Update Your Transformer to the Latest Release: Re-Basin of Task Vectors

Filippo Rinaldi [*1] **Giacomo Capitani** [*1] **Lorenzo Bonicelli** [1] **Donato Crisostomi** [2] **Federico Bolelli** [1]
Elisa Ficarra [1] **Emanuele Rodolà** [2] **Simone Calderara** [1] **Angelo Porrello** [1]

Abstract

Foundation models serve as the backbone for numerous specialized models developed through fine-tuning. However, when the underlying pre-trained model is updated or retrained (e.g., on larger and more curated datasets), the fine-tuned model becomes obsolete, losing its utility and requiring retraining. This raises the question: is it possible to transfer fine-tuning to a new release of the model? In this work, we investigate how to transfer fine-tuning to a new checkpoint without having to re-train, in a data-free manner. To do so, we draw principles from model re-basin and provide a recipe based on weight permutations to re-base the modifications made to the original base model, often called task vector. In particular, our approach tailors model re-basin for Transformer models, taking into account the challenges of residual connections and multi-head attention layers. Specifically, we propose a two-level method rooted in spectral theory, initially permuting the attention heads and subsequently adjusting parameters within select pairs of heads. Through extensive experiments on visual and textual tasks, we achieve the seamless transfer of fine-tuned knowledge to new pre-trained backbones without relying on a single training step or datapoint. Code is available at https://github.com/aimagelab/TransFusion.

1. Introduction

Recently, there has been a notable shift among researchers and practitioners towards fine-tuning pre-trained models, rather than building them from scratch. This method leverages backbones developed on large-scale datasets, considerably decreasing the amount of data and training time needed to tailor models for specific downstream tasks. For this reason, pre-trained backbones such as OpenAI's CLIP (Radford et al., 2021) are being extensively utilized as base foundation models. As a result, the corresponding fine-tuned versions play a crucial role in numerous real-world applications like medical imaging (Lu et al., 2024) and satellite image analysis (Mall et al., 2024).

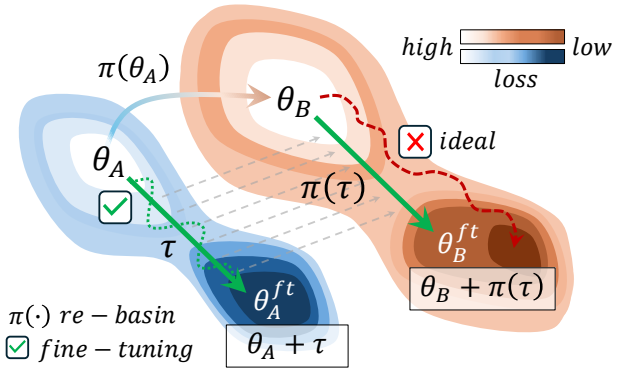

Figure 1: Transporting task vector τ from a fine-tuned base model $\theta_A^{ft} = \theta_A + \tau$ to a new release θ_B.

However, while these pre-trained backbones are widely adopted, their evolution poses new challenges, with tech companies and academic institutions frequently releasing updated checkpoints. Often, these updates do not modify the underlying architecture but simply consist of new weights, trained on increasingly large datasets compared to their predecessors (Ilharco et al., 2021). Moreover, the additional training data may be more curated or specifically tailored to specialized domains, boosting their zero-shot capabilities considerably.

To take advantage of newly released checkpoints, the typical approach is to retrain them on the downstream task. This means fine-tuning the new checkpoint on the same data already used to adapt the original model. Besides the considerable costs associated with re-training the new model, this strategy is also unviable in certain scenarios. Indeed, the data for the downstream task might be no longer available due to compliance with privacy or storage constraints.

[*]Equal contribution [1]AImageLab, University of Modena and Reggio Emilia, Italy. [2]Sapienza, University of Rome, Italy. Correspondence to: Filippo Rinaldi <filippo.rinaldi@unimore.it>.

Proceedings of the 42^{st} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

This raises an important question: can we **re-use** the fine-tuning that has already been performed on the newly released model? Precisely, the overall aim of this paper is to investigate whether we can *transport* the previous fine-tuning, in a training-free manner. To understand the idea of *transport*, we consider the weights of the original base model as θ_A, and their fine-tuning as $\theta_A^{ft} = \theta_A + \tau$. The task vector (Ilharco et al., 2023; Ortiz-Jimenez et al., 2024) $\tau = \theta_A^{ft} - \theta_A$ represents a direction from θ_A that embodies all adjustments made during the fine-tuning process. Hence, our goal is to find a procedure $\pi(\cdot)$ that can transport the task vector τ into an appealing basin of the newly released model θ_B (see Fig. 1). In this approach, the procedure $\pi(\cdot)$ must be designed to ensure that the modified weights $\theta_B^{ft} = \theta_B + \pi(\tau)$ achieve low loss on the downstream task.

When designing the transportation function $\pi(\cdot)$, the ideal approach should be data-free and training-free to meet the concerns above. Nevertheless, if the two base models θ_A and θ_B differ significantly (due to varying initialization, training strategies, or datasets), the knowledge acquired during fine-tuning of θ_A may not transfer to θ_B with a mere addition of the original task vector (*i.e.*, $\theta_B^{ft} = \theta_B + \tau$). To bridge the gap in representation spaces and facilitate the transfer, intuitively, we have to make the two base models "compatible", such that they "speak the same language". To address this challenge, we could *rebase* one of the two models (for instance, θ_A), such that any linear interpolation between the weights of the edited θ'_A and θ_B yields an intermediate model that performs comparably to both θ_A and θ_B. This indicates that the models are now aligned and thus share a common low-loss basin. Notably, this concept of re-basin models shares similarities with the approach described in (Ainsworth et al., 2023), where alignment is achieved by finding optimal permutations of the rows in the weight matrices. In this paper, we build upon this idea and explore its application in the context of fine-tuning, with a task vector being permuted and finally applied to θ_B.

While model re-basin presents an appealing framework, it currently faces several technical hindrances. To date, successful applications of model re-basin have been limited to Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) (Ainsworth et al., 2023). Unfortunately, the application of model re-basin to multi-head attention layers, despite their widespread use in Transformer-based architectures, has been largely overlooked, with a few very recent attempts based on Optimal Transport (see Sec. 5). However, these recent methods do not guarantee **functional equivalence** between the permuted model and the original, unpermuted model. The primary obstacle lies in managing the weights associated with multiple attention heads. As we discuss in Sec. 3, indeed, to apply standard permutation-based approaches (Ainsworth et al., 2023; Singh & Jaggi, 2020; Imfeld et al., 2024), the heads must be concatenated and treated as a single unified projection. This way, after applying permutations, each head of the edited model may incorporate rows from different original heads — an issue we refer to as *head contamination*. This is problematic because, without preserving the logical separation of heads during permutation, it becomes impossible to invert the permutation process and recover the original, unpermuted output of the attention block. Furthermore, existing methods struggle when two addends in the computational graph rely on distinct permutation matrices, a situation common in residual connections such as $h' = h + f(h)$. Differently, we avoid averaging the respective permutation matrices, thereby preserving their discrete nature.

To address these issues, we propose a structured two-level approach for effective re-basin of Transformer-based models, called TransFusion. To avoid *head contamination*, TransFusion firstly seeks optimal mappings between pairs of heads (*inter-head* permutations); subsequently, we restrict permutations to only the rows within these coupled heads (*intra-head* permutations).We mathematically prove that this two-level permutation strategy prevents head contamination and preserves **functional equivalence** between the original and permuted models. Notably, the *inter-head* permutations are optimized leveraging a distance metric that is *invariant* to permutations of the rows and columns within the heads. Such a metric is founded on spectral theory (Jovanović & Stanić, 2012) and employs the singular values of the weight matrices, which are unaffected by orthogonal transformations like those induced by permutations.

We show that transporting task vectors allows the transfer of knowledge into a new checkpoint in a data-free manner. In practice, this means we can improve the zero-shot performance of the new version on the downstream task. We also demonstrate that the transport retains the generalization capabilities on a support set — a crucial factor to justify updating the base model to the new release. To summarize:

Motivation. Given a fine-tuned Transformer model $f(\cdot, \theta_A^{ft} = \theta_A + \tau)$ we aim to transfer the task vector τ into a new release $f(\cdot, \theta_B)$ in a data-free manner.

Contributions. *i)* We introduce a novel data-free method for aligning Transformers. *ii)* We propose a permutation-invariant spectral measure to manage multi-head attention layers. *iii)* We reveal the practicality of applying a task vector from one base model θ_A to a new release θ_B.

2. Background

Given two models with weights θ_A and θ_B, **model re-basin** (Ainsworth et al., 2023) investigates how to permute the units of one model to facilitate the alignment of two mod-

els. The two models are then merged in the weight space, resulting in an interpolated model that achieves performance comparable to that of the two original ones.

Following the notation of (Ainsworth et al., 2023), **re-basin** is defined as any operation defined on the weights of two models θ_A and θ_B that maps one of the two models onto the local loss region (*basin*) of the other one. To assess the effectiveness of re-basin, one common approach is to check the property of linear mode connectivity (Frankle et al., 2020; Entezari et al., 2022) between the permuted model and the other, reference model. Informally, this involves checking if the model weights laying on the linear path connecting θ_A and θ_B also result in a low loss value.

To reach such a property, existing model re-basin techniques leverage the permutation symmetries inherent in neural networks (Entezari et al., 2022). These symmetries allow the swapping of the units within a layer without changing the functionality of the network. To show that, we consider the activation of the ℓ-th layer of an MLP:

$$z_{\ell+1} = \sigma(W_\ell z_\ell + b_\ell), \quad z_0 = x, \tag{1}$$

where W_ℓ and b_ℓ are the weight matrix and bias vector and σ denotes an element-wise activation function. In this case, the following relation holds for any permutation matrix P:

$$\begin{aligned} z_{\ell+1} &= P^\top P z_{\ell+1} = P^\top P \sigma(W_\ell z_\ell + b_\ell), & (2)\\ &= P^\top \sigma(P W_\ell z_\ell + P b_\ell), \quad \text{where } P \in S_d, & (3) \end{aligned}$$

with S_d denoting the set of $d \times d$ permutation matrices. Thanks to this relation, we can essentially permute the weights and biases of a layer using a matrix P. Therefore, when we apply the permutation P to the parameters of a layer, the resulting output undergoes the same permutation. However, to ensure that the transformed model remains functionally equivalent to the original, the next layer must process the output in its original, unpermuted form. This can be achieved equivalently by permuting the weights of the subsequent layer using the inverse permutation $P^\top$. Accordingly, we define a transformed set of weights θ' as:

$$W'_\ell = P W_\ell, \quad b'_\ell = P b_\ell, \quad W'_{\ell+1} = W_{\ell+1} P^\top. \tag{4}$$

Git Re-Basin. Ainsworth et al. (2023) exploit Eq. (2) to induce weight alignment between θ_A and θ_B. Formally, we consider the ℓ-th feed-forward layer, with weight matrices $W_\ell^{(A)}$ and $W_\ell^{(B)}$ for θ_A and θ_B respectively. Given that each row of $W_\ell^{(A)}$ and $W_\ell^{(B)}$ represents a distinct feature, if $[W_\ell^{(A)}]_{i,:} \approx [W_\ell^{(B)}]_{j,:}$, then it makes sense to associate the units i and j. Therefore, we could formalize the alignment as finding the permutation matrix that maximizes the dot product between $P_\ell W_\ell^{(A)}$ and $W_\ell^{(B)}$. However, to preserve functional equivalence, we have to account for the term $P_{\ell-1}^\top$ related to the permutation of the previous layer —see Eq. (4). This results in a global optimization across layers:

$$\begin{aligned} \underset{\pi=\{P_\ell\}_1^L}{\operatorname{argmax}} \left\langle W_1^{(B)}, P_1 W_1^{(A)} \right\rangle + \left\langle W_2^{(B)}, P_2 W_2^{(A)} P_1^\top \right\rangle + \\ + \cdots + \left\langle W_L^{(B)}, W_L^{(A)} P_{L-1}^\top \right\rangle. \end{aligned} \tag{5}$$

where $\langle A, B \rangle = \sum_{i,j} A_{i,j} B_{i,j}$ is the inner product between real-valued matrices. As discussed in (Ainsworth et al., 2023), the optimization problem described in Eq. (5) corresponds to the Symmetric Orthogonal Bilinear Assignment Problem (SOBLAP), which is unfortunately NP-hard. Its relaxation re-casts it as a series of Linear Assignment Problems (LAPs), focusing on one permutation $\boldsymbol{P_\ell}$ at a time while keeping the others fixed. In formal terms:

$$\underset{P_\ell}{\operatorname{argmax}} \left\langle W_\ell^{(B)}, P_\ell W_\ell^{(A)} P_{\ell-1}^\top \right\rangle + \left\langle W_{\ell+1}^{(B)}, P_{\ell+1} W_{\ell+1}^{(A)} P_\ell{}^\top \right\rangle. \tag{6}$$

Notably, each LAP can be solved with efficient, polynomial-time methods like the Hungarian algorithm (Jonker & Volgenant, 1988). The outcome is a set of permutation matrices $\pi = \{P_\ell\}_1^L$, which, when applied to model θ_A, result in a new model $\theta_{A'} = \pi(\theta_A)$. Notably, this model is functionally equivalent and, theoretically, it resides in the low-loss basin of θ_B. However, as optimizing a series of LAPs is a coarse approximation of the SOBLAP, there are no strong guarantees regarding the optimality of the solution.

3. TransFusion: An Approach For Re-basin Transformer Models

Objective. Our approach, named TransFusion, is designed to transfer task-specific knowledge between transformer-based models that have undergone different pre-training. Specifically, it starts with an initial weight set θ_A and a task vector $\tau = \theta_A^{ft} - \theta_A$, derived after fine-tuning on a downstream task. The goal is to adapt τ to a new parameter configuration θ_B. This process aims to preserve the inherent properties of θ_B — for example, its superior zero-shot capabilities compared to θ_A — and to integrate specialized knowledge carried out by τ for the downstream task. Finally, we aim to enable model transfer in a data-free manner.

Weight Matching. To achieve these objectives, we start by aligning the weights of θ_A with those of θ_B. This is accomplished with a novel *data-free weight matching strategy* tailored for Transformer architectures. The procedure is deeply discussed in Sec. 3.1. Briefly, we tackle various shortcomings of existing methods and handle two building blocks of attention-based networks, namely the residual paths and the multi-head attention mechanism. To manage the latter, we introduce a novel two-step process that employs a permutation-invariant spectral metric to match pairs

of heads within the same layer of θ_A and θ_B. Subsequently, we permute features within the matched heads to optimize weight alignment, as detailed in Sec. 2.

Transport. We end up with a functionally equivalent model $\theta'_A = \pi(\theta_A)$, where $\pi(\cdot)$ yields a permutation of every layer in θ_A. Afterwards, $\pi(\cdot)$ is used to transport the task vector $\tau = \theta_{ft} - \theta_A$ into the low-loss basin of θ_B (see Sec. 3.2).

3.1. Attention Alignment for Transformer Models

A Transformer-based block consists of a multi-head attention layer and an MLP block, connected through residual connections. Considering the MLP, this builds upon standard linear projections, which we treat as discussed in Sec. 2. Instead, we adopt a novel, tailored approach for multi-head attention layers addressing a common pitfall. Considering multiple heads, current methods view their projections as a whole linear layer, thereby joining the corresponding weight matrices before applying permutations. However, such an approach does not reflect the organization of attention in distinct, parallel heads. For example, this can result in artifacts, where units from separate heads in the original model are mixed together — an issue we call *head contamination*. This compromises the structural separability of attention heads and precludes the preservation of *functional equivalence*, that is, the ability to permute and subsequently unpermute the weight matrices while yielding identical model outputs.

In the following, we present our proposal against head contamination (see **Step 1** and **2**) and a practical approach to handle residual connections (**Step 3**). The complete methodology is outlined in Algorithm 1.

Step 1: Inter-Head Alignment
Consider the q(uery), k(ey), and v(alue) projection matrices W_q, W_k, and $W_v \in \mathbb{R}^{d_m \times d_m}$ — with d_m denoting the total embedding dimension of the attention module. We partition each matrix into $H = \#\text{heads}$ matrices (one for each head) of shape $d_k \times d_m$, where $d_k = \frac{d_m}{H}$. This results in a tensor $\tilde{W}_q = \text{split}(W_q, H) \in \mathbb{R}^{H \times d_k \times d_m}$ for the query projection matrix W_q. The same operation is applied for W_k and W_v to obtain $\tilde{W}_k$ and $\tilde{W}_v$.

The first step involves defining a distance metric between pairs of heads, such that we can identify and execute the optimal swap between heads in θ_A and θ_B (see Fig. 2). We employ a distance metric that is **invariant** to permutations of rows and columns within the H sub-matrices in $\tilde{W}_q$, $\tilde{W}_k$, and $\tilde{W}_v$. In this respect, one might question why invariance is crucial for comparisons between different heads. We note that the initial, natural order of units does not necessarily correspond to the optimal alignment that could be achieved. Consequently, the metric used in this initial phase must be insensitive to the specific ordering of head features, thereby ensuring an agnostic comparison of the heads.

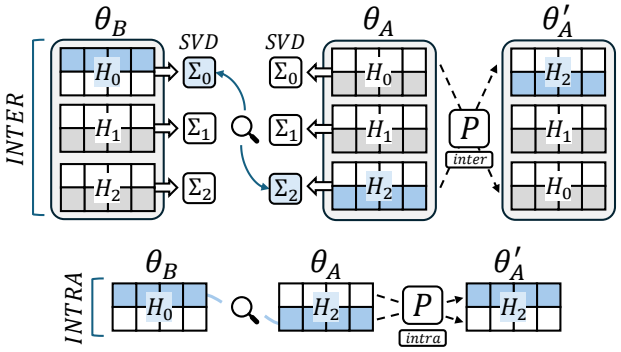

Figure 2: Inter- (Step 1) and intra-head alignment (Step 2).

To achieve the permutation-invariance property, we employ a distance based on the **singular values** of the sub-matrices representing the heads. Specifically, given two heads $h_i^B = [\tilde{W}]^B_{i,:,:} \in \mathbb{R}^{d_k \times d_m}$ from model θ_B and $h_j^A = [\tilde{W}]^A_{j,:,:}$ from model θ_A, we compute the distance as:

$$d_{ij} = \|\Sigma_i - \Sigma_j\|, \tag{7}$$

where Σ_i and Σ_j denote the singular values of h_i^B and h_j^A respectively. These can be computed through the Singular Value Decomposition (SVD); in formal terms, considering the i-th head, the SVD decompose its weight $h_i^B = U_i \Sigma_i V_i^T$, where U_i and V_i are orthogonal matrices, and Σ_i is a diagonal matrix containing the singular values of h_i. As demonstrated in Appendix A.1, the Euclidean distance between singular values remains invariant to permutations.

To take into account the distance for query, key and value projections jointly, we construct a distance matrix $D \in \mathbb{R}^{H \times H}$, where each element $D_{ij} = d^q_{ij} + d^k_{ij} + d^v_{ij}$ represents an inter-head alignment cost that is calculated as the sum of the pairwise distances across q, k and v matrices. We hence employ D to find the optimal inter-head permutation:

$$P_{\text{inter\_head}} = \underset{P \in S_H}{\text{argmin}} \sum_{i=1}^{H} D_{i,P[i]}, \tag{8}$$

where $D_{i,P[i]}$ is the distance between the i-th head of model θ_B and the $P[i]$-th (candidate) head of model θ_A. The solution $P_{\text{inter\_head}}$ can be practically determined with the Hungarian algorithm. The corresponding permutation is applied to each $\tilde{W}_q^A$, $\tilde{W}_k^A$, and $\tilde{W}_v^A$, thereby reordering the heads of θ_A to increase alignment with those of θ_B.

Step 2: Intra-Head Alignment
After matching each pair of heads $(h_i^B, h^A_{P[i]})$, we aim to swap individual units between h_i^B and $h^A_{P[i]}$. To do so, as in Git Re-Basin (Ainsworth et al., 2023), we seek for permutations that maximize the inner product between the

Algorithm 1 Weight Matching

Require: $\theta_A = \left\{W_\ell^{(A)}\right\}_{\ell=1}^{L}$ and $\theta_B = \left\{W_\ell^{(B)}\right\}_{\ell=1}^{L}$
Ensure: A permutation $\pi = \{P_1, \ldots, P_{L-1}\}$ of θ_A.

Initialize: $P_1 \leftarrow I, \ldots, P_{L-1} \leftarrow I$
repeat
  for $\ell \in 1, \ldots, L-1$ **do**
    if ℓ is a MHA layer **then**
      $P_\ell \leftarrow$ Multi-Head Attention Alignment (Alg. 2)
    else
      $P_\ell \leftarrow$ Solving LAP as in Eq. (6)
until convergence

corresponding H sub-portions of the projection weights h_i^B and $h_{P[i]}^A$, as follows:

$$P_{\text{intra\_head}} = \{P_{\text{intra\_head}}^{(i)}\}_{i=1}^{H},$$
$$\text{where } P_{\text{intra\_head}}^{(i)} = \underset{P \in S_{d_k}}{\operatorname{argmax}} \left\langle h_i^B, P h_{P[i]}^A \right\rangle. \quad (9)$$

In this formula, the cost considers the dot products across query, key, and value sub-matrices. In the end, the optimization yields H permutation matrices $P_{\text{intra\_head}}^{(i)} \in \mathcal{S}_{d_k}$.

In summary, this two-step process ensures both global inter-head and local intra-head alignment while preserving the structural integrity of the multi-head attention mechanism. The comprehensive procedure for aligning multi-head attention layers, combining these alignment stages, is formalized in Algorithm 2. Specifically, these individual alignment steps are unified into a single composed permutation, denoted as P_{attn}, which is applied directly to the projection matrices of the multi-head attention layer. Crucially, it can be shown that our structured composed permutation preserves functional equivalence: despite reordering and permuting the heads and their internal dimensions, the attention computation remains unchanged (up to a corresponding permutation of its outputs). We formalize this in the following theorem.

Theorem 3.1 (Equivariance of Multi-Head Attention to Structured Permutations). *Let $P_{inter\_head} \in S_H$ be a permutation over the H attention heads, and let $P_{intra\_head} = \{P_{intra\_head}^{(i)}\}_{i=1}^{H}$ be a set of independent permutations acting within each head (of size $d_k = \frac{d_m}{H}$). Then applying the composed block permutation P_{attn} to each of the projection matrices $W_q, W_k, W_v \in \mathbb{R}^{d_m \times d_m}$ is functionally equivalent to permuting the output of the multi-head attention module. The resulting attention output O' satisfies: $O' = OP_{attn}$, where O is the unpermuted output.*

A complete proof of Theorem 3.1 is provided in Appendix A.2.

Algorithm 2 Attention Alignment

Require: Weights $\tilde{W}_q^{(A)} P_{\ell-1}^\top$, $\tilde{W}_k^{(A)} P_{\ell-1}^\top$, $\tilde{W}_v^{(A)} P_{\ell-1}^\top \in \theta_A$ and $\tilde{W}_q^{(B)}, \tilde{W}_k^{(B)}, \tilde{W}_v^{(B)} \in \theta_B$ for multi-head attention projection layer ℓ and previous layer $\ell - 1$.
Ensure: Permutation $P_{\ell\text{-attn}}$ for $\tilde{W}_q^{(A)}$, $\tilde{W}_k^{(A)}$, $\tilde{W}_v^{(A)}$.

Step 1: Inter-Head Alignment
Create spectral distance matrix D (Eq. (7)).
$P_{\text{inter}} \leftarrow$ Solve LAP on D (Eq. (8)).
Step 2: Intra-Head Alignment
for $h = 1$ **to** H head pairs from P_{inter} **do**
  $P_{\text{intra}}^{(h)} \leftarrow$ Solve LAP for head pair h (Eq. (9)).
$P_{\ell\text{-attn}} \leftarrow P_{\text{inter}} \circ \left\{P_{\text{intra}}^{(h)}\right\}_{h=1}^{H}$ ▷ compose permutations

Step 3: Managing of Residual Connections
Each transformer block incorporates two residual connections: the first bypasses the multi-head attention layer, and the second bypasses the feed-forward network:

$$\begin{aligned} \mathbf{z}_{attn} &= W_0\,\text{MHA}(\mathbf{x}), \\ \mathbf{z}_i &= \mathbf{z}_{attn} + \mathbf{x}, \\ \mathbf{z}_f &= W_2 \text{ReLU}(W_1 \mathbf{z}_i), \\ \mathbf{z}_{out} &= \mathbf{z}_f + \mathbf{z}_i, \end{aligned} \quad (10)$$

where $\mathbf{x}$ is the input, W_0 is the weight of the attention mechanism, and W_1 and W_2 those of the feed-forward layer. For simplicity, we omit layer normalization as it can be regarded as a standard linear projection.

In each residual block, the input and the intermediate output are summed to produce the final output. However, we note that there are several sources of potential mismatch between the two addends: intuitively, if the two addends have undergone different permutations, it is reasonable to suspect a potential mismatch in their representations.

To clarify the interaction between permutations in residual blocks, consider Fig. 3, which represents the first residual $\mathbf{z}_i = \mathbf{z}_{attn} + \mathbf{x}$. When the weights of the model are permuted, the input $\mathbf{x}$ comes with its own permutation P_{in}, which has to be accounted using $P_{\text{in}}^\top$. Moreover, the attention projection W_0 adds its own permutation matrix P_{W_0}. This leads to the following relation:

$$\mathbf{z}_i = P_{W_0} \mathbf{z}_{\text{attn}} + P_{\text{in}} \mathbf{x}. \quad (11)$$

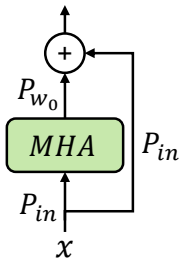

Figure 3: Residuals block and permutations.

When examining the permutations/summations that impact on $\mathbf{z}_i$, there are two main issues: *issue I)* the residual branch lacks transformations that could account for the matrix P_{in};

Table 1: Comparison of permutation-based methods on visual tasks, in terms of task accuracy [↑] and support accuracy [↑].

Method	EUROSAT		DTD		GTSRB		SVHN	
	TASK	SUPP.	TASK	SUPP.	TASK	SUPP.	TASK	SUPP.
θ_B *zero-shot*	49.02	68.73	47.50	68.73	43.42	68.73	45.97	68.73
$\theta_B + \tau$	-7.62	-16.15	-0.15	-0.10	-5.39	-0.70	-22.00	-16.45
$\theta_B + \pi(\tau)$ *(Optimal Transport)*	-14.05	-5.28	-0.53	-1.18	-2.43	-1.30	-12.30	-2.70
$\theta_B + \pi(\tau)$ *(GiT Re-Basin)*	+0.95	-0.48	-0.91	**-0.02**	+0.76	**-0.05**	+0.79	**+0.30**
TRANSFUSION (OURS)	**+4.95**	**-0.06**	**+0.21**	-0.08	**+1.10**	-0.40	**+3.64**	-0.48

issue II) the projection W_0 adds its own permutation P_{W_0} of which the residual branch has no information about.

To maintain coherence between the two addends, they must be transformed under identical permutations. To enforce this consistency, we redefine the identity mapping made by the residual connection. We replace it with a composition, $\mathcal{I}_i = P_{W_0} P_{\text{in}}^\top$, consisting of two permutations — one to address *issue I* and another for *issue II* — as follows:

$$\mathbf{z}_i = P_{W_0}\mathbf{z}_{\text{attn}} + \mathcal{I}_i P_{\text{in}}\mathbf{x} = P_{W_0}\mathbf{z}_{\text{attn}} + P_{W_0}\mathbf{x}, \qquad (12)$$

which highlights how the two addends now share the same permutation. An analogous process applies to the second residual connection yielding $\mathbf{z}_{\text{out}}$ (see Appendix A.3 for the full procedure). As a final technical note, we remark that the permutation matrix P_{W_2} associated to the second residual block in Eq. (10) has to be considered as input permutation for the subsequent layer.

3.2. Transporting Task Vectors from θ_A to θ_B

By applying π to model θ_A, we would have a functionally equivalent model $\theta'_A = \pi(\theta_A)$ with stronger linear-mode connectivity with θ_B compared to the original θ_A. However, to allow knowledge transfer from the fine-tuned model $\theta_A^{ft} = \theta_A + \tau$ to θ_B, we do not apply the permutations directly on θ_A, but rather on the task vector τ, as follows:

$$\text{task vector}: \quad \tau = \theta_A^{ft} - \theta_A, \qquad (13)$$

$$\text{transport}: \quad \tilde{\theta}_B^{ft} = \theta_B + \alpha\pi(\tau), \qquad (14)$$

where α is a non-negative scaling factor (Wortsman et al., 2022b) modulating the influence of $\pi(\tau)$ on θ_B.

By leveraging the concept of transporting task vectors, we have several notable advantages, especially in a scenario with multiple models fine-tuned on distinct tasks from the same base model θ_A. In this scenario, the weight matching process between θ_A and θ_B needs to be conducted only **once**. Indeed, a permutation set π can be established and reused to transfer any number of task vectors. This approach avoids the additional computational costs associated with learning separate transport functions for each transfer. Moreover, transporting multiple task vectors using the same reference model θ_A allows their combination at destination θ_B, which basically means we could still apply model merging (Wortsman et al., 2022a) after re-basin.

3.3. Complexity Analysis

In this subsection, we assess the computational complexity of the proposed weight matching procedure. The key insight is that the method is highly efficient compared to full retraining, and scales polynomially with model size.

Proposition 3.2. *Let L be the number of layers and d_m the embedding dimension of each transformer block. The overall computational complexity of our weight matching procedure is dominated by $O(L\, d_m^3)$. This complexity matches that of Git Re-Basin, making our approach comparably efficient in terms of computational cost.*

The proof is provided in Appendix A.5 and illustrates the per-layer contribution of both MLP and attention blocks.

4. Experiments

This section is structured into three main parts. Initially, we empirically assess the transportation of task vectors, involving extensive experiments across both visual and natural language processing (NLP) tasks (Sec. 4.1). Subsequently, we examine the capability of our methodology to align the weights of two Transformer models while maintaining functional equivalence (Sec. 4.2). Finally, several ablative studies show the impact of our techniques on addressing multi-head attention layers and residual connections (Sec. 4.3).

4.1. TransFusion of Task Vectors

Visual Classification Tasks. As reference architecture, we consider the CLIP ViT-B/16 Vision Transformer (Radford et al., 2021) from Open-CLIP (Cherti et al., 2023). We refer to θ_A as the original pre-training weights and θ_B as those used for the re-basin. We use CommonPool pre-training for θ_A and Datacomp for θ_B, both cited in (Gadre et al., 2024).

Considering the base model θ_A, we fine-tune the corresponding model on several computer vision tasks (Radford et al., 2021; Ilharco et al., 2023). We employ DTD (Cimpoi et al.,

Table 2: Comparison of permutation-based methods on NLP tasks, in terms of task accuracy [↑].

Method	QQP	SST2	RTE	CoLA
θ_B	55.00	50.69	54.51	40.94
$\theta_B + \tau$	-8.29	+0.23	-2.53	-0.77
$\theta_B + \tau$ *(OT)*	-8.31	+5.39	-1.08	-1.25
$\theta_B + \tau$ *(GiT Re-Basin)*	+3.58	+5.73	+2.17	+1.44
TransFusion (Ours)	**+6.50**	**+5.96**	**+3.61**	**+2.49**

2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2011), and SVHN (Netzer et al., 2011) and obtain multiple, independent fine-tuned models like $\theta_A^{ft} = \theta_A + \tau$. Afterwards, we empirically assess the transportation of τ to the new weights θ_B. In this respect, we adopt two metrics to characterize the quality of the transported model $\theta_B + \pi(\tau)$: *i)* the zero-shot performance on the original task (*specialized knowledge*), and *ii)* the zero-shot performance on a support, unseen set to evaluate the preservation of *broader capabilities*. In our experiments, ImageNet-R (Hendrycks et al., 2021) serves as a support dataset.

We report the results in Tab. 1 as drops (-) or gains (+) in accuracy relative to the zero-shot performance of θ_B. As baselines, we provide the results of *vanilla transportation* (no permutations applied on τ) and those of Git Re-Basin (Ainsworth et al., 2023) and Optimal Transport (OT) (Imfeld et al., 2024), two existing methods for model re-basin. Specifically, the comparison with OT is noteworthy since this approach is designed for Transformer models (like ours).

As can be seen, our method enhances zero-shot performance on the downstream tasks and preserves generalization on the support dataset, outperforming existing permutation-based methods. Considering the results of our approach, it is particularly noteworthy that we enhance performance on the downstream task while maintaining generalization, all achieved without the use of any data.

In the experiments shown in Tab. 1, we consistently set the scaling coefficient for the (permuted) task vector as $\alpha = 1$ (see Sec. 3.2). To investigate sensitivity and performance changes while varying α, we kindly refer the reader to Fig. 4 (more datasets are in Appendix A.4). This illustrates the drop/gain in accuracy for $\theta_B + \alpha\tau$ (blue) and our $\theta_B + \alpha\pi(\tau)$ (red). This drop/gain is measured w.r.t. the zero-shot accuracy of θ_B, and α varies within the range $[0.01, 2.0]$. The outcome is that applying the permuted $\pi(\tau)$ to θ_B leads to tangible improvements in the downstream task (top row), especially $\alpha \approx 1$. Moreover, when $\alpha \geq 0.5$, the permuted task vector is considerably more reliable in terms of generalization (higher accuracy on the support set).

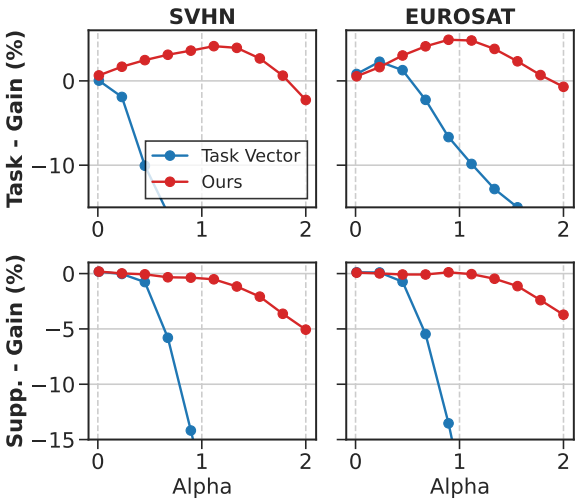

Figure 4: Zero-shot gain/drop relative to θ_B of naive $\theta_B + \alpha\tau$ (blue) and our strategy $\theta_B + \alpha\pi(\tau)$ (red) varying α.

NLP Classification Tasks. Herein, we investigate a different setting that involves closed-vocabulary text classification — specifically, a set of tasks from the GLUE benchmark (Wang et al., 2019). We consider a model $\theta = \{\phi, \omega\}$ composed of a pre-trained Transformer encoder ϕ and a classification head ω. We then evaluate the transport of the learned task vector $\tau_\phi = \phi_A^{ft} - \phi_A$ on a new feature extractor ϕ_B. As access to data of the downstream task is restricted, we are unable to train a new classifier for θ_B: consequently, we re-use the originally fine-tuned classifier, denoted as ω^{ft}. The goal is to evaluate whether transporting the task vector τ_ϕ aligns the representation yielded by $\phi_B + \pi(\tau_\phi)$ with the original, fine-tuned classifier ω^{ft}. In our experiments, we employed two variants of the ViT-B-16 text encoder, pretrained respectively on the `commonpool-l-s1b-b8k` (θ_A) and `datacomp-l-s1b-b8k` (θ_B) (Gadre et al., 2024).

Tab. 2 presents the evaluation for the GLUE benchmark. Unexpectedly, simply applying the classification head from the original feature extractor ϕ_A yields poor performance (see first line of Tab. 2, θ_B). On the other hand, transporting τ_ϕ with Git Re-Basin and Optimal Transport performs reasonably, with good gains on QQP and SST2. Moreover, our approach leads to the highest and more consistent performance gains, highlighting the potential of our framework.

4.2. TransFusion Improves Alignment and Preserves Functional Equivalence

While the previous analyses focus on transferring task vectors, we now delve into the effectiveness of our approach in terms of weight alignment. In detail, we consider two ViT-B/16 models (Dosovitskiy et al., 2021) A and B trained

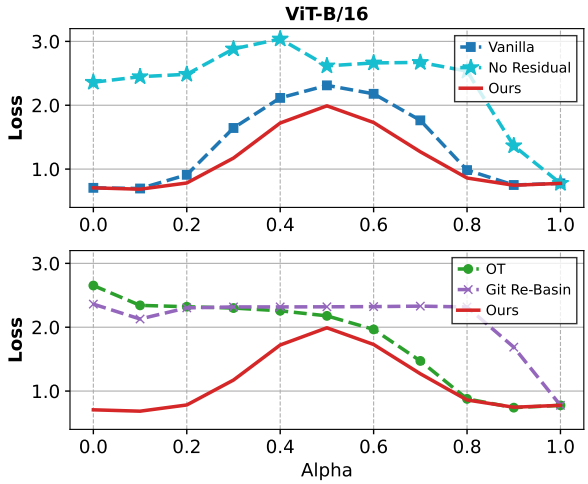

Figure 5: Loss values on CIFAR-10 test set during model interpolation. Top: Our permutation approach vs. vanilla interpolation and no residual variant. Bottom: Comparison with Optimal Transport and Git Re-Basin methods, which fail to preserve functional equivalence as $\alpha \to 0$.

independently on CIFAR-10 (Krizhevsky et al., 2009) from scratch, which means they underwent different initializations and batch orders. After training, we apply our permutation strategy to θ_A and analyze the resulting alignment of $\pi(\theta_A)$ and θ_B in terms of *linear mode connectivity*: following (Ainsworth et al., 2023), we evaluate the loss landscape $\mathcal{L}\left((1-\alpha)\pi\left(\theta_A\right)+\alpha\theta_B\right), \alpha \in [0,1]$ while interpolating between the two models.

As shown by Fig. 5 (top), applying the permutation π yields an interpolated model that exhibits consistent lower loss compared to the vanilla approach that does not permute θ_A ($\pi(\theta_A) = \theta_A$). Moreover, the *no residual* approach underscores the critical role of properly handling residual connections in our method — both during the permutation of θ_A and throughout the interpolation process between models. Similarly, in Fig. 5 (bottom) we assess the loss landscape using π derived from the Optimal Transport (OT) hard alignment method (Imfeld et al., 2024) and Git Re-Basin (Ainsworth et al., 2023). While these works show a more favorable loss landscape than naive interpolation, we observe that the resulting interpolated model struggles and achieves high loss when $\alpha \to 0$, highlighting that both OT and Git Re-Basin do not preserve functional equivalence. We conjecture that such a lack stems from potential weaknesses in effectively permuting layers featuring residual connections and multi-head attention blocks. In virtue of the results achieved by our approach, we can claim it represents the first successful data-free method to interpolate between two Transformer models in weight space, while ensuring the functional equivalence of $\pi(\theta_A)$.

Table 3: TransFusion results by head alignment strategy.

Head Align.	EuroSAT		GTSRB		SVHN	
	Task	Supp.	Task	Supp.	Task	Supp.
θ_B *zero-shot*	49.02	68.73	43.42	68.73	45.97	68.73
Brute Force	+1.32	-0.21	+0.60	-0.46	+3.39	-0.40
No Att-Align	+2.22	-0.47	+0.71	**+0.05**	+0.24	**-0.08**
Ours (Full)	**+4.95**	**-0.06**	**+1.10**	-0.40	**+3.64**	-0.48

4.3. Ablative Analysis

On the Strategy to Manage Multiple Heads. We herein explore the significance of an appropriate policy for permuting the projection layers within multi-head attention mechanisms. Specifically, we present the outcomes of transferring τ with varying strategies to permute attention projection layers. As potential alternatives, we firstly consider **brute force alignment**, which considers all possible head pair combinations within each attention layer. Then, for each candidate pair, the *intra-head* alignment cost is computed by optimizing the objective in Eq. (9). The final permutation is then derived with the Hungarian algorithm, which selects pairs with the highest intra-alignment scores.

After, we compare our approach with one that pairs heads in A and B according to their natural order, thereby avoiding *head contamination* by design. Nevertheless, the preservation of original head ordering comes at the cost of ignoring functional mismatches between attention units. We refer to this further baseline as **no attention alignment**.

The results of these experiments are detailed in Tab. 3. Our attention-alignment strategy achieves superior performance on the downstream task ($\theta_B + \pi(\tau)$) compared to alternative approaches, while maintaining comparable zero-shot capabilities. The comparison with the brute force approach underscores the effectiveness of our permutation-invariant costs in modeling inter-head relationships, demonstrating superior performance over a brute force alignment that optimizes for the best match within each candidate pair of heads. Furthermore, our results suggest that preserving the original head ordering (as in the no-alignment strategy) yields better performance than brute-force inter-head matching for two of the three experimental tasks. This underscores the menace of head contamination and the importance of structure-aware alignment methodologies over indiscriminate similarity maximization.

Few Shot Fine-tuning. There are practical scenarios in which retaining data is infeasible. If such constraints are not present, our proposed method can be effectively combined with fine-tuning. To illustrate this, we follow (Zhang et al., 2024) and start with a small subset consisting of 10 shots per class, learning a scaling coefficient per layer, denoted

Table 4: Fine-tuning results with permuted task vectors.

Method	EuroSAT	DTD	GTSRB	SVHN
θ_B *zero-shot*	49.02	47.50	43.42	45.97
$\theta_B + \alpha\tau$	+7.93	-1.44	+4.70	-15.98
$\theta_B + \alpha\pi(\tau)$	**+10.00**	**+1.21**	**+6.80**	**+10.52**

as $\alpha = [\alpha_1, \ldots, \alpha_{|L|}]$. The results in Tab. 4 clearly indicate a substantial improvement when fine-tuning a model that has undergone re-basin using our approach, represented as $\theta_B + \alpha\pi(\tau)$. In contrast, fine-tuning directly from $\theta_B + \alpha\tau$, without permutation, yields inferior outcomes. This emphasizes that re-basing and fine-tuning should not be considered mutually exclusive but complementary strategies.

5. Related Work

Mode Connectivity occurs when paths of nearly constant loss connect different solutions within the loss landscape of neural networks (NNs) (Garipov et al., 2018; Freeman & Bruna, 2017; Garipov et al., 2018; Draxler et al., 2018). When such paths are linear, we refer to *linear* mode connectivity (LMC) (Frankle et al., 2020). Entezari et al. (2022) conjecture that solutions found by Stochastic Gradient Descent (SGD) can be linearly connected when accounting for permutation symmetries. Motivated by this, several works first align the models into a shared optimization space by permuting their neurons, and then merge them through a simple average (Ainsworth et al., 2023; Jordan et al., 2023; Stoica et al., 2024; Peña et al., 2023; Crisostomi et al., 2024; Navon et al., 2023; Singh & Jaggi, 2020). Most relevant to our work, Imfeld et al. (2024) applies optimal transport to align activations in transformer-based networks. However, unlike the latter, our method accounts for the logical division of multi-head attention projections, preserving the functional equivalence of the permuted models.

Weight Interpolation and Task Arithmetic. Emerging research reveals that the output of NNs can be manipulated through algebraic operations in weight space (Ilharco et al., 2022; Wortsman et al., 2022a). Central to this paradigm are task vectors τ (Ilharco et al., 2023), which encode task-specific knowledge and exhibit compositional properties: combining vectors through addition enables multi-task generalization while their negation can suppress learned behaviors without significantly impacting unrelated tasks.

Beyond arithmetic, weight interpolations further unlock unexpected capabilities: blending fine-tuned and pre-trained weights often yields single-task performance superior to standalone fine-tuning (Frankle et al., 2020; Izmailov et al., 2018; Matena & Raffel, 2022; Ramé et al., 2023; Rame et al., 2022; Wortsman et al., 2022b), suggesting a reconciliation of specialized adaptation with generalization capabilities. Multi-task merging via parameter averaging (Ilharco et al., 2022; 2023; Wortsman et al., 2022a; Yadav et al., 2024) not only circumvents catastrophic forgetting (French, 1999; McCloskey & Cohen, 1989; Porrello et al., 2025) but synthesizes models that retain diverse expertise, even serving as superior starting points for future adaptation (Choshen et al., 2022). The benefits of weight ensembles and interpolations extend beyond just fine-tuned models; they also apply to models that are trained from scratch. Techniques such as those proposed by (Ainsworth et al., 2023; Singh & Jaggi, 2020), leverage permutation symmetries to facilitate coherent interpolation between models trained in different ways. Collectively, these findings position weight-space manipulation as a scalable toolkit for resource-efficient model engineering, where arithmetic and interpolation replace brute-force retraining.

6. Discussion and Conclusions

For TransFusion to succeed, the source expert must perform strongly on the target task. This insight directly explains the diminished results on DTD in Tab. 1: while our fine-tuning achieves performance well above 95% on all other datasets, DTD reaches only 75% accuracy. We attribute this degraded task vector quality to DTD's inherently challenging characteristics — featuring merely 40 examples per class on average, compared to roughly 1,000 examples per class in our other datasets. Although our proposal delivers substantial gains without requiring access to data from the original task, we acknowledge significant room for improvement. We believe most of the lost performance could be recovered through two complementary directions: incorporating a modest amount of additional data, or employing more sophisticated matching metrics for permutation discovery that better reflect activation distributions rather than relying on simple dot product similarity for linear layers. Such extensions represent promising avenues for future investigation.

Acknowledgments

This work was supported by the Italian Ministerial grants PRIN 2022: "B-Fair: Bias-Free Artificial Intelligence Methods for Automated Visual Recognition" (CUP E53D23008010006) and by the University of Modena and Reggio Emilia and Fondazione di Modena through the "Fondo di Ateneo per la Ricerca - FAR 2024" (CUP E93C24002080007). The work also received funding from DECIDER, the European Union's Horizon 2020 research and innovation programme under GA No. 965193 and "AIDA: explAinable multImodal Deep learning for person-Alized oncology" (Project Code 20228MZFAA). We acknowledge the CINECA award under the ISCRA initiative for providing high-performance computing resources.

Impact Statement

Our approach to transferring fine-tuning between model versions without requiring re-training or additional data holds the potential to significantly lower the barriers to maintaining state-of-the-art AI technologies. By enabling seamless updates, this method could empower organizations, especially those with limited resources, to keep pace with technological advancements, thereby democratizing access to sophisticated AI tools. This democratization can foster innovation across diverse sectors, potentially leveling the playing field between large entities and smaller, resource-constrained organizations.

The idea of aligning models with minor variations in architecture, such as different layer counts, is also worth exploring. One simple approach could involve selectively pruning layers from the model with more layers to match its counterpart—removing redundant or unimportant layers—while an alternative strategy might replicate the last block of the smaller network multiple times to achieve parity. We plan to investigate these strategies further in future work.

However, as we facilitate easier transitions between model versions, it is crucial to ensure that these updates do not compromise the integrity of the models. Dependence on outdated or poorly validated models poses significant risks, particularly when used in critical applications. Therefore, it is imperative that as this technology is adopted, continuous efforts are made to monitor, validate, and refine these models to safeguard against biases and errors that may arise from rapid model evolution. Future research should focus on developing robust frameworks for evaluating the ethical implications, fairness, and transparency of AI models as they evolve, ensuring that advancements in AI technology are implemented responsibly and ethically.

References

Ainsworth, S., Hayase, J., and Srinivasa, S. Git Re-Basin: Merging Models modulo Permutation Symmetries. In *International Conference on Learning Representations*, 2023.

Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023.

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

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. Describing Textures in the Wild. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014.

Crisostomi, D., Fumero, M., Baieri, D., Bernard, F., and Rodola, E. C^2M^3: Cycle-Consistent Multi-Model Merging. *Advances in Neural Information Processing Systems*, 2024.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. Essentially No Barriers in Neural Network Energy Landscape. In *International Conference on Machine Learning*, 2018.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The Role of Permutation Invariance in Linear Mode Connectivity of Neural Networks. In *International Conference on Learning Representations*, 2022.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear Mode Connectivity and the Lottery Ticket Hypothesis. In *International Conference on Machine Learning*, 2020.

Freeman, C. D. and Bruna, J. Topology and Geometry of Half-Rectified Network Optimization. In *International Conference on Learning Representations*, 2017.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4), 1999.

Gadre, S. Y., Ilharco, G., Fang, A., Hayase, J., Smyrnis, G., Nguyen, T., Marten, R., Wortsman, M., Ghosh, D., Zhang, J., et al. Datacomp: In search of the next generation of multimodal datasets. *Advances in Neural Information Processing Systems*, 2024.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. *Advances in Neural Information Processing Systems*, 2018.

Helber, P., Bischke, B., Dengel, A., and Borth, D. EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7), 2019.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., and Gilmer, J. The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization. *IEEE International Conference on Computer Vision*, 2021.

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., and Schmidt, L. OpenCLIP, 2021.

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

Ilharco, G., Ribeiro, M. T., Wortsman, M., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing Models with Task Arithmetic. In *International Conference on Learning Representations*, 2023.

Imfeld, M., Graldi, J., Giordano, M., Hofmann, T., Anagnostidis, S., and Singh, S. P. Transformer Fusion with Optimal Transport. In *International Conference on Learning Representations*, 2024.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging Weights Leads to Wider Optima and Better Generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Jonker, R. and Volgenant, T. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*, 1988.

Jordan, K., Sedghi, H., Saukh, O., Entezari, R., and Neyshabur, B. REPAIR: REnormalizing Permuted Activations for Interpolation Repair. In *International Conference on Learning Representations*, 2023.

Jovanović, I. and Stanić, Z. Spectral distances of graphs. *Linear Algebra and its Applications*, 436(5), 2012.

Krizhevsky, A., Hinton, G., et al. Learning Multiple Layers of Features from Tiny Images. *Technical Report, University of Toronto*, 2009.

Lu, M. Y., Chen, B., Williamson, D. F., Chen, R. J., Liang, I., Ding, T., Jaume, G., Odintsov, I., Le, L. P., Gerber, G., et al. A visual-language foundation model for computational pathology. *Nature Medicine*, 30, 2024.

Mall, U., Phoo, C. P., Liu, M. K., Vondrick, C., Hariharan, B., and Bala, K. Remote Sensing Vision-Language Foundation Models without Annotations via Ground Remote Alignment. In *International Conference on Learning Representations*, 2024.

Matena, M. S. and Raffel, C. A. Merging Models with Fisher-Weighted Averaging. *Advances in Neural Information Processing Systems*, 35, 2022.

McCloskey, M. and Cohen, N. J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Academic Press, 1989.

Navon, A., Shamsian, A., Achituve, I., Fetaya, E., Chechik, G., and Maron, H. Equivariant Architectures for Learning in Deep Weight Spaces. In *International Conference on Machine Learning*, 2023.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., et al. Reading Digits in Natural Images with Unsupervised Feature Learning. In *Neural Information Processing Systems Workshops*. Granada, 2011.

Ortiz-Jimenez, G., Favero, A., and Frossard, P. Task Arithmetic in the Tangent Space: Improved Editing of Pre-Trained Models. *Advances in Neural Information Processing Systems*, 2024.

Peña, F. A. G., Medeiros, H. R., Dubail, T., Aminbeidokhti, M., Granger, E., and Pedersoli, M. Re-basin via implicit Sinkhorn differentiation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023.

Porrello, A., Bonicelli, L., Buzzega, P., Millunzi, M., Calderara, S., and Cucchiara, R. A Second-Order Perspective on Model Compositionality and Incremental Learning. In *International Conference on Learning Representations*, 2025.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning*, 2021.

Rame, A., Kirchmeyer, M., Rahier, T., Rakotomamonjy, A., patrick gallinari, and Cord, M. Diverse Weight Averaging for Out-of-Distribution Generalization. In *Advances in Neural Information Processing Systems*, 2022.

Ramé, A., Ahuja, K., Zhang, J., Cord, M., Bottou, L., and Lopez-Paz, D. Model Ratatouille: Recycling Diverse Models for Out-of-Distribution Generalization. In *International Conference on Machine Learning*, 2023.

Singh, S. P. and Jaggi, M. Model Fusion via Optimal Transport. *Advances in Neural Information Processing Systems*, 2020.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *The 2011 international joint conference on neural networks*. IEEE, 2011.

Stoica, G., Bolya, D., Bjorner, J., Ramesh, P., Hearn, T., and Hoffman, J. ZipIt! Merging Models from Different Tasks without Training. In *International Conference on Learning Representations*, 2024.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*, 2019.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, 2022a.

Wortsman, M., Ilharco, G., Kim, J. W., Li, M., Kornblith, S., Roelofs, R., Lopes, R. G., Hajishirzi, H., Farhadi, A., Namkoong, H., et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2022b.

Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. TIES-Merging: Resolving Interference When Merging Models. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhang, F. Z., Albert, P., Rodriguez-Opazo, C., van den Hengel, A., and Abbasnejad, E. Knowledge Composition using Task Vectors with Learned Anisotropic Scaling. *Advances in Neural Information Processing Systems*, 2024.

A. Appendix

A.1. On the Invariance to Permutations of our Metric for Inter-head Alignment

Proposition A.1. *Let $h \in \mathbb{R}^{m\times n}$ be arbitrary. For any h, denote its singular values by $\sigma(h) = (\sigma_1(h), \sigma_2(h), \ldots, \sigma_{\min(m,n)}(h))$, where $\sigma_1(h) \geq \sigma_2(h) \geq \cdots \geq 0$. For two matrices h_1, h_2 of the same shape, define*

$$d_p(h_1, h_2) = \|\sigma(h_1) - \sigma(h_2)\|_p\,, \tag{15}$$

where $\|\cdot\|_p$ is the usual p-norm for vectors. Then, for any permutation matrices $P_r \in \mathbb{R}^{m\times m}$ and $P_c \in \mathbb{R}^{n\times n}$, the row- and column-permuted matrix

$$h' = P_r h P_c \tag{16}$$

has exactly the same singular values as h. In particular,

$$d_p(h, h') = 0 \tag{17}$$

for every p, making d invariant under row- and column-permutations of h.

Proof. If P is a permutation matrix, then $P^\top P = I$, *i.e.* it is orthogonal. Furthermore, the singular values of any matrix h are given by the square root of the eigenvalues of $h^\top h$. If $h' = P_r\, h\, P_c$, then

$$(h')^\top (h') = (P_r\, h\, P_c)^\top\ (P_r\, h\, P_c) \tag{18}$$

$$= P_c^\top\, h^\top\, P_r^\top\ P_r\, h\, P_c \tag{19}$$

$$= P_c^\top\, h^\top\, h\, P_c\,. \tag{20}$$

Since $P_c^\top\, h^\top h\, P_c$ is a similarity transform of $h^\top h$, which does not change the eigenvalues, $h^\top h$ and $(h')^\top h'$ have the same eigenvalues, and in turn h and h' share the same singular values. hence $\sigma(h') = \sigma(h)$, and therefore

$$d_p(h, h') = \|\sigma(h) - \sigma(h')\|_p = 0\,, \tag{21}$$

proving that, for any row or column permutation of h, the distance $d(h, h')$ remains unchanged. □

A.2. Proof of Equivariance of Multi-Head Attention to Structured Permutations 3.1

Proof. We provide a detailed, step-by-step proof showing that our two-stage alignment procedure—inter-head reordering followed by intra-head permutations—preserves the functionality of a multi-head self-attention layer. Let:

- $X \in \mathbb{R}^{S\times d_{\text{model}}}$ be the input sequence.
- $W_q, W_k, W_v \in \mathbb{R}^{d_{\text{model}}\times d_{\text{model}}}$ be the query, key, and value projection matrices.
- H be the number of attention heads, each of dimensionality $d_k = d_{\text{model}}/H$.

Define:

$$Q = XW_q, \quad K = XW_k, \quad V = XW_v, \tag{22}$$

and split them by head:

$$Q = [Q_1, Q_2, \ldots, Q_H], \quad Q_i \in \mathbb{R}^{S\times d_k}, \tag{23}$$

and similarly for K and V. Let P_{inter} be an inter-head permutation in $\mathcal{S}_H$, with induced permutation vector π, and let $P_{\text{intra}}^{(i)} \in \mathcal{S}_{d_k}$ be the intra-head permutation for head i. We form the block-permutation matrix:

$$P_{\text{attn}} = \sum_{i=1}^{H} E^{i,\pi(i)} \otimes P_{\text{intra}}^{(i)}, \tag{24}$$

where $E^{i,\pi(i)}$ is a binary $H \times H$ matrix with a single 1 at $(i, \pi(i))$, and $\otimes$ denotes the Kronecker product.

Step 1: Permuting the projection weights

Applying P_{attn} to the query projections gives:

$$\begin{aligned} Q' &= XW_qP_{\text{attn}} = QP_{\text{attn}} \\ &= \left[\sum_{j=1}^{H} Q_j P_{\text{attn}}[j,i]\right]_{i=1}^{H} \\ &= \left[Q_{\pi^{-1}(i)} P_{\text{intra}}^{\pi^{-1}(i)}\right]_{i=1}^{H} \end{aligned}$$

Hence,

$$Q'_i = Q_{\pi^{-1}(i)}\, P_{\text{intra}}^{\pi^{-1}(i)}, \tag{25}$$

where the new head Q'_i corresponds to the head designated by the inter-head permutation $\pi^{-1}(i)$, modified according to $P_{\text{intra}}^{\pi^{-1}(i)}$. The same applies to:

$$K'_i = K_{\pi^{-1}(i)} P_{\text{intra}}^{\pi^{-1}(i)}, \quad V'_i = V_{\pi^{-1}(i)} P_{\text{intra}}^{\pi^{-1}(i)}. \tag{26}$$

Step 2: Permuting the attention scores

Because each $P_{\text{intra}}^{(i)}$ is orthogonal ($PP^T = I$), the attention scores satisfy:

$$\begin{aligned} A'_i &= \text{softmax}\left(\frac{Q'_i {K'_i}^T}{\sqrt{d_k}}\right) \\ &= \text{softmax}\left(\frac{Q_{\pi^{-1}(i)} P_{\text{intra}}^{\pi^{-1}(i)} (P_{\text{intra}}^{\pi^{-1}(i)})^T K^T_{\pi^{-1}(i)}}{\sqrt{d_k}}\right) \\ &= \text{softmax}\left(\frac{Q_{\pi^{-1}(i)} K^T_{\pi^{-1}(i)}}{\sqrt{d_k}}\right) = A_{\pi^{-1}(i)}. \end{aligned}$$

Thanks to the orthogonality of the intra-head permutation blocks, the attention scores are only influenced by the inter-head permutation.

Step 3: Permuting the value outputs

For each head,

$$O'_i = A'_i V'_i = A_{\pi^{-1}(i)} V_{\pi^{-1}(i)} P_{\text{intra}}^{\pi^{-1}(i)} = O_{\pi^{-1}(i)} P_{\text{intra}}^{\pi^{-1}(i)}. \tag{27}$$

Step 4: Reconstructing the final output

Concatenating all heads yields:

$$O' = [O'_1, O'_2, \ldots, O'_H] = O P_{\text{attn}}. \tag{28}$$

Conclusion. Applying P_{attn} to the projection matrices is thus equivalent to permuting the output of multi-head attention. The self-attention layer remains functionally equivalent, and the original output can be recovered via $O = O' P_{\text{attn}}^T$. □

A.3. Full Procedure to Manage Residual Connections

Proof. We begin with the standard formulation of a transformer block, ignoring LayerNorm for simplicity:

$$\begin{aligned} \mathbf{z}_{attn} &= W_0\,\text{MHA}(\mathbf{x}), \\ \mathbf{z}_i &= \mathbf{z}_{attn} + \mathbf{x}, \\ \mathbf{z}_f &= W_2 \text{ReLU}(W_1 \mathbf{z}_i), \\ \mathbf{z}_{out} &= \mathbf{z}_f + \mathbf{z}_i. \end{aligned} \tag{29}$$

Ignoring the ReLU activation function as well, we examine the impact of applying a permutation to one layer within a transformer block and then reversing it in the subsequent layer. This transformation leads to:

$$\begin{aligned} \mathbf{z}_{\text{attn}} &= P_{\text{W}_0} W_0 P_{\text{attn}}^\top \Big(P_{\text{attn}}\,\text{MHA}\, P_{\text{in}}^\top (P_{\text{in}} \mathbf{x}) \Big), \\ \mathbf{z}_i &= P_{\text{W}_0} \mathbf{z}_{\text{attn}} + P_{\text{in}} \mathbf{x}, \\ \mathbf{z}_f &= P_{\text{W}_2} W_2 P_{\text{W}_1}^\top \Big(P_{\text{W}_1} W_1 P_{\text{W}_0}^\top (\mathbf{z}_i) \Big), \\ \mathbf{z}_{\text{out}} &= P_{\text{W}_2} \mathbf{z}_f + P_{\text{W}_0} \mathbf{z}_i. \end{aligned} \tag{30}$$

To ensure consistency in the permutation applied to both addends within each residual block, we replace the identity mapping with a permutation composition $\mathcal{I}$, where $\mathcal{I}_i = P_{\text{W}_0} P_{\text{in}}^\top$ and $\mathcal{I}_{\text{out}} = P_{\text{W}_2} P_{\text{W}_0}^\top$. This results in:

$$\begin{aligned} \mathbf{z}_i &= P_{\text{W}_0} \mathbf{z}_{\text{attn}} + \mathcal{I}_i P_{\text{in}} \mathbf{x} = P_{\text{W}_0} \mathbf{z}_{\text{attn}} + P_{\text{W}_0} \mathbf{x}, \\ \mathbf{z}_{\text{out}} &= P_{\text{W}_2} \mathbf{z}_f + \mathcal{I}_{\text{out}} P_{\text{W}_0} \mathbf{z}_i = P_{\text{W}_2} \mathbf{z}_f + P_{\text{W}_2} \mathbf{z}_i. \end{aligned} \tag{31}$$

After incorporating these compositions, the permutations remain consistent across each residual path, simplifying the block equations to:

$$\begin{aligned} \mathbf{z}_{\text{attn}} &= P_{\text{W}_0} W_0 \Big(\text{MHA}(\mathbf{x}) \Big), \\ \mathbf{z}_i &= P_{\text{W}_0} \mathbf{z}_{\text{attn}} + P_{\text{W}_0} \mathbf{x}, \\ \mathbf{z}_f &= P_{\text{W}_2} W_2 \Big(W_1 (\mathbf{z}_i) \Big), \\ \mathbf{z}_{\text{out}} &= P_{\text{W}_2} \mathbf{z}_f + P_{\text{W}_2} \mathbf{z}_i. \end{aligned} \tag{32}$$

With P_{W_2} serving as the input permutation for the subsequent layer. □

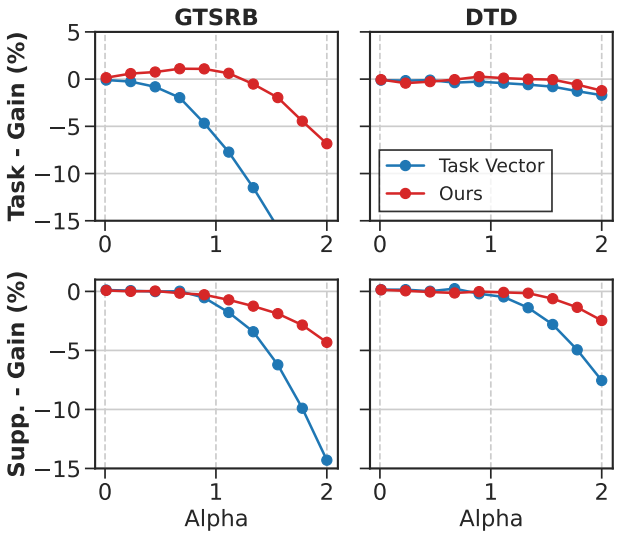

Figure 6: Zero-shot gain/drop relative to θ_B of naive $\theta_B + \alpha\tau$ (blue) and our strategy $\theta_B + \alpha\pi(\tau)$ (red) varying α.

A.4. Extended Comparison on the Application of the Task Vector

In Fig. 6, we extend the sensitivity analysis of the scaling coefficient α — originally presented in Fig. 4 — to additional visual classification tasks.

A.5. Proof of Proposition 3.2

Proposition A.2. *Proof.* To assess how computational complexity scales with model size, we define:

- L: number of layers, evenly divided into MLP ($\frac{L}{2}$) and self-attention ($\frac{L}{2}$).
- H: number of attention heads.
- Each MLP layer contains two linear projections with dimension (d_m, d_h) and (d_h, d_m), assuming $d_m = d_h$.
- Self-attention layers have Q, K, and V matrices, each of size (d_m, d_m).

We now estimate the complexity for a single iteration of the weight-matching algorithm.

MLP Layers. The main computational cost comes from computing a pairwise similarity matrix between rows of projection matrices ($O(d_m^3)$), and solving a (d_m, d_m) assignment via Hungarian algorithm ($O(d_m^3)$). Hence, per-layer cost is:

$$O(d_m^3) \tag{33}$$

Self-Attention Layers. Split into two steps: inter-head and intra-head permutation.

Inter-head permutation:

- $6H$ SVDs over matrices of size $(\frac{d_m}{H}, d_m)$

$$O\left(\frac{6d_m^3}{H}\right) \tag{34}$$

- Distance matrix over heads:

$$O\left(\frac{3H^2 d_m}{2}\right) \tag{35}$$

- Hungarian algorithm over (H, H) matrix:

$$O(H^3) \tag{36}$$

Intra-head permutation:

- Per-head similarity:

$$O\left(\left(\frac{d_m}{H}\right)^2 d_m\right) = O\left(\frac{d_m^3}{H^2}\right) \tag{37}$$

- Hungarian algorithm per head:

$$O\left(\left(\frac{d_m}{H}\right)^3\right) \tag{38}$$

Summed over H heads:

$$O\left(H\left(\frac{d_m^3}{H^2} + \left(\frac{d_m}{H}\right)^3\right)\right) = O\left(\frac{d_m^3}{H} + \frac{d_m^3}{H^2}\right) \tag{39}$$

Total Self-Attention Cost per Layer.

$$O\left(\frac{6d_m^3}{H} + \frac{3H^2 d_m}{2} + H^3 + \frac{d_m^3}{H} + \frac{d_m^3}{H^2}\right) \tag{40}$$

Final Complexity. Summing across $\frac{L}{2}$ MLP and $\frac{L}{2}$ attention layers:

$$O\left(\frac{L}{2} d_m^3 + \frac{L}{2}\left(\frac{6d_m^3}{H} + \frac{3H^2 d_m}{2} + H^3 + \frac{d_m^3}{H} + \frac{d_m^3}{H^2}\right)\right). \tag{41}$$

This expression can be algebraically simplified to a more compact equivalent form:

$$O\left(L\left(d_m^3 + \frac{d_m^3}{H} + \frac{d_m^3}{H^2} + H^3 + H^2 d_m\right)\right). \tag{42}$$

So, the complexity scales polynomially with d_m and H, and remains significantly lower than data-based fine-tuning. □