY OF MODENA AND REGGIO EMILIA

"Enzo Ferrari" Department of Engineering
Master's Degree in Artificial Intelligence Engineering (LM-32)

Exploring Task Analogies in Deep Models:
Mitigating Subtractive Interference in Weight Space

Supervisor:
Dr. Angelo Porrello
Co-supervisor:
Prof. Simone Calderara

Candidate:
Fabio Bozzoli
Student ID 200648

ACADEMIC YEAR 2025/2026

Contents

List of Figures

Abstract

Background While Foundation Models like CLIP achieve unprecedented generalization, adapting them to specific downstream tasks via traditional fine-tuning is computationally expensive and fragments their generalized knowledge into isolated models. Task Arithmetic offers a powerful alternative by treating fine-tuning as a directional weight-space shift, or task vector. Manipulating these vectors through simple algebra enables zero-shot multi-task composition and domain adaptation without the need for retraining.

Problem A highly ambitious application of this framework is the zero-shot task analogy, which attempts to transfer learned capabilities from a source domain to a completely unobserved target domain. However, applying simple algebraic summation in a highly non-linear weight space inevitably causes high level of interference. This geometric misalignment entangles the analogical vector with source-specific artifacts. Although recent literature attempts to mitigate this by performing arithmetic in linearized regimes (e.g., the tangent space of the Neural Tangent Kernel), this thesis demonstrates that strict linearization artificially bottlenecks the model's expressivity, rendering it incapable of handling severe structural domain shifts. Furthermore, we identify the unconstrained subtraction of the source task vector (τ_A) as the primary cause of performance degradation, as it inadvertently erodes foundational, low-level feature extractors.

Methodology This thesis proposes solving the interference directly within the expressive non-linear space. To achieve this, we adapt GradFix, a gradient-guided masking mechanism rooted in model rebasin, to the novel context of zero-shot task analogies. Assuming domain shifts modify largely disjoint parameter subsets, we utilize a minimal target support set (few-shot oracle) to compute a binary mask via gradient sign-consensus. This discrete directional mask is applied asymmetrically (exclusively to the subtractive term) filtering out geometrically misaligned weights and safely decoupling domain-specific noise from transferable semantic knowledge prior to the algebraic projection.

Results Extensive empirical evaluations on the DomainNet benchmark validate both our theoretical diagnosis and algorithmic solution. Our targeted intervention effectively navigates the expressivity versus linearity trade-off, yielding zero-shot accuracy gains of up to +15% from the pre-trained baseline. Furthermore, the asymmetric masking actively preserves the linear separability and the topological integrity of the latent manifold, successfully addressing the geometric limitations of task arithmetic under complex domain shifts.

1. Introduction

Deep learning has fundamentally shifted toward massive Foundation Models, like the Vision Transformer (ViT). While they achieve unprecedented generalization, adapting them to specific downstream tasks via traditional fine-tuning is computationally expensive and fractures their generalized knowledge into isolated, task-specific models.

Model Merging and *Task Arithmetic* offer a powerful alternative to address this fragmentation. Fine-tuning a foundation model induces a directional weight-space shift known as a *task vector*. These vectors can be manipulated via simple arithmetic (addition, subtraction) to enable zero-shot multi-task composition and domain adaptation without retraining.

A highly ambitious application of this framework is the *zero-shot task analogy*. Similar to word embeddings, task analogies attempt to transfer a learned capability from a source domain to a completely unobserved target domain. For example, adding the shift between "sketch" and "quickdraw" (learned on man-made objects) to a "food sketch" model theoretically generates a "food quickdraw" classifier zero-shot. However, applying simple algebraic summation in a highly non-linear weight space inevitably causes catastrophic interference. The source domain shift rarely aligns perfectly with the optimal target direction. This geometric misalignment, which we formally define as the *closure error*, entangles the naive analogical vector with source-specific artifacts and drives the model into suboptimal loss basins.

Recent literature proposes performing task arithmetic within a linearized regime, specifically the *tangent space* (Neural Tangent Kernel). While a first-order Taylor expansion mathematically preserves arithmetic properties and mitigates interference, this theoretical protection comes at a severe cost. As this thesis demonstrates, strict linearization artificially bottlenecks the model's expressivity, lacking the representational capacity required for severe structural shifts (e.g., from detailed sketches to minimalist quickdraws).

This thesis proposes solving the closure error directly within the expressive non-linear space. To achieve this, we adapt **GradFix** [34], a gradient-guided masking mechanism rooted in model rebasin, to the novel context of zero-shot task analogies. Assuming domain shifts modify largely disjoint parameter subsets, we use a minimal target support set (few-shot) to compute a binary mask via gradient sign-consensus. This filters out geometrically misaligned weights, safely decoupling domain-specific noise from transferable semantic knowledge before the algebraic projection occurs.

1.1 Main contributions

This thesis addresses the geometric and topological limitations of task arithmetic under complex domain shifts, offering both a theoretical diagnosis and an algorithmic solution. Our core contributions are:

- **Expressivity vs. linearity trade-off:** we demonstrate that while linearized tangent spaces

protect against interference, they create a severe expressivity bottleneck. Conversely, non-linear spaces retain necessary capacity but suffer from geometric misalignment.

- **Diagnosis of subtractive interference:** we identify the unconstrained subtraction of the source task vector (τ_A) as the primary cause of performance degradation, showing that it inadvertently erodes foundational, low-level feature extractors.
- **Asymmetric masking via GradFix:** we apply *GradFix* [34] to mitigate this degradation. By using a few-shot oracle, we apply a discrete directional mask exclusively to the subtractive term, uncoupling domain noise from core semantic knowledge.
- **Empirical validation:** we establish that our targeted intervention yields zero-shot accuracy gains of up to +15% and actively preserves the linear separability and topological integrity of the latent manifold.

1.2 Thesis outline

The remainder of this thesis progressively builds the theoretical foundation necessary to solve the task analogy problem.

Chapter 2 reviews the fundamentals of model fine-tuning, focusing on the geometric and spectral properties of the Hessian matrix.

Chapter 3 delves into the topology of the loss landscape, exploring mode connectivity, the absence of loss barriers (analogous to spin glass models), Linear Mode Connectivity (LMC), and the Lottery Ticket Hypothesis.

Chapter 4 introduces Task Arithmetic, detailing the Neural Tangent Kernel (NTK), task arithmetic within the tangent space, weight disentanglement, and τJP regularization, formalizes the zero-shot Task Analogy as a complex transfer learning problem,.

Chapter 5 highlights task analogy geometric challenges, and introduces our adapted solution applying *GradFix*.

Chapter 6 presents the experimental evaluation on the DomainNet benchmark, comparing non-linear and linearized regimes, and concludes with an extensive geometric and topological diagnosis.

Chapter 7 summarizes the key findings, discusses broader implications, and outlines potential future research directions.

2. Fine-tuning and loss landscapes

2.1 Training networks as an optimization problem

Training a neural network can be interpreted as an optimization problem over a high dimensional parameter space. It is well known that certain network architectures are easier to train than others, and that the choice of the initial hyperparameters (such as the batch size, the learning rate and the optimizer) can have a significant impact on the training process [23].
We can formalize the training process as follows. Given:

- a training set $\mathcal{D} = \{(x_i, y_i) : x_i \in \mathbb{R}^{d_{\text{in}}}, y_i \in \mathbb{R}^{d_{\text{out}}}, i = 1, ..., N\}$, where x_i are sampled from a possibly unknown distribution ν;
- a prediction function parametrized by a vector θ, $h \in \mathcal{H} = \{h(\cdot, \theta) : \theta \in \mathbb{R}^M, h : \mathbb{R}^{d_{\text{in}}} \times \mathbb{R}^M \to \mathbb{R}^{d_{\text{out}}}\}$;
- a loss function $\ell : \mathbb{R}^{d_{\text{out}}} \times \mathbb{R}^{d_{\text{out}}} \to \mathbb{R}$, that measures the error between the predictions and the true labels;

the goal is to find the parameters θ^* that minimize the loss function $\mathcal{L}(\theta)$, defined as:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta) = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i, \theta), y_i) \tag{2.1}$$

The function $\mathcal{L} : \mathbb{R}^M \to \mathbb{R}$ defines a scalar field over the parameter space. The geometry of this scalar field, commonly referred to as the loss landscape, is characterized by its critical points, curvature, and connected components. The behavior of gradient-based optimization algorithms is fundamentally determined by the geometric properties of this landscape.

For solving this optimization problem, we typically use a variant of the Stochastic Gradient Descent (SGD) algorithm, which iteratively updates the parameters in the direction of the negative gradient of the loss function. If we use the standard SGD update rule (steepest descent), the parameters are updated as follows:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \ell(h(x_i, \theta_t), y_i)$$

where η_t is the learning rate, and i is a randomly sampled index from the training set. In the case of mini-batch SGD, we compute the gradient over a batch of samples as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta_t}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} \nabla_\theta \ell(h(x_i, \theta_t), y_i)$$

where i is a randomly sampled index from the training set and $\mathcal{S}_t \subseteq \{1, ..., N\}$ is a mini-batch of indices [10].

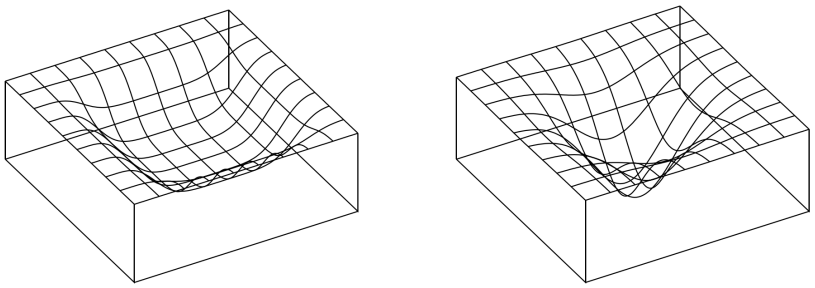


Figure 2.1: Visualization of a flat minimum (left) and a sharp minimum (right) in the loss landscape.

2.2 The fine-tuning setting

In the fine-tuning setting, however, optimization does not start from a random initialization, but from a pre-trained parameter configuration θ_0, which already lies in a structured region of the loss landscape. So the great difference between standard training and fine-tuning procedures is the fact that the first method starts from a random initialization, while the second starts from a pre-trained configuration. For this reason we can consider a parameter perturbation such that $\theta = \theta_0 + \Delta_\theta$ and under sufficient regularity conditions a second order Taylor expansion of the loss function around θ_0 gives us the following approximation:

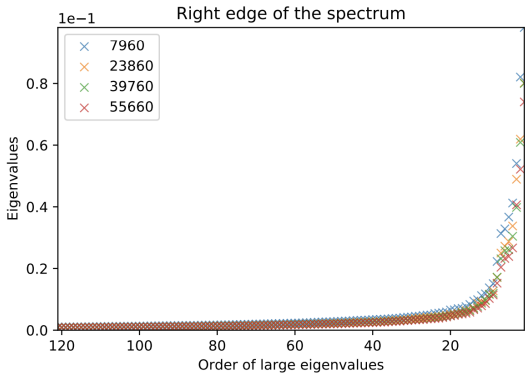
$$\mathcal{L}(\theta + \Delta_\theta) \approx \mathcal{L}(\theta_0) + \nabla_\theta \mathcal{L}(\theta_0)^\top \Delta_\theta + \frac{1}{2} \Delta_\theta^\top H(\theta_0) \Delta_\theta \tag{2.2}$$

where $H(\theta_0)$ is the Hessian matrix of the loss function evaluated at θ_0.

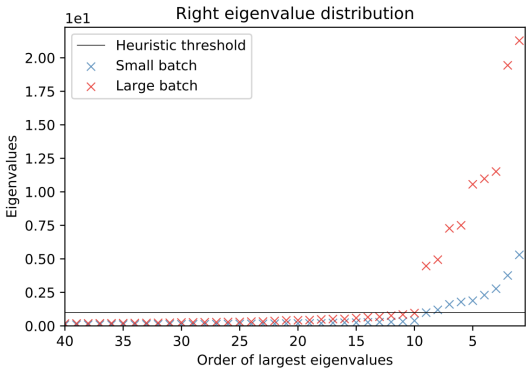
2.3 Some interesting observations about the Hessian

The relationship between curvature and generalization has been studied since the late 1990s. Hochreiter and Schmidhuber (1997) [17] introduced the notion of flat minima, arguing that solutions lying in wide basins of the loss landscape exhibit improved robustness and generalization. We can see a good representation of the difference between a flat minimum and a sharp minimum in Figure 2.1. Moreover it has been observed by Sagun et al. [36] that the Hessians are affected by the training data: if the data have k classes with relatively small deviations among them, then the Hessian will have k large eigenvalues, while the remaining eigenvalues will be close to zero.
In addition it has been tested the effect of growing size of the network when data, algorithm and architecture are fixed, and it has been observed that the number of large eigenvalues does not grow with the size of the network, but remains approximately equal to the number of classes in the data.

Another interesting observation is the study of the comparison between a large batch size and a small batch size. It has been observed that the large batch size leads to sharper minima, while the small batch size leads to flatter minima. This is because the large batch size provides a more accurate estimate of the gradient, which allows the optimization process to converge to a sharper minimum.



(a) Effect of network size on the number of large eigenvalues in the Hessian matrix. x-axis rank of the eigenvalues, y-axis value of the eigenvalues.



(b) Comparison between the loss landscapes of a large batch size (LB) and a small batch size (SB). The LB landscape is sharper than the SB landscape. x-axis rank of the eigenvalues, y-axis value of the eigenvalues.

On the other hand, the small batch size introduces more noise in the gradient estimation, which can help the optimization process to escape from sharp minima and converge to flatter minima.

As shown in Figures 2.2a and 2.2b, the Hessian spectrum exhibits a clear separation: a dense bulk centered near zero and a small set of isolated large eigenvalues. These results suggest that trained neural networks converge to geometrically structured regions of the parameter space, where the effective dimensionality of the curvature is significantly lower than the ambient parameter dimension.

This observation can be explained by supposing that the loss function can be expressed as a composition of two functions:

- the model function $h_\bullet : \mathbb{R}^M \to \mathbb{R}^{d_{\text{out}}}$, which maps the parameters to the predictions;
- the loss function $\ell_\bullet : \mathbb{R}^{d_{\text{out}}} \to \mathbb{R}^+$, which measures the error between the predictions and the true labels and is a convex function.

Here $\bullet$ refers to the given example.

From simple computations we can see that the gradient and the Hessian of the loss function can be expressed as follows (note that $J_\theta h(\theta) \in \mathbb{R}^{M \times d_{\text{out}}}$ is the Jacobian matrix of the model function with respect to the parameters) [46]:

$$\nabla_\theta \mathcal{L}(\theta) = J_\theta h(\theta) \nabla_h \ell(h(\theta)) \tag{2.3}$$

$$\nabla^2_\theta \ell(h(\theta)) = J_\theta h(\theta) \nabla^2_h \ell(h(\theta)) J_\theta h(\theta)^\top + \sum_{c=1}^{d_{\text{out}}} \frac{\partial \ell}{\partial h_c}(h(\theta)) \nabla^2_\theta h_c(\theta) \tag{2.4}$$

Observation 2.3.1. The term $J_\theta h(\theta) \nabla^2_h \ell(h(\theta)) J_\theta h(\theta)^\top$ is called **Gauss-Newton matrix** and it is a positive semi-definite matrix that approximates the Hessian of the loss function. It is often used in optimization algorithms, to find the optimal parameters of a model.

Since $\ell_\bullet$ is a convex function, we have that $\nabla^2_h \ell(h(\theta)) \succeq 0$, so it is a symmetric positive semi-definite matrix. Therefore we can decompose it using the eigendecomposition for symmetric matrices [1] as follows: $\nabla^2_\theta \ell(h(\theta)) = Q\Lambda Q^\top = Q\sqrt{\Lambda}\sqrt{\Lambda}^\top Q^\top = Q\sqrt{\Lambda}(Q\sqrt{\Lambda})^\top$, where Q is an orthogonal matrix, Λ is a diagonal matrix with non-negative entries and $\sqrt{\Lambda} = \text{diag}(\sqrt{\lambda_1}, ..., \sqrt{\lambda_{d_{\text{out}}}})$. This means that the loss function can be expressed as:

$$\nabla^2_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(J_i S_i)(J_i S_i)^\top + \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{d_{\text{out}}} \frac{\partial \ell_i}{\partial h_c}(h_{x_i}(\theta))\nabla^2_\theta h_c(\theta) \tag{2.5}$$

where $S_i = Q_i\sqrt{\Lambda_i}$ and $J_i = J_\theta h_{x_i}(\theta)$. At a point close to a local minimum the gradient of the loss function is close to zero, which means that the second term of the Hessian is close to zero. For this reason we can approximate the Hessian as follows:

$$\nabla^2_\theta \mathcal{L}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}(J_i S_i)(J_i S_i)^\top \tag{2.6}$$

This formulation leads to two fundamental geometric and statistical observations:

1. **Low-rank structure:** the Hessian is expressed as a sum of outer products of matrices with inner dimension d_{out}. Consequently, its rank is strictly bounded by:

   $$\text{rank}(H) \leq N \cdot d_{out}.$$

   In the overparameterized regime ($M \gg N \cdot d_{out}$), the Hessian is necessarily rank-deficient. This implies that only a small fraction of directions in parameter space exhibit significant curvature (the "spikes"), while the vast majority correspond to nearly flat directions (the "bulk") [36].

2. **Statistical interpretation:** the matrix $\sum(J_i S_i)(J_i S_i)^\top$ can be interpreted as the uncentered sample covariance matrix of the weighted gradients. This means that the eigenvectors corresponding to large eigenvalues (high curvature) align with the directions where the gradients of the training examples vary the most. Conversely, the null space of this matrix represents directions where the model outputs are insensitive to parameter perturbations as reported by Sagun et al. [36].

This spectral degeneracy explains why trained neural networks concentrate in low dimensional curved subspaces of the ambient parameter space. As we will discuss in the next chapters, this property is the key enabler for arithmetic operations in parameter space, such as *Linear Mode Connectivity* and *Task Arithmetic*, as updates tend to remain confined to these structured, meaningful manifolds.

3. Mode connectivity

In the previous chapter we have seen that the loss landscape of deep neural networks is highly non-convex, and that there are many local minima. Moreover, the loss is often high along a line segment connecting two optima. These two observations suggest that the loss landscape of deep neural networks is very complex, and that there are many local minima that are not connected by a simple path.

3.1 Connection procedure

Garipov et al. (2018) [13] proposed a method to minimize the training loss along a curve connecting two optima. Let $\hat{\theta}_1, \hat{\theta}_2 \in \mathbb{R}^M$ be two sets of weights obtained by training a neural network with different initializations. The objective is to find a curve $\phi_\alpha(t) : [0,1] \to \mathbb{R}^M$ such that $\phi_\alpha(0) = \hat{\theta}_1$ and $\phi_\alpha(1) = \hat{\theta}_2$, and that minimizes the training loss along the curve:

$$\hat{\ell}(\alpha) = \frac{\int_0^1 \mathcal{L}(\phi_\alpha(t))dt}{\int_0^1 ||\phi'_\alpha(t)||dt} = \frac{\int_0^1 \mathcal{L}(\phi_\alpha(t))||\phi'_\alpha(t)||dt}{\int_0^1 ||\phi'_\alpha(t)||dt} = \int_0^1 \mathcal{L}(\phi_\alpha(t))q_\alpha(t)dt = \mathbb{E}_{t\sim q_\alpha}[\mathcal{L}(\phi_\alpha(t))] \tag{3.1}$$

where $q_\alpha(t) = ||\phi'_\alpha(t)|| \cdot \left(\int_0^1 ||\phi'_\alpha(t)||dt\right)^{-1}$ is the distribution on $t \in [0,1]$ induced by the curve ϕ_α. The numerator corresponds to the line integral of the loss function along the path with respect to arc-length measure. Therefore, the normalized quantity represents the average loss encountered along the curve in parameter space. Dealing with the gradient of $\hat{\ell}(\alpha)$ is intractable because also the distribution q_α depends on α. Therefore it has been proposed to optimize the following surrogate objective:

$$\ell(\alpha) = \int_0^1 \mathcal{L}(\phi_\alpha(t))dt = \mathbb{E}_{t\sim U(0,1)}[\mathcal{L}(\phi_\alpha(t))] \tag{3.2}$$

where $U(0,1)$ is the uniform distribution on $[0,1]$. We know that these two formulations are equivalent when the curve ϕ_α has constant speed, i.e. $||\phi'_\alpha(t)||$ is constant for all $t \in [0,1]$. This is achieved for example when $\phi_\alpha(t)$ defines a polygonal chain with two line segments of the same length and when their parametrizations are linear in t. In practice, it has been observed that optimizing $\ell(\alpha)$ also leads to a decrease in $\hat{\ell}(\alpha)$, and that the resulting curve ϕ_α has approximately constant speed.

To minimize Equation (3.2) it is sufficient to sample $\tilde{t}$ from the uniform distribution and to compute the stochastic gradient step of $\mathcal{L}(\phi_\alpha(t))$ with respect to α. In this way we obtain an unbiased estimate of:

$$\nabla_\alpha \mathcal{L}(\phi_\alpha(\tilde{t})) \approx \mathbb{E}_{t\sim U(0,1)}[\nabla_\alpha \mathcal{L}(\phi_\alpha(t))] = \nabla_\alpha \mathbb{E}_{t\sim U(0,1)}[\mathcal{L}(\phi_\alpha(t))] = \nabla_\alpha \ell(\alpha) \tag{3.3}$$

3.1.1 Empirical results

Garipov et al. [13] have tested the above procedure on different architectures. Here are presented the results obtained with a ResNet-164 on CIFAR-100.

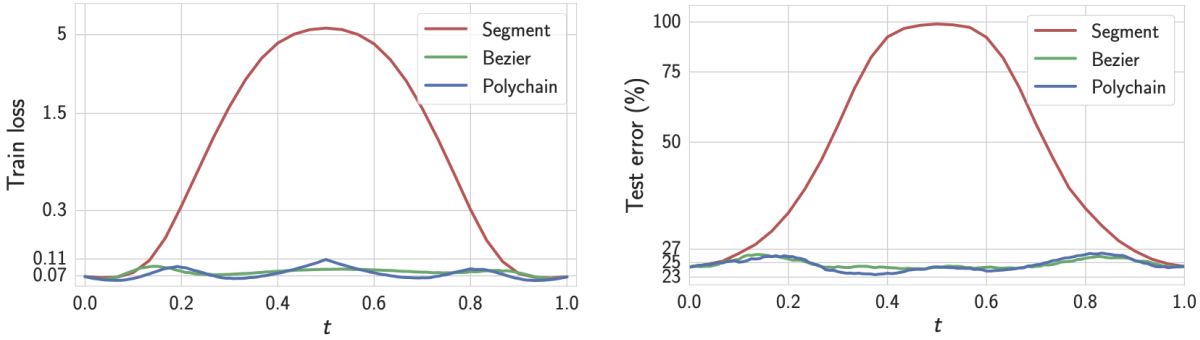


Figure 3.1: **Training loss and test error across different paths.** The left panel displays the ℓ_2-regularized cross-entropy training loss as a function of the parameter t. The right panel shows the test error for varying values of t. Both plots compare the performance of three interpolation methods: the polygonal chain (blue), the Bézier curve (green), and the linear segment (red).

From Figure 3.1 it is possible to see that the linear segment connecting the two optima has a high loss and a high test error for intermediate values of t. On the other hand the *polychain* and the *bezier* curves have a low loss and a low test error for all values of t. This means that there is a curve connecting the two optima along which the loss is low, and that this curve can be found by optimizing the surrogate objective Equation (3.2).

Another good way to see the effectiveness of the proposed method is by plotting the same ℓ_2-regularized cross entropy train loss along the curve $\phi_\alpha(t)$ in a 3D plot.

This time consider three different weight vectors: $\theta_1, \theta_2, \theta_3 \in \mathbb{R}^M$. Set

$$u = (\theta_2 - \theta_1),\ v = (\theta_3 - \theta_1) - \frac{\langle \theta_3 - \theta_1, \theta_2 - \theta_1 \rangle}{||\theta_2 - \theta_1||^2} \cdot (\theta_2 - \theta_1)$$

which is a simple Gram-Schmidt orthogonalization of $\theta_3 - \theta_1$ with respect to $\theta_2 - \theta_1$. Then we normalize the two vectors $\hat{u} = \frac{u}{||u||}$ and $\hat{v} = \frac{v}{||v||}$ obtaining an orthonormal basis of the plane defined by $\theta_1, \theta_2, \theta_3$. Then it is possible to visualize the loss landscape by evaluating the networks corresponding to each of the points in the grid. A point $P(x, y)$ in the plane would then be given by $P(x, y) = \theta_1 + x \cdot \hat{u} + y \cdot \hat{v}$.

These experiments (see Figure 3.2) provide empirical evidence that the optima of deep neural networks are connected by simple pathways.

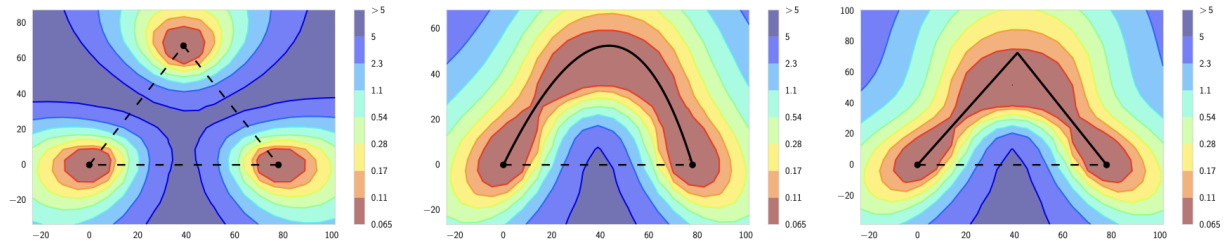


Figure 3.2: **Visualization of the ℓ_2-regularized cross-entropy loss surface** for a ResNet-164 on CIFAR-100 within two-dimensional weight subspaces. In all panels, the horizontal axis is defined by the segment connecting the optima of two independently trained networks. The vertical axis varies across panels according to the different planes defined in the text. *Left:* Relative positions of three independent optima. *Middle and Right:* A quadratic Bézier curve and a polygonal chain (with one vertex) connecting the two lower optima along a path of near-constant loss. Notably, a direct linear interpolation between modes crosses a region of significantly higher loss in every projection.

3.2 No barriers in loss landscape

Building on the work of Garipov et al. [13], Draxler et al. [8] introduced a method to identify a *minimum energy path* between two optima. Within their framework, the terms "energy" and "training loss" are used interchangeably to describe the optimization landscape. The goal is to find a continuous path p^* from θ_1 to θ_2 through parameter space with the lowest maximum energy:

$$p^*(\theta_1, \theta_2) = \underset{p \text{ from } \theta_1 \text{ to } \theta_2}{\arg\min} \{\max_{\theta \in p} \mathcal{L}(\theta)\} \tag{3.4}$$

The lowest path p^* is called the *minimum energy path* (MEP), and the parameter set with the maximum loss along this path is called the *saddle point*, in the ideal MEP setting it corresponds to a first-order saddle of the loss surface.

Obviously the problem of finding the MEP is intractable, but it has been proposed a method to find an approximation of the MEP. The method is called Nudged Elastic Band (NEB) and it is based on the idea of discretizing the path into a finite set of pivots and iteratively update their positions. Each pivot is then subjected to two forces:

- a loss force derived from the gradient of the loss, projected perpendicular to the path;
- a spring force between neighboring pivots, acting only parallel to the path to maintain an approximately uniform spacing.

This construction prevents the path from sliding back to the minima and yields an approximation of a minimum-energy path. The maximum loss along the resulting path provides an upper bound on the saddle energy between the two minima.

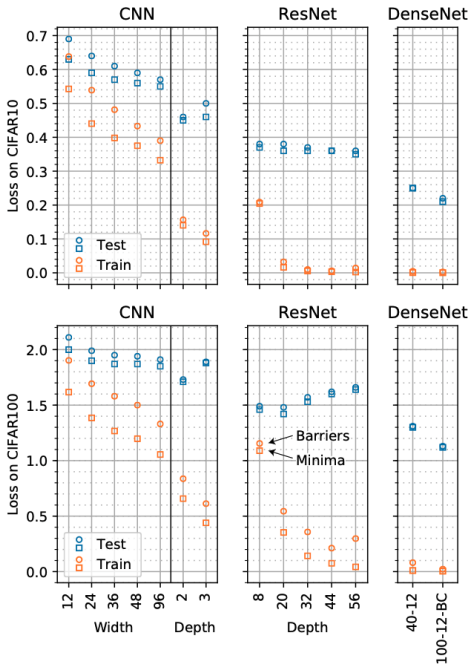


Figure 3.3: Here are represented the minimum (■) and the saddle point (●) on training and test set. It's possible to see that the ce on loss values between the minima and the saddle point is very narrow for very deep architectures (middle and right plot) while it remains a small difference for shallow architectures. Another interesting insight is that in a simple architecture like a CNN, by increasing the depth, the difference between the minima and the saddle point becomes smaller.

3.2.1 Results

In Figure 3.3 it is possible to see the differences obtained by comparing the saddle points obtained by Auto-NEB and the minima on the training and test set. It has been tested with different architectures (CNN, ResNet with different depths and DenseNet) and datasets (CIFAR-10 and CIFAR-100).

Another interesting result is the fact that the loss value at the saddle point found by Auto-NEB is usually smaller than the loss value during the most part of the training procedure, as we can see in Figure 3.4. This means that the loss landscape of deep neural networks is very smooth and that there are essentially no significant energy barriers in highly overparameterized architectures. Moreover, it has been observed that the loss value of the saddle point found by Auto-NEB is usually very similar to the one observed at the minimum. This remark can imply that the loss surface can be interpreted as a large connected component with a very flat landscape, where the minima are just different points in the same connected component. This is consistent with the low-rank structure of the Hessian matrix observed in the previous chapter: if curvature concentrates in a low-dimensional subspace, flat directions can enable continuous transitions between minima.

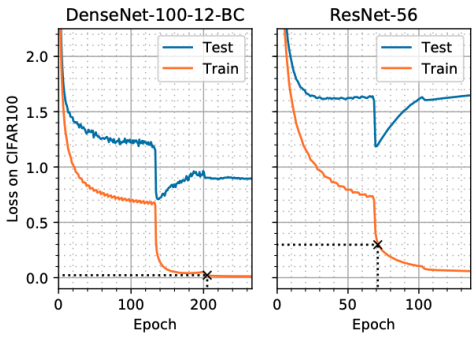


Figure 3.4: Here are represented the learning curves of the two deepest models and the saddle points found by Auto-NEB with a dotted line.

3.2.2 Discussion of the Results

The empirical findings presented above can be interpreted through two complementary mechanisms: *resilience* and *redundancy*. Together, they help explain why modern deep architectures exhibit extremely low or even vanishing loss barriers between minima.

Resilience. SOTA architectures have hundreds of parameters per layer and skip connections between non adjacent layers. This means that low perturbations of one or few parameters can be compensated by coordinated adjustments of the other parameters. This type of resilience is exploited by procedures like Dropout.

Redundancy. Redundancy provides a constructive explanation. Consider a two-layer network solving the XOR problem (as in Figure 3.5): swapping two hidden neurons yields an equivalent solution corresponding to a different minimum. A direct interpolation between the two configurations leads to high loss. However, introducing a single additional neuron enables a continuous reallocation of functional roles, allowing a path between the two minima without increasing the loss. In large deep networks, where redundancy is pervasive, similar mechanisms can eliminate barriers between minima, explaining the existence of near-constant-loss connecting paths.

3.3 Theoretical perspective: the spin glass analogy

The previous empirical results show that barriers can be circumvented; here we discuss a complementary theoretical view explaining why many good solutions cluster in a narrow energy band.

The complex topological properties of the loss landscape of deep neural networks have been studied through the lens of statistical physics. In particular, it has been shown that the loss landscape of deep neural networks can be modeled as a **spin glass model** as studied by Choromanska et al. [7].

A spin glass is a disordered magnet characterized by frustrated interactions. It consists of a set of magnetic moments (spins) whose interaction couplings are randomly distributed, leading to a

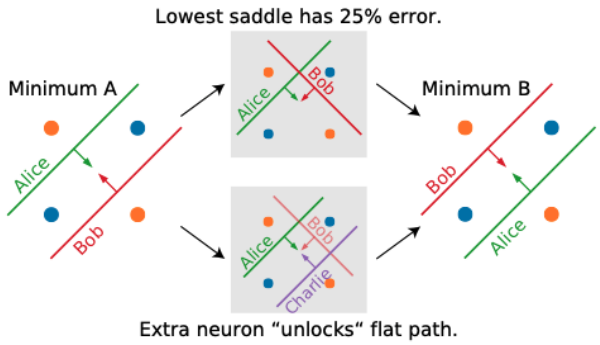


Figure 3.5: The continuous transition between two equivalent configurations of a two layer perceptron that can fit the XOR example. The two configurations are obtained by swapping the two neurons in the first hidden layer.

highly non-convex energy landscape. Typically they are solved by starting from the Hamiltonian of the system, which is a function that describes the energy of the system in terms of the configuration of the spins. The spin glass model compared in the work of Choromanska et al. [7] to the loss landscape of DNNs is the H-spin spherical spin glass model. In this section we follow the spin-glass literature and denote by $\tilde{\mathbf{w}}$ the (normalized) configuration variable of the H-spin model; this should not be confused with the network parameters θ used elsewhere in the thesis. The Hamiltonian is:

$$\mathcal{L}_{\Lambda,H}(\tilde{\mathbf{w}}) = \frac{1}{\Lambda^{(H-1)/2}} \sum_{i_1,i_2,\ldots,i_H=1}^{\Lambda} X_{i_1,\ldots,i_H} \prod_{k=1}^{H} \tilde{w}_{i_k} \tag{3.5}$$

with the spherical constraint $\frac{1}{\Lambda}\sum_{i=1}^{\Lambda} \tilde{w}_i^2 = 1$.

The following derivation relies on strong simplifying assumptions that make the loss analytically tractable. The resulting model should be interpreted as a qualitative theoretical analogy rather than an exact description of modern architectures. The assumptions are the following:

- **Variable independence:** the network's inputs and activation states (the ReLU gating along each path) are modeled as independent random variables, effectively decoupling them from the network's weights. In statistical mechanics, this provides the necessary "quenched disorder" to render the Hamiltonian analytically tractable, enabling a Gaussian random field approximation.

- **Redundancy:** as we have seen before, this assumption reflects the highly over-parameterized regime typical of modern deep learning. It ensures the system is large enough to apply asymptotic limits from statistical physics (the thermodynamic limit), allowing the vast number of parameters to be treated as a continuous statistical ensemble.

- **Uniformity:** it is assumed that the unique weights are evenly distributed across the network's connection graph. This guarantees that every possible combination of H weights appears

with roughly the same frequency, transforming the discrete graph topology into a continuous, symmetric polynomial where no single connection disproportionately dominates.

Under these assumptions it is possible to show (see Appendix A for more mathematical details) that the loss function of a DNN can be expressed as follows:

$$\mathcal{L}_{\Lambda,H}(\tilde{\mathbf{w}}) = \mathcal{C}_1 + \mathcal{C}_2 q \sum_{i_1,i_2,\dots,i_H=1}^{\Lambda} X_{i_1,i_2,\dots,i_H} \prod_{k=1}^{H} \tilde{w}_{i_k} \tag{3.6}$$

where $\mathcal{C}_1$ and $\mathcal{C}_2$ are some constants, q is a normalization factor that depends on the number of parameters and the depth of the network, $X_{i_1,i_2,\dots,i_H}$ are i.i.d. standard Gaussian random input variables, $\tilde{w}_{i_k}$ is the k-th component of the weight vector $\tilde{\mathbf{w}}$, H is the number of layers in the architecture, and $\Lambda = \sqrt[H]{\prod_{k=0}^{H-1} n_k}$ is the geometric mean of the number of neurons in each layer.

When minimizing the loss function it is possible to ignore the constant terms $\mathcal{C}_1$ and $\mathcal{C}_2$ and to substitute $q = \frac{1}{\Lambda^{(H-1)/2}}$. In this way we obtain the Hamiltonian of the H-spin spherical spin glass model and apply the results of Auffinger et al. [4].

3.3.1 Theoretical results

To understand the theoretical results from the Auffinger's studies it is necessary to introduce the following definition:

Definition 3.3.1. Let $u \in \mathbb{R}$ and k be an integer such that $0 \leq k \leq \Lambda$. We will denote as $\mathcal{C}_{\Lambda,k}(u)$ a random number of critical values of $\mathcal{L}_{\Lambda,H}(w)$ in the set $\Lambda B = \{\Lambda x : x \in (-\infty, u)\}$ with index[1] equal to k. Similarly we will denote as $\mathcal{C}_{\Lambda}(B)$ a random total number of critical values of $\mathcal{L}_{\Lambda,H}(w)$.

We assume that $\Lambda \to \infty$ (which is possible if and only if $N \to \infty$) and we call $E_\infty = E_\infty(H) = 2\sqrt{\frac{H-1}{H}}$ **energy barrier**.

Thanks to these definitions and using the Theorem 2.12 proved by Auffinger et al. [4] it is possible to say that any critical point that lies above the energy barrier is a high index saddle point with overwhelming probability. So for this reason all critical points with non diverging index must be in the interval $(-\Lambda E_0(H), -\Lambda E_\infty(H))$, where $E_0(H)$ is some real number.

Moreover, following Theorem 2.15 in Auffinger et al. [4], finding a critical value with an index grater than or equal to k below the energy threshold $-\Lambda E_k(H)$ is exponentially unlikely. This threshold naturally lies within the interval $(-\Lambda E_0(H), -\Lambda E_\infty(H))$, since the underlying sequence $\{E_i(H)\}_{i=0}^{\infty}$ is strictly decreasing and converges to $E_\infty(H)$.

These findings reveal a highly structured and layered topology of the loss landscape. In particular, slightly above the global minimum, there is a narrow energy band $(-\Lambda E_0(H), -\Lambda E_1(H))$ containing exclusively local minima (critical points of index 0). As we move to higher energy levels, the landscape becomes increasingly populated by saddle points of higher indices. Specifically, the subsequent band $(-\Lambda E_1(H), -\Lambda E_2(H))$ contains only local minima and index-1 saddle points.

[1] The number of negative eigenvalues of the Hessian matrix $\nabla^2 \mathcal{L}_{\Lambda,H}$ at $\mathbf{w}$ is also called index of $\nabla^2 \mathcal{L}_{\Lambda,H}$ at $\mathbf{w}$.

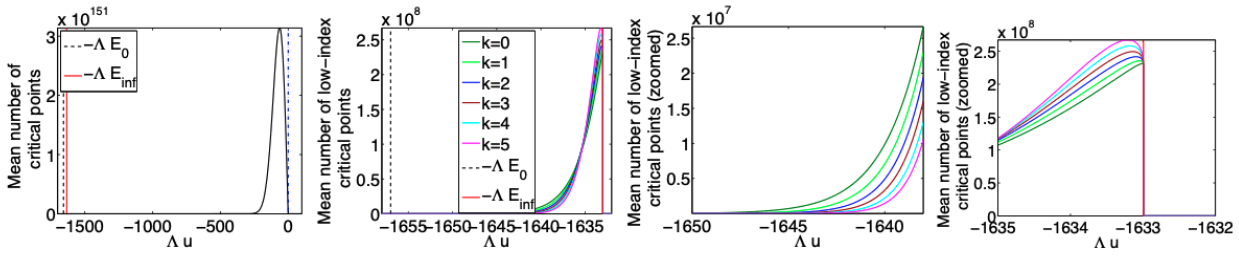


Figure 3.6: Distribution of the mean number of critical points, local minima and saddle points with low index. The first picture represents the original distribution, the following 3 the zoomed version of the energy band $(-\Lambda E_0(H), -\Lambda E_\infty(H))$. For this simulation the parameters Λ and H have been set to 1000 and 3 respectively.

The next interval, $(-\Lambda E_2(H), -\Lambda E_3(H))$, introduces index-2 saddle points alongside those of lower indices, a hierarchical pattern that generalizes systematically for all higher energy bands.

Another interesting insight is that the number of critical points in $(-\Lambda E_0(H), -\Lambda E_\infty(H))$ grows exponentially with Λ. This means that the number of local minima grows exponentially with the number of parameters, and the probability of finding a saddle point in that interval, rather than a local minima, goes to 0. This behaviour is exemplified in Figure 3.6.

The optimizer easily avoids the band of high index saddle points which have a high number of negative curvature directions, and reaches a local minimum or a low index saddle point in the energy band $(-\Lambda E_0(H), -\Lambda E_\infty(H))$. For this reason finding a bad solution (a critical point far away from the global minimum) is unlikely, specially in the overparametrized regime where $\Lambda \to \infty$.

Moreover finding the global minimum is also unlikely, because if we suppose to be at a local minimum with energy $-E_\infty(H) - \delta$, in order to find a futher low lying minimum, we would have to pass across a saddle point with higher energy. This process takes an exponential time in Λ and it is therefore unlikely to happen. For this reason, the local minima found by the optimizer are likely to be good solutions with low loss value.

This theory does not directly imply mode connectivity; rather, it supports the view that many good solutions exist in a narrow low-energy band, making connectivity phenomena plausible in overparameterized regimes.

3.4 Linear mode connectivity

Frankle et al. [12] adopt a methodology akin to the one proposed by Garipov et al. [13] to analyze training instability. Specifically, they train two identical copies of a network, starting from the same initialization but subjected to different realizations of SGD noise (e.g., different mini-batch orderings). At the end of the training procedure, this yields two distinct sets of optimized weights, θ_T^1 and θ_T^2. The connectivity between these two solutions is then evaluated by measuring the error barrier along the linear interpolation connecting them, using the following function:

- $\varepsilon_{\text{sup}}(\theta_T^1, \theta_T^2) = \sup_{t\in[0,1]} \varepsilon(t\theta_T^1 + (1-t)\theta_T^2)$ is the supremum of the test error along the linear

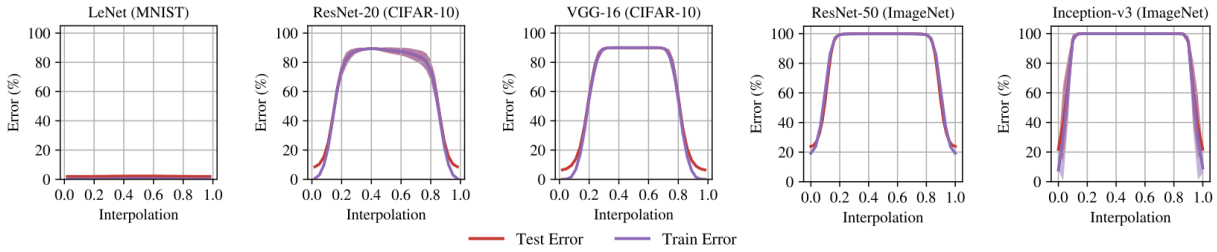


Figure 3.7: The graphs displays the curve of the train and test error along the linear interpolation between two solutions obtained by training the same network with different samples of SGD noise

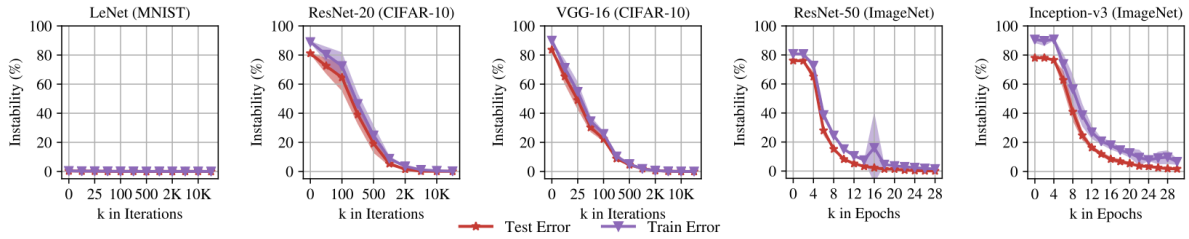


Figure 3.8: In the graphs are represented the curve of the train and test error along the linear interpolation between two solutions obtained by training the same network with different samples of SGD noise. On the x-axis is represented the epoch k at which the two copies of the network are created.

path connecting θ_T^1 and θ_T^2 ($\varepsilon(x)$ is the test error of the network with weights x).;

- $\bar{\varepsilon}(\theta_T^1, \theta_T^2) = \text{mean}(\varepsilon(\theta_T^1), \varepsilon(\theta_T^2))$ is the average of the test error at the two endpoints.

The error barrier height on the linear path between θ_T^1 and θ_T^2 is $\varepsilon_{\text{sup}}(\theta_T^1, \theta_T^2) - \bar{\varepsilon}(\theta_T^1, \theta_T^2)$. We call this quantity the *linear interpolation instability* (shorthand *instability*) of $\mathcal{N}$ to SGD noise. We say that θ_T^1 and θ_T^2 are *mode connected* if there exists a path connecting them where the error barrier height is ≈ 0. They are *linearly mode connected* if this is true along the linear path. We consider a network $\mathcal{N}$ stable to SGD noise when the networks that result from instability analysis are linearly mode connected. Otherwise, it is unstable.

Frankle er al. [12] have tested two different scenarios:

Instability analysis from initialization. They have trained two copies of the same, randomly initialized network with different samples of SGD noise. Figure 3.7 shows the train and test error when interpolating between the two solutions; there we can see that only LeNet is stable to SGD noise, while all the other architectures are unstable.

Instability analysis during training. They have trained a network with the same sample of SGD noise until a certain epoch k. Then they have built two copies of the network and continued the training with different samples of SGD noise until the end of the training procedure.

Figure 3.8 illustrates the train and test error when interpolating between the two solutions for different values of k (the reported value is the highest found during interpolation, the peak in Figure 3.7). We can see that for all the architectures there is a critical epoch k_c such that for $k > k_c$ the two

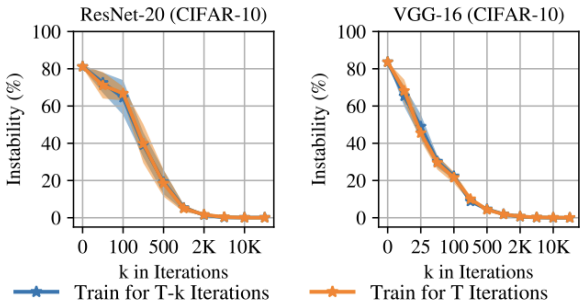


Figure 3.9: Linear interpolation instability on the test set when making two copies of the state of the network at step k and either (1) training for the remaining $T - k$ steps (blue) or (2) training for T steps with the learning rate schedule reset to step 0 (orange).

solutions are linearly mode connected, while for $k < k_c$ they are not. Moreover, the value of k_c is very small compared to the total number of epochs, which means that the instability to SGD noise arises very early during training.

We know that varying the step k at which the two copies of the network are created has two effects:

1. it changes the state of the network from which we train two copies to completion on different SGD noise;

2. it changes the amount of training time remaining after the two copies are created: when we run instability analysis from step k, we train the two copies for $T - k$ steps. So as k increases, the two copies are trained for fewer steps, which can make them more similar and therefore more likely to be linearly mode connected.

To disentangle the role of the training time remaining after the two copies are created, they have run an additional experiment: they reset the learning rate schedule to iteration 0 after creating the two copies and trained them for an additional T steps. The different behavior of the two procedures is exemplified in Figure 3.9.

3.4.1 Lottery ticket hypothesis

We know that sparse networks are more difficult to train from scratch. Beyond trivial sparsities where many weights remain and random subnetworks can train to full accuracy, sparse networks trained in isolation are generally less accurate than the corresponding dense networks. However, the lottery ticket hypothesis states that within a dense, randomly initialized network there exists a subnetwork that can be trained in isolation to achieve comparable accuracy to the original network. This subnetwork is called a *winning ticket* and it is obtained by pruning the original network after training and resetting the remaining weights to their initial values.

Iterative magnitude pruning (IMP) is a procedure to find winning tickets. Given an SGD procedure modelled by function: $\mathcal{A}^{s \to t} : \mathbb{R}^M \times U \to \mathbb{R}^M$ that maps weights $\theta_s \in \mathbb{R}^M$ at step s and

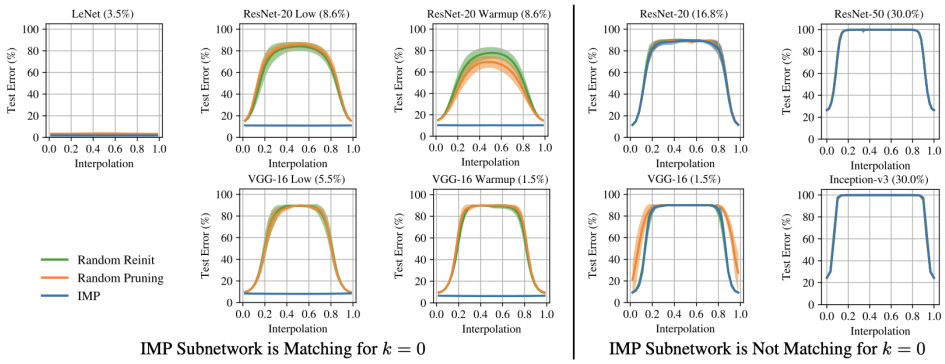


Figure 3.10: Linear interpolation test error when making two copies of the state of the network at initialization. Lines are means and standard deviations over three initializations and data orders.

SGD randomness $u \sim U$ to weights $\theta_t \in \mathbb{R}^M$ at step t by training for $t - s$ steps, IMP consists in the steps explained in Algorithm 1.

Algorithm 1 Iterative magnitude pruning (IMP)

1: **Input:** initial weights $\theta_0 \in \mathbb{R}^M$, initialize pruning mask to $m = 1^M$
2: **Train:** θ_0 to θ_k with noise $u \sim U$: $\theta_k = \mathcal{A}^{0 \to k}(\theta_0, u)$
3: **for** $n = 1$ to N **do**
4: Train $m \odot \theta_k$ to $m \odot \theta_T$ with noise $u' \sim U$: $\theta_T = \mathcal{A}^{k \to T}(m \odot \theta_k, u')$
5: Prune the lowest magnitude entries of θ_T that remain. Let $m[i] = 0$ if $\theta_T[i]$ is pruned.
6: **end for**
7: **Output:** winning ticket θ_k and mask m

This version of IMP is not the one originally proposed by Frankle and Carbin [11], but it is a more general version that allows to find winning tickets at any step of the training procedure. The original version of IMP corresponds to the case where $k = 0$.

Instability analysis of subnetworks at initialization. In this scenario they have found out that these subnetworks are only matching when they are stable to SGD noise as we can see in Figure 3.10.

Instability analysis of subnetworks during training. Frankle et al. [12] have found out that networks whose IMP subnetworks were stable when rewinding to iteration 0 remain stable at all other rewinding points. Furthermore, networks whose IMP subnetworks were unstable when rewinding to iteration 0 become stable when rewinding to later iterations. Crucially, the authors demonstrate that the exact point at which these sparse subnetworks become stable closely coincides with the point where they become matching (i.e., capable of training to the full accuracy of the unpruned network). This insight is well exemplified in Figure 3.11.

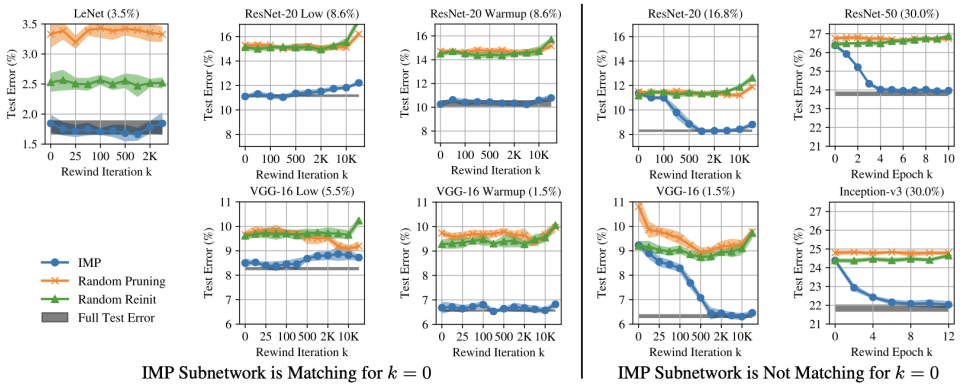


Figure 3.11: Linear interpolation test error when making two copies of the state of the network at a later training step. Lines are means and standard deviations over three initializations and data orders.

3.5 Summary and implications

The empirical and theoretical studies discussed in this chapter paint a coherent picture of the loss landscape of deep neural networks. Globally, the landscape is highly non-convex and rugged, resembling a spin-glass system where independent solutions are separated by high-loss barriers. However, locally, the geometry changes dramatically.

As demonstrated by Frankle et al. [12], when optimization trajectories share a common initialization point, whether it is a specific epoch k_c in the Lottery Ticket Hypothesis or a shared pre-trained checkpoint, the resulting minima lie in a region of parameter space that is approximately linearly connected, in the sense that the linear interpolation between solutions exhibits negligible loss/error barriers. Within this basin, solutions are linearly mode connected, meaning the direct linear path between them does not incur any loss penalty.

This geometric property is the fundamental enabler for the techniques we will explore in the remainder of this thesis. If parameter vectors obtained from a shared pre-trained model lie in a linearly connected subspace, we can safely perform linear algebraic operations on them without destroying their predictive capabilities. This intuition forms the theoretical foundation for **Task Arithmetic**, which we will formally define and analyze in the next chapter.

4. Task Arithmetic & Task Analogy

Changing how pre-trained models behave, for example for improving their performance on a specific task or mitigating biases or unwanted behaviors, is a common practice in deep learning. It is a field with a growing interest as demonstrated by recent works [38, 35, 32, 30, 29, 25]. In this chapter we will explore how we can edit a variety of models with **Task Arithmetic** as exposed by Ilharco et al. [18] , which is a technique that allows to perform arithmetic operations in parameter space using **task vectors**. Formally we define a task vector as follows:

Definition 4.0.1. Let $\theta_{\text{pre}} \in \mathbb{R}^M$ be a pre-trained model and $\theta_{\text{ft}}^t \in \mathbb{R}^M$ be the same model fine-tuned on a task t. The task vector $\tau_t \in \mathbb{R}^M$ is defined as: $\tau_t = \theta_{\text{ft}}^t - \theta_{\text{pre}}$. When the task is clear from the context we will simply refer to τ as the task vector.

4.1 Editing models with task arithmetic

We will explore three different arithmetic operations, but later on we will focus on the analogy setting, which is the most interesting one from a theoretical perspective. In Figure 4.1 we can see a visual representation of the three different arithmetic operations that can be performed with task vectors.

Learning via addition. The objective is to add more task vectors to obtain either a better performance on a specific task or to obtain a model that performs well on multiple tasks at the same time. From a theoretical standpoint, following Ortiz-Jimenez et al. [28], we can define the idealized mathematical behavior of perfect task addition through the following formal property:

Property 4.1.1. Consider a network $f : \mathcal{X} \times \Theta \to \mathcal{Y}$ and a set of task vectors $\mathcal{T} = \{\tau_t\}_{t\in[T]}$ with associated non-intersecting task supports $\mathcal{D} = \{\mathcal{D}_t \subset \mathcal{X}\}_{t\in[T]}$, i.e., $\forall t, t'$, if $t \neq t'$ then $\mathcal{D}_t \cap \mathcal{D}_{t'} = \emptyset$. We say a network f satisfies the task arithmetic property around θ_0 with respect to $\mathcal{T}$ and $\mathcal{D}$ if:

$$f\left(\mathbf{x}; \theta_0 + \sum_t^T \alpha_t \tau_t\right) = \begin{cases} f(\mathbf{x}; \theta_0 + \alpha_t \tau_t) & \mathbf{x} \in \mathcal{D}_t \\ f(\mathbf{x}; \theta_0) & \mathbf{x} \notin \bigcup_{t=1}^T \mathcal{D}_t \end{cases} \tag{4.1}$$

with $(\alpha_1, \alpha_2, \ldots, \alpha_T) \in \mathcal{A} \subseteq \mathbb{R}^T$

It is crucial to note that this property describes an idealized scenario of zero interference. In practice, due to the highly non-linear nature of deep neural networks and the inevitable overlap in real-world task data distributions, exact equality is rarely achieved, which leads to interference between tasks.

In Figure 4.2 we can see that adding more task vectors in CLIP leads to better multi-task vectors, which is consistent with the task arithmetic property.

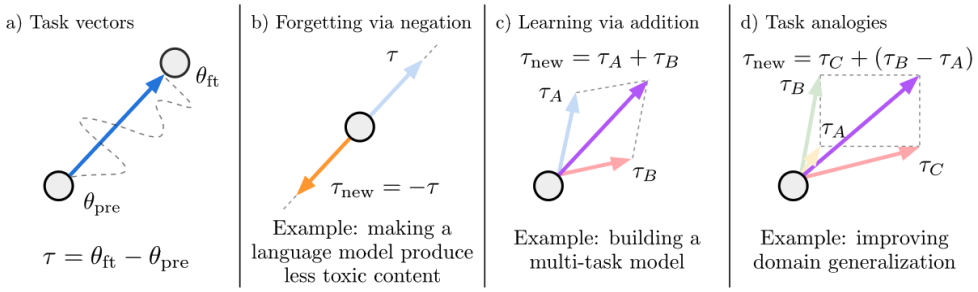


Figure 4.1: The three different arithmetic operations that can be performed with task vectors.

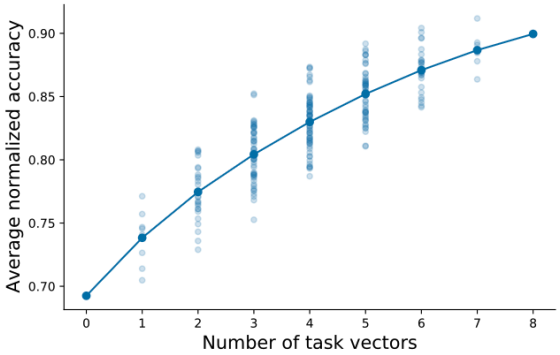


Figure 4.2: Experiments are done for vision tasks using CLIP on eight datasets (CARS, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397, SVHN). Accuracy is averaged across all tasks. When more task vectors are added, better multi-task vectors are built. Each point represents an experiment with a subset of the eight tasks.

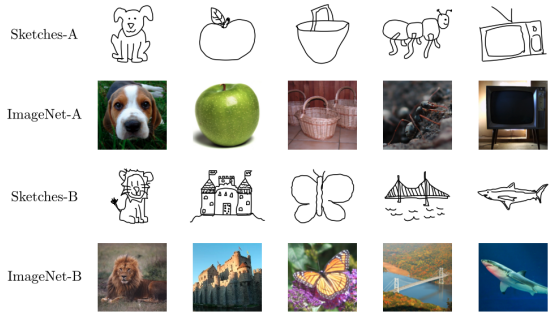
Forgetting via negation. The objective is to reduce the performance of a model on a specific task by subtracting the corresponding task vector, without altering the performance on other tasks. To evaluate the specificity of this operation, Ilharco et al. [18] measured the accuracy on *control tasks* as we can easily see in Figure 4.1.

Task analogies. The objective is to explore task analogies in the form "A is to B as C is to D" and to use these relationships to generate entirely new task vectors. This approach allows for the creation of task vectors that are not directly observed in the training data but are inferred from known domain shifts. For this reason, we can see Task Analogy as an advanced form of Transfer Learning [40], where the directional knowledge extracted from known tasks is algebraically transferred to infer the weights for a novel, unseen task (e.g., $\tau_D = \tau_C + (\tau_B - \tau_A)$).

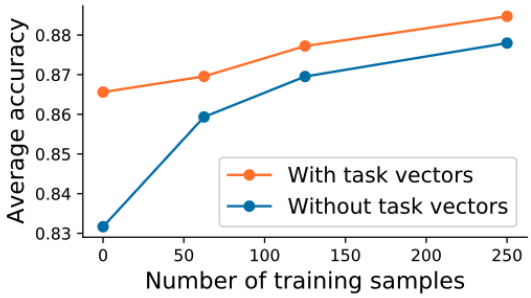
Because task analogy involves complex domain shifts and is highly sensitive to the geometric properties of the parameter space, a deeper theoretical and empirical analysis of this operation—along with novel methods to improve its robustness—will be extensively discussed in Chapter 5.

Table 4.1: Experiments are done for vision tasks using CLIP on eight datasets they wished to forget (CARS, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397, SVHN), and the control task (ImageNet). Negating task vectors reduces the performance on the target task while maintaining the performance on the control task.

Method	ViT-B/32		ViT-B/16		ViT-L/14	
	Target (↓)	Control (↑)	Target (↓)	Control (↑)	Target (↓)	Control (↑)
Pre-trained	48.3	63.4	55.2	68.3	64.8	75.5
Fine-tuned	90.2	48.2	92.5	58.3	94.0	72.6
Gradient ascent	2.73	0.25	1.93	0.68	3.93	16.3
Random vector	45.7	61.5	53.1	66.0	60.9	72.9
Negative task vector	24.0	60.9	21.3	65.4	19.0	72.9



(a) Experiments are done for vision task using 125 overlapping classes between ImageNet and a dataset of human sketches. The analogy is for example "real dog is to real lion as sketch dog is to sketch lion".



(b) Three different task vectors are generated by fine-tuning three models independently and then they are combined using the analogy. Accuracy is averaged over the four target subpopulations and three CLIP models.

4.1.1 From weight arithmetic to local linearity

The empirical effectiveness of task arithmetic [18] and weight-space interpolation [42] suggests that neural networks exhibit a surprisingly regular behavior along certain directions in parameter space. Interpolating between a pre-trained model and its fine-tuned counterpart $\theta_\alpha = (1-\alpha)\theta_{\text{zs}} + \alpha\theta_{\text{ft}}$ often results in smooth and approximately monotonic changes in loss as shown by Lucas et al. [24], despite the non-convexity of the optimization landscape.

The Monotonic Linear Interpolation (MLI) property, as defined by Lucas et al. [24], indicates that the direction from the pretrained to the fine-tuned model is such that movement in that direction directly translates to performance gains on the fine-tuning task.

Furthermore, Wortsman et al. [42] show that linearly interpolating between zero-shot and fine-tuned models can improve robustness under distribution shift, while preserving or even improving accuracy on the target distribution as shown in Figure 4.5.

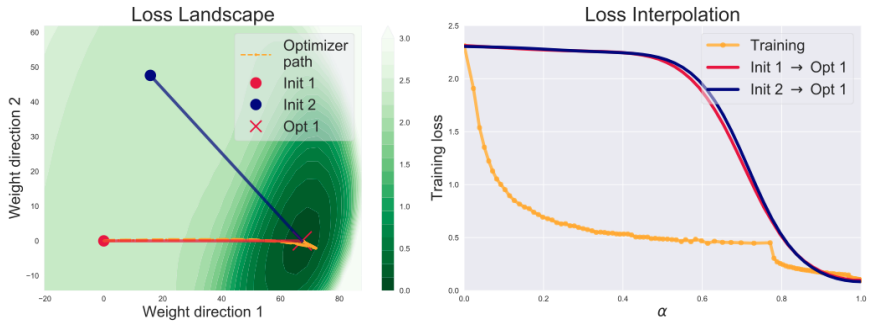


Figure 4.4: Monotonic linear interpolation (MLI) between a zero-shot and a fine-tuned ResNet-20 on CIFAR-10 from two different unrelated initializations to optimum. On the left there is a 2D slice of the loss landscape, defined by the two initializations and optimum, along with the optimization trajectory projected onto the plane. On the right, the interpolated loss curves are represented, with training loss shown relative to the proportion of distance travelled to the optimum.

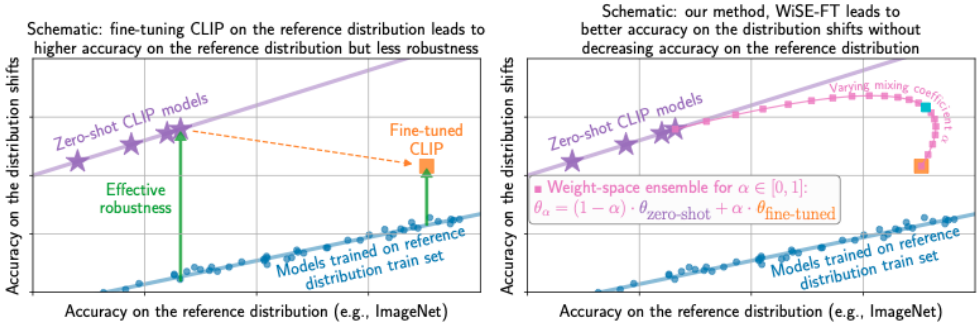


Figure 4.5: Weight-space interpolation between a zero-shot and a fine-tuned CLIP model on ImageNet. Interpolating between the two models can improve robustness under distribution shift, while improving accuracy on the target distribution.

These empirical findings are consistent with the hypothesis that, in a neighborhood of the pre-trained parameters, the functional behavior of the network can be well approximated by a first-order expansion, as formalized in the Neural Tangent Kernel (NTK) framework by Jacot et al. [20]. Under such a local linear approximation, parameter updates act as elements of a tangent space, and their composition corresponds to linear combinations of first-order functional variations.

This perspective motivates a transition from heuristic weight arithmetic to a principled formulation of model editing in tangent space, where task vectors are interpreted and combined through their induced first-order functional effects.

4.1.2 Task vectors as accumulated gradients

The empirical success of weight-space interpolation and task arithmetic can be rigorously explained by analyzing the optimization dynamics that generate task vectors. Recent work by Gargiulo et al. [47] provides a fundamental theoretical bridge, demonstrating that task vectors are not mere spatial

displacements, but are mathematically equivalent to accumulated gradients.

Specifically, they proved that a task vector τ, generated via standard gradient descent fine-tuning with learning rate η, corresponds to the accumulated negative gradient of the loss function over the optimization steps. Furthermore, they demonstrated that performing task addition diverges from the true trajectory of joint multi-task optimization only by a second-order error term $\mathcal{O}(\eta^2)$. Specifically, they have proved the following theorem (for the full proof see Appendix E):

Theorem 4.1.1. Let $\theta_{\text{TA}}^{(k)} = \theta_{\text{base}} + \alpha \sum_{t \in T} \tau_t^{(k)}$ be the model obtained using vanilla task arithmetic with scaling parameter α. Let $\{\theta_t^{(k)}\}_{t \in T}$ be produced by running k full-batch GD epochs with step size η on each task, and let $\theta_{\text{MT}}^{(k)}$ be the model obtained from k GD epochs with step size $\alpha\eta$ on the aggregated loss $\mathcal{L} = \sum_{t \in T} \overline{\ell}_t$. It holds that:

$$\theta_{\text{TA}}^{(1)} = \theta_{\text{MT}}^{(1)}$$
$$\theta_{\text{TA}}^{(k)} = \theta_{\text{MT}}^{(k)} + \eta^2 C(\{\theta_{\text{MT}}^{(j)}\}_{j=1}^{k-2}) + \mathcal{O}(\eta^3) \quad \text{for } k > 1$$

$$\text{where} \quad C(\{\theta_{\text{MT}}^{(j)}\}_{j=1}^{h}) = \sum_{t \in T} \sum_{e=0}^{h} \nabla^2 \ell_t(\theta_{\text{MT}}^{(e)}) \sum_{m=0}^{e} r_t(\theta_{\text{MT}}^{(m)})$$

$$\text{with} \quad r_t(\theta) := \alpha \sum_{t' \in T \setminus \{t\}} \nabla \ell_{t'}(\theta) + (\alpha - 1) \nabla \ell_t(\theta)$$

Moreover, they bounded the second-order error term $C(\{\theta_{\text{MT}}^{(j)}\}_{j=1}^{h})$ in feed-forward networks, assuming bounded weights and activation functions with bounded derivatives.

This formal equivalence is a cornerstone for understanding task arithmetic. It implies that when we perform addition, we are effectively linearly combining the gradient descent trajectories of the individual tasks.

In addition to this theoretical insight, Gargiulo et al. [47] empirically verified that task vectors remain highly effective even beyond the first epoch. This occurs because the initial gradient updates largely dictate the overall optimization trajectory. To demonstrate this, they compared the performance of task arithmetic under two conditions: merging models fine-tuned to full convergence versus merging models fine-tuned for just a single epoch.

Remarkably, the multi-task model obtained after only one epoch performed competitively with the fully converged counterpart, as shown in Figure 4.6. This empirical observation perfectly aligns with the theoretical findings: early task vectors are closer to the true task-loss gradients, prioritizing gradient approximation over task-specific overfitting. Consequently, for the purpose of multi-task model merging, performing a single epoch of fine-tuning is often sufficient to capture the essential geometric shift required for successful arithmetic composition.

To further understand this phenomenon, they empirically analyzed how much the first epoch of fine-tuning contributes to the final task vector. Specifically, Figure 4.7a illustrates the epoch-wise normalized gradient norms $\frac{\|\nabla_\theta \ell_t(\theta^{(k)})\|}{\sum_{k'=1}^{K} \|\nabla_\theta \ell(\theta^{(k')})\|}$ across different datasets. It is evident that the first epoch contributes the largest share to the total gradient norm, which is consistent with the theoretical derivations. Even in datasets where the first epoch's relative contribution is less pronounced, such

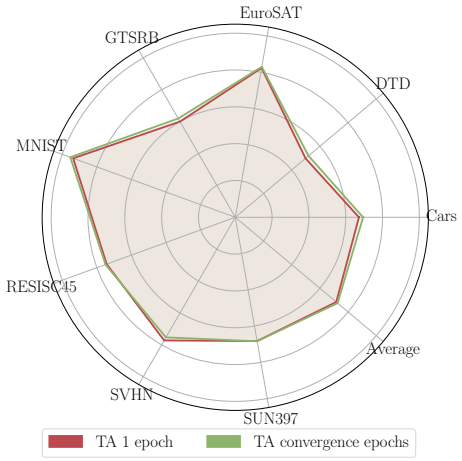


Figure 4.6: Comparison of multi-task accuracy obtained by merging models via task arithmetic. The multi-task model derived from merging single-epoch fine-tuned models performs competitively against the model obtained by merging fully converged models, highlighting the dominance of early gradient directions.

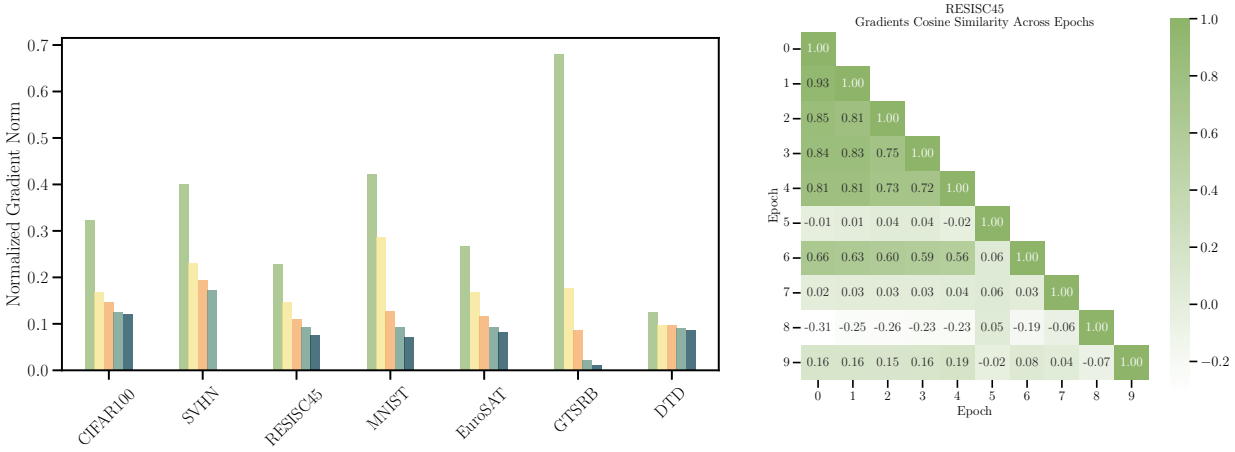
as RESISC45 and DTD, the task vector obtained after one epoch still dictates the direction of the entire optimization trajectory.

This is corroborated by computing the pairwise cosine similarity of gradients from the first 10 epochs. As shown in Figure 4.7b, the gradients from the subsequent 5 epochs maintain a high cosine similarity (> 0.8) with the initial epoch's gradient. This strong directional persistence confirms that fully trained task vectors favor task-specific performance through localized refinement, whereas first-epoch task vectors prioritize a pure gradient approximation. Both yield similar multi-task merging capabilities, but relying on the single-epoch task vector provides a highly efficient and theoretically grounded proxy for the true gradient trajectory.

4.2 Local linearity and the tangent space

As extensively discussed in Chapter 3, models optimized from a shared initialization tend to converge into the same linearly connected basin of attraction. In the specific context of transfer learning, Neyshabur et al. [27] provide a structural explanation for this geometric phenomenon. They show that models fine-tuned from pre-trained weights remain remarkably close in ℓ_2 distance within the parameter space. This bounded movement occurs because fine-tuning predominantly alters the higher layers to adapt to the target task, while the lower layers, which extract general, low-level features, remain strongly anchored to their pre-trained values.

From Figure 4.2 we can see that the similarity between the features of the pre-trained model and the fine-tuned model is higher in the lower layers than in the higher layers, which supports this hypothesis. This observation does not hold for models trained from random initialization. This structural property of fine-tuning suggests that the parameter space around the pre-trained model



(a) Normalized gradient norms after 5 epochs of finetuning.

(b) Pairwise cosine similarities of gradients from the first 10 epochs.

Figure 4.7: Analysis of first-epoch gradients demonstrating the optimization trajectory dominance.

Table 4.2: Feature similarity for different layers of a ResNet-50 on the target domain CheXpert. The similarity is measured with *Central Kernel Alignment* (CKA) as discussed in [21]. P-T stands for model trained/fine-tuned on target domain starting from pre-trained weights; P stands for pre-trained model; RI-T stands for model trained on target domain from random initialization.

models/layer	conv1	layer 1	layer 2	layer 3	layer 4
P-T & P	0.6225	0.4592	0.2896	0.1877	0.0453
P-T & P-T	0.6710	0.8230	0.6052	0.4089	0.1628
P-T & RI-T	0.0036	0.0011	0.0022	0.0003	0.0808
RI-T & RI-T	0.0016	0.0088	0.0004	0.0004	0.0424

is organized in a way that allows for local linear approximations to be effective, as the fine-tuned models do not deviate significantly from the pre-trained parameters, especially in the lower layers.

From the Neural Tangent Kernel (NTK) theory (for more details see Appendix B), we know that for a sufficiently wide neural network, the function it represents can be approximated by a linear function in the parameter space around its initialization. So it can be approximated with a first-order Taylor expansion as follows:

$$f(\mathbf{x};\theta) \approx f(\mathbf{x};\theta_0) + (\theta - \theta_0)^\top \nabla_\theta f(\mathbf{x};\theta_0) \tag{4.2}$$

There are many cases in which this local linear approximation is not accurate. In such cases we say that training occurs in *non-linear regime*. Conversely, when the local linear approximation is accurate, we say that training occurs in *linear regime*. The linear regime is more likely to occur when the model is sufficiently wide and when the fine-tuning process does not deviate significantly from the pre-trained parameters, which is often the case in transfer learning scenarios.

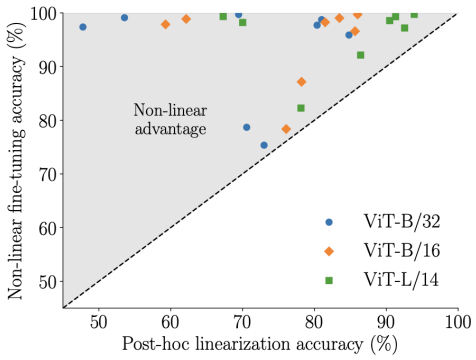


Figure 4.8: Single task accuracies of CLIP non-linearly fine-tuned $f(\cdot;\theta)$ and its post-hoc linearized version $f_{\text{lin}}(\cdot;\theta)$. Markers represent different ViT models. The target task are 8 datasets (CARS, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397, SVHN). The post-hoc linearized version of the model performs worse than the non-linear fine-tuned model.

4.2.1 Post-hoc linearization

In general, if a pre-trained network $f(\cdot;\theta_0)$ demonstrates kernel behavior during fine-tuning, the following property must hold:

Property 4.2.1. The change in the network output after training can be approximated by its first-order Taylor expansion: $f(\mathbf{x};\theta^*) - f(\mathbf{x};\theta_0) \approx (\theta^* - \theta_0)^\top \nabla_\theta f(\mathbf{x};\theta_0)$, where θ^* is the fine-tuned parameters and θ_0 is the pre-trained parameters.

In other words, the approximation of the network in the tangent space must hold after fine-tuning. We will call the post-hoc linearized version of f, f_{lin}. This is obtained by applying the fine-tuned task vector $\tau = \theta^* - \theta_0$ to the first-order Taylor expansion of f around θ_0:

$$f_{\text{lin}}(\mathbf{x};\tau) = f(\mathbf{x};\theta_0) + \tau^\top \nabla_\theta f(\mathbf{x};\theta_0) \tag{4.3}$$

The objective in Figure 4.8 is to verify whether the post-hoc linearized version f_{lin} can perform the same as the fine-tuned model $f(\cdot;\theta^*)$ on the target task. All the post-hoc linearized models in the figure perform worse than their non-linear counterparts. We call this advantage *non-linear advantage*.

It's important to note that even if the non-linear components are important for single task performance, they might not be crucial for the arithmetic properties of task vectors. For verifying this behavior, Ortiz-Jimenez et al. [28] performed task addition and task negation with both the post-hoc linearized version of the model and the non-linear fine-tuned version as done by Ilharco et al. [18]. In particular the experiments done are the following two:

- **Task addition.** The sum of the task vectors $\tau = \sum_t \tau_t$ is added to the pre-trained model and the performance of the resulting model is measured as the maximum average accuracy across all tasks. The results are shown in Table 4.3.

Table 4.3: **Task addition.** Average absolute (%) and normalized accuracies (%) of different CLIP 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	$f(\cdot\,;\,\theta_0)$	48.4	–	55.2	–	64.4	–
Non-lin. FT	$f(\cdot\,;\,\theta_0+\tau)$	71.4	76.5	75.5	80.0	85.1	88.8
Post-hoc lin.	$f_{\text{lin}}(\cdot\,;\,\theta_0+\tau)$	57.1	81.9	65.0	85.2	75.2	90.0
Linear. FT	$f_{\text{lin}}(\cdot\,;\,\theta_0+\tau_{\text{lin}})$	**76.5**	**85.4**	**81.3**	**86.0**	**88.5**	**93.5**

- **Task negation.** The task vector τ of a target task is negated from the pre-trained model, while retaining performance on a control task. The results are shown in Table 4.4.

In both cases, they have used a single mixing coefficient $\alpha = \alpha_1 = \cdots = \alpha_T$ optimized separately for the non-linear and the post-hoc linearized version of the model. The results confirm that task arithmetic properties don't rely only on the linear components of the model. Specifically it is possible to see a notable decrease in absolute performance in task addition when using the post-hoc linearized model and highlights that task arithmetic in non-linear models leverages the non-linear components.

Although these results reject the linear hypothesis, it is still remarkable that the post-hoc linearized version of the model performs better than the non-linear one in task negation. Furthermore, the post-hoc linearized version of the model performs better than the non-linear one in task addition by using the normalized performance, which can be formalized as follows.

Definition 4.2.1. The normalized performance of a model f on a task t is defined as the average ratio between the performance of f on the sum of all the T task vectors $\tau_1, \ldots, \tau_T$ and the performance of the fine-tuned model on t:

$$\text{Normalized accuracy} = \frac{1}{T}\sum_{t=1}^{T}\frac{\text{acc}_{\mathbf{x}\sim\mu_t}[f(\mathbf{x};\theta_0+\sum_{t'}\tau_{t'})]}{\text{acc}_{\mathbf{x}\sim\mu_t}[f(\mathbf{x};\theta_0+\tau_t)]}$$

So we can state that the post-hoc linearized version of the model is more consistent with the task arithmetic Property 4.1.1 than the non-linear one, even if it performs worse in absolute performance. This suggests that the non-linear components of the model are crucial for single task performance, but they might not be crucial for the arithmetic properties of task vectors.

4.2.2 Weight Disentanglement

So far we have seen that the linear regime is not necessary for task arithmetic properties to hold. But what are the necessary conditions that allow it? Ortiz-Jimenez et al. [28] hypothesize that a key condition for task arithmetic to hold is the disentanglement of the weight space, which can be formalized as follows.

Table 4.4: **Task negation.** Minimum accuracy (%) of different CLIP ViTs edited by negating a task vector from a target task while retaining 95% of their performance on the control task. We report average performances over eight tasks on non-linear and linearized models as introduced before.

Method		ViT-B/32		ViT-B/16		ViT-L/14	
		Targ. (↓)	Cont. (↑)	Targ. (↓)	Cont. (↑)	Targ. (↓)	Cont. (↑)
Pre-trained	$f(\cdot\,;\,\theta_0)$	48.4	63.4	55.2	68.3	64.4	75.5
Non-lin. FT	$f(\cdot\,;\,\theta_0 - \tau)$	24.0	60.7	19.2	64.6	18.0	**72.5**
Post-hoc lin.	$f_{\text{lin}}(\cdot\,;\,\theta_0 - \tau)$	14.8	60.3	**10.8**	**64.8**	12.1	71.8
Linear. FT	$f_{\text{lin}}(\cdot\,;\,\theta_0 - \tau_{\text{lin}})$	**10.9**	**60.8**	11.3	**64.8**	**7.9**	**72.5**

Definition 4.2.2. A parametric function $f : \mathcal{X} \times \Theta \to \mathcal{Y}$ is weight disentangled with respect to a set of task vectors $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_T\}$ and the corresponding support $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T\}$ if:

$$f\left(\mathbf{x}; \theta_0 + \sum_{t}^{T} \alpha_t \tau_t\right) = \sum_{t=1}^{T} f(\mathbf{x}; \theta_0 + \alpha_t \tau_t) \mathbb{1}_{(\mathbf{x} \in \mathcal{D}_t)} + f(\mathbf{x}; \theta_0) \mathbb{1}_{(\mathbf{x} \notin \bigcup_{t=1}^{T} \mathcal{D}_t)} \tag{4.4}$$

This definition captures the idea that the effect of each task vector τ_t on the output of the model is localized to its corresponding support $\mathcal{D}_t$, and that the effects of different task vectors do not interfere with each other. It is trivial to see that satisfying Property 4.2.2 implies satisfying Property 4.1.1, but the converse is not true. It is important to highlight that weight disentanglement doesn't imply that the model is linear.

Another important observation is that even if a model is well weight disentangled, it might be still perform poorly on single tasks. This is because the performance of a model on a task is not only determined by the disentanglement of the weight space.

It is possible to visualize the level of weight disentanglement of a model by visualizing the weight disentanglement error, defined as follows.

Definition 4.2.3. Given two tasks the weight disentanglement error ξ is defined as the expected distance between the output of the model when both task vectors are added and the output of the model when only one task vector is added:

$$\xi(\alpha_1, \alpha_2) = \sum_{t=1}^{2} \mathbb{E}_{\mathbf{x} \sim \mu_t}[\text{dist}(f(\mathbf{x}; \theta_0 + \alpha_t \tau_t), f(\mathbf{x}; \theta_0 + \alpha_1 \tau_1 + \alpha_2 \tau_2))] \tag{4.5}$$

where dist is any distance metric between the outputs of the model. Since we are dealing with classification tasks we use the following notion of distance: $\text{dist}(y_1, y_2) = \mathbb{1}_{(y_1 \neq y_2)}$.

Consequently, a lower weight disentanglement error indicates a higher degree of disentanglement at the scaling points (α_1, α_2). In Figure 4.9 we can see the weight disentanglement error for a CLIP ViT-B/32 model fine-tuned on different example task pairs. The post-hoc linearized version of

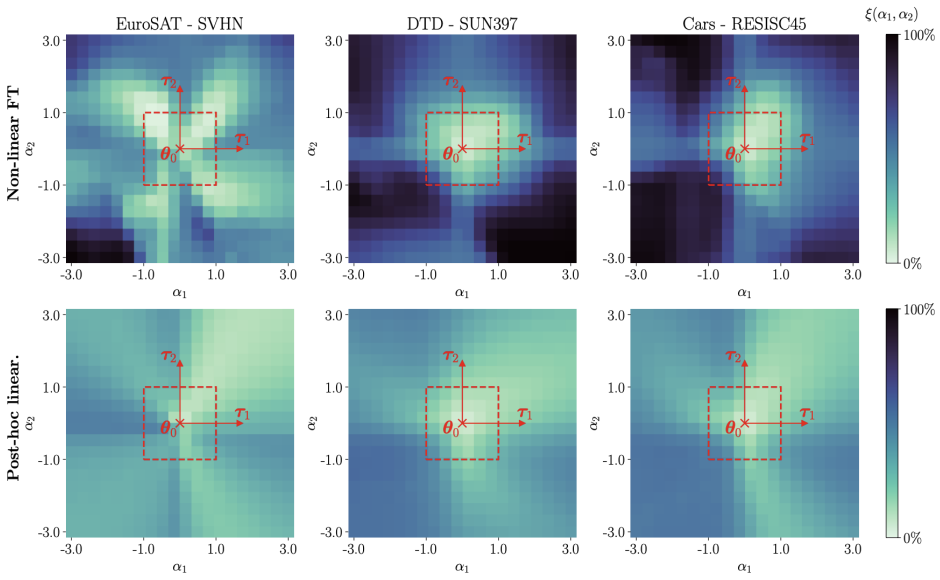


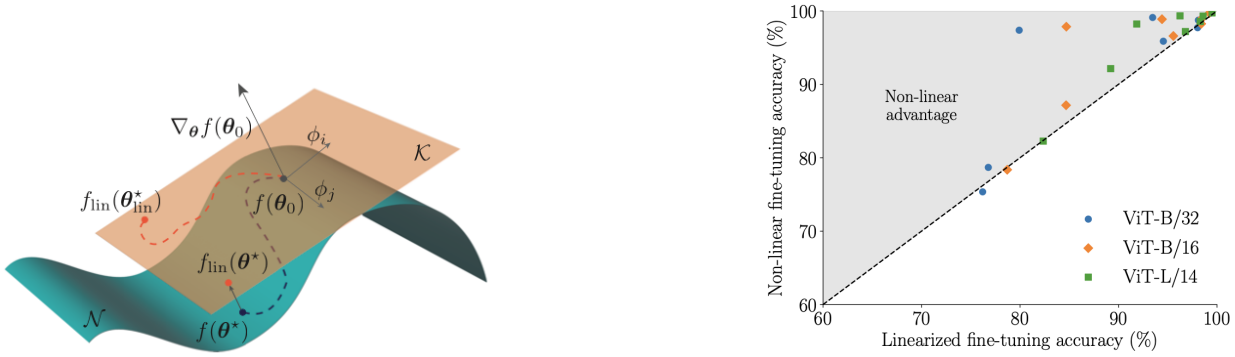
Figure 4.9: Weight disentanglement error ξ for a CLIP ViT-B/32 non-linear model and its post-hoc linearized version on different example task pairs. The model is more weight disentangled when the mixing coefficients α_1 and α_2 are close to 0, which means that the model is more weight disentangled when it is closer to the pre-trained model.

the model is more weight disentangled than the non-linear one, which suggests that the non-linear components of the model are not crucial for weight disentanglement. It is possible to see that the error increases when the mixing coefficients α_1 and α_2 are above 1. This explains why post-hoc linearization achieved better performance in task addition when using the normalized performance and managed to forget more through task negation, which are sensitive to the weight disentanglement error.

4.2.3 Enhancing task arithmetic with linearization

Ortiz-Jimenez et al. [28] have shown that the post-hoc linearized version of the model is more consistent with the task arithmetic properties than the non-linear one, but it performs worse in absolute performance. For this reason, instead of applying the non-linear task vector $\tau = \theta^* - \theta_0$ to f_{lin}, they have proposed to directly obtain the task vectors through explicit linear fine-tuning as illustrated in Figure 4.10a. So the fine-tuning process can follow the same protocols used before but with the network parametrization dictated by Eq. (4.3). In this way, it is possible to obtain task vectors $\tau_{\text{lin}} = \theta^*_{\text{lin}} - \theta_0$.

It is important to note that computing the Jacobian $\nabla_\theta f(\mathbf{x}; \theta_0)$ is computationally expensive, especially for large models, because it has a size of the number of parameters multiplied by the embedding space dimensionality. For this reason, Ortiz-Jimenez et al. [28] have proposed to use the fast Jacobian-vector product (JVP) to compute the output of the linearized model without explicitly computing the Jacobian (for more details see Appendix C).



(a) Tangent space concept.

(b) Single-task accuracies.

Figure 4.10: **Linearized fine-tuning and performance.** **(a)** Linear fine-tuning consists in optimizing the task vector τ directly in the tangent space $\mathcal{K}$ of the pre-trained model $f(\cdot;\theta_0)$, which is equivalent to optimizing a linearized version of the model f_{lin}. Here $\mathcal{N}$ represents the space of neural network functions non-linearly parametrized by $\theta \in \Theta$. **(b)** Single-task accuracies of non-linearly fine-tuned $f(\cdot;\theta^*)$ and linearly fine-tuned $f_{\text{lin}}(\cdot;\theta^*_{\text{lin}})$ CLIP ViTs. By explicitly optimizing the task vector for the linearized model rather than applying a post-hoc expansion, linearized fine-tuning inherently surpasses standard approximations and effectively neutralizes the non-linear advantage.

As the considered model do not exhibit inherently a linear behavior during fine-tuning, the linear fine-tuning process yields significant different results compared to post-hoc linearization: $f_{\text{lin}}(\mathbf{x};\theta_0+\tau) \neq f_{\text{lin}}(\mathbf{x};\theta_0+\tau_{\text{lin}})$. Despite sharing the same kernel formulation, linearized fine-tuning actively restricts the optimization trajectory within the tangent space, directly tuning the vector τ_{lin} for the linearized architecture. By construction, this active optimization yields strictly superior results compared to post-hoc linearization. Notably, as demonstrated by Ortiz-Jimenez et al. [28], this methodology effectively neutralizes the traditional advantage of non-linear models, bringing the performance of the linearized model on par with standard non-linear fine-tuning as we can see also in Figure 4.10b. Another interesting insight is that bigger models tend to benefit more from linearized fine-tuning, which suggests that the linear regime is more likely to occur in bigger models, as we can expect from NTK theory.

This boost in single-task performance doesn't compromise the weight disentanglement error, which remains the same as the post-hoc linearized version of the model as we can see in Figure 4.11. Moreover in Table 4.3 and Table 4.4 we can see that the linearly fine-tuned model performs better than the non-linear one in both task addition and task negation, even in absolute performance. In particular the linearized fine-tuned models achieve higher multi task accuracies through task addition and can forget more through task negation, while maintaining a similar level of accuracy on the control task. This suggests that linearized fine-tuning is a promising approach for enhancing task arithmetic properties while maintaining or even improving single-task performance.

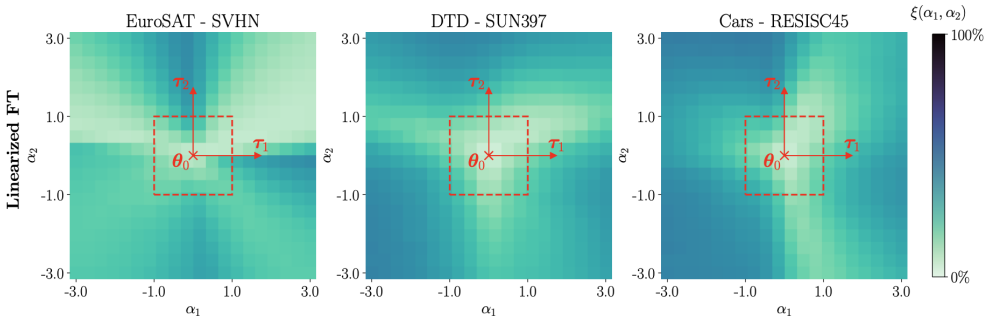


Figure 4.11: Heatmaps illustrate the disentanglement error $\xi(\alpha_1, \alpha_2)$ for a ViT-B/32 linearly fine-tuned across different task pairs. Lighter areas indicate regions in the parameter space with stronger disentanglement. The red bounding box delineates the search space evaluated to compute α in our experiments.

4.2.4 Towards understanding task arithmetic

Generally (see the Appendix D for more details), a kernel admits a decomposition in terms of its eigenfunctions and eigenvalues $\{(\Phi_\rho, \lambda_\rho)\}_{\rho \in \mathbb{N}}$. This implies that k can represent functions of the form $f(\mathbf{x}) = \sum_{\rho=1}^{\infty} c_\rho \Phi_\rho(\mathbf{x})$ with a finite kernel norm: $\|f\|_{\mathcal{H}}^2 = \sum_{\rho=1}^{\infty} \frac{c_\rho^2}{\lambda_\rho} < \infty$. Specifically, $\{c_\rho\}_{\rho \in \mathbb{N}}$ is a representation of the function f in the eigenbasis of the kernel $\{\Phi_\rho\}_{\rho \in \mathbb{N}}$. The following proposition establishes a sufficient condition for the task arithmetic property, based on how tasks are represented in this basis.

Proposition 4.2.1. Suppose that the task functions $\{f_t^*\}_{t \in [T]}$ belong to the RKHS of the kernel k and their coefficients in the kernel eigenbasis are $\{(c_{t,\rho}^*)_{\rho \in \mathbb{N}}\}_{t \in [T]}$. If $\forall t, \rho$ either $c_{t,\rho}^* = 0$ or $\text{supp}(\Phi_\rho) \subseteq \mathcal{D}_t$, then the kernel k has the task arithmetic property with respect to the tasks $\{f_t^*\}_{t \in [T]}$ and the corresponding supports $\{\mathcal{D}_t\}_{t \in [T]}$.

We can sum up the Proposition 4.2.1 by saying that if each task is represented with eigenfunctions that vanishes outside the support of the task, the functions corresponding to different tasks do not interfere with each other and the task arithmetic property holds. Ortiz-Jimenez et al. [28] have proved also a more general condition for task arithmetic to hold, which is the following.

Proposition 4.2.2. Suppose that the task functions $\{f_t^*\}_{t \in [T]}$ belong to the RKHS of the kernel k and their coefficients in the kernel eigenbasis are $\{(c_{t,\rho}^*)_{\rho \in \mathbb{N}}\}_{t \in [T]}$. Furthermore, let the kernel eigenfunctions be either zero or linearly independent over each domain $\mathcal{D}_t$. The kernel k has the task arithmetic property with respect to the tasks $\{f_t^*\}_{t \in [T]}$ and the corresponding supports $\{\mathcal{D}_t\}_{t \in [T]}$ if and only if $\forall t, \rho$ either $c_{t,\rho}^* = 0$ or $\text{supp}(\Phi_\rho) \subseteq \mathcal{D}_t$.

It is crucial to understand that these conditions are not satisfied with random initializations. Indeed we know that when using a simple network as a fully connected one or a convolutional one and when the data distribution is uniform on a ring or a torus, the NTK can be diagonalized with Fourier series, so with a series of sine and cosine functions as well studied by Basri et al.

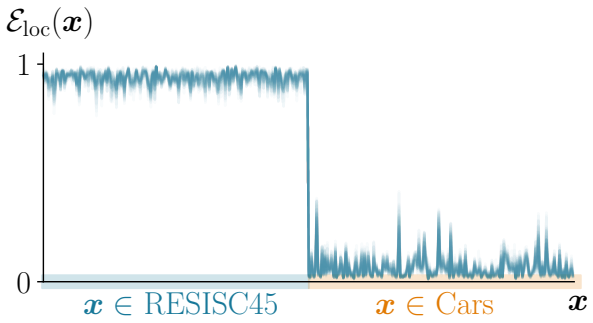


Figure 4.12: Estimated support of the eigenfunctions of the NTK of a ViT-B/32 CLIP model trained on RESISC45. The plot shows the sum of the local energy of the eigenfunctions over a random subset of the training and control supports (RESISC45 and Cars, respectively).

[5]. These functions are linearly independent over any domain but are not localized. Consequently, according to Proposition 4.2.2, these architectures cannot perform task arithmetic within such settings. This means that the task arithmetic properties of pre-trained models are not a consequence of the architecture itself, but rather a consequence of the pre-training process, which shapes the kernel in a way that allows for task arithmetic to hold.

To explore whether CLIP models use localized eigenfunctions for task arithmetic, Ortiz-Jimenez et al. [28] have performed a spectral analysis of the matrix of the kernel of CLIP ViTs $(K_{NTK})_{ij} = k_{NTK}(\mathbf{x}_i, \mathbf{x}_j)$ with $\mathbf{x}_i \in \mathcal{D}_t$, where t is the task on which they trained and $\mathbf{x}_j \in \mathcal{D}_t \cup \mathcal{D}_{t'}$, where $\mathcal{D}_{t'}$ is the support of a control task. They have introduced the concept of *local energy* $\varepsilon_{\text{loc}}(\mathbf{x}) = \sum_\rho \Phi_\rho^2$ to measure the power of the eigenvectors of k_{NTK} in the points of the dataset used for training.

As illustrated in Figure 4.12, analyzing a ViT-B/32 CLIP model fine-tuned on the RESISC45 dataset reveals a striking phenomenon. When comparing the target task against a control dataset (such as Cars), the local energy of the task-specific eigenfunctions is massively concentrated on the data points belonging to the training domain. This sharp drop in energy outside the target support provides empirical confirmation that pre-trained CLIP models naturally discover and exploit localized eigenfunctions. Consequently, the network can effectively disentangle the representations of different tasks, allowing task-specific operations (like addition and negation) to be carried out algebraically without catastrophic interference.

While empirical evidence highlights the crucial role of localized eigenfunctions, a subtle mathematical nuance exists regarding their strict necessity. Theoretically, the task arithmetic property could still hold even if individual eigenfunctions are not perfectly localized, provided that their linear combination perfectly cancels out outside the target domain.

By definition, the eigenfunctions of the NTK are globally linearly independent over the entire input space. However, they do not necessarily maintain this linear independence when evaluated on a restricted sub-domain (i.e., a single task dataset). If the eigenfunctions *do* maintain their linear independence on each individual task domain (a property known as local linear independence), then strict spatial localization becomes an absolute mathematical requirement for task arithmetic.

Conversely, if a network possesses locally linearly independent but non-localized eigenfunctions,

task arithmetic cannot work. A prime example of this failure mode occurs in neural networks at random initialization. As discussed in Appendix B, the NTK eigenfunctions of un-trained networks often take the form of global Fourier atoms. Since Fourier bases are locally independent but oscillate globally across the entire space without localization, zero-shot task addition is mathematically impossible in such models, further proving that task arithmetic is an emergent property of the pre-training phase.

4.2.5 Predicting weight disentanglement in practice with τJp

As discussed in Section 4.2.2, weight disentanglement is the fundamental property that allows task arithmetic to succeed without catastrophic interference. However, evaluating the exact disentanglement error $\xi(\alpha_1, \alpha_2)$ for every possible combination of tasks requires instantiating multiple models and evaluating them on their respective validation sets. In scenarios involving a massive number of tasks, this brute-force evaluation becomes computationally prohibitive.

For this reason, Yoshida et al. [45] have proposed a new metric called τJp (Task Vector Jacobian Product) which is based on the product of the task vector τ and the Jacobian of the pre-trained model with respect to its weights.

They have started from a task arithmetic involving two tasks τ_A and τ_B. In the NTK regime, the model's output can be approximated as:

$$f(\mathbf{x}; \theta_0 + \alpha_A\tau_A + \alpha_B\tau_B) \approx f(\mathbf{x}; \theta_0) + \alpha_A\tau_A^\top \nabla_\theta f(\mathbf{x}; \theta_0) + \alpha_B\tau_B^\top \nabla_\theta f(\mathbf{x}; \theta_0) \tag{4.6}$$

with $\alpha_A, \alpha_B \in \mathbb{R}$. In this case for inputs $\mathbf{x}_A$ and $\mathbf{x}_B$ belonging to the supports of task A and task B respectively, achieving a weight disentanglement error $\xi(\alpha_A, \alpha_B) = 0$ requires that the following condition holds:

$$f(\mathbf{x}_A; \theta_0 + \alpha_A\tau_A + \alpha_B\tau_B) \approx f(\mathbf{x}_A; \theta_0) + \alpha_A\tau_A^\top \nabla_\theta f(\mathbf{x}_A; \theta_0) + \mathbf{0} \approx f(\mathbf{x}_A; \theta_0 + \alpha_A\tau_A) \tag{4.7}$$

$$f(\mathbf{x}_B; \theta_0 + \alpha_A\tau_A + \alpha_B\tau_B) \approx f(\mathbf{x}_B; \theta_0) + \mathbf{0} + \alpha_B\tau_B^\top \nabla_\theta f(\mathbf{x}_B; \theta_0) \approx f(\mathbf{x}_B; \theta_0 + \alpha_B\tau_B) \tag{4.8}$$

Equations (4.7) and (4.8) suggest that the weight disentanglement error $\xi(\alpha_A, \alpha_B)$ is closely related to the magnitude of the Jacobian products $\tau_A^\top \nabla_\theta f(\mathbf{x}_B; \theta_0)$ and $\tau_B^\top \nabla_\theta f(\mathbf{x}_A; \theta_0)$. In particular they imply the following conditions:

$$\tau_A^\top \nabla_\theta f(\mathbf{x}_B; \theta_0) = \mathbf{0}$$
$$\tau_B^\top \nabla_\theta f(\mathbf{x}_A; \theta_0) = \mathbf{0}$$

So, the task vector of a given task must be orthogonal to the Jacobian of the pre-trained model evaluated on the data points of the other task. Based on this insight, Yoshida et al. [45] have proposed to use the following metric to predict the weight disentanglement error $\xi(\alpha_A, \alpha_B)$:

$$\tau\text{Jp} = \frac{1}{2}\left(\|\tau_A^\top \nabla_\theta f(\mathbf{x}_B; \theta_0)\|^2 + \|\tau_B^\top \nabla_\theta f(\mathbf{x}_A; \theta_0)\|^2\right) \tag{4.9}$$

According to the theoretical analysis, a lower τJp value should correspond to a lower weight disentanglement error $\xi(\alpha_A, \alpha_B)$, which in turn should correspond to better performance in task

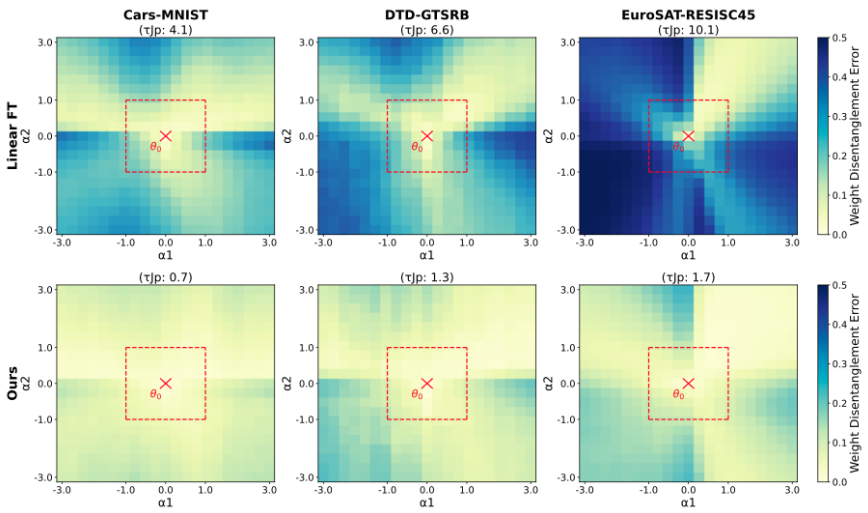


Figure 4.13: Visualization of weight disentanglement in a ViT-B/32 model with respect to τJp. The top row illustrates the baseline linearized model without regularization, while the bottom row displays the model fine-tuned with the proposed τJp penalty. The heatmaps reveal a clear correlation: a large τJp makes weight disentanglement highly sensitive to the mixing coefficients, whereas a lower τJp yields significantly greater robustness. Notably, the proposed regularization effectively expands this robust region across the coefficient space. The red cross at the origin marks the pre-trained model, and the red bounding box delineates the standard search space for task arithmetic coefficients.

arithmetic operations. This fact has been empirically verified by them as we can see in Figure 4.13. In particular, they have shown that as τJp decreases, the error tends to become more robust to changes in the mixing coefficients α_A and α_B, which is consistent with the theoretical analysis.

They have also analyzed the correlation between normalized accuracy and τJp. The results are presented in Figure 4.14. Each data point represents a model trained by performing task addition on two out of the eight image tasks. Normalized accuracy here is defined as the accuracy of each task after applying task arithmetic relative to its accuracy before task arithmetic. Across all models scaled, they have observed a consistent trend where a lower τJp value corresponds to a higher normalized accuracy, which confirms that τJp is a good predictor of the performance of task arithmetic operations.

τJp regularization

Based on the insights provided by τJp, Yoshida et al. [45] have proposed a novel regularization technique for fine-tuning models that explicitly minimizes τJp during the training process. The objective is to promote learning to occur in a subspace where τ is orthogonal to the Jacobian of the pre-trained model evaluated on the data points of other tasks. Inspired by this requirement, they

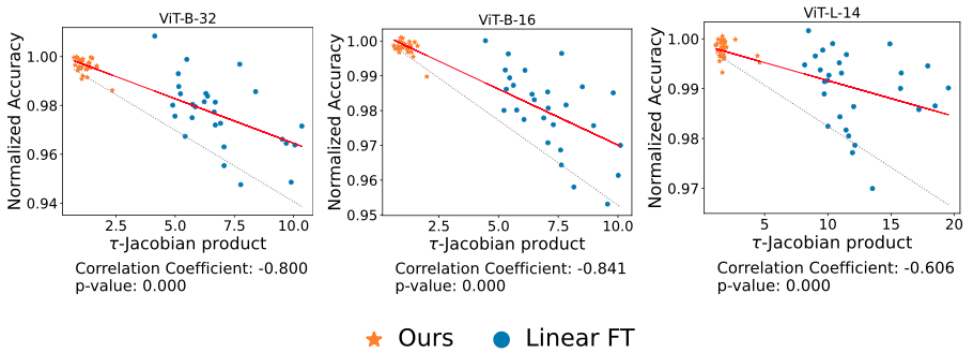


Figure 4.14: Relationship between τJp and normalized accuracy in task addition. The scatter plot displays all 28 pairwise combinations derived from the usual set of eight distinct tasks. A clear inverse correlation emerges: lower τJp values consistently correspond to higher normalized accuracies. Blue dots represent the standard linearized task addition baseline, whereas orange stars indicate task vectors fine-tuned with the proposed τJp penalty. The distinct separation between the two clusters demonstrates that the proposed regularization successfully minimizes τJp, directly mitigating task interference and improving overall multi-task performance.

have proposed the following τJp-based regularized loss function:

$$\mathcal{L}_{\tau\text{Jp}}(f_{\text{lin}}(\mathbf{x};\theta_0),y) = \mathcal{L}(f_{\text{lin}}(\mathbf{x};\theta_0),y) + \lambda \sum_{t\in T_{\text{orth}}} \|(\theta-\theta_0)^\top \nabla_\theta f_{\text{lin}}(\mathbf{x}_t;\theta_0)\|^2 \tag{4.10}$$

where T_{orth} denotes the set containing indices of other tasks for which we aim to suppress interference. The hyperparameter λ controls the strength of the regularization, and it can be tuned to find the right balance between minimizing the original loss and minimizing τJp.

However, in practical applications, when there are numerous tasks, computing the Jacobian $\nabla_\theta f_{\text{lin}}(\mathbf{x}_t;\theta_0)$ for every task can be computationally expensive. To address this issue, Yoshida et al. [45] have proposed to use data from one task for each iteration of training, which significantly reduces the computational burden while still effectively minimizing τJp across all tasks over the course of training:

$$\mathcal{L}^{(i)}_{\tau\text{Jp}}(f_{\text{lin}}(\mathbf{x};\theta_0),y) = \mathcal{L}(f_{\text{lin}}(\mathbf{x};\theta_0),y) + \lambda\|(\theta-\theta_0)^\top \nabla_\theta f_{\text{lin}}(\mathbf{x}_{i \bmod |T_{\text{orth}}|};\theta_0)\|^2 \tag{4.11}$$

where i is the iteration index. They have also done an empirical comparison between the two methods (4.10) and (4.11), showing that the cyclic method achieves comparable performance to the strict method while being significantly more computationally efficient. In Table 4.5 we can see the results of the comparison between the two approaches and linear fine-tuning without regularization using ViT-B-32. From the perspective of accuracy, the strict regularization slightly outperforms the cyclic one, indicating that the strict implementation of the proposed regularization can nearly eliminate task interference, but the difference is not significant. In terms of computation times, it achieves an 80% reduction. Furthermore, despite having a runtime comparable to linear fine-tuning, it demonstrates a significant improvement in performance.

Table 4.5: Comparison of the strict regularization and the efficient regularization in task addition on ViT-B-32

Method	Abs. (↑)	Norm. (↑)	Sec. / Iter.(↓)
No reg. (Linear FT)	74.3	85.0	0.361
Cyclical reg. (4.11)	84.5	97.6	0.374
Strict reg. (4.10)	86.4	99.3	2.027

Table 4.6: Performance of task addition across the usual eight vision tasks. The 'Task vector coef.' column specifies the method used to determine coefficients; '1.0' denotes that all coefficients remained fixed at unity, with no further adjustment.

Method	Task vector coef.	ViT-B-32		ViT-B-16		ViT-L-14	
		Abs. (↑)	Norm. (↑)	Abs. (↓)	Norm. (↑)	Abs. (↓)	Norm. (↑)
Pre-trained	-	47.3	-	54.5	-	65.1	-
Individual	-	89.9	-	92.2	-	93.7	-
MTL	-	87.8	-	90.8	-	92.6	-
Non-lin. FT (Ilharco et al., [18])	1.0	19.9	20.5	19.1	19.7	37.6	39.0
	Grid-searched	70.4	78.0	75.5	81.5	84.0	89.3
Linear FT (Ortiz-Jimenez et al., [28])	1.0	55.4	61.7	58.2	63.6	80.5	86.7
	Grid-searched	74.3	85.0	78.7	87.6	85.8	92.8
Ties-Merging (Yadav et al., [43])	1.0	74.2	84.8	78.6	87.6	85.0	91.9
	Grid-searched	74.2	84.8	78.6	87.6	85.0	91.9
AdaMerging (Yang et al., [44])	Trained	80.1	88.5	84.9	92.1	**90.8**	96.4
Ours	1.0	84.2	97.2	87.5	98.4	**90.8**	**99.0**
	Grid-searched	**84.5**	**97.6**	**87.6**	**98.5**	**90.8**	**99.0**

Yoshida et al. [45] have conducted experiments to evaluate the effectiveness of τJp regularization in improving task arithmetic performance comparing their method with standard fine-tuning as done by Ilharco et al. [18], linear fine-tuning as done by Ortiz-Jimenez et al. [28], Ties-Merging as done by Yadav et al. [43] and AdaMerging as done by Yang et al. [44]. The results of the experiments are reported in Table 4.6 for task addition and in Table 4.7 for task negation. In both cases, their method outperforms the baselines in terms of both absolute and normalized performance, demonstrating that τJp regularization is an effective technique for enhancing task arithmetic properties while maintaining or even improving single-task performance.

4.3 Task analogy

4.3.1 The geometry of directional shifts

Task analogy formalizes the intuitive concept of "A is to B as C is to D" within the high-dimensional weight space of neural networks. Instead of merely combining skills, analogies allow for the *zero-shot* generation of entirely new task vectors by transferring a learned relationship from a source

Table 4.7: Results of task negation for the usual eight tasks. We report the minimum accuracy achieved on target tasks while preserving at least 95% of the pre-trained model's performance on control tasks. Values in parentheses (·) denote reference cases where control task accuracy fell below the 95% threshold. Our approach demonstrates superior forgetting of target tasks while maintaining higher control task performance relative to existing methods.

Method	Task vector coef.	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 (Ilharco et al., [18])	1.0	(10.9)	(44.7)	(10.8)	(51.6)	(15.2)	(68.6)
	Grid-searched	24.0	60.7	20.3	64.7	18.4	72.4
Linear FT (Ortiz-Jimenez et al., [28])	1.0	(6.3)	(57.2)	(5.4)	(62.2)	(3.0)	(67.9)
	Grid-searched	11.8	60.6	8.8	65.0	8.3	72.2
Ties-Merging (Yadav et al., [43])	1.0	21.8	61.6	24.3	67.0	26.6	74.4
	Grid-searched	21.8	61.7	24.3	67.0	26.6	74.4
Ours	1.0	11.8	**62.5**	11.8	**67.8**	15.1	**75.1**
	Grid-searched	**6.7**	60.8	**4.7**	66.0	**3.7**	73.0

domain to a target domain.

Formally, if we have a pre-trained model θ_0 and four distinct but relationally connected tasks (A, B, C, and D), we can define their respective task vectors as $\tau_i = \theta_i - \theta_0$. The task analogy hypothesizes that the geometric transformation required to move from A to B is parallel to the transformation required to move from C to D. Consequently, the unseen task vector τ_D can be approximated algebraically without any direct optimization on the dataset $\mathcal{D}_D$:

$$\tau_D \approx \tau_C + (\tau_B - \tau_A) \tag{4.12}$$

In this formulation, the term $\Delta = \tau_B - \tau_A$ acts as a **directional shift vector**. It encodes pure, domain-agnostic transformation rules (e.g., changing a background, altering a style, or shifting a modality) decoupled from the specific semantic content of the original tasks.

4.3.2 Real-world applications

Initial demonstrations of task analogies by Ilharco et al. [18] relied on specific, curated subsets of data—such as inferring the weights for "indoor lions" by extracting the spatial shift from "outdoor dogs" to "indoor dogs". However, recent advancements have proven that this algebraic property extends far beyond semantic visual concepts, successfully capturing complex, systemic shifts across various modalities.

To fully grasp the potential of task analogies, it is highly instructive to observe how the shift vector has been recently deployed to solve major open problems in modern deep learning:

- **Mitigating the synthetic-to-real gap:** Synthetic data has long been used for training automatic speech recognition (ASR) models due to its cost-effectiveness and scalability for the

availability of text-to-speech systems. However these models have been plagued by a significant performance drop when evaluated on real-world recordings because of the distributional shift, commonly referred to as the "synthetic-to-real gap". For this reason, Su et al. [39] proposed to leverage task analogy to bridge this gap by learning the directional shift from a synthetic ASR model to a real ASR model and applying it to a new synthetic ASR model, effectively enabling zero-shot adaptation to real-world data without any direct training on it. Specifically, they have firstly trained two ASR models: one on synthetic data (θ^S_{syn}) and another on real data (θ^S_{real}). The acoustic disparity between real and synthetic speech is quantified by subtracting the parameter sets of these models:

$$\tau = \theta^S_{real} - \theta^S_{syn}$$

Then, this shift vector τ is applied to a new synthetic ASR model (θ^T_{syn}) to generate a new model ($\theta^T_{new\_real}$) that is expected to perform well on real data:

$$\theta^T_{syn\_new} = \theta^T_{syn} + \lambda\tau$$

where λ is a scaling factor that controls the magnitude of the shift. Moreover, in scenarios where multiple source models are available, they have also proposed to learn multiple shift vectors and combine them using the average to further enhance the performance of the new model on real data:

$$\tau_i = \theta^{S_i}_{real} - \theta^{S_i}_{syn}; \qquad \theta^T_{syn\_new} = \theta^T_{syn} + \frac{\lambda}{|S|}\sum_{i=1}^{|S|}\tau_i$$

This approach has shown significant improvements in ASR performance on real-world recordings, demonstrating the effectiveness of task analogy in addressing the synthetic-to-real gap.

- **Zero-shot quantization:** Deep Learning models are typically trained in high-precision floating-point formats, which can be computationally expensive and memory-intensive for deployment on resource-constrained devices. Quantization is a common technique to reduce the precision of model parameters, thereby decreasing the model size and inference latency. In particular, among all quantization methods, post-training quantization (PTQ) is widely used due to its simplicity and effectiveness. However, its convenience comes at a cost: the quantized model often suffers from a significant drop in performance, especially when quantizing to very low bit-widths (e.g., 3-bit). Quantization aware training (QAT) proposed by Jacob et al. [19] mitigates the performance degradation by exposing the model to PTQ effects during optimization, allowing it to adapt to the perturbations induced by extreme low-bit quantization. This method is very effective, yielding quantized models that closely match the performance of their full-precision counterparts. However it is more demanding than PTQ, as it requires training and computational resources that may not be available in all scenarios. For this reason, Solombrino et al. [37] proposed to leverage task analogy to achieve zero-shot quantization. They have firstly trained two models: one in full precision and another in low

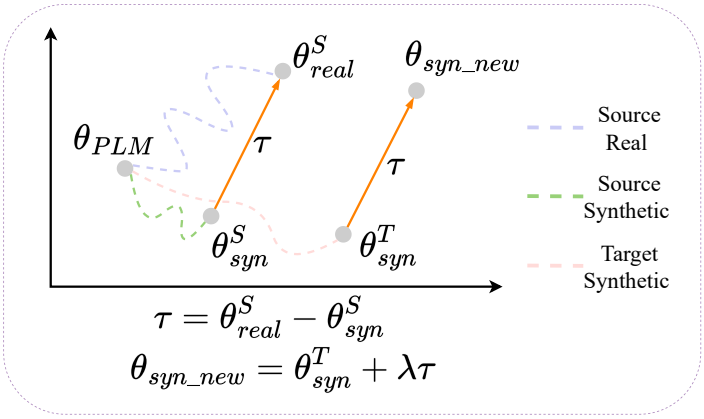


Figure 4.15: The framework illustrates the process of creating the SYN2REAL task vector by subtracting the parameter differences between a model fine-tuned on synthetic speech (Source Synthetic) and a model fine-tuned on real speech (Source Real) from pretrained ASR (PASR). This task vector is then applied to the target synthetic domain (Target Synthetic) to improve ASR performance by bridging the gap between synthetic and real speech data.

precision using QAT on the same *donor* dataset $\mathcal{D}$. The quantization shift vector is then computed by subtracting the parameters of the two models:

$$\tau_{\mathcal{D}} = \theta_{\mathcal{D},\text{QAT}} - \theta_{\mathcal{D}}$$

This shift vector $\tau_{\mathcal{D}}$ captures the transformation required to adapt a full-precision model to a low-precision format. By applying this shift vector to a new full-precision model on a *receiver* task, they can generate a new model that is expected to perform well in low precision without any direct training on it:

$$\theta_{\mathcal{R}\leftarrow\mathcal{D}} = \theta_{\mathcal{R}} + \lambda\tau_{\mathcal{D}}$$

where λ is a scaling factor that controls the magnitude of the shift. This approach has shown promising results in achieving zero-shot quantization, significantly reducing the performance gap between full-precision and quantized models without the need for additional training, making it a practical solution for deploying models on resource-constrained devices. The results are presented in Figure 4.16, where it is possible to observe also the importance of the scaling factor λ. Modulating the magnitude of the shift vector is crucial to avoid destructive interference and maximize the performance gains.

- **Zero-shot synthetic-to-real handwritten text recognition:** Handwritten Text Recognition (HTR) is a challenging task due to the high variability in handwriting styles, which often leads to a significant performance drop when models trained on synthetic data are evaluated on real-world handwritten text. Despite notable advances in recent years, HTR remains an open problem. Differences in writing styles, ligatures, slant and pressure cause models to generalize poorly from synthetic to real data. Overcoming this gap typically requires

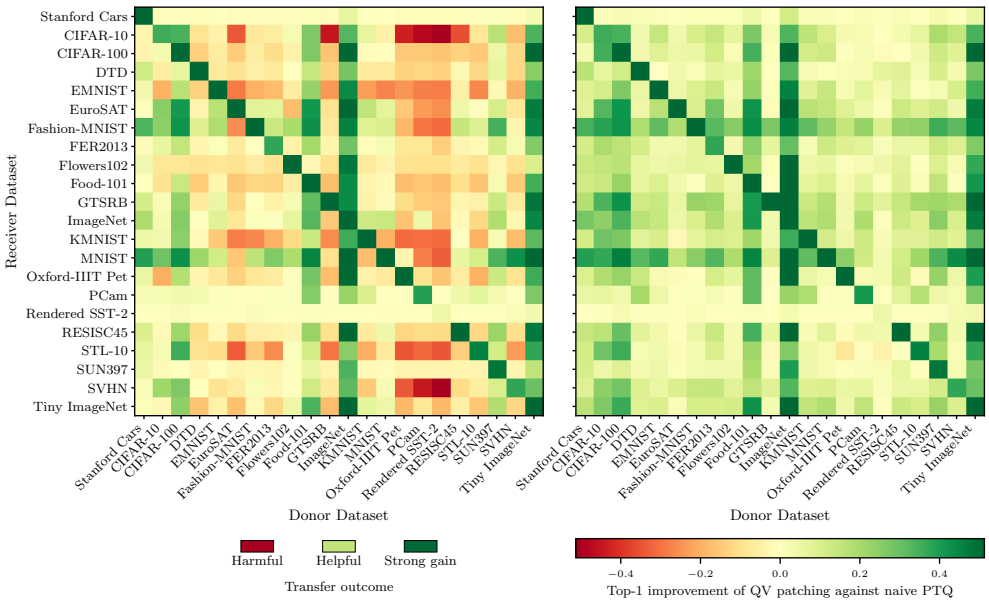


Figure 4.16: **Quantization vector traìnsferability for ViT/B-16**. Top-1 accuracy change (Δ) from patching receiver r with donor d quantization vector, relative to vanilla 3-bit PTQ. Left shows transfer with a constant scaling factor, while right demonstrates that modulating the magnitude λ eliminates destructive interference and maximizes gains.

collecting and annotating large amounts of real handwritten data, which is costly and time-consuming. To address this issue, Garrido-Munoz et al. [14] proposed to adopt task analogy to achieve zero-shot synthetic-to-real HTR. In particular they have advanced two variants of the task analogy framework: a *monolingual analogy*, which relies on a single source language domain and a *multilingual analogy*, which leverages multiple source language domains to learn a more robust and generalizable shift vector. Let θ_0 be the ancestor model, and θ^S_{syn} and θ^T_{syn} be the models obtained by fine-tuning the ancestor model θ_0 on synthetic data for the source and target languages, respectively. In this way we obtain the synthetic task vectors for the source and target languages:

$$\tau^S_{syn} = \theta^S_{syn} - \theta_0; \qquad \tau^S_{real} = \theta^S_{real} - \theta_0$$

Their difference $\Delta^S_{\text{s2r}} = \tau^S_{real} - \tau^S_{syn}$ represents the synthetic-to-real shift vector for the source language. Following the analogical reasoning $\tau^S_{syn} : \tau^S_{real} :: \tau^T_{syn} : \tau^T_{real}$, the synthetic-to-real shift vector for the target language can be approximated as:

$$\theta^T_{real} = \theta_0 + \tau^T_{syn} + \Delta^S_{\text{s2r}}$$

A graphical representation is presented in Figure 4.17

In the multilingual analogy variant, multiple source languages are used to compute multiple synthetic-to-real shift vectors, which are then averaged to obtain a more robust and

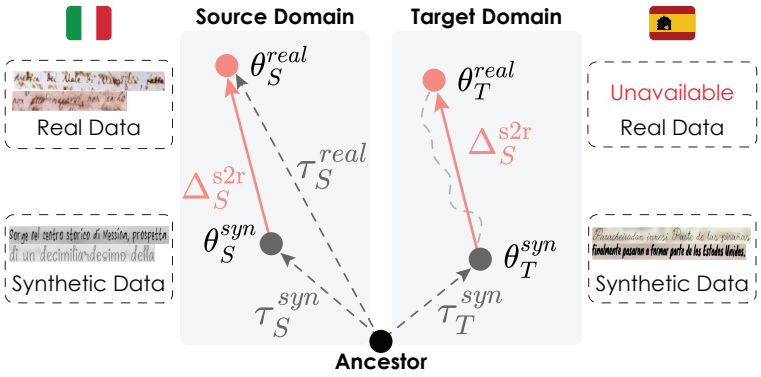


Figure 4.17: **Example of single analogy merging:** The Spanish real-domain model (θ^{es}_{real}) is estimated by applying the synthetic-to-real displacement vector (Δ^{it}_{s2r}) derived from the Italian domain.

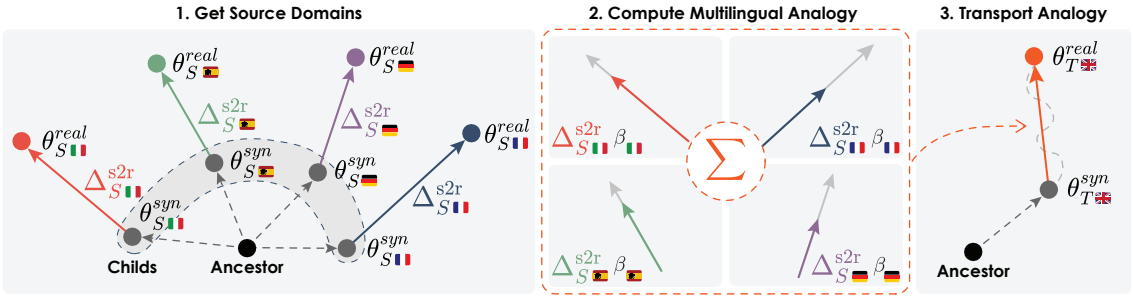


Figure 4.18: **Example of multilingual analogy merging:** The Spanish real-domain model (θ^{es}_{real}) is estimated by applying a weighted average of the synthetic-to-real displacement vectors derived from multiple source languages (e.g., Italian, French, and English), where the weights are determined by the similarity between each source language and the target language.

generalizable shift vector for the target language:

$$\theta^{T}_{real} = \theta^{T}_{syn} + \alpha \sum_{s \in \mathcal{S}} \beta_{(\cdot)}(s, T) \Delta^{s}_{\text{s2r}}$$

where $\mathcal{S}$ is the set of all source languages, and $\beta_{(\cdot)}(s, T)$ is an importance coefficient for each source language s that quantifies its relevance to the target language T. β may come from any similarity measure between the source and target languages, such as KL divergence, Hellinger Distance and Jaccard Similarity. Intuitively languages that are more similar to the target language should have an higher weight. The multilingual framework is presented in Figure 4.18.

These diverse applications demonstrate that if a network is properly regularized and operates within a linear or well-disentangled regime, the parameter space inherently understands complex relational transformations. Building upon this premise, the remainder of this chapter will focus on the specific challenges of generating and filtering these analogical vectors...

5. The geometry of misalignment: why zero-shot analogy fails

While the algebraic formulation of the analogy equation $\tau_D \approx \tau_C + (\tau_B - \tau_A)$ is theoretically appealing, deploying this framework in practice reveals a fundamental limitation. The assumption that the difference vector $\Delta = \tau_B - \tau_A$ extracts a pure, strictly parallel directional shift is highly idealized.

In Chapter 4, we established that arithmetic properties hold reliably only under strict weight disentanglement. However, in standard non-linear fine-tuning, models do not achieve this state. When a pre-trained model is fine-tuned on a source task B, its parameters adjust to capture the defining characteristics of domain B, but they inevitably co-adapt to spurious correlations and idiosyncratic conditions unique to the training set $\mathcal{D}_B$.

Consequently, the calculated shift vector Δ is inherently noisy and can be conceptually decomposed into two distinct components:

$$\Delta = \Delta_{\text{ideal}} + \varepsilon_{\text{interference}} \tag{5.1}$$

where Δ_{ideal} represents the true, generalizable semantic transformation, and $\varepsilon_{\text{interference}}$ encapsulates the entangled, task-specific artifacts causing the directional misalignment.

5.1 Beyond statistical noise: geometric and semantic limits

The limitations of pure zero-shot analogy cannot be attributed solely to the statistical noise of $\varepsilon_{\text{interference}}$. The observed misalignment is also rooted in the non-linear geometry of the parameter space and the semantic asymmetry of the domains themselves.

First, the translation of the shift vector from the source domain to the target anchor implicitly assumes a globally flat parameter space. In reality, the directional shift $\tau_B - \tau_A$ is optimized for the local tangent space around the source task. Due to the curvature of the loss landscape in deep non-linear networks, transporting this vector to a different region (around τ_C) inevitably induces geometric distortion.

Second, assuming a universal Δ_{ideal} implies that a semantic shift is independent of the underlying object. However, the transformation required to stylize or adapt one class of objects is often fundamentally different from the operations required for another class. Thus, the semantic shift is conditionally dependent on the anchor domain.

Because of these compounding factors, when the raw shift vector is added to the anchor task, the resulting model lands in a region of the parameter space that diverges from any valid target minimum. Due to the highly non-convex nature of neural network loss landscapes, there is no single true target vector, but rather a manifold of valid local minima (basins of attraction) that successfully solve the target task.

This structural misalignment induces a quantifiable **closure error**, mathematically measurable as the Euclidean distance between the algebraically predicted task vector $\tau_C + (\tau_B - \tau_A)$ and an empirical target vector τ_D obtained via direct fine-tuning. The magnitude of this error indicates that the naive algebraic projection fails to land inside the valid target basin, reducing the model's core competence and leading to sub-optimal zero-shot performance.

5.2 Calibrating the shift: the dual scaling baseline

To mitigate the interference caused by entangled noise and reduce the geometric closure error, a standard practice is to modulate the magnitude of the vectors before addition. Instead of a direct summation, the target model parameters are computed using a weighted combination, introducing scaling coefficients to balance the contribution of the anchor and the shift vector.

We refer to this approach as the **Dual Scaling Baseline**. By decoupling the scaling factors, the target model θ_D is formulated as:

$$\theta_D = \theta_0 + \alpha_1 \tau_C + \alpha_2(\tau_B - \tau_A) = \theta_0 + \alpha_1 \tau_C + \alpha_2 \Delta \tag{5.2}$$

where $\alpha_1, \alpha_2 \in [0, 1]$ are hyperparameters typically found via grid search on a validation set.

The primary advantage of dual scaling is its ability to protect the anchor task. By setting $\alpha_2 < \alpha_1$, we can suppress the overall magnitude of the shift vector Δ, reducing the impact of the interference term $\varepsilon_{\text{interference}}$ while allowing the anchor task τ_C to dominate the network's behavior.

However, from a geometric perspective, dual scaling acts as a coarse global adjustment. While adjusting the scalar α_2 modifies the length of the vector, it cannot fundamentally correct its direction. If the raw shift vector is inherently misaligned, scaling it down merely places the model in a sub-optimal region between the anchor and the target basin. Because α_2 is applied uniformly across the entire parameter space, it equally penalizes both the noise $\varepsilon_{\text{interference}}$ and the desired semantic shift Δ_{ideal}. The network is forced into a trade-off: either accept interference to gain the full domain shift, or suppress the interference at the cost of attenuating the analogical transformation.

This structural limitation highlights a critical need in zero-shot transfer methodologies. To close the geometric gap and navigate the non-convex parameter space, global scalar tuning is mathematically insufficient. A parameter-wise filtering mechanism is required to realign the shift vector and steer the optimization trajectory into the correct basin of attraction before the algebraic addition takes place.

5.3 Gradient-guided task vector filtering (GradFix)

Addressing the geometric misalignment of the directional shift Δ requires a parameter-wise filtering mechanism. To understand why this filtering must be guided by the model's loss gradients, we refer to the theoretical foundations of task arithmetic.

As established in Section 4.1.2, the task vector τ, generated via standard gradient descent fine-tuning, is mathematically equivalent to the accumulated negative gradient of the loss, diverging

from a joint multi-task optimization trajectory only by a second-order error term $\mathcal{O}(\eta^2)$. Because task vectors are essentially surrogate gradients, computing the analogical shift $\Delta = \tau_B - \tau_A$ is equivalent to manipulating optimization trajectories. Consequently, any geometric correction applied to Δ must respect the actual gradient descent directions of the target loss landscape to ensure effective convergence.

To achieve this gradient-aware filtering, we draw inspiration from the work of Rinaldi et al. [34], who introduced **GradFix**. Originally designed to solve a *Model Rebasin* problem—transporting a task vector across models with distinct pre-training initializations—GradFix uses the gradient of the target model as a proxy for its local loss geometry. By computing the gradient on a few samples and masking the source task vector, they aligned the transported vector with the descent directions of the new pre-trained backbone.

In this work, we adapt the core mechanism of GradFix to solve **analogical alignment**. Instead of bridging the gap between different pre-trained initializations, we operate within the tangent space of a single foundation model θ_0. We leverage the gradient-sign masking mechanism to filter geometric noise out of the analogical shift vector $\Delta = \tau_B - \tau_A$ before applying it to the anchor task τ_C.

5.3.1 Few-shot oracle and sign consensus

To construct the filter, we assume access to a *few-shot oracle* consisting of a minimal subset of labeled data from the target domain, denoted as $\mathcal{D}_{few} \subset \mathcal{D}_D$ (e.g., K samples per class).

Instead of computing a standard average gradient over a batch, which is sensitive to outlier magnitudes, we evaluate the gradient of the loss for each individual sample n:

$$g^{(n)} = \nabla_\theta \ell(f(\mathbf{x}_n; \theta_C), y_n)$$

To extract the structural direction of descent, we isolate the sign of each gradient. Following a robust estimation strategy, we aggregate these signs using a majority vote across all samples in $\mathcal{D}_{few}$. For each parameter coordinate i, the anti-gradient sign estimator $\hat{s}_i$ is defined as:

$$\hat{s}_i = \text{sign}\left(- \sum_{n=1}^{|\mathcal{D}_{few}|} \text{sign}(g_i^{(n)}) \right)$$

The vector $\hat{s} \in \{-1, 0, 1\}^M$ represents the consensus descent direction. If $\hat{s}_i$ is positive, the majority of the target samples require that parameter to increase to minimize the loss. If $\hat{s}_i = 0$, there is a perfect tie, indicating high uncertainty.

5.3.2 Directional masking for targeted negation

Once the consensus direction is established, we use it to construct a structural filter. Rather than indiscriminately masking the entire analogical shift $\Delta = \tau_B - \tau_A$, empirical analysis reveals that interference is primarily driven by the subtractive term, τ_A. Unconstrained subtraction inadvertently erodes foundational, task-agnostic features.

To mitigate this, we construct a binary mask $m \in \{0,1\}^M$ designed to evaluate sign consistency specifically against the source domain shift. For each parameter i, the mask is strictly positive only if the proposed negation agrees with the empirical consensus descent direction:

$$m_i = \mathbb{I}\big(\text{sign}(-\tau_{A,i}) == \hat{s}_i \wedge \hat{s}_i \neq 0\big)$$

This rigorous masking zeroes out the subtraction of any parameter where removing the source task conflicts with the local geometry of the target basin, as well as parameters where the few-shot oracle is undecided ($\hat{s}_i = 0$).

Finally, we inject this filtered, locally aligned negation into our algebraic framework. The target model θ_D is formulated as:

$$\theta_D = \theta_0 + \alpha_1(\tau_C + \tau_B) - \alpha_2(\tau_A \odot m)$$

where $\odot$ denotes the element-wise Hadamard product, and (α_1, α_2) decouple the scaling of the additive and subtractive components.

By applying this gradient-guided mask exclusively to the source vector, we perform a selective negation. This reduces the geometric closure error caused by subtractive interference. The resulting target model absorbs the constructive semantic transformations (τ_C and τ_B) while safely rejecting harmful domain-specific artifacts, ensuring that the anchor's core competences are preserved.

Having established this theoretical and algorithmic framework, the next step is empirical validation. In the following chapter, we will rigorously test this masking mechanism on complex visual domains, demonstrating how gradient-guided targeted negation mitigates interference and improves the reliability of zero-shot task analogies.

6. Experimental evaluation

6.1 Experimental setup

In this chapter, we evaluate the effectiveness of *GradFix* in the context of zero-shot task analogies. Our objective is to demonstrate that gradient-sign masking can successfully align analogical shift vectors even in the presence of significant domain gaps and non-linear distortions. This section details the datasets, the architectural configuration, the baselines, and the implementation framework used for all experiments.

6.1.1 Datasets and tasks

To test the robustness of our method across diverse visual distributions, we utilize **DomainNet** [31], currently one of the largest and most challenging benchmarks for multi-source domain adaptation. DomainNet consists of approximately 0.6 million images distributed across 345 categories and 6 distinct domains: *Real, Sketch, Quickdraw, Clip Art, Infograph*, and *Painting*. In Figure 6.1 we provide some images from all the domains to illustrate the significant visual and structural differences that characterize this dataset.

To systematically evaluate task analogies across conceptually distinct datasets, we aggregated the original 345 fine-grained categories into 9 broader, semantically cohesive super-classes, expanding upon the semantic taxonomy provided in the original paper's appendix [31]. These custom super-classes are: *Living Beings, Nature & Environment, Food & Agriculture, Transportation, Man-made Objects / Constructions, Tools & Devices, Human-related, Arts & Leisure*, and *Abstract / Others*.

To rigorously evaluate our analogical framework, we isolate two highly controlled transfer scenarios. We specifically target severe structural and morphological domain shifts—such as transitioning from the **Sketch** or **Painting** domains to the highly abstracted **Quickdraw** domain. These transitions provide a compelling testbed: while paintings and sketches contain rich textures or detailed, human-drawn illustrations, quickdraws are reduced to minimalist, conceptual strokes.

Within this context, we define two primary task analogies, structured as $A : B \approx C : D$ (i.e., "A is to B as C is to D"):

- **Sketch $\rightarrow$ Quickdraw transfer:** we extract the structural shift from the *Man-made Objects* (MMO) domain and apply it to the *Food & Agriculture* (FA) domain. This yields the following relation:

$$\texttt{FAS} : \texttt{MMOS} \approx \texttt{FAQ} : \texttt{MMOQ}$$

- **Painting $\rightarrow$ Quickdraw transfer:** similarly, we transfer the shift extracted from *Man-made Objects* to the *Tools & Devices* (TD) domain, formulated as:

$$\texttt{TDP} : \texttt{MMOP} \approx \texttt{TDQ} : \texttt{MMOQ}$$

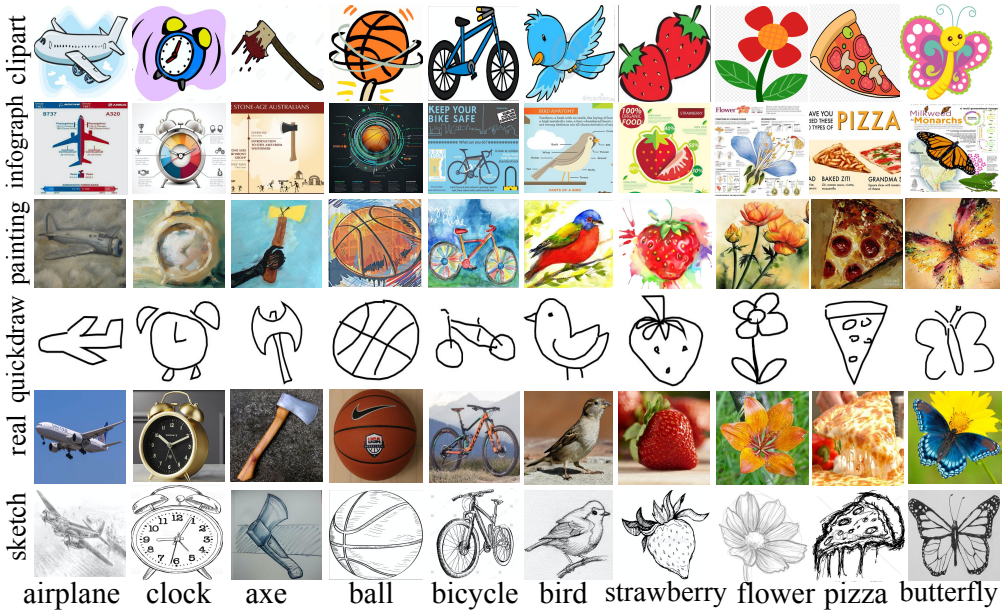


Figure 6.1: Sample images from the six domains of the DomainNet dataset. The significant visual and structural differences between domains (e.g., Real vs. Sketch) present a challenging testbed for zero-shot task analogies.

This specific configuration allows us to mathematically isolate the components of the analogical transfer. In a naive task arithmetic framework, synthesizing a *Quickdraw* food classifier zero-shot would involve extracting the "sketch-to-quickdraw" transformation strictly from the *Man-made Objects* macro-task ($\Delta = \tau_{\text{MMOQ}} - \tau_{\text{MMOS}}$) and directly adding it to a *Food & Agriculture* anchor model (τ_{FAS}).

However, to actively mitigate the destructive negation caused by the unconstrained subtraction of the source domain, we redefine this algebraic projection. We apply the gradient-guided mask, M, exclusively to the subtractive term:

$$\tau_{\text{FAQ}} \approx \tau_{\text{FAS}} + \tau_{\text{MMOQ}} - (M \odot \tau_{\text{MMOS}})$$

Through this *masked negation*, our objective is to successfully synthesize the target classifier zero-shot. This surgical formulation allows us to filter out source-specific interference while preserving the foundational weights, achieving the domain shift without ever explicitly training the anchor model on the target distribution.

6.1.2 Architecture and optimization regimes

Our framework is built upon the **CLIP ViT-B/32** architecture [33], utilizing the pre-trained weights provided by OpenAI. Contrastive Language-Image Pre-training (CLIP) is uniquely suited for this study: its robust, multimodal latent space encodes the highly generalizable foundational features that our surgical negation mechanism explicitly aims to protect during analogical transfer.

A central debate in contemporary task arithmetic is whether weight-space algebraic operations should be executed in the standard non-linear parameter space or within a linearized regime. To systematically evaluate the trade-off between representational capacity and arithmetic interference, we derive our base task vectors (τ_A, τ_B, and τ_C) under two strictly distinct optimization paradigms:

- Standard non-linear fine-tuning: the model is optimized using standard cross-entropy. This regime preserves the full expressivity of the deep network, which is strictly necessary to learn profound morphological shifts (e.g., abstracting detailed sketches into minimalist quickdraws). However, as our analysis reveals, the resulting vectors are highly susceptible to destructive negation and geometric misalignment during algebraic composition.
- Tangent space linearization: following the formalization by Ortiz-Jimenez et al. [28], the network is approximated via a first-order Taylor expansion, operating in the Neural Tangent Kernel (NTK) regime. This transforms the optimization into a strictly linear problem, theoretically guaranteeing perfect vector arithmetic. Yet, this mathematical protection artificially induces a severe expressivity bottleneck, constraining the model's ability to map complex structural domain gaps.

For rigorous comparison and reproducibility, all experiments involving the linearized regime were conducted integrating the official *Task Arithmetic in the Tangent Space* framework[1].

6.1.3 Evaluation Baselines and Benchmarks

To rigorously quantify the efficacy of our surgical negation and to assess the fundamental limitations of different parameter spaces, we evaluate our approach against a comprehensive set of baselines. For the analogical synthesis ($\theta_D \approx \theta_0 + \tau_C + \tau_B - \tau_A$), we compare the following configurations:

- **Zero-shot anchor (θ_0):** the performance of the foundational pre-trained model evaluated directly on the target domain D, establishing the base transferability prior to any analogical modification.
- **Standard non-linear analogy:** the naive algebraic projection $\theta_0 + \tau_C + \tau_B - \tau_A$ computed using vectors derived from standard fine-tuning. This baseline exposes the raw catastrophic interference and destructive negation inherent in unconstrained non-linear arithmetic.
- **Tangent space linearization:** the analogical projection executed entirely within the linearized (NTK) regime. While mathematically designed to mitigate interference, this baseline tests the severity of the expressivity bottleneck when mapping severe structural shifts.
- **Regularized linear analogy (τ_{JP}):** a specialized linearized baseline explicitly regularized using the Jacobian penalty (τ_{JP}). By preserving vector orthogonality relative to the pre-training distribution (e.g., ImageNet), this serves as an empirically strengthened upper bound for linear task arithmetic.

[1] Source code available at: `https://github.com/gortizji/tangent_task_arithmetic`

- **GradFix:** the proposed gradient-guided masking approach, applied within the highly expressive non-linear space. By strictly filtering the subtractive term (τ_A), this configuration aims to preserve foundational features while safely absorbing constructive semantic shifts.

For each arithmetic configuration, we investigate two continuous modulation strategies: a *single-scaling* variant (applying a global scalar α to the composite analogical shift) and a *dual-scaling* variant (applying decoupled scalars, α_1 to the additive components and α_2 to the subtractive component). This comparison is critical to empirically validate whether interference is asymmetric, and whether simple scalar decoupling can sufficiently mitigate destructive negation without requiring discrete masking.

Specifically for the GradFix framework, the reference point for computing the gradient-sign consensus mask adapts to the structural composition. While a single-scaling baseline evaluates the local geometry at the foundational model (θ_0), our optimal dual-scaling approach computes the gradient mask by evaluating the loss directly at the intermediate anchor model ($\theta_0 + \tau_C + \tau_B$). Because the anchor already occupies a shifted region of the loss landscape, deriving the gradients at this specific initialization point ensures that the subsequent masked negation accurately reflects the local geometric requirements of the target basin.

6.1.4 GradFix configuration

For the implementation of our proposed GradFix filtering, we assume a *few-shot* scenario where a minimal subset of K samples per class (we evaluate $K = 5$ and $K = 50$) is available from the target domain D to compute the sign-consensus vector $\hat{s}$.

Crucially, these samples are used exclusively to compute the directional gradients for the majority vote and are never used to update the model parameters via backpropagation. By applying the resulting binary mask exclusively to the subtractive source vector τ_A, GradFix effectively uncouples destructive domain interference from constructive semantic knowledge. This rigorous separation ensures that any performance gain is strictly derived from geometric realignment and targeted amnesia prevention, allowing the zero-shot analogy to safely leverage the full capacity of the non-linear foundation model.

6.1.5 Validation of simple task addition

Before evaluating the highly complex analogical transfers, it is critical to rigorously validate the structural integrity of our generated checkpoints. To empirically verify that our task vectors are correctly optimized and capable of algebraic composition, we conducted a preliminary evaluation focused exclusively on standard multi-task addition. Specifically, we evaluated the direct composition of all the distinct task vectors of the analogy setting ($\theta_{multi} = \theta_0 + \tau_1 + \tau_2 + \tau_3 + \tau_4$) across all three optimization regimes: standard non-linear, linearized, and regularized linear (τ_{JP}).

The empirical results reported in Table 6.1 and Table 6.2 confirm that all checkpoints behave as expected within the context of simple addition. The standard non-linear vectors successfully compose to perform all the four tasks, despite the natural presence of minor non-linear interference.

Table 6.1: Validation of simple multi-task addition (Top-1 Accuracy %) for the *Sketch-Quickdraw* analogy. The table reports the performance of the composed model ($\theta_{multi} = \theta_0 + \tau_1 + \tau_2 + \tau_3 + \tau_4$) evaluated on the individual target domains. Results are compared across the standard non-linear, linearized, and regularized linear (τ_{JP}) optimization regimes. This baseline experiment verifies the structural integrity and composability of the task vectors prior to their use in complex analogical transfers.

Method	Metric / Variant	FoodAgricultureSketch	ManMadeObjectsSketch	FoodAgricultureQuickdraw	ManMadeObjectsQuickdraw	Mean
Zero-shot baseline		70.32	52.29	24.97	17.52	–
Nonlinear	FT accuracy	76.78	84.07	78.67	78.57	–
	Addition	85.05	61.52	66.00	51.18	65.94
Linear	FT accuracy	83.37	73.88	77.87	74.34	–
	Addition (fixed)	80.42	63.33	63.23	56.19	65.79
Linear Reg. orth ImageNet	FT accuracy	87.58	74.97	78.00	73.71	–
	Addition (fixed)	87.16	67.37	63.23	54.87	68.16

Table 6.2: Validation of simple multi-task addition (Top-1 Accuracy %) for the *Painting-Quickdraw* analogy. The table reports the performance of the composed model ($\theta_{multi} = \theta_0 + \tau_1 + \tau_2 + \tau_3 + \tau_4$) evaluated on the individual target domains. Results are compared across the standard non-linear, linearized, and regularized linear (τ_{JP}) optimization regimes. This baseline experiment verifies the structural integrity and composability of the task vectors prior to their use in complex analogical transfers.

Method	Metric / Variant	ManMadeObjectsPainting	ToolsDevicesPainting	ManMadeObjectsQuickdraw	ToolsDevicesQuickdraw	Mean
Zero-shot baseline		65.06	63.70	17.52	20.79	–
Nonlinear	FT accuracy	88.00	73.58	77.03	76.77	–
	Addition	71.27	76.05	55.95	68.34	67.90
Linear	FT accuracy	80.54	82.96	71.37	74.57	–
	Addition (fixed)	75.68	80.00	37.78	53.92	61.85
Linear Reg. orth ImageNet	FT accuracy	80.22	84.44	73.83	77.85	–
	Addition (fixed)	77.30	83.70	53.69	62.68	69.34

Furthermore, both the standard and regularized linearized checkpoints exhibit excellent multi-task composability, effectively minimizing geometric interference as mathematically predicted by existing task arithmetic literature.

This validation serves a crucial role in our broader analysis. By demonstrating that our checkpoints flawlessly execute simple additive composition across all regimes, we systematically isolate the root causes of performance degradation in the analogical setting. It proves that the subsequent failures in the "Sketch $\rightarrow$ Quickdraw" transfer are not artifacts of poor fine-tuning.

6.2 Main quantitative results

To comprehensively evaluate the zero-shot analogical transfer, Table 6.3 reports the performance across all four quadrants of the domain shift. It is crucial to note how these results are structured: each column represents a distinct target dataset. The accuracy reported in a specific cell is obtained by formulating the analogical shift to explicitly target that column's dataset, optimizing the scaling

coefficients for that specific transfer, and evaluating the resulting synthesized model on that domain.

The empirical data reveals two fundamental insights regarding both the optimization regime and the underlying cause of catastrophic interference:

The expressivity vs. linearity trade-off. Contrary to the theoretical idealization of task arithmetic, our results demonstrate that strict linearization in the tangent space fundamentally bottlenecks performance. As shown in Table 6.3 and Table 6.4, the linearized analogy (dual scaling) achieves only 44.03% accuracy on average and 44.95% respectively, underperforming the standard non-linear analogy. Even when the linear regime is explicitly regularized (τ_{JP}) towards a common multi-task optimum, it reaches only 46.56% and 46.12%, struggling to recover the capacity of standard fine-tuning. This provides strong empirical evidence that severe morphological shifts (such as abstracting detailed sketches into minimalist quickdraws) require the full non-linear representational capacity of the network. While the tangent space mathematically preserves algebraic addition, it lacks the expressivity required to solve complex domain adaptations.

GradFix: targeted mitigation of subtractive interference. Having established that the standard non-linear space is necessary for task performance, we address the catastrophic interference that plagues naive arithmetic. Our analysis identifies unconstrained subtraction (τ_A) as the primary driver of this interference, causing the destructive negation of foundational features.

When the GradFix mask ($K = 50$) is applied to the non-linear source vector τ_A, target accuracy on FAQ drastically improves from 47.03% (naive dual-scaling baseline) to 55.23% (+8.20%). Even in the extreme $K = 5$ few-shot scenario, GradFix achieves 54.00% (+6.97%), and on average it improves the baseline by $\approx +4\%$. By utilizing the few-shot oracle to filter out geometrically misaligned parameters exclusively during subtraction, GradFix successfully suppresses non-linear interference without crippling the network's capacity. It effectively bridges the gap between the naive projection and the joint-training upper bound, proving that gradient-guided discrete filtering of the source domain is significantly more effective than simple uniform scalar modulation.

6.2.1 Few-shot adaptability and feature quality

While zero-shot accuracy provides a static measure of analogical transfer, it does not fully capture the topological quality and plasticity of the resulting parameter space. To rigorously evaluate whether our targeted masking places the model in a genuinely advantageous region of the loss landscape, we conduct two complementary few-shot probing experiments ($K = 5$ and $K = 50$ target samples per class): *latent space linear separability* and *basin proximity*.

Latent space linear separability (Linear Probing). First, we evaluate the representational quality of the extracted visual features. In this setup, we completely freeze the synthesized image encoder (θ_D) and train only a new linear classification head using the few-shot target support set.

If the analogical shift successfully avoids destructive negation, the extracted features should be highly discriminative and linearly separable for the target classes. As detailed in Table 6.5 and Table 6.7, the naive non-linear analogy yields highly entangled representations, achieving only 62.43% and 62.41% linear probing accuracy under the 50-shot regime on average respectively. This indicates that the unconstrained subtraction of the source vector severely degrades the foundational

Table 6.3: Zero-shot top-1 accuracy (%) for the *Sketch-Quickdraw* analogical transfer. Note: each column represents a distinct target dataset. The reported accuracy corresponds to a model specifically formulated and optimized (via coefficients and masks) to target that column's domain. For the dual scaling variants, each cell reports the test accuracy followed by the target-specific optimized coefficients (α_1, α_2).

Method	Metric / Variant	FoodAgricultureSketch	ManMadeObjectsSketch	FoodAgricultureQuickdraw	ManMadeObjectsQuickdraw	Mean
Zero-shot baseline		70.32	52.29	24.97	17.52	–
Nonlinear	FT accuracy	88.00	73.58	77.03	76.77	–
	Analogy (fixed)	74.74 (0.35)	53.86 (0.15)	40.58 (0.65)	28.36 (0.55)	49.39
	Analogy (dual)	75.58 $(0.15, 0.05)$	55.19 $(0.15, 0.00)$	47.03 $(0.55, 0.10)$	29.59 $(0.60, 0.00)$	51.85
Linear	FT accuracy	83.37	73.88	77.87	74.34	–
	Analogy (fixed)	73.47 (0.15)	53.56 (0.20)	29.61 (0.70)	21.01 (0.40)	44.42
	Analogy (dual)	70.53 $(0.05, 0.05)$	52.29 $(0.00, 0.00)$	30.77 $(0.65, 0.00)$	22.55 $(0.10, 0.00)$	44.03
Linear Reg. orth ImageNet	FT accuracy	87.58	74.97	78.00	73.71	–
	Analogy (fixed)	72.21 (0.25)	53.26 (0.15)	33.10 (0.75)	19.83 (0.25)	43.75
	Analogy (dual)	73.26 $(0.30, 0.25)$	53.62 $(0.30, 0.15)$	36.71 $(0.45, 0.05)$	22.63 $(0.50, 0.00)$	46.56
GradFix $K = 50$	Analogy (fixed)	**78.74** (0.05)	**57.12** (0.05)	**35.10** (0.05)	**23.06** (0.05)	48.51
	Analogy (dual)	**76.84** $(0.30, 0.05)$	**55.61** $(0.25, 0.05)$	**55.23** $(0.45, 0.10)$	**35.64** $(0.60, 0.10)$	55.83
GradFix $K = 5$	Analogy (fixed)	**76.21** (0.05)	**57.12** (0.05)	**30.71** (0.05)	**21.23** (0.05)	46.14
	Analogy (dual)	**74.74** $(0.40, 0.10)$	**55.07** $(0.20, 0.05)$	**54.00** $(0.55, 0.15)$	**33.37** $(0.50, 0.15)$	54.30

feature space, effectively causing amnesia of the low-level features required to distinguish classes.

Conversely, applying GradFix exclusively to filter the subtractive interference improves linear probing accuracy to 64.80% and 65.43 (+2.37% and +2.92%) respectively on average. This provides strong empirical evidence that our asymmetric masking strategy does not merely calibrate output logits, but fundamentally preserves the integrity of the latent manifold. By restricting the surgical intervention to the negative term (τ_A), GradFix safely removes domain-specific artifacts while actively shielding the foundation model's core feature extractors from destruction, yielding a highly separable feature space perfectly primed for the target task.

Table 6.4: Zero-shot Top-1 Accuracy (%) for the *Painting-Quickdraw* analogical transfer. Note: each column represents a distinct target dataset. The reported accuracy corresponds to a model specifically formulated and optimized (via coefficients and masks) to target that column's domain. For the dual scaling variants, each cell reports the test accuracy followed by the target-specific optimized coefficients (α_1, α_2).

Method	Metric / Variant	ManMadeObjectsPainting	ToolsDevicesPainting	ManMadeObjectsQuickdraw	ToolsDevicesQuickdraw	Mean
Zero-shot baseline		65.06	63.70	17.52	20.79	–
Nonlinear	FT accuracy	76.78	84.07	78.67	78.57	–
	Analogy (fixed)	66.34 (0.10)	68.02 (0.25)	29.57 (0.45)	37.13 (0.55)	50.27
	Analogy (dual)	66.47 $(0.15, 0.00)$	68.64 $(0.25, 0.10)$	30.48 $(0.50, 0.30)$	39.55 $(0.65, 0.20)$	51.29
Linear	FT accuracy	80.54	82.96	71.37	74.57	–
	Analogy (fixed)	65.06 (0.00)	66.91 (0.05)	17.81 (0.05)	25.21 (0.05)	43.77
	Analogy (dual)	64.46 $(0.05, 0.00)$	67.90 $(0.05, 0.00)$	20.07 $(0.05, 0.00)$	27.36 $(0.10, 0.00)$	44.95
Linear Reg. orth ImageNet	FT accuracy	80.22	84.44	73.83	77.85	–
	Analogy (fixed)	66.47 (0.30)	67.65 (0.40)	21.16 (0.30)	28.26 (0.45)	45.89
	Analogy (dual)	65.82 $(0.45, 0.20)$	67.28 $(0.45, 0.20)$	21.61 $(0.30, 0.05)$	29.77 $(0.40, 0.15)$	46.12
GradFix $K = 50$	Analogy (fixed)	**69.13** (0.05)	**69.38** (0.05)	**23.61** (0.05)	**20.79** (0.00)	45.73
	Analogy (dual)	**67.06** $(0.30, 0.10)$	**69.51** $(0.30, 0.05)$	**35.59** $(0.55, 0.10)$	**47.42** $(0.75, 0.10)$	54.90
GradFix $K = 5$	Analogy (fixed)	**67.64** (0.10)	**68.27** (0.05)	**22.29** (0.05)	**23.32** (0.05)	45.38
	Analogy (dual)	**65.89** $(0.30, 0.15)$	**68.27** $(0.35, 0.10)$	**35.83** $(0.60, 0.20)$	**44.91** $(0.65, 0.10)$	53.73

Table 6.5: Latent space linear separability (Top-1 Accuracy %) evaluated via 50-shot head refinement (+ FS) on the *Sketch-Quickdraw* analogy. **Note:** Each column represents a distinct target dataset. The reported accuracy corresponds to a feature space generated by an analogical shift specifically formulated for that column's domain. For the dual scaling variants, the target-specific optimized coefficients (α_1, α_2) are reported.

Method	Metric / Variant	FoodAgricultureSketch	ManMadeObjectsSketch	FoodAgricultureQuickdraw	ManMadeObjectsQuickdraw	Mean
Zero-shot baseline	Zero-shot accuracy	70.32	52.29	24.97	17.52	–
	Zero-shot + FS	77.26	63,63	46,19	43,23	57.58
Nonlinear	FT accuracy	88.00	73.58	77.03	76.77	–
	Analogy (fixed) + FS	80.00	64.72	58.77	45.52	62.25
	Analogy (dual) + FS	79.37	64.17	58.97	47.20	62.43
GradFix $K = 50$	Analogy (fixed) + FS	**82,53**	**64.66**	**53,74**	**43.52**	61.11
	Analogy (dual) + FS	**81.47**	**63.99**	**64.13**	**49.61**	64.80
GradFix $K = 5$	Analogy (fixed) + FS	**82.11**	**64.54**	**52.00**	**43.40**	60.51
	Analogy (dual) + FS	**80.21**	**64.23**	**62.71**	**49.45**	64.15

Table 6.6: Latent space linear separability (Top-1 Accuracy %) evaluated via 5-shot head refinement (+ FS) on the *Sketch-Quickdraw* analogy. **Note:** Each column represents a distinct target dataset. The reported accuracy corresponds to a feature space generated by an analogical shift specifically formulated for that column's domain. For the dual scaling variants, the target-specific optimized coefficients (α_1, α_2) are reported.

Method	Metric / Variant	FoodAgricultureSketch	ManMadeObjectsSketch	FoodAgricultureQuickdraw	ManMadeObjectsQuickdraw	Mean
Zero-shot baseline	Zero-shot accuracy	70.32	52.29	24.97	17.52	–
	Zero-shot + FS	72.21	55.85	32.26	24.27	46.15
Nonlinear	FT accuracy	88.00	73.58	77.03	76.77	–
	Analogy (fixed) + FS	74.95	56.51	47.10	29.78	52.09
	Analogy (dual) + FS	76.00	57.24	49.35	32.48	53.77
GradFix $K = 50$	Analogy (fixed) + FS	**78.95**	**58.87**	**39.16**	**28.63**	51.40
	Analogy (dual) + FS	**77.89**	**57.66**	**56.13**	**38.22**	57.48
GradFix $K = 5$	Analogy (fixed) + FS	**78.11**	**58.62**	**36.71**	**27.08**	50.13
	Analogy (dual) + FS	**76.00**	**57.24**	**55.23**	**36.24**	56.18

Table 6.7: Latent space linear separability (Top-1 Accuracy %) evaluated via 50-shot head refinement (+ FS) on the *Painting-Quickdraw* analogy. **Note:** Each column represents a distinct target dataset. The reported accuracy corresponds to a feature space generated by an analogical shift specifically formulated for that column's domain. For the dual scaling variants, the target-specific optimized coefficients (α_1, α_2) are reported.

Method	Metric / Variant	ToolsDevicesPainting	ManMadeObjectsPainting	ToolsDevicesQuickdraw	ToolsDevicesQuickdraw	Mean
Zero-shot baseline	Zero-shot accuracy	65.06	63.70	17.52	20.79	–
	Zero-shot + FS	74.45	76.17	43.28	42.26	59.04
Nonlinear	FT accuracy	76.78	84.07	78.67	78.57	–
	Analogy (fixed) + FS	74.71	76.17	49.33	52.34	63.14
	Analogy (dual) + FS	74.06	76.17	48.99	50.42	62.41
GradFix $K = 50$	Analogy (fixed) + FS	**75.75**	**77.53**	**43.88**	**42.23**	59.85
	Analogy (dual) + FS	**73.61**	**76.05**	**52.89**	**59.17**	65.43
GradFix $K = 5$	Analogy (fixed) + FS	**75.75**	**76.05**	**43.69**	**46.49**	60.50
	Analogy (dual) + FS	**73.61**	**75.06**	**53.13**	**58.38**	65.05

Table 6.8: Latent space linear separability (Top-1 Accuracy %) evaluated via 5-shot head refinement (+ FS) on the *Painting-Quickdraw* analogy. **Note:** Each column represents a distinct target dataset. The reported accuracy corresponds to a feature space generated by an analogical shift specifically formulated for that column's domain. For the dual scaling variants, the target-specific optimized coefficients (α_1, α_2) are reported.

Method	Metric / Variant	FoodAgricultureSketch	ManMadeObjectsSketch	FoodAgricultureQuickdraw	ManMadeObjectsQuickdraw	Mean
Zero-shot baseline	Zero-shot accuracy	65.06	63.70	17.52	20.79	–
	Zero-shot + FS	67.19	65.06	24.39	24.15	45.20
Nonlinear	FT accuracy	76.78	84.07	78.67	78.57	–
	Analogy (fixed) + FS	67.77	68.77	32.65	38.75	51.99
	Analogy (dual) + FS	67.64	69.01	32.41	35.02	51.02
GradFix $K = 50$	Analogy (fixed) + FS	**69.71**	**71.23**	**28.80**	**24.34**	48.52
	Analogy (dual) + FS	**68.61**	**70.62**	**38.55**	**48.94**	56.68
GradFix $K = 5$	Analogy (fixed) + FS	**68.94**	**69.88**	**27.88**	**27.96**	48.67
	Analogy (dual) + FS	**67.44**	**69.88**	**38.82**	**46.53**	55.67

7. Conclusions and future work

7.1 Summary of contributions

In this thesis, we investigated the geometric and topological challenges of zero-shot task analogies within the weight space of foundation models. While task arithmetic provides an elegant, training-free mechanism to steer model behavior, our analysis demonstrated that naive algebraic composition suffers from significant geometric interference when subjected to severe structural domain shifts, such as the transition from detailed sketches to abstract quickdraws.

Through a rigorous empirical and topological evaluation, we formulated two fundamental contributions to the literature of model merging and transfer learning:

- **Identification of the expressivity bottleneck and interference negation:** we demonstrated that linearizing the optimization trajectory in the tangent space (NTK regime) induces a severe expressivity bottleneck, rendering the model less capable of mapping complex morphological shifts. Conversely, while the standard non-linear parameter space provides the necessary capacity, we proved that it suffers from asymmetric interference. Specifically, we identified the unconstrained subtraction of the source task (τ_A) as a primary driver of performance degradation. This operation causes the *destructive negation* of foundational, domain-agnostic feature extractors, structurally limiting the full potential of the analogical shift.

- **Surgical negation via GradFix:** to resolve this geometric misalignment, we have applied GradFix [34], a gradient-guided masking mechanism. By utilizing a few-shot oracle to compute a sign-consensus mask, we applied a discrete directional filter exclusively to the subtractive term of the analogy. This *surgical negation* effectively uncouples domain-specific artifacts from core semantic knowledge. Our extensive evaluation via linear probing and zero-shot testing proved that GradFix consistently improves the analogical baseline (yielding accuracy gains of up to +15% from the pretrained model). It does not merely calibrate output logits, but actively preserves the integrity and separability of the latent manifold, safely bridging the gap between naive analogical projection and the joint-training upper bound.

7.2 Limitations

Despite its empirical success and topological robustness, the current formulation of our framework exhibits certain limitations that warrant acknowledgment:

- **Reliance on a few-shot oracle:** while the analogical transfer itself is evaluated in a zero-shot manner on the target test set, the computation of the GradFix mask strictly requires a small support set ($K = 5$ or $K = 50$) from the target distribution. Consequently, the masking method is not a purely zero-data mathematical projection, but rather a few-shot geometrically guided intervention.

- **Rigidity of binary masking:** the GradFix filter operates as a hard boolean mask ($m \in \{0, 1\}$). While this heuristic is computationally efficient and effectively shields the network from destructive gradients, binary truncation is a naturally aggressive operation. It does not account for the magnitude of the interference, potentially zeroing out parameters that might have been useful if only marginally scaled down.
- **Architectural specificity:** our empirical evaluation was conducted utilizing the Vision Transformer architecture (CLIP ViT-B/32). While ViTs exhibit highly structured and predictable representational spaces, the exact dynamics of destructive negation might manifest differently in alternative architectures, such as deep Convolutional Neural Networks (CNNs) or dense mixture-of-experts models.

7.3 Future directions

The insights derived from the success of targeted subtractive mitigation open several promising avenues for future research in the domain of parameter-efficient model merging:

- **Continuous mask optimization:** transitioning from the discrete boolean logic of GradFix to a continuous optimization paradigm represents a natural evolution of this work. Future studies could explore learning continuous, layer-wise, or channel-wise scaling coefficients (conceptually similar to AdaMerging [44]) utilizing the few-shot oracle. This would allow for a softer, mathematically optimal surgical negation rather than a rigid cutoff.
- **Expansion to generative AI and LLMs:** the principles of surgical negation hold significant potential beyond classification tasks. As Large Language Models (LLMs) and diffusion models increasingly rely on weight merging for concept unlearning or multi-skill integration, the unconstrained subtraction of vectors (e.g., to remove toxicity or a specific visual style) often induces catastrophic degradation of linguistic or generative coherence. Applying our asymmetric masking framework to these massive generative architectures could provide a highly efficient mechanism for safe and targeted concept erasure.
- **Analogical synthesis as a warm-start initialization:** while this thesis primarily investigates the zero-shot capabilities of task analogies, the topologically favorable parameter space generated by GradFix presents a highly promising initialization point for further fine-tuning. Because GradFix preserves the integrity of the latent manifold and teleports the weights into a basin geometrically proximate to the target optimum, initializing a model with the GradFix synthesis, rather than the standard pre-trained foundation model, could significantly accelerate convergence. Future research should investigate this "warm-start" paradigm to evaluate whether it improves sample efficiency in strictly limited data regimes.

A. Mapping neural networks to spin glass models

In this section we will show how to derive the loss landscape of a deep neural network as a spin glass model. We will follow the derivation from Choromanska et al. [7].

We consider a binary classification problem and a simple MLP. Let:

- X be the random input vector of dimensionality d;
- $H-1$ be the number of hidden layers (we will refer to 0^{th} layer as the input layer and to H^{th} layer as the output layer);
- n_k be the number of neurons in the k^{th} layer ($n_0 = d$ and $n_H = 1$);
- W_i be the matrix of all the weights between $(i-1)^{\text{th}}$ layer and i^{th} layer;
- σ be the activation function ($\sigma(x) = \max(0, x)$).

Therefore we can express the output of the network as follows:

$$Y = q\sigma(W_H^\top \sigma(W_{H-1}^\top ... \sigma(W_1^\top X))) \tag{A.1}$$

where is a normalization factor that depends on the number of parameters and the depth of the network.

It is possible to re-express the output of the network also in the following way:

$$Y = q \sum_{i=1}^{n_0} \sum_{j=1}^{\gamma} X_{i,j} A_{i,j} \prod_{k=1}^{H} w_{i,j}^{(k)} \tag{A.2}$$

where the first sum is over the input neurons, the second is over all possible paths (each composed of exactly H weights) from a given network input to the output, γ is the total number of such paths (so $\gamma = n_1 n_2 ... n_H$). Note that for all $i = \{1, 2, ..., n_0\} : X_{i,1} = X_{i,2} = ... = X_{i,\gamma}$. Moreover, $w_{i,j}^{(k)}$ is the weight of the k^{th} segment of path indexed with (i, j) which connects the layer $k-1$ to the layer k. Finally $A_{i,j}$ is an indicator variable that denotes whether the path (i, j) is active ($A_{i,j} = 1$) or not ($A_{i,j} = 0$) depending on the value of the activation function.

If we consider the case in which every path is equally likely to be active, we can model $A_{i,j}$ as a Bernoulli random variable with parameter ρ, which represents the success probability. In this way we can reformulate the Equation (A.2) as follows:

$$\mathbb{E}_A[Y] = q \sum_{i=1}^{n_0} \sum_{j=1}^{\gamma} X_{i,j} \rho \prod_{k=1}^{H} w_{i,j}^{(k)} \tag{A.3}$$

Definition A.0.1 (Size of the network)**.** The size of the network is defined as $N = \sum_{k=1}^{H} n_k n_{k-1}$, which is the total number of parameters in the network.

Let:

- $\mathcal{W} = \{w_1, w_2, ..., w_N\}$ the set of all the weights in the network;
- $\mathcal{A}$ the set of all H-length configuration of the weights (so $|\mathcal{A}| = N^H$);
- $\mathcal{B} = \{(w_{1,1}^{(1)}, w_{1,1}^{(2)}, ..., w_{1,1}^{(H)}), (w_{1,2}^{(1)}, w_{1,2}^{(2)}, ..., w_{1,2}^{(H)}), ..., (w_{n_0,\gamma}^{(1)}, w_{n_0,\gamma}^{(2)}, ..., w_{n_0,\gamma}^{(H)})\}$ the set of all the paths in the network (note that each single weight comes from $\mathcal{W}$ and $\mathcal{B} \subset \mathcal{A}$).

In this way we can re-express the output of the network as follows:

$$Y_N \stackrel{\text{def}}{=} \mathbb{E}_A[Y] = q \sum_{i_1,i_2,...,i_H=1}^{N} \sum_{j=1}^{r_{i_1,i_2,...,i_H}} X_{i_1,i_2,...,i_H}^{(j)} \rho \prod_{k=1}^{H} w_{i_k} \tag{A.4}$$

where $r_{i_1,i_2,...,i_H} \in \{0, 1\}$ and denotes whether a specific configuration of weights appears in Equation (A.3). So the following condition holds: $\sum_{i_1,i_2,...,i_H=1}^{N} r_{i_1,i_2,...,i_H} = \prod_{i=0}^{H-1} n_i \stackrel{\text{def}}{=} \Psi$.

Definition A.0.2. The mass of the network Ψ is the total number of paths from the input to the output, so $\Psi = \prod_{i=0}^{H-1} n_i$. Let be Λ the H^{th} root of Ψ, so $\Lambda = \sqrt[H]{\Psi}$.

Observation A.0.1. The mass of the network Ψ is different from the total number of paths for a given input γ. In particular: $\Psi = n_0 \cdot \gamma$.

Thanks to the following Definition A.0.3 we can use the redundancy hypotesis to re-express the output of the network as a spin glass model.

Definition A.0.3. A network $\mathcal{M}$ which has the same graph of connections as the original network $\mathcal{N}$ and s unique weights satisfying $s \leq N$ is called a (s, ϵ)-reduction image of $\mathcal{N}$ for some $\epsilon \in [0, 1]$ if the prediction accuracy of $\mathcal{N}$ and $\mathcal{M}$ differ by no more than ϵ.

Choromanska et al. [7] formally prove that reducing the network to a smaller set of unique weights $s \ll N$ does not destroy its predictive power. Specifically, they demonstrate that the predictions of the reduced network maintain a strictly bounded positive correlation with the predictions of the original network, justifying this approximation step.

Under also the uniformity hypotesis, if we consider the $\mathcal{M}$ network to be (s, ϵ)-reduction image of the original network $\mathcal{N}$ for some $s \ll N$, we can re-express the output of the network as follows:

$$Y_s = q \sum_{i_1,i_2,...,i_H=1}^{s} \sum_{j=1}^{t_{i_1,i_2,...,i_H}} X_{i_1,i_2,...,i_H}^{(j)} \rho \prod_{k=1}^{H} w_{i_k} \tag{A.5}$$

where $t_{i_1,i_2,...,i_H} \in \{\mathbb{Z}^+ \cup 0\}$ is the number of times each configuration of weights $(w_{i_1}, w_{i_2}, ..., w_{i_H})$ appears in Equation (A.4) and $\sum_{i_1,i_2,...,i_H=1}^{s} t_{i_1,i_2,...,i_H} = \Psi$.

In this network $\mathcal{M}$ the total possible number of configurations of weights is s^H and the total number of paths is Ψ. Therefore, if we assume that each configuration of weights appears in the

same number of paths (so called *uniformity assumption*, very common with real networks), we can re-express the output of the network as follows:

$$Y_s = q \sum_{i_1,i_2,\ldots,i_H=1}^{s} \sum_{j=1}^{\frac{\Psi}{s^H}} X^{(j)}_{i_1,i_2,\ldots,i_H} \rho \prod_{k=1}^{H} w_{i_k} \tag{A.6}$$

Moreover if we consider $s = \Lambda$ we can re-express the output of the network as:

$$\hat{Y} = \hat{Y}_{(s=\Lambda)} = q \sum_{i_1,i_2,\ldots,i_H=1}^{\Lambda} X_{i_1,i_2,\ldots,i_H} \rho \prod_{k=1}^{H} w_{i_k} \tag{A.7}$$

By assuming that for some positive constant $\mathcal{C}$ weights satisfy the spherical condition $\frac{1}{\Lambda}\sum_{i=1}^{\Lambda} w_i^2 = \mathcal{C}$, and that the loss of the network is the absolute loss $\mathcal{L}^a_{\Lambda,H}(\mathbf{w}) = \mathbb{E}_A[|Y - \hat{Y}|]$ or the hinge loss $\mathcal{L}^h_{\Lambda,H}(\mathbf{w}) = \mathbb{E}_A[\max(0, 1 - Y\hat{Y})]$, we can re-express the loss of the network as follows:

$$\mathcal{L}_{\Lambda,H}(\tilde{\mathbf{w}}) = \mathcal{C}_1 + \mathcal{C}_2 q \sum_{i_1,i_2,\ldots,i_H=1}^{\Lambda} X_{i_1,i_2,\ldots,i_H} \prod_{k=1}^{H} \tilde{w}_{i_k} \tag{A.8}$$

where $\mathcal{C}_1$ and $\mathcal{C}_2$ are some constants and weights $\tilde{w}_i$ are rescaled versions of the original weights w_i satisfying $\frac{1}{\Lambda}\sum_{i=1}^{\Lambda} \tilde{w}_i^2 = 1$.

B. Some math behind Neural Tangent Kernel

B.1 Notation and setting

We will consider a fully connected neural network with parameters θ, $f(\cdot;\theta) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$. Layers are indexed from 0 to L, where the 0^{th} layer is the input layer and the L^{th} layer is the output layer. So in total there are $M = \sum_{l=1}^{L} n_l n_{l-1}$ parameters in the network.

The training dataset is $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \{x_i\}_{i=1}^{N}$ is the input space and $\mathcal{Y} = \{y_i\}_{i=1}^{N}$ is the output space (*label space*).

We will call the activation function σ and $\mathbf{w}^{(l)}$ the weights of the l^{th} layer and $\mathbf{b}^{(l)}$ the biases of the l^{th} layer, respectively. So we have the following recursive definition for the output of the network:

$$\begin{aligned} A^{(0)} &= \mathbf{x}; \\ \tilde{A}^{(l+1)} &= \frac{1}{\sqrt{n_l}} {\mathbf{w}^{(l)}}^{\top} A^{(l)} + \beta \mathbf{b}^{(l)}; \\ A^{(l+1)} &= \sigma(\tilde{A}^{(l+1)}) \end{aligned} \tag{B.1}$$

for all $l = \{0, 1, ..., L-1\}$. The output of the network is $f(\mathbf{x};\theta) = A^{(L)}$.

NTK parametrization applies a rescale weight of $\frac{1}{\sqrt{n_l}}$ for avoiding divergence for infinite width networks. The constant β controls the relative scale of the weights and the biases. All the network parameters are initialized with i.i.d. Gaussian $\mathcal{N}(0,1)$.

B.2 Neural Tangent Kernel

Neural Tangent Kernel (NTK) defined by Jacot et al. [20] is an important tool for understanding the training dynamics of neural networks. As explained by Weng [41], it is possible to understand the intuition behind NTK by starting from the definition of the empirical loss function $\mathcal{L} : \mathbb{R}^M \to \mathbb{R}_+$, using a per sample loss function $\ell : \mathbb{R}^{n_L} \times \mathbb{R}^{n_L} \to \mathbb{R}_+$, as follows:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(\mathbf{x}^{(i)}); \theta, y^{(i)})$$

According to the chain rule, the gradient of the loss function with respect to the parameters θ can be expressed as follows:

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta f(\mathbf{x}^{(i)};\theta) \nabla_f \ell(f(\mathbf{x}^{(i)};\theta), y^{(i)})$$

where $\nabla_\theta f(\mathbf{x}^{(i)};\theta) \in \mathbb{R}^{M \times n_L}$ is the Jacobian of the network output with respect to the parameters θ for the input $\mathbf{x}^{(i)}$ and $\nabla_f \ell(f(\mathbf{x}^{(i)};\theta), y^{(i)}) \in \mathbb{R}^{n_L} \times 1$ is the gradient of the loss function ℓ with respect to the network output.

During training each gradient descent update step introduces a small incremental change in the parameters θ. We can see it as a continuous time process and express the evolution of the parameters θ during training as follows:

$$\frac{d\theta}{dt} = -\nabla_\theta \mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta f(\mathbf{x}^{(i)};\theta)\nabla_f \ell(f(\mathbf{x}^{(i)};\theta), y^{(i)})$$

With the chain rule we can also express the evolution of the network output during training as follows:

$$\frac{df(\mathbf{x};\theta)}{dt} = \frac{df(\mathbf{x};\theta)}{d\theta}\frac{d\theta}{dt} = -\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta f(\mathbf{x};\theta)^\top \nabla_\theta f(\mathbf{x}^{(i)};\theta)\nabla_f \ell(f(\mathbf{x}^{(i)};\theta), y^{(i)})$$

where the term in blue is the Neural Tangent Kernel (NTK) defined as follows:

Definition B.2.1 (Neural Tangent Kernel)**.** The Neural Tangent Kernel (NTK) is defined as the inner product of the Jacobian of the network output with respect to the parameters θ for two different inputs $\mathbf{x}$ and $\mathbf{x}'$ as follows:

$$K(\mathbf{x}, \mathbf{x}';\theta) = \nabla_\theta f(\mathbf{x};\theta)^\top \nabla_\theta f(\mathbf{x}';\theta)$$

where each entry in the output matrix at location (m, n) with $1 \leq m, n \leq n_L$ is:

$$K_{m,n}(\mathbf{x}, \mathbf{x}';\theta) = \sum_{i=1}^{M} \frac{\partial f_m(\mathbf{x};\theta)}{\partial \theta_i}\frac{\partial f_n(\mathbf{x}';\theta)}{\partial \theta_i}$$

Since it is written as a dot product between mapped inputs, we are guaranteed that the NTK is a positive semi-definite kernel.

B.3 NTK in the infinite width limit

When $n_1, \ldots, n_L \to \infty$ the NTK converges to be:

1. deterministic at initialization, so it does not depend on the random initialization of the parameters θ;

2. constant during training.

The proof is based on mathematical induction on the layer index l.

We always have $K^{(0)} = 0$ since the input layer does not have any parameters. When $L = 1$, we can get the representation of NTK at initialization as follows:

$$K^{(1)}(\mathbf{x}, \mathbf{x}') = \frac{1}{n_0}\mathbf{x}^\top \mathbf{x}' + \beta^2$$

In fact, $f(\mathbf{x};\theta) = \tilde{A}^{(1)}(\mathbf{x}) = \frac{1}{\sqrt{n_0}}\mathbf{w}^{(0)\top}\mathbf{x} + \beta\mathbf{b}^{(0)}$ and $f(\mathbf{x}';\theta) = \tilde{A}^{(1)}(\mathbf{x}') = \frac{1}{\sqrt{n_0}}\mathbf{w}^{(0)\top}\mathbf{x}' + \beta\mathbf{b}^{(0)}$. So we have:

$$K^{(1)}(\mathbf{x},\mathbf{x}';\theta) = \left(\frac{\partial f(\mathbf{x};\theta)}{\partial \mathbf{w}^{(0)}}\right)^\top \frac{\partial f(\mathbf{x}';\theta)}{\partial \mathbf{w}^{(0)}} + \left(\frac{\partial f(\mathbf{x}';\theta)}{\partial \mathbf{b}^{(0)}}\right)^\top \frac{\partial f(\mathbf{x};\theta)}{\partial \mathbf{b}^{(0)}} = \frac{1}{n_0}\mathbf{x}^\top\mathbf{x}' + \beta^2$$

Note that since we have only one layer, there is nothing to take on infinite width.

Induction step. We assume that for some $l \in \{1, 2, ..., L-1\}$ layer network with $\tilde{M}$ parameters in total, with $\tilde{\theta} = (\mathbf{w}^{(0)}, \ldots, \mathbf{w}^{l-1}, \mathbf{b}^{(0)}, \ldots, \mathbf{b}^{(l-1)}) \in \mathbb{R}^{\tilde{M}}$ as parameters, has a NTK converging to a deterministic limit when $n_1, \ldots, n_{l-1} \to \infty$.

$$K^{(l)}(\mathbf{x},\mathbf{x}';\tilde{\theta}) = \nabla_{\tilde{\theta}}\tilde{A}^{(l)}(\mathbf{x})^\top \nabla_{\tilde{\theta}}\tilde{A}^{(l)}(\mathbf{x}') \to K_\infty^{(l)}(\mathbf{x},\mathbf{x}')$$

Next let's check the case $L = l + 1$. This layer has additional weight matrix $\mathbf{w}^{(l)}$ and bias vector $\mathbf{b}^{(l)}$ and thus total parameters are $\theta = (\tilde{\theta}, \mathbf{w}^{(l)}, \mathbf{b}^{(l)})$. The output function of this $(l+1)$-layer network is:

$$f(\mathbf{x};\theta) = \tilde{A}^{(l+1)}(\mathbf{x};\theta) = \frac{1}{\sqrt{n_l}}\mathbf{w}^{(l)\top}\sigma(\tilde{A}^{(l)}(\mathbf{x};\theta)) + \beta\mathbf{b}^{(l)}$$

Now we can compute the derivatives:

$$\begin{aligned}
\nabla_{\mathbf{w}^{(l)}} f(\mathbf{x};\theta) &= \frac{1}{\sqrt{n_l}}\sigma(\tilde{A}^{(l)})^\top \in \mathbb{R}^{1\times n_l} \\
\nabla_{\mathbf{b}^{(l)}} f(\mathbf{x};\theta) &= \beta \\
\nabla_{\tilde{\theta}} f(\mathbf{x};\theta) &= \frac{1}{\sqrt{n_l}}\nabla_{\tilde{\theta}}\sigma(\tilde{A}^{(l)})\mathbf{w}^{(l)} \\
&= \frac{1}{\sqrt{n_l}}\begin{bmatrix} \dot{\sigma}(\tilde{A}_1^{(l)})\frac{\partial \tilde{A}_1^{(l)}}{\partial \tilde{\theta}_1} & \ldots & \dot{\sigma}(\tilde{A}_{n_l}^{(l)})\frac{\partial \tilde{A}_{n_l}^{(l)}}{\partial \tilde{\theta}_1} \\ \vdots & & \vdots \\ \dot{\sigma}(\tilde{A}_1^{(l)})\frac{\partial \tilde{A}_1^{(l)}}{\partial \tilde{\theta}_{\tilde{M}}} & \ldots & \dot{\sigma}(\tilde{A}_{n_l}^{(l)})\frac{\partial \tilde{A}_{n_l}^{(l)}}{\partial \tilde{\theta}_{\tilde{M}}} \end{bmatrix}\mathbf{w}^{(l)} \in \mathbb{R}^{\tilde{M}\times n_{l+1}}
\end{aligned}$$

where $\dot{\sigma}$ is the derivative of the activation function σ and each entry at location $(p, m), 1 \le p \le \tilde{M}, 1 \le m \le n_{l+1}$ in the matrix can be written as:

$$\frac{\partial f_m(\mathbf{x};\theta)}{\partial \tilde{\theta}_p} = \sum_{i=1}^{n_l} w_{im}^{(l)}\dot{\sigma}(\tilde{A}_i^{(l)})\nabla_{\tilde{\theta}_p}\tilde{A}_i^{(l)}$$

So putting altogether we have:

$$
\begin{aligned}
&K^{(l+1)}(\mathbf{x},\mathbf{x}';\theta) = \nabla_\theta f(\mathbf{x};\theta)^\top \nabla_\theta f(\mathbf{x}';\theta) \\
&= \nabla_{\mathbf{w}^{(l)}} f(\mathbf{x};\theta)^\top \nabla_{\mathbf{w}^{(l)}} f(\mathbf{x}';\theta) + \nabla_{\mathbf{b}^{(l)}} f(\mathbf{x};\theta)^\top \nabla_{\mathbf{b}^{(l)}} f(\mathbf{x}';\theta) + \nabla_{\tilde{\theta}} f(\mathbf{x};\theta)^\top \nabla_{\tilde{\theta}} f(\mathbf{x}';\theta) \\
&= \frac{1}{n_l}\Bigg[\sigma(\tilde{A}^{(l)})\sigma(\tilde{A}^{(l)})^\top + \beta^2 \\
&+ \mathbf{w}^{(l)\top}\begin{bmatrix} \dot\sigma(\tilde{A}_1^{(l)})\dot\sigma(\tilde{A}_1^{(l)})\sum_{p=1}^{\tilde{M}} \frac{\partial \tilde{A}_1^{(l)}}{\partial \tilde{\theta}_p}\frac{\partial \tilde{A}_1^{(l)}}{\partial \tilde{\theta}_p} & \dots & \dot\sigma(\tilde{A}_1^{(l)})\dot\sigma(\tilde{A}_{n_l}^{(l)})\sum_{p=1}^{\tilde{M}} \frac{\partial \tilde{A}_1^{(l)}}{\partial \tilde{\theta}_p}\frac{\partial \tilde{A}_{n_l}^{(l)}}{\partial \tilde{\theta}_p} \\ \vdots & \ddots & \vdots \\ \dot\sigma(\tilde{A}_{n_l}^{(l)})\dot\sigma(\tilde{A}_1^{(l)})\sum_{p=1}^{\tilde{M}} \frac{\partial \tilde{A}_{n_l}^{(l)}}{\partial \tilde{\theta}_p}\frac{\partial \tilde{A}_1^{(l)}}{\partial \tilde{\theta}_p} & \dots & \dot\sigma(\tilde{A}_{n_l}^{(l)})\dot\sigma(\tilde{A}_{n_l}^{(l)})\sum_{p=1}^{\tilde{M}} \frac{\partial \tilde{A}_{n_l}^{(l)}}{\partial \tilde{\theta}_p}\frac{\partial \tilde{A}_{n_l}^{(l)}}{\partial \tilde{\theta}_p} \end{bmatrix}\mathbf{w}^{(l)}\Bigg] \\
&= \frac{1}{n_l}\Bigg[\sigma(\tilde{A}^{(l)})\sigma(\tilde{A})^\top + \beta^2 \\
&+ \mathbf{w}^{(l)\top}\begin{bmatrix} \dot\sigma(\tilde{A}_1^{(l)})\dot\sigma(\tilde{A}_1^{(l)})K_{11}^{(l)} & \dots & \dot\sigma(\tilde{A}_1^{(l)})\dot\sigma(\tilde{A}_{n_l}^{(l)})K_{1n_l}^{(l)} \\ \vdots & \ddots & \vdots \\ \dot\sigma(\tilde{A}_{n_l}^{(l)})\dot\sigma(\tilde{A}_1^{(l)})K_{n_l1}^{(l)} & \dots & \dot\sigma(\tilde{A}_{n_l}^{(l)})\dot\sigma(\tilde{A}_{n_l}^{(l)})K_{n_ln_l}^{(l)} \end{bmatrix}\mathbf{w}^{(l)}\Bigg]
\end{aligned}
$$

where each individual entry at location (m,n), $1 \le m,n \le n_{l+1}$ in the matrix $K^{(l+1)}$ can be written as follows:

$$
K_{m,n}^{(l+1)} = \frac{1}{n_l}\left[\sigma(\tilde{A}_m^{(l)})\sigma(\tilde{A}_n^{(l)})^\top + \beta^2 + \sum_{i=1}^{n_l}\sum_{j=1}^{n_l} w_{im}^{(l)} w_{jn}^{(l)} \dot\sigma(\tilde{A}_i^{(l)})\dot\sigma(\tilde{A}_j^{(l)})K_{ij}^{(l)}\right]
$$

As $n_l \to \infty$, Lee et al. [22] demonstrate that the empirical covariance terms (highlighted in blue and green) converge to a deterministic limit $\Sigma^{(l+1)}$. Concurrently, by applying the inductive hypothesis to the term in red, we can replace the empirical kernel $K_{ij}^{(l)}$ with its infinite width counterpart. Thus, we obtain the following limits for the individual components:

$$
\begin{aligned}
\frac{1}{n_l}\left[\sigma(\tilde{A}^{(l)})\sigma(\tilde{A}^{(l)})^\top + \beta^2\right] &\to \Sigma^{(l+1)} \\
\frac{1}{n_l}\sum_{i=1}^{n_l}\sum_{j=1}^{n_l} w_{im}^{(l)} w_{jn}^{(l)} \dot\sigma(\tilde{A}_i^{(l)})\dot\sigma(\tilde{A}_j^{(l)})K_{ij}^{(l)} &\to \frac{1}{n_l}\sum_{i=1}^{n_l}\sum_{j=1}^{n_l} w_{im}^{(l)} w_{jn}^{(l)} \dot\sigma(\tilde{A}_i^{(l)})\dot\sigma(\tilde{A}_j^{(l)})K_{\infty,ij}^{(l)}
\end{aligned}
$$

In conclusion, the deterministic nature of the limit relies fundamentally on the initialization properties of the weights. Because the weights are initialized as independent, zero-mean Gaussian variables and are entirely independent of the previous layer's activations, all cross-terms in the summation strictly vanish. Through the Law of Large Numbers, the remaining empirical sum converges to its theoretical expectation, replacing the explicit weight dependency with a deterministic integral over the derivatives, often denoted as $\dot\Sigma^{(l+1)}$. This deterministic factor then simply multiplies the asymptotic kernel of the previous layer, $K_\infty^{(l)}$.

Later, works such as Arora et al. [3] provided proofs with weaker limits, demonstrating that this convergence holds even when the layers are not strictly infinitely wide, but sufficiently large. This concludes the induction step. For a full network with L layers, the global empirical NTK converges to a deterministic kernel that remains constant during the entire training process.

B.4 Linearized models

Thanks to the previous results, we can express the evolution of the network output during training as follows:

$$\frac{df(\theta)}{dt} = -\eta \nabla_\theta f(\theta)^\top \nabla_\theta f(\theta) \nabla_f \mathcal{L} = -\eta K_\infty \nabla_f \mathcal{L}$$

Now we can introduce the linearized model of the network as follows (for brevity we omit the input $\mathbf{x}$ in the notation):

Definition B.4.1 (Linearized model)**.** The linearized model of a neural network $f(\cdot;\theta)$ around the initialization θ_0 is defined as follows:

$$f^{\text{lin}}(\theta(t)) = f(\theta(0)) + \nabla_\theta f(\theta)|_{\theta=\theta(0)}(\theta(t) - \theta(0))$$

Since we have that:

$$\theta(t) - \theta(0) = -\eta \nabla_\theta \mathcal{L}(\theta) = -\eta \nabla_\theta f(\theta)^\top \nabla_f \mathcal{L}$$

we can express the linearized model as follows:

$$f^{\text{lin}}(\theta(t)) - f(\theta(0)) = -\eta \nabla_\theta f(\theta)|_{\theta=\theta(0)} \nabla_\theta f(\theta)|^\top_{\theta=\theta(0)} \nabla_f \mathcal{L}$$

So if we compute the evolution of the linearized model during training we have:

$$\frac{df^{\text{lin}}(\theta)}{dt} = -\eta K(\theta(0)) \nabla_f \mathcal{L} = -\eta K_\infty \nabla_f \mathcal{L}$$

This implies that the evolution of the linearized model during training is the same as the evolution of the original model during training, since they are both governed by the same kernel K_∞. Therefore, in the infinite width limit, the training dynamics of a neural network can be fully described by its linearized model around the initialization.

If the empirical loss function is a simple mean squared error (MSE) loss, $\nabla_\theta \mathcal{L} = f(\mathcal{X};\theta) - \mathcal{Y}$, the dynamics of the network becomes a simple linear ODE and it can be solved in closed form as follows:

$$\begin{aligned}
\frac{df(\theta)}{dt} &= -\eta K_\infty (f(\theta) - \mathcal{Y}) \\
\frac{dg(\theta)}{dt} &= -\eta K_\infty g(\theta) \text{ where } g(\theta) = f(\theta) - \mathcal{Y} \\
\int \frac{dg(\theta)}{g(\theta)} &= -\eta \int K_\infty dt \\
g(\theta) &= Ce^{-\eta K_\infty t}
\end{aligned}$$

We can observe that when $t = 0$ we have $g(\theta) = f(\theta(0)) - \mathcal{Y}$, so $C = f(\theta(0)) - \mathcal{Y}$. Therefore, the solution of the ODE is:

$$f(\theta) = (f(\theta(0)) - \mathcal{Y})e^{-\eta K_\infty t} + \mathcal{Y} = f(\theta(0))e^{-\eta K_\infty t} + (I - e^{-\eta K_\infty t})\mathcal{Y}$$

This solution shows that the network output converges to the target $\mathcal{Y}$ at a rate determined by the eigenvalues of the kernel K_∞.

This exact property of local linearity and continuous equivalence to a linearized model is what formally justifies the algebraic manipulation of weights used throughout this thesis. By defining the task vector as $\tau = \theta - \theta_0$, we can safely perform operations like addition and analogy directly in the parameter space, with the mathematical guarantee that these operations translate linearly into the functional space of the model.

B.5 Recent bounds on the linear regime and spectral stability

It is well known that the infinite width limit is an idealized setting and that real-world neural networks are finite. However, recent works by Afzal et al. [2] have proposed theoretical bounds confirming that the linearized model is a good approximation of the original model even for finite width models (such as LLMs via Parameter-Efficient Fine-Tuning (PEFT)), provided that an appropriate inductive bias restricts the optimization trajectory.

Specifically, they formally bound the deviation of the non-linear network from its linearized approximation:

Theorem B.5.1 (Distance to linearized model, informal (Theorems 3 and 4 in [2]))**.** Assume the neural network function and its gradients are Lipschitz continuous. Under an ℓ_2-distance regularization towards the pre-trained weights θ_0, the distance between the inference of the fully non-linear fine-tuned model $f_{\theta_t}(X)$ and its linearized counterpart $\overline{f}_{\bar{\theta}_t}(X)$ is strictly bounded.

Ghojogh et al. [15] have proven the following Theorem which helps Afzal et al. [2] for showing the equivalence of the fine-tuning problem to a kernel regression problem in the Reproducing Kernel Hilbert Space (RKHS) defined by the NTK.

Theorem B.5.2 (Representer Theorem)**.** For a set of data $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, consider a RKHS $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$ with kernel function k. For any loss function $\ell = \mathbb{R}^2 \to \mathbb{R}$ consider the optimization problem:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) + \sigma\Omega(\|f\|_k),$$

where $\eta \geq 0$ is the regularization parameter and $\Omega(\|f\|_k)$ is a penalty term (such as $\|f\|_k^2$). The solution of this optimization problem is:

$$f^* = \sum_{i=1}^{n} \alpha_i k(\cdot, \mathbf{x}_i)$$

Proof. Assume we project the function f onto the subspace spanned by the kernel functions evaluated at the training data points $\{k(\mathbf{x}_i, \cdot)\}_{i=1}^n$. The function f can be decomposed into components along and orthogonal to this subspace, respectively denoted as follows

$$f = f_\parallel + f_\perp \implies \|f\|_k^2 = \|f_\parallel\|_k^2 + \|f_\perp\|_k^2 \geq \|f_\parallel\|_k^2 \tag{B.2}$$

Moreover, using the reproducing property of the RKHS (exposed in D.0.4), we have:

$$f(\mathbf{x}_i) = \langle f, k(\mathbf{x}_i, \cdot)\rangle_k = \langle f_\parallel, k(\mathbf{x}_i, \cdot)\rangle_k + \langle f_\perp, k(\mathbf{x}_i, \cdot)\rangle_k = \langle f_\parallel, k(\mathbf{x}_i, \cdot)\rangle_k = f_\parallel(\mathbf{x}_i) \tag{B.3}$$

The second equivalence is due to the fact that $f_\perp$ is orthogonal to the subspace spanned by $\{k(\mathbf{x}_i, \cdot)\}_{i=1}^n$, so the inner product between them is zero. According to Equation B.3, we have:

$$\sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) = \sum_{i=1}^{n} \ell(f_\parallel(\mathbf{x}_i), y_i) \tag{B.4}$$

And now, using Equations B.2 and B.4, we can say:

$$\min_{f\in\mathcal{H}} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) + \sigma\Omega(\|f\|_k) = \min_{f\in\mathcal{H}} \sum_{i=1}^{n} \ell(f_\parallel(\mathbf{x}_i), y_i) + \sigma\Omega(\|f_\parallel\|_k^2) \tag{B.5}$$

Hence, for this minimization problem, we only require the component lying in the subspace spanned by the kernel of RKHS. Therefore, we can represent the solution as a linear combination of basis vectors $\{k(\mathbf{x}_i, \cdot)\}_{i=1}^n$. □

Observation B.5.1. In Chapter D we say that an Hilbert space can be infinite dimensional. According to the Definition D.0.3, RKHS is an Hilbert space, so it can be infinite dimensional. However, the Representer Theorem B.5.2 states that the solution of the optimization problem lies in the subspace spanned by the kernel functions evaluated at the training data points $\{k(\mathbf{x}_i, \cdot)\}_{i=1}^n$. This subspace is finite dimensional (with dimension at most n), so we can say that the solution of the optimization problem lies in a finite dimensional subspace of the RKHS; although, that finite number of dimensions can be very large.

Afzal et al. [2] have considered $\Omega(\|f\|_k) = \|f\|_k^2$ and $\ell(f(\mathbf{x}_i), y_i) = \|f(\mathbf{x}_i) - y_i\|_2^2$ and the minimization problem can be written as follows:

$$\min_{f\in\mathcal{H}} \sum_{(\mathbf{x},y)\in\mathcal{D}} \|f(\mathbf{x}) - y\|^2 + \sigma\|f\|_k^2 \tag{B.6}$$

and have shown that the solution of the fine-tuning problem is a kernel regression problem in the RKHS defined by the NTK. For this purpose they have used the Representer Theorem B.5.2 to show that the solution of the fine-tuning problem can be expressed as a linear combination of the kernel functions evaluated at the training data points. So we have:

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i K(\cdot, \mathbf{x}_i) = \alpha^\top \mathbf{K}(\cdot, \mathbf{X}),$$

where $\mathbf{K}(\cdot, \mathbf{X}) = [k(\cdot, \mathbf{x}_1), \ldots, k(\cdot, \mathbf{x}_n)] \in \mathbb{R}^{1\times n}$. Substituting Equation B.5 into Equation B.6, we can express the optimization problem in terms of the kernel matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ as follows:

$$\min_{\alpha} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\|\alpha^\top \mathbf{K}(\cdot, \mathbf{X}) - \mathbf{Y}\|^2\right] + \sigma\|f\|_k^2$$

which is a convex problem with a closed form solution: $\alpha^* = (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma\mathbf{I})^{-1}\mathbf{Y}$. Equivalently:

$$f^*(\cdot) = \mathbf{K}(\cdot, \mathbf{X})[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma\mathbf{I}]^{-1}\mathbf{Y},$$

where $[\mathbf{K}(\mathbf{X}, \mathbf{X})]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$

Furthermore, their work explicitly connects the eigenvalue spectrum of the NTK to the empirical risk $\mathcal{R}(\theta)$ of the fine-tuned model, providing a direct metric to predict adaptation performance:

Theorem B.5.3 (Empirical risk and spectral perturbation, informal (Theorems 5 and 6 in [2]))**.** In the linearized fine-tuning regime, the empirical risk of the model is strictly bounded by the maximum $\lambda_{\text{max}}(\mathbf{K})$ and minimum $\lambda_{\text{min}}(\mathbf{K})$ eigenvalues of the NTK matrix $\mathbf{K}$. Furthermore, if a new subset of trainable parameters is added (inducing a perturbation kernel $\mathbf{S}$), the eigenvalues of the new kernel $\mathbf{K} + \mathbf{S}$ are bounded within a narrow multiplicative factor $(1 \pm \eta_{\text{pert}})$ of the original eigenvalues $\lambda_i(K)$.

In conclusion, these recent theoretical frameworks bridge the gap between idealized infinite-width limits and the practical fine-tuning of finite modern architectures. By rigorously bounding the deviation from the linear regime and linking the empirical risk directly to the NTK spectrum, they mathematically validate the core premise of this work: analyzing the geometric and spectral properties of the tangent space provides a robust, predictable mechanism to understand why and how task arithmetic operates without catastrophic interference.

C. Jacobian vector product

As demonstrated in Chapter B, the functional behavior of a fine-tuned network in the infinite-width limit can be approximated by its linearized model:

$$f_{\text{lin}}(\theta(t)) = f(\theta_0) + \nabla_\theta f(\theta_0)(\theta(t) - \theta_0)$$

where $\tau = \theta(t) - \theta_0$ is the task vector. While mathematically elegant, evaluating this expression directly is computationally intractable for modern deep neural networks. The Jacobian matrix $\nabla_\theta f(\theta_0) \in \mathbb{R}^{C \times M}$ (where C is the output dimension and M is the number of parameters) requires hundreds of gigabytes of memory to be explicitly instantiated, causing immediate Out-Of-Memory (OOM) errors.

However, we do not need the full Jacobian matrix; we only need its product with the task vector τ. This problem can be elegantly solved using Automatic Differentiation (AD) as exposed by Baydin et al. [6]. Unlike numerical differentiation, which is prone to truncation errors, or symbolic differentiation, which suffers from expression swell, AD evaluates derivatives at machine precision with ideal asymptotic efficiency.

Forward mode AD provides a highly efficient and "matrix-free" way of computing Jacobian-vector products (JVPs). By initializing the tangent variables with the elements of our task vector τ (i.e., setting the initial derivative seeds to $\dot{\theta} = \tau$), the forward accumulation mode allows us to compute the exact directional derivative (the JVP) in just one forward pass of the computational graph.

To understand how a single forward pass calculates both the output and the derivative, we must introduce the underlying mechanism of *dual numbers* (or primal-tangent pairs). In forward mode AD, every variable v in the program is internally replaced by a tuple $(v, \dot{v})$, where v is the primal value and $\dot{v}$ is its corresponding tangent value. Elementary mathematical operations are then overloaded to operate simultaneously on these pairs using the chain rule. For instance, a multiplication operation $A = X_1 \times X_2$ is automatically evaluated as $(v_A, \dot{v}_A) = (v_{X_1} v_{X_2}, \dot{v}_{X_1} v_{X_2} + v_{X_1} \dot{v}_{X_2})$.

Let's consider for example the evaluation trace (Fig. C.1) of the function $f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ at the point $(x_1, x_2) = (2, 5)$ with the task vector $\tau = (1, 0)$. The forward mode AD would compute the sequence of augmented operations in Table C.1. Extending this concept to a neural network, it is easy to see that a run of the code computes the directional derivative $\dot{y}_j = \sum_{i=1}^{M} \frac{\partial y_j}{\partial \theta_i} \tau_i$ for all output dimensions $j = 1, \ldots, C$ simultaneously. This yields the complete JVP in just one forward pass, without ever instantiating the full Jacobian matrix:

$$\mathbf{J}_f \tau = \begin{bmatrix} \frac{\partial y_1}{\partial \theta_1} & \cdots & \frac{\partial y_1}{\partial \theta_M} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_C}{\partial \theta_1} & \cdots & \frac{\partial y_C}{\partial \theta_M} \end{bmatrix} \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_M \end{bmatrix}$$

Modern deep learning frameworks implement these JVPs via forward-mode autodiff or equivalent reverse-on-forward tricks. This guarantees that the computational complexity of evaluating the

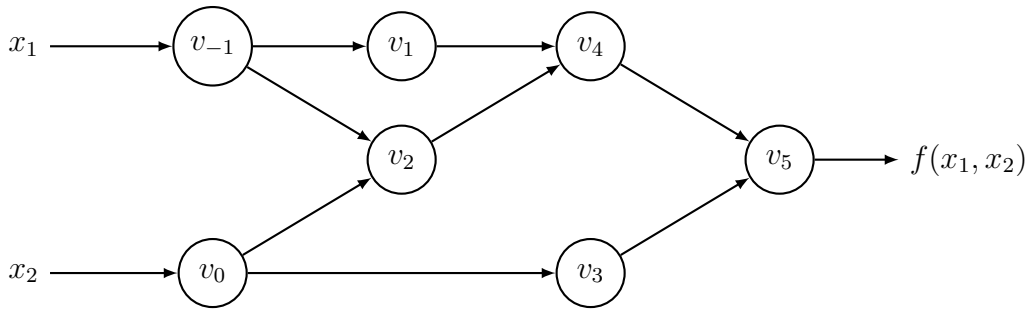


Figure C.1: Computational graph of the example evaluated function. Here we have that variables $v_{i-2} = x_i$ for $i = 1, 2$; variables v_i for $i = 1, 2, 3, 4$ are the intermediate variables; and variable v_5 is the output of the function.

Forward Primal Trace			Forward Tangent (Derivative) Trace		
v_{-1}	$= x_1$	$= 2$	$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
v_0	$= x_2$	$= 5$	$\dot{v}_0$	$= \dot{x}_2$	$= 0$
v_1	$= \ln v_{-1}$	$= \ln 2$	$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$	$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
v_3	$= \sin v_0$	$= \sin 5$	$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$	$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$	$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
y	$= v_5$	$= 11.652$	$\dot{\boldsymbol{y}}$	$= \dot{\boldsymbol{v}}_5$	$= \mathbf{5.5}$

Table C.1: Forward mode AD example, with $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ evaluated at $(x_1, x_2) = (2, 5)$ and setting $\dot{x}_1 = 1$ and $\dot{x}_2 = 0$. The original forward evaluation of the primals on the left is augmented by the tangent operations on the right, where each line complements the original directly to its left.

linearized model f_{lin} is essentially bounded to $\mathcal{O}(M)$, which is practically equivalent to the cost of a standard single forward pass of the non-linear network. This makes the training and evaluation of linearized models like CLIP ViT strictly feasible on standard GPU hardware.

D. Reproducing Kernel Hilbert Space

To properly understand the eigenfunction localization theory proposed by Ortiz-Jimenez et al. [28], we must shift our perspective from the parameter space of the neural network to its function space. In this context, we need to introduce the mathematical framework of **Reproducing Kernel Hilbert Spaces** (RKHS) as well exposed by Meneghetti [26].

We begin by recalling the foundational structures of functional analysis. The most general complete spaces are Banach spaces, but to measure angles and projections between functions, we require the geometric structure of a Hilbert space:

Definition D.0.1 (Banach space and Hilbert space)**.** A Banach space is a complete normed vector space $(\mathbb{V}, \|\cdot\|)$. So it is a vector space endowed with a norm such that every Cauchy sequence

$$(x_n)_{n=0}^{\infty} \subset \mathbb{V},\ \exists x \in \mathbb{V} : \lim_{n\to\infty} x_n = x$$

A Hilbert space $\mathcal{H}$ is a Banach space endowed with a dot-product operation, therefore is a Banach space whose norm derives from an internal product.

An example of an Hilbert space is the $\mathcal{L}_2$ space of square integrable functions, where the dot product of functions from $\mathbb{R}^d$ to $\mathbb{R}$ is defined as: $\langle f, g\rangle = \int_{-\infty}^{\infty} f(x)g(x)dx$ and therefore $\|f\|_{\mathcal{L}_2} = (\int_{-\infty}^{\infty} |f(x)|^2 dx)^{1/2}$.

When dealing with spaces of functions, a fundamental operation is evaluating a function f (e.g., our neural network) at a specific data point x. In general Hilbert spaces (such as the standard $\mathcal{L}_2$ space), point evaluation is not always well-behaved or even mathematically well-defined. To ensure that evaluating our model on a specific input is a stable operation, we focus on the evaluation functional:

Definition D.0.2 (Dirac Evaluation functional)**.** Let $\mathcal{H}$ be the $\mathcal{L}_2$ space of functions from $\mathcal{X}$ to $\mathbb{R}$ for some non empty set $\mathcal{X}$. For an element $x \in \mathcal{X}$, a **Dirac evaluation functional** at x is a functional $\delta_x : \mathcal{H} \to \mathbb{R}$, s.t. $\delta_x(f) = f(x)$ for all $f \in \mathcal{H}$.

In this case, let $\mathcal{X}$ be $\mathbb{R}^d$ for some d. We say that δ_x is bounded if $\exists M > 0, M \in \mathbb{R}$:

$$\|\delta_x(f)\| = |f(x)| \leq M\|f\|_{\mathcal{L}_2}\ \forall f \in \mathcal{H} \text{ and } \forall x \in \mathbb{R}^d$$

This specific stability property is exactly what defines our space of interest:

Definition D.0.3 (Reproducing Kernel Hilbert Space)**.** A **Reproducing Kernel Hilbert Space** (RKHS) is an Hilbert space $\mathcal{H}$ where all the Dirac evaluation functionals δ_x are bounded and continuous.

The continuity of the evaluation functional might seem like a minor technicality, but it unlocks a profound geometric property thanks to the Riesz Representation Theorem:

Theorem D.0.1 (Riesz Representation Theorem)**.** If Φ is a bounded linear functional on a Hilbert space $\mathcal{H}$, then there exists a unique element $u \in \mathcal{H}$ such that $\Phi(f) = \langle f, u \rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$.

Applying this theorem to our bounded Dirac functional δ_x, we deduce that there must exist a unique function $k_x \in \mathcal{H}$ such that evaluating f at x is perfectly equivalent to taking the inner product between f and k_x, i.e., $f(x) = \langle f, k_x \rangle_{\mathcal{H}}$. This unique element acts as the representative of the point x within the function space, leading to the final definition:

Definition D.0.4 (Reproducing kernel)**.** Let $\mathcal{H}$ be an Hilbert space of function from $\mathcal{X}$ to $\mathbb{R}(\mathcal{L}_2(\mathcal{X}))$ with $\mathcal{X}$ non empty. A function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a **reproducing kernel** of $\mathcal{H}$ if it satisfies:

- $\forall x \in \mathcal{X}$ $k_x = \mathcal{K}(\cdot, x) \in \mathcal{H}$, k_x is called **representer at** x.
- $\forall x \in \mathcal{X}$, $\forall f \in \mathcal{H}$ $f(x) = \langle f, k_x \rangle_{\mathcal{H}} = \langle f, \mathcal{K}(\cdot, x) \rangle_{\mathcal{H}}$ (**reproducing property**).

It is important to note that $k_x \in \mathcal{H}$ is a function from $\mathcal{X}$ to $\mathbb{R}$, such that $k_x(y) = \mathcal{K}(x, y)$. In particular $\forall x, x' \in \mathcal{X}$ we have:

$$\mathcal{K}(x, x') = \langle \mathcal{K}(\cdot, x), \mathcal{K}(\cdot, x') \rangle_{\mathcal{H}} = \langle k_x, k_{x'} \rangle_{\mathcal{H}}$$

where $k_x, k_{x'}$ are respectively the unique representatives of δ_x and $\delta_{x'}$. This means that the kernel function $\mathcal{K}$ is symmetric and positive definite, from the definition of dot product.

From the first property of the Reproducing Kernel we know that $\mathcal{H}$ contains all function of the form $f = \sum_{j=1}^{d} \alpha_j \mathcal{K}(\cdot, x_j)$ if $x_j \in \mathcal{X}$ and we have

$$\|f\|_{\mathcal{H}}^2 = \sum_{j=1}^{d} \sum_{i=1}^{d} \alpha_j \alpha_i \langle \mathcal{K}(\cdot, x_j), \mathcal{K}(\cdot, x_i) \rangle_{\mathcal{H}} = \sum_{j=1}^{d} \sum_{i=1}^{d} \alpha_j \alpha_i \mathcal{K}(x_j, x_i)$$

This demonstrates that every RKHS uniquely defines a symmetric, positive-definite kernel $\mathcal{K}$, and conversely, the Moore-Aronszajn theorem guarantees that every positive-definite kernel uniquely defines an RKHS. To understand the complexity and the structure of the functions learned by the network, it is highly beneficial to decompose the kernel into its fundamental components. If the positive-definite kernel $\mathcal{K}$ is continuous and has a finite trace, the *Mercer-Hilbert-Schmidt theorem* states that it admits an absolutely and uniformly convergent spectral decomposition:

Theorem D.0.2 (Mercer-Hilbert-Schmidt)**.** If $\mathcal{K}$ is a positive definite kernel (that is continuous with finite trace), then there exists an infinite sequence of eigenfunctions $(\Phi_i)_{i=1}^{\infty}$ and eigenvalues $(\lambda_i)_{i=1}^{\infty}$ with $\lambda_1 \geq \lambda_2 \geq \dots$ and $\mathcal{K}$ can be written as:

$$\mathcal{K}(x, x') = \sum_{i=1}^{\infty} \lambda_i \Phi_i(x) \Phi_i(x')$$

In this way we can construct a RKHS $\mathcal{H}_{\mathcal{K}}$ such that: $\mathcal{H}_{\mathcal{K}} := \{f : f = \sum_{i=1}^{\infty} c_i \Phi_i\}$. So for $f \in \mathcal{L}_2$, we will denote f in terms of its coefficients in the eigenfunctions:

$$f_i = \langle f, \Phi_i \rangle_{\mathcal{L}_2} = \int f(x) \Phi_i(x) dx$$

It is a basic result of Fourier analysis that such representation exists and is unique. Given all this, we can now define the inner product on $\mathcal{H}_\mathcal{K}$ as follows:

$$\langle f, g \rangle = \sum_{i=1}^{\infty} \frac{f_i g_i}{\lambda_i}$$

where we used the $\mathcal{H}_\mathcal{K}$-ortogonality $\langle \Phi_i, \Phi_j \rangle_{\mathcal{H}_\mathcal{K}} = \frac{\delta_{i,j}}{\sqrt{\lambda_i}\sqrt{\lambda_j}}$.

It's possible to note that $\mathcal{K}$ is the Reproducing Kernel of $\mathcal{H}_\mathcal{K}$ since the eigenfunction expansion of $\mathcal{K}$ given by Theorem D.0.2 and the orthogonality of the eigenfunctions imply that

$$\langle f, \mathcal{K}(\cdot, x) \rangle_{\mathcal{H}_\mathcal{K}} = \left\langle \sum_{j=1}^{\infty} c_j \Phi_j, \sum_{i=1}^{\infty} \lambda_i \Phi_i \Phi_i(x) \right\rangle_{\mathcal{H}_\mathcal{K}} = \sum_{i=1}^{\infty} \frac{c_i \lambda_i \Phi_i(x)}{\lambda_i} = \sum_{i=1}^{\infty} c_i \Phi_i(x) = f(x)$$

Building upon this inner product, the squared RKHS norm of a function $f = \sum_{i=1}^{\infty} c_i \Phi_i$ is naturally defined by taking the inner product of the function with itself:

$$\|f\|^2_{\mathcal{H}_\mathcal{K}} = \langle f, f \rangle_{\mathcal{H}_\mathcal{K}} = \sum_{i=1}^{\infty} \frac{c_i^2}{\lambda_i}$$

This specific formulation is the cornerstone of the theoretical analysis presented by Ortiz-Jimenez et al. [28]. The RKHS norm essentially measures the "complexity" or the "energy cost" of learning a specific function f using the kernel $\mathcal{K}$. Because the squared coefficients c_i^2 are divided by the eigenvalues λ_i, the norm heavily penalizes functions that rely on eigenfunctions associated with small eigenvalues.

In the context of the Neural Tangent Kernel (NTK), large eigenvalues (λ_i) correspond to patterns or features that the neural network learns quickly and efficiently, whereas small eigenvalues correspond to high-frequency noise that is hard to fit. Therefore, the requirement that a target task function f^* must have a finite kernel norm ($\|f^*\|^2_{\mathcal{H}_\mathcal{K}} < +\infty$) mathematically translates the inductive bias of neural networks: they naturally favor smooth functions aligned with the principal components of their empirical kernel.

Ultimately, this theoretical framework justifies why task arithmetic is feasible. As demonstrated in Chapter 4, when the principal eigenfunctions (Φ_i) of the NTK are strictly localized within specific task domains, fine-tuning on a given task modifies only a specific subset of the kernel's basis. This structural property ensures that the RKHS representations of different tasks remain orthogonal and decoupled, allowing the linearized models to be algebraically added without catastrophic cross-task interference.

E. Proof of Theorem 4.1.1

For clarity reasons, we will report the statement of Theorem 4.1.1 here again:

Theorem E.0.1. Let $\theta_{\text{TA}}^{(k)} = \theta_{\text{base}} + \alpha \sum_{t \in T} \tau_t^{(k)}$ be the model obtained using vanilla task arithmetic with scaling parameter α. Let $\{\theta_t^{(k)}\}_{t \in T}$ be produced by running k full-batch GD epochs with step size η on each task, and let $\theta_{\text{MT}}^{(k)}$ be the model obtained from k GD epochs with step size $\alpha\eta$ on the aggregated loss $\mathcal{L} = \sum_{t \in T} \ell_t$. It holds that:

$$\theta_{\text{TA}}^{(1)} = \theta_{\text{MT}}^{(1)} \tag{E.1}$$

$$\theta_{\text{TA}}^{(k)} = \theta_{\text{MT}}^{(k)} + \eta^2 C(\{\theta_{\text{MT}}^{(j)}\}_{j=1}^{k-2}) + \mathcal{O}(\eta^3) \quad \text{for } k > 1 \tag{E.2}$$

$$\text{where} \quad C(\{\theta_{\text{MT}}^{(j)}\}_{j=1}^{h}) = \sum_{t \in T} \sum_{e=0}^{h} \nabla^2 \ell_t(\theta_{\text{MT}}^{(e)}) \sum_{m=0}^{e} r_t(\theta_{\text{MT}}^{(m)}) \tag{E.3}$$

$$\text{with} \quad r_t(\theta) := \alpha \sum_{t' \in T \setminus \{t\}} \nabla \ell_{t'}(\theta) + (\alpha - 1) \nabla \ell_t(\theta)$$

All heavy computation is done once inside Lemma E.0.1, which is the core technical result of this section. Before stating the lemma, we need to introduce some notation to keep track of the accumulated gradients and curvature effects across epochs. For $t \in T$ and $k \geq 0$, we define:

- $r_t(\cdot)$ quantifies the mismatch between the two gradients at one step.

$$r_t(\theta) = \alpha \sum_{t' \in T \setminus \{t\}} \nabla \ell_{t'}(\theta) + (\alpha - 1) \nabla \ell_t(\theta) = \alpha \sum_{t' \in T} \nabla \ell_{t'}(\theta) - \nabla \ell_t(\theta) \tag{E.4}$$

- $p_t^k(\cdot)$ represents the accumulation of those mismatches over many steps.

$$p_t^k(\theta_{\text{base}}, \theta_{\text{MT}}^{(1)}, \ldots, \theta_{\text{MT}}^{(k)}) = \sum_{j=0}^{k} r_t(\theta_{\text{MT}}^{(j)}) \tag{E.5}$$

- $s_t^k(\cdot)$ accounts for the extra bending introduced by curvature once the paths have parted.

$$s_t^k(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(k)}) = \sum_{j=0}^{k} \nabla^2 \ell_t(\theta_{\text{MT}}^{(j)}) [p_t^j(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(j-1)})].$$

Lemma E.0.1. Using the notation just introduced, it holds that

$$\theta_i^{(1)} = \theta_{\text{MT}}^{(1)} + \eta p_i^0(\theta_{\text{base}})$$

and for $m \geq 2$

$$\begin{aligned} \theta_i^{(m+1)} = \theta_{\text{MT}}^{(m+1)} &+ \eta p_i^m(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(m)}) \\ &- \eta^2 s_i^{m-1}(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(m-1)}) + O(\eta^3) \end{aligned}$$

Proof. The proof is by induction on m.

$m = 1$**. First epoch.** We have $\forall i \in T$ from the previous definitions that:

$$\theta_i^{(1)} = \theta_{\text{base}} - \eta \nabla \ell_i(\theta_{\text{base}}) \text{ while } \theta_{\text{MT}}^{(1)} = \theta_{\text{base}} - \alpha\eta \sum_{t \in T} \nabla \ell_t(\theta_{\text{base}})$$

Consequently, we have that:

$$\begin{aligned}\theta_i^{(1)} &= \theta_{\text{MT}}^{(1)} + \eta \left[\alpha \sum_{j \neq i} \nabla \ell_j(\theta_{\text{base}}) + (\alpha - 1) \nabla \ell_i(\theta_{\text{base}}) \right] \\ &= \theta_{\text{MT}}^{(1)} + \eta r_i(\theta_{\text{base}}) = \theta_{\text{MT}}^{(1)} + \eta p_i^0(\theta_{\text{base}}).\end{aligned}$$

$m = 2$**. Second epoch.**

$$\begin{aligned}\theta_i^{(2)} &= \theta_i^{(1)} - \eta \nabla \ell_i(\theta_i^{(1)}) \\ &\overset{(1)}{=} \theta_{\text{MT}}^{(1)} + \eta r_i(\theta_{\text{base}}) - \eta \nabla \ell_i \left(\theta_{\text{MT}}^{(1)} + \eta r_i(\theta_{\text{base}}) \right) \\ &\overset{Taylor}{\approx} \theta_{\text{MT}}^{(1)} + \eta r_i(\theta_{\text{base}}) - \eta \nabla \ell_i(\theta_{\text{MT}}^{(1)}) - \eta^2 \nabla^2 \ell_i(\theta_{\text{MT}}^{(1)}) r_i(\theta_{\text{base}}) + O(\eta^3) \\ &= \theta_{\text{MT}}^{(1)} - \eta \nabla \ell_i(\ theta a_{\text{MT}}^{(1)}) + \eta r_i(\theta_{\text{base}}) - \eta^2 \nabla^2 \ell_i(\theta_{\text{MT}}^{(1)}) r_i(\theta_{\text{base}}) + O(\eta^3)\end{aligned}$$

where in (1) we used the result of the first epoch.
Adding and subtracting $\eta\alpha \sum_{t \in T} \nabla \ell_i(\ theta a_{\text{MT}}^{(1)})$

$$\begin{aligned}\theta_i^{(2)} &= \underbrace{\theta_{MT}^{(1)} - \eta\alpha \sum_t \nabla \bar{\ell}_t(\theta_{MT}^{(1)})}_{= \theta_{MT}^{(2)}} + \underbrace{\eta\alpha \sum_{t \in T} \nabla \ell_i(\ theta a_{\text{MT}}^{(1)}) - \eta \nabla \ell_i(\ theta a_{\text{MT}}^{(1)})}_{\eta r_i(\theta_{\text{MT}}^{(1)})} + \eta r_i(\theta_{\text{base}}) \\ &\quad - \eta^2 \nabla^2 \ell_i(\theta_{\text{MT}}^{(1)}) r_i(\theta_{\text{base}}) + O(\eta^3) \\ &= \theta_{\text{MT}}^{(2)} + \eta r_i(\theta_{\text{MT}}^{(1)}) + \eta r_i(\theta_{\text{base}}) - \eta^2 \nabla^2 \ell_i(\theta_{\text{MT}}^{(1)}) r_i(\theta_{\text{base}}) + O(\eta^3) \\ &= \theta_{\text{MT}}^{(2)} + \eta p_i^1(\theta_{\text{base}}, \theta_{\text{MT}}^{(1)}) - \eta^2 s_i^0(\theta_{\text{base}}) + O(\eta^3)\end{aligned}$$

Inductive step Let us assume that

$$\theta_i^{(m)} = \theta_{\text{MT}}^{(m)} + \eta p_i^{m-1}(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(m-1)}) - \eta^2 s_i^{m-2}(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(m-2)}) + O(\eta^3)$$

We can derive that

$$
\begin{aligned}
\theta_i^{(m+1)} &= \theta_i^{(m)} - \eta\nabla\ell_i(\theta_i^{(m)}) \\
&\overset{(2)}{=} \theta_{\text{MT}}^{(m)} + \eta p_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)}) - \eta^2 s_i^{m-2}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-2)}) - \eta\nabla\ell_i(\theta_i^{(m)}) + \mathcal{O}(\eta^3) \\
&\overset{(3)}{=} \theta_{\text{MT}}^{(m)} + \eta p_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)}) - \eta^2 s_i^{m-2}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-2)}) \\
&\quad -\eta\nabla\ell_i\left(\theta_{\text{MT}}^{(m)} + \eta p_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)}) - \eta^2 s_i^{m-2}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-2)})\right) \\
&\quad + \mathcal{O}(\eta^3) \\
&\overset{Taylor}{\approx} \theta_{\text{MT}}^{(m)} + \eta p_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)}) - \eta^2 s_i^{m-2}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-2)}) \\
&\quad -\eta\nabla\overline{L}_i(\theta_{\text{MT}}^{(m)}) - \eta\nabla^2\overline{L}_i(\theta_{\text{MT}}^{(m)})\left(\eta p_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)}) - \eta^2 s_i^{m-2}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-2)})\right) \\
&\quad + \mathcal{O}(\eta^3) \\
&= \theta_{\text{MT}}^{(m)} + \eta p_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)}) - \eta^2 s_i^{m-2}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-2)}) - \eta\nabla\ell_i(\theta_{\text{MT}}^{(m)}) \\
&\quad - \eta^2\nabla^2\ell_i(\theta_{\text{MT}}^{(m)}) p_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)}) + \mathcal{O}(\eta^3) \\
&\overset{(4)}{=} \underbrace{\theta_{\text{MT}}^{(m)} - \eta\alpha\sum_{t\in T}\nabla\ell_t(\theta_{\text{MT}}^{(m)})}_{=\theta_{\text{MT}}^{(m+1)}} + \eta p_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)}) \\
&\quad + \underbrace{\eta\alpha\sum_{t\in T}\nabla\ell_t(\theta_{\text{MT}}^{(m)}) - \eta\nabla\ell_i(\theta_{\text{MT}}^{(m)})}_{\eta r_i(\theta_{\text{MT}}^{(m)})} \\
&\quad \underbrace{- \eta^2\left(s_i^{m-2}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-2)}) + \nabla^2\ell_i(\theta_{\text{MT}}^{(m)}) p_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)})\right)}_{\eta^2 s_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)})} + \mathcal{O}(\eta^3) \\
&= \theta_{\text{MT}}^{(m+1)} + \eta p_i^{m}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m)}) - \eta^2 s_i^{m-1}(\theta_{\text{base}},\dots,\theta_{\text{MT}}^{(m-1)}) + \mathcal{O}(\eta^3)
\end{aligned}
$$

where in (2) and (3) we used the inductive hypothesis, while in (4) we added and subtracted the red term as we have done previously. The last line follows from the definition of p_i^m and s_i^{m-1}. This concludes the inductive step and therefore the proof of the lemma. □

Proof Theorem. After one epoch of fine-tuning on each task, the task vectors are simply the negative gradients of the losses at the initialization point:

$$\theta_i^{(1)} = \theta_{\text{base}} - \eta\nabla\ell_i(\theta_{\text{base}}) \implies \tau_i^{(1)} = -\eta\nabla\ell_i(\theta_{\text{base}}).$$

Therefore for the first epoch:

$$\theta_{\text{TA}}^{(1)} = \theta_{\text{base}} + \alpha\sum_{i\in T}\tau_i^{(1)} = \theta_{\text{base}} - \eta\alpha\sum_{i\in T}\nabla\ell_i(\theta_{\text{base}})$$

Choosing $\alpha\eta$ as learning rate for the loss $\sum_{i\in T}\ell_i$:

$$\theta_{\text{MT}}^{(1)} = \theta_{\text{base}} - \alpha\eta\sum_{i\in T}\nabla\ell_i(\theta_{\text{base}}).$$

So $\theta_{\text{MT}}^{(1)} = \theta_{\text{TA}}^{(1)}$. For $k \geq 2$, notice that

$$\theta_{\text{MT}}^{(k)} = \theta_{\text{base}} - \alpha\eta \sum_{j=0}^{k-1} \nabla \sum_{t \in T} \ell_t(\theta_{\text{MT}}^{(j)}). \tag{E.6}$$

Now, using Lemma E.0.1, we get:

$$\begin{aligned}
&-\alpha\eta \sum_{j=0}^{k-1} \nabla \sum_{t \in T} \ell_t(\theta_{\text{MT}}^{(j)}) + \eta p_i^{k-1}(\eta^0, \ldots, \eta_{\text{MT}}^{k-1}) \\
&\overset{(1)}{=} -\alpha\eta \sum_{j=0}^{k-1} \nabla \sum_{t \in T} \ell_t(\theta_{\text{MT}}^{(j)}) + \eta \sum_{j=0}^{k-1} r_i(\theta_{\text{MT}}^{(j)}) \\
&\overset{(2)}{=} -\alpha\eta \sum_{j=0}^{k-1} \nabla \sum_{t \in T} \ell_t(\theta_{\text{MT}}^{(j)}) + \eta \sum_{j=0}^{k-1} \alpha \sum_{t \in T} \nabla \ell_t(\theta_{\text{MT}}^{(j)}) - \nabla \ell_i(\theta_{\text{MT}}^{(j)}) \\
&\overset{(3)}{=} -\eta \sum_{j=0}^{k-1} \nabla \ell_i(\theta_{\text{MT}}^{(j)}) \,.
\end{aligned}$$

where in (1) we used the definition of $p_i^{(k)}$ E.5 and in (2) we used the definition of $r_t(\theta)$ E.4. In (3) the first two terms cancel out and we are left with the last term. For Lemma E.0.1 we have that:

$$\begin{aligned}
\theta_i^{(m+1)} &= \theta_{\text{MT}}^{(m+1)} + \eta p_i^m(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(m)}) - \eta^2 s_i^{m-1}(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(m-1)}) + O(\eta^3) \\
&\overset{(1)}{=} \theta_{\text{base}} - \alpha\eta \sum_{j=0}^{m} \nabla \sum_{t \in T} \ell_i(\theta_{\text{MT}}^{(j)}) + \eta p_i^m(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(m)}) \\
&\quad - \eta^2 s_i^{m-1}(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(m-1)}) + O(\eta^3) \\
&= \theta_{\text{base}} - \eta \sum_{j=0}^{m} \nabla \ell_i(\theta_{\text{MT}}^{(j)}) - \eta^2 s_i^{m-1}(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(m-1)}) + O(\eta^3)
\end{aligned}$$

where in (1) we used Equation (E.6) and then the reult we just derived. By simply rearraging the terms we can write the task vectors as follows:

$$\begin{aligned}
\tau_i^{(k)} &= \theta_i^{(k)} - \theta_{\text{base}} \\
&= -\eta \sum_{j=0}^{k-1} \nabla \ell_i(\theta_{\text{MT}}^{(j)}) - \eta^2 s_i^{k-2}(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(k-2)}) + O(\eta^3)
\end{aligned}$$

Consequently the model obtained with TA is

$$\begin{aligned}
\theta_{\text{TA}}^{(k)} &= \theta_{\text{base}} + \alpha \sum_{i \in T} \tau_i^{(k)} \\
&\overset{(1)}{=} \theta_{\text{base}} - \eta\alpha \sum_{j=0}^{k-1} \sum_{i \in T} \nabla \ell_i(\theta_{\text{MT}}^{(j)}) - \alpha \sum_{i \in T} \eta^2 s_i^{k-2}(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(k-2)}) + O(\eta^3) \\
&= \theta_{\text{MT}}^{(k)} - \alpha \sum_{i \in T} \eta^2 s_i^{k-2}(\theta_{\text{base}}, \ldots, \theta_{\text{MT}}^{(k-2)}) + O(\eta^3) \,.
\end{aligned}$$

□

References

[1] Hervé Abdi. "The Eigen-Decomposition : Eigenvalues and Eigenvectors". In: 2006. URL: https://api.semanticscholar.org/CorpusID:14829978.

[2] Zahra Rahimi Afzal, Tara Esmaeilbeig, Mojtaba Soltanalian, and Mesrob I Ohannessian. "Linearization explains fine-tuning in large language models". In: *Advances in Neural Information Processing Systems 32* (2026).

[3] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. "On exact computation with an infinitely wide neural net". In: *Advances in neural information processing systems* 32 (2019).

[4] Antonio Auffinger, Gérard Ben Arous, and Jiří Černỳ. "Random matrices and complexity of spin glasses". In: *Communications on Pure and Applied Mathematics* 66.2 (2013), pp. 165–201.

[5] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. "The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies. arXiv e-prints, page". In: *arXiv preprint arXiv:1906.00425* (2019).

[6] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic differentiation in machine learning: a survey". In: *Journal of machine learning research* 18.153 (2018), pp. 1–43.

[7] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. "The loss surfaces of multilayer networks". In: *Artificial intelligence and statistics*. PMLR. 2015, pp. 192–204.

[8] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. "Essentially no barriers in neural network energy landscape". In: *International conference on Machine Learning*. PMLR. 2018, pp. 1309–1318.

[9] Mete Erdogan. *Tangent Space Fine-Tuning for Directional Preference Alignment in Large Language Models*. 2026. arXiv: 2602.01128 [cs.LG]. URL: https://arxiv.org/abs/2602.01128.

[10] Giorgia Franchini, Federica Porta, and Simone Rebegoldi. *Ottimizzazione numerica per l'intelligenza artificiale*. 2024.

[11] Jonathan Frankle and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". In: *International Conference on Learning Representations* (2018).

[12] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. "Linear mode connectivity and the lottery ticket hypothesis". In: *International conference on Machine Learning*. PMLR. 2020, pp. 3259–3269.

[13] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. “Loss surfaces, mode connectivity, and fast ensembling of dnns”. In: *Advances in neural information processing systems* 31 (2018).

[14] Carlos Garrido-Munoz, Aniello Panariello, Silvia Cascianelli, Angelo Porrello, Simone Calderara, Jorge Calvo-Zaragoza, and Rita Cucchiara. “Zero-Shot Synthetic-to-Real Handwritten Text Recognition via Task Analogies”. In: *Computer Vision and Pattern Recognition* (2026).

[15] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. “Reproducing Kernel Hilbert Space, Mercer’s Theorem, Eigenfunctions, Nyström Method, and Use of Kernels in Machine Learning: Tutorial and Survey”. In: *arXiv preprint arXiv:2106.08443* (2021).

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[17] Sepp Hochreiter and Jürgen Schmidhuber. “Flat minima”. In: *Neural computation* 9.1 (1997), pp. 1–42.

[18] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. “Editing models with task arithmetic”. In: *Proceedings of the 11th International Conference on Learning Representations* (2022).

[19] Benoit Jacob, S Kligys, Bo Chen, M Zhu, M Tang, A Howard, H Adam, and Dmitry Kalenichenko. “Quantization and training of neural networks for efficient integer-arithmetic-only inference. arXiv”. In: *Computer Vision and Pattern Recognition* (2017).

[20] Arthur Jacot, Franck Gabriel, and Clément Hongler. “Neural tangent kernel: Convergence and generalization in neural networks”. In: *Advances in neural information processing systems* 31 (2018).

[21] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. “Similarity of neural network representations revisited”. In: *International conference on Machine Learning*. PMlR. 2019, pp. 3519–3529.

[22] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. “Deep neural networks as gaussian processes”. In: *International Conference on Learning Representations* (2017).

[23] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. “Visualizing the loss landscape of neural nets”. In: *Advances in neural information processing systems* 31 (2018).

[24] James Lucas, Juhan Bae, Michael R Zhang, Stanislav Fort, Richard Zemel, and Roger Grosse. “Analyzing monotonic linear interpolation in neural network loss landscapes”. In: *Proceedings of the 38th International Conference on Machine Learning* (2021).

[25] Gianluca Mancusi, Mattia Bernardi, Aniello Panariello, Angelo Porrello, Rita Cucchiara, and Simone Calderara. “Is multiple object tracking a matter of specialization?” In: *Advances in Neural Information Processing Systems*. 2024.

[26] Laura Meneghetti. *Reproducing Kernel Hilbert Spaces*. 2017.

[27] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. “What is being transferred in transfer learning?” In: *Advances in neural information processing systems* 33 (2020), pp. 512–523.

[28] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. “Task arithmetic in the tangent space: Improved editing of pre-trained models”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 66727–66754.

[29] Aniello Panariello, Emanuele Frascaroli, Pietro Buzzega, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. “Modular Embedding Recomposition for Incremental Learning”. In: *British Machine Vision Conference*. 2025.

[30] Aniello Panariello, Daniel Marczak, Simone Magistri, Angelo Porrello, Bartlomiej Twardowski, Andrew Bagdanov, Simone Calderara, and Joost van de Weijer. “Accurate and Efficient Low-Rank Model Merging in Core Space”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen. Vol. 38. Curran Associates, Inc., 2025, pp. 61793–61825.

[31] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. “Moment matching for multi-source domain adaptation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1406–1415.

[32] Angelo Porrello, Pietro Buzzega, Felix Dangel, Thomas Sommariva, Riccardo Salami, Lorenzo Bonicelli, and Simone Calderara. “Dataless weight disentanglement in task arithmetic via kronecker-factored approximate curvature”. In: *International Conference on Learning Representations* (2026).

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. “Learning transferable visual models from natural language supervision”. In: *International conference on Machine Learning*. PmLR. 2021, pp. 8748–8763.

[34] Filippo Rinaldi, Aniello Panariello, Giacomo Salici, Fengyuan Liu, Marco Ciccone, Angelo Porrello, and Simone Calderara. “Gradient-Sign Masking for Task Vector Transport Across Pre-Trained Models”. In: *International Conference on Learning Representations* (2025).

[35] Filippo Rinaldi, Aniello Panariello, Giacomo Salici, Angelo Porrello, and Simone Calderara. “Transporting task vectors across different architectures without training”. In: *International Conference on Machine Learning* (2026).

[36] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. "Empirical analysis of the hessian of over-parametrized neural networks". In: *International Conference on Learning Representations workshop* (2017).

[37] Daniele Solombrino, Antonio Andrea Gargiulo, Adrian Robert Minut, Luca Zhou, Alessandro Zirilli, and Emanuele Rodola. *Zero-Shot Quantization via Weight-Space Arithmetic*. 2026. arXiv: 2604.03420 [cs.CV]. URL: https://arxiv.org/abs/2604.03420.

[38] Thomas Sommariva, Francesca Morandi, Simone Calderara, and Angelo Porrello. "Distilling Linearized Behavior for Effective Task Arithmetic". In: *International Conference on Machine Learning* (2026).

[39] Hsuan Su, Hua Farn, Fan-Yun Sun, Shang-Tse Chen, and Hung yi Lee. *Task Arithmetic can Mitigate Synthetic-to-Real Gap in Automatic Speech Recognition*. 2024. arXiv: 2406.02925 [eess.AS]. URL: https://arxiv.org/abs/2406.02925.

[40] Karl Weiss, Taghi Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big Data* 3 (May 2016). DOI: 10.1186/s40537-016-0043-6.

[41] Lilian Weng. "Some Math behind Neural Tangent Kernel". In: *Lil'Log* (2022). URL: https://lilianweng.github.io/posts/2022-09-08-ntk/.

[42] 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. 2022 IEEE". In: *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 7949–7961.

[43] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. *TIES-Merging: Resolving Interference When Merging Models*. 2023. arXiv: 2306.01708 [cs.LG]. URL: https://arxiv.org/abs/2306.01708.

[44] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. *AdaMerging: Adaptive Model Merging for Multi-Task Learning*. 2024. arXiv: 2310.02575 [cs.LG]. URL: https://arxiv.org/abs/2310.02575.

[45] Kotaro Yoshida, Yuji Naraki, Takafumi Horie, Ryosuke Yamaki, Ryotaro Shimizu, Yuki Saito, Julian McAuley, and Hiroki Naganuma. "Mastering Task Arithmetic: τJp as a Key Indicator for Weight Disentanglement". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=1VwWi6zbxs.

[46] Jim Zhao, Sidak Pal Singh, and Aurelien Lucchi. "Theoretical characterisation of the gauss newton conditioning in neural networks". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 114965–115000.

[47] Luca Zhou, Daniele Solombrino, Donato Crisostomi, Maria Sofia Bucarelli, Giuseppe Alessio D'Inverno, Fabrizio Silvestri, and Emanuele Rodola. "On Task Vectors and Gradients". In: *CoRR* abs/2508.16082 (2025). URL: https://doi.org/10.48550/arXiv.2508.16082.