ALIGN WHAT FORMS THE UPDATE: BILINEAR FACTOR ALIGNMENT FOR CROSS-MODEL TASK-VECTOR TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting large-scale pre-trained models to specialized tasks has been one of the prevalent paradigms in modern machine learning, with fine-tuning being one of the most widely studied approaches. However, when a new version of a pre-trained model becomes available, expertise acquired through fine-tuning cannot be directly reused because it is tied to the parameterization of the original model, requiring another costly fine-tuning. To address this inefficiency, recent work seeks to transfer expertise across models in the form of task vectors—defined as the parameter difference between a fine-tuned model and its base model. However, the connection between task-vector formation and cross-model alignment remains underexplored. In this work, we revisit task-vector formation and derive an alignment principle directly from the structure of the parameter update. Specifically, each layer-wise parameter gradient factorizes as an outer product between an input-side activation and an output-side gradient, revealing the bilinear structure underlying the update. This structure exposes two distinct factor spaces and motivates aligning their coordinate systems across models. Building on our observation, we propose BIFA (**Bi**linear **F**actor **A**lignment), a training-free framework that separately aligns the activation and gradient factor spaces across source and target models to transfer task vectors. Across extensive vision and language-model benchmarks, BIFA consistently outperforms existing transfer methods across models that differ in width, depth, and pre-training configuration, further extending to cross-family LLM transfer with greater architectural mismatch.

1 INTRODUCTION

Modern machine learning has increasingly adopted a paradigm in which large-scale pre-trained models (Achiam et al., 2023; Touvron et al., 2023; Dosovitskiy et al., 2021) are adapted to downstream tasks, with fine-tuning as one of widespread approaches (Brown et al., 2020; Wei et al., 2022; Wortsman et al., 2022). While fine-tuning may be effective, it becomes inefficient as pre-trained model families are continually expanded with new models, even when they share a common architecture backbone (e.g., Transformer (Vaswani et al., 2017)) but differ in model size, architectural configuration, or pre-training recipes (Cherti et al., 2023; Gadre et al., 2023). Because the resulting expertise is encoded in the parameter space of a particular pre-trained model on which fine-tuning is performed, a newly released model variant often requires separate fine-tuning to acquire similar capabilities. This leads to repeated computational cost and motivates the problem: transferring task expertise across pre-trained model variants without fine-tuning the target model.

Recently, there has been a growing line of research work that aims to transfer task-specific knowledge across pre-trained model variants without requiring additional fine-tuning (Rinaldi et al., 2025; 2026a;b). Many methods employ task vectors (Ilharco et al., 2023) as a way to represent fine-tuned expertise, defining them as the parameter difference between a fine-tuned model and its base model. In particular, given a task vector obtained by fine-tuning a source pre-trained model, the goal is to transfer this task vector to a separate target pre-trained model so that the target acquires the corresponding task expertise without full fine-tuning. Permutation-based methods exploit weight-space symmetries to re-base task vectors across models (Ainsworth et al., 2023; Rinaldi et al., 2025), while GradFix (Rinaldi et al., 2026a) uses target-model gradients to mask task-vector coordinates

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

according to the local loss landscape. However, such parameter-level approaches typically rely on strong architectural compatibility and matching parameter dimensions, limiting its applicability to models that differ in scale or width. To move beyond such identical-architecture requirement, THESEUS (Rinaldi et al., 2026b) transports task vectors across models with mismatched hidden dimensions by estimating transformation maps between their intermediate activation spaces, enabling transfer across models with greater architectural mismatch. While Rinaldi et al. (2026b) have shown the potential of correspondence in task-vector transfer of disparate models, a significant performance gap remains relative to direct fine-tuning. Further, prior work has largely treated task-vector formation and cross-model alignment as separate problems, leaving their relationship relatively underexplored.

Therefore, we take a different starting point: rather than introducing another transformation mapping estimator, we revisit the formation of the task vector and study whether the structure of the update itself can identify the spaces that should be aligned across models. In particular, our revisited formulation reveals that a task vector can be formulated as the accumulation of the bilinear interactions coupling input-side activations with output-side gradients. This perspective exposes two factor spaces underlying task-vector formation and suggests a factor-wise view of cross-model alignment: the activation and gradient spaces should be related separately across source and target models.

To this end, we propose **BIFA**, a training-free framework that leverages **Bi**linear **F**actor **A**lignment to transfer task-specific expertise across pre-trained models. For each layer, BIFA estimates an input-side alignment from paired pre-trained activations and an output-side alignment from paired task-conditioned gradients. Importantly, the required alignments are estimated from a small calibration set using a single forward–backward pass through frozen pre-trained models, without requiring target-model parameter updates. We demonstrate the effectiveness of BIFA across a wide range of benchmarks in computer vision and language models, showing that it enables robust knowledge transfer even across models with architectural and pre-training variations, further extending to cross-family LLM transfer where structural discrepancies extend well beyond model size. Our contributions are summarized as follows:

- **We revisit task-vector formation and expose its bilinear factor structure**, identifying input activations and output gradients as the two distinct factors.
- **We introduce BIFA, a training-free factor-wise alignment framework** that separately aligns activation and gradient spaces.
- **We show that BIFA consistently improves transfer across multiple mapping estimators**, supporting the effectiveness of explicitly aligning the factor spaces revealed by the bilinear update structure.

2 Related work

Task vectors and model merging. Task vectors provide a parameter-space representation of task-specific expertise and have been widely studied for model merging and composition (Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024; Wang et al., 2024; Gargiulo et al., 2025). A representative approach is Task Arithmetic (Ilharco et al., 2023), which represents fine-tuned expertise as task vectors in parameter space and manipulates them through linear operations such as addition and negation. Subsequent methods improve task-vector composition and model merging by reducing interference among task vectors (Yadav et al., 2023; Yu et al., 2024), identifying consensus directions (Wang et al., 2024), or exploiting low-rank structure (Gargiulo et al., 2025). These approaches establish task vectors as a useful abstraction for reusing task-specific knowledge, but typically assume that the vectors inhabit a shared parameter space. This exact-base-model assumption fundamentally limits their applicability across differing architectures or pre-training recipes, necessitating a new approach for cross-model alignment.

Cross-model alignment. A separate line of work studies how to establish correspondence between independently trained models. From the parameter-space perspective, Re-basin methods (Ainsworth et al., 2023; Rinaldi et al., 2025) are based on the observation that models with the same architecture but different initializations can learn similar functions while representing them with different hidden-unit orderings. They address this parameter-space mismatch by exploiting permutation symmetries to map models into a shared basin. However, these techniques are generally restricted to models

with identical architectures and widths. Representation-level studies provide another perspective by comparing models through functional similarity rather than parameter correspondence (Kornblith et al., 2019; Csiszárik et al., 2021; Bansal et al., 2021; Mai et al., 2026). SVCCA (Raghu et al., 2017) reveals substantial linear correspondence between representations learned by independently trained networks, while model stitching (Csiszárik et al., 2021; Bansal et al., 2021; Mai et al., 2026) shows that such representations can sometimes be connected through simple mappings. Together, these findings motivate establishing cross-model correspondence between internal model spaces.

Cross-model task-vector transfer. Recent work has explored transferring task-specific parameter update from a source pre-trained model to a different target pre-trained model without repeating full fine-tuning. TransFusion (Rinaldi et al., 2025) re-bases task vectors through permutation-based parameter alignment, while GradFix (Rinaldi et al., 2026a) uses target-model gradients to mask conflicting parameter coordinates. However, both rely on parameter-level compatibility, limiting transfer across structurally different models. THESEUS (Rinaldi et al., 2026b) moves beyond this limitation by characterizing a task update through the functional effect it induces on intermediate representations, using layer input and output activations to align update responses across models. This enables task-vector transfer across models with greater architectural mismatch. In contrast, we take a complementary perspective: rather than defining transfer correspondence from parameter coordinates or representation responses alone, we revisit how the parameter update itself is formed. In particular, we show that the layer-wise gradient factorizes into an input activation and an output-side gradient, exposing two distinct factor spaces underlying the bilinear update. BIFA explicitly aligns these activation and gradient factor spaces across source and target models.

3 PRELIMINARIES

3.1 SETUP AND NOTATION

Consider a pre-trained model with parameters $\boldsymbol{\theta}_{\text{pre}}$. When this model is adapted to a downstream task via fine-tuning, the resulting parameters are denoted as $\boldsymbol{\theta}_{\text{ft}}$. Following Ilharco et al. (2023), we define the task vector $\boldsymbol{\tau}$ as the difference between the expert model and the pre-trained model:

$$\boldsymbol{\tau} = \boldsymbol{\theta}_{\text{ft}} - \boldsymbol{\theta}_{\text{pre}}. \tag{1}$$

In this work, we consider the problem of transferring task expertise in the form of a task vector, available for a source pre-trained model A, to a different target pre-trained model B. We assume both models share the same architectural backbone (e.g., Transformer) but may differ in width, depth, architectural configuration, or pre-training recipes. For a l-th layer, the source task vector is defined as:

$$\boldsymbol{\tau}_A^{(l)} = \boldsymbol{\theta}_{A,\text{ft}}^{(l)} - \boldsymbol{\theta}_{A,\text{pre}}^{(l)}, \tag{2}$$

where $\boldsymbol{\tau}_A^{(l)} \in \mathbb{R}^{d_{\text{out},A} \times d_{\text{in},A}}$ with $d_{\text{in},A}$ and $d_{\text{out},A}$ denoting the input and output dimensions of layer l in the source model A. We aim to transfer the task-specific knowledge from A to B by constructing a target task vector $\hat{\boldsymbol{\tau}}_B^{(l)} \in \mathbb{R}^{d_{\text{out},B} \times d_{\text{in},B}}$, where $d_{\text{in},B}$ and $d_{\text{out},B}$ denote the input and output dimensions of layer l in the target model B. The goal is to ensure that the updated target layer, $\boldsymbol{\theta}_{B,\text{pre}}^{(l)} + \hat{\boldsymbol{\tau}}_B^{(l)}$, induces a functional effect analogous to the source expert layer, without requiring additional fine-tuning on model B.

3.2 SEQUENCE MATCHING

Following the protocol in prior work (Rinaldi et al., 2026b), we further consider source and target models with disparate architectures (especially, different widths and depths). To this end, we adopt the pre-processing steps described in Rinaldi et al. (2026b). Given N calibration inputs, we define the input activation of layer l as:

$$\boldsymbol{X}_A^{(l)} \in \mathbb{R}^{N \times L_A \times d_{\text{in},A}}, \quad \boldsymbol{X}_B^{(l)} \in \mathbb{R}^{N \times L_B \times d_{\text{in},B}}, \tag{3}$$

where L_A and L_B represent the respective sequence lengths. For vision models, if $L_A \neq L_B$, we employ bilinear interpolation to align the sequence length of the source model to match L_B. By

aggregating the batch and sequence dimensions into a total of $M = N \times L_B$ tokens, we obtain the flattened activation matrices $\bar{\boldsymbol{X}}_A^{(l)} \in \mathbb{R}^{M \times d_{\text{in},A}}, \quad \bar{\boldsymbol{X}}_B^{(l)} \in \mathbb{R}^{M \times d_{\text{in},B}}$.

For language models with different tokenizers, each model first processes the same calibration text using its own tokenizer. We then establish token-level correspondence after the forward/backward computation by grouping representations associated with the same lexical span and averaging those corresponding to multiple subword tokens. This yields M matched token-level signals for both models, which are used to estimate the input- and output-side alignment maps.

4 METHOD

4.1 REVISITING TASK VECTOR: BILINEAR FACTORIZATION OF TASK-VECTOR FORMATION

To understand how a task vector is formed and transferred across models, we revisit the layer-wise parameter updates induced during fine-tuning. Specifically, we consider a layer l within a Transformer block, such as an MLP or attention projection. At optimization step t, let $\boldsymbol{y}_t^{(l)} = \boldsymbol{x}_t^{(l)} (\boldsymbol{\theta}_t^{(l)})^\top$, where $\boldsymbol{\theta}_t^{(l)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is the weight matrix and $\boldsymbol{x}_t^{(l)} \in \mathbb{R}^{1 \times d_{\text{in}}}$ is its input activation.

The gradient of the loss $\mathcal{L}_t$ with respect to $\boldsymbol{\theta}_t^{(l)}$ can be written by the chain rule as the outer product of the output-side gradient $\boldsymbol{g}_t^{(l)} = \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{y}_t^{(l)}} \in \mathbb{R}^{1 \times d_{\text{out}}}$ and the input activation $\boldsymbol{x}_t^{(l)}$:

$$\nabla_{\boldsymbol{\theta}_t^{(l)}} \mathcal{L}_t = \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{y}_t^{(l)}} \left(\frac{\partial \boldsymbol{y}_t^{(l)}}{\partial \boldsymbol{\theta}_t^{(l)}} \right) = (\boldsymbol{g}_t^{(l)})^\top \boldsymbol{x}_t^{(l)}. \tag{4}$$

To see how this local bilinear structure accumulates into a task vector, we first consider the vanilla gradient-descent update $\Delta \boldsymbol{\theta}_t^{(l)} = -\eta_t \nabla_{\boldsymbol{\theta}_t^{(l)}} \mathcal{L}_t$, where η_t denotes the learning rate at step t. Accumulating the updates over T optimization steps gives

$$\boldsymbol{\tau}^{(l)} = \boldsymbol{\theta}_{\text{ft}}^{(l)} - \boldsymbol{\theta}_{\text{pre}}^{(l)} = \sum_{t=1}^{T} \Delta \boldsymbol{\theta}_t^{(l)} = -\sum_{t=1}^{T} \eta_t \nabla_{\boldsymbol{\theta}_t^{(l)}} \mathcal{L}_t = -\sum_{t=1}^{T} \eta_t (\boldsymbol{g}_t^{(l)})^\top \boldsymbol{x}_t^{(l)}. \tag{5}$$

Equation 5 expresses the layer-wise task vector as an accumulation of bilinear interactions between two factors: the input-side activation $\boldsymbol{x}_t^{(l)}$ and the output-side gradient $\boldsymbol{g}_t^{(l)}$. In Appendix B, we show that this bilinear structure is preserved under momentum-based updates. We further show that the bilinear structure still captures the dominant part of the resulting task vector, even under adaptive preconditioning (e.g., AdamW used in our experiments). This factorization exposes distinct input- and output-side factor spaces underlying the update, motivating factor-wise cross-model alignment of the activation and gradient spaces.

4.2 TASK-VECTOR TRANSFER VIA BILINEAR FACTOR ALIGNMENT

The bilinear structure in Equation 5 suggests a factor-wise transfer rule: separately relate the input- and output-side factor spaces across models. Let $\boldsymbol{R}_{\text{in}}^{(l)} \in \mathbb{R}^{d_{\text{in},A} \times d_{\text{in},B}}$ and $\boldsymbol{R}_{\text{out}}^{(l)} \in \mathbb{R}^{d_{\text{out},A} \times d_{\text{out},B}}$ denote linear maps from the source input- and output-side factor spaces to their target counterparts. We define the transported factors as $\hat{\boldsymbol{x}}_B^{(l)} = \boldsymbol{x}_A^{(l)} \boldsymbol{R}_{\text{in}}^{(l)}$ and $\hat{\boldsymbol{g}}_B^{(l)} = \boldsymbol{g}_A^{(l)} \boldsymbol{R}_{\text{out}}^{(l)}$. Applying these maps to each bilinear contribution in the source task vector defines the transported task vector as

$$\begin{aligned}
\hat{\boldsymbol{\tau}}_B^{(l)} &:= -\sum_{t=1}^{T} \eta_t \hat{\boldsymbol{g}}_{B,t}^{(l)\top} \hat{\boldsymbol{x}}_{B,t}^{(l)} \\
&= -\sum_{t=1}^{T} \eta_t \left(\boldsymbol{g}_{A,t}^{(l)} \boldsymbol{R}_{\text{out}}^{(l)} \right)^\top \left(\boldsymbol{x}_{A,t}^{(l)} \boldsymbol{R}_{\text{in}}^{(l)} \right) \\
&= (\boldsymbol{R}_{\text{out}}^{(l)})^\top \left(-\sum_{t=1}^{T} \eta_t (\boldsymbol{g}_{A,t}^{(l)})^\top \boldsymbol{x}_{A,t}^{(l)} \right) \boldsymbol{R}_{\text{in}}^{(l)} \\
&= (\boldsymbol{R}_{\text{out}}^{(l)})^\top \boldsymbol{\tau}_A^{(l)} \boldsymbol{R}_{\text{in}}^{(l)}.
\end{aligned} \tag{6}$$

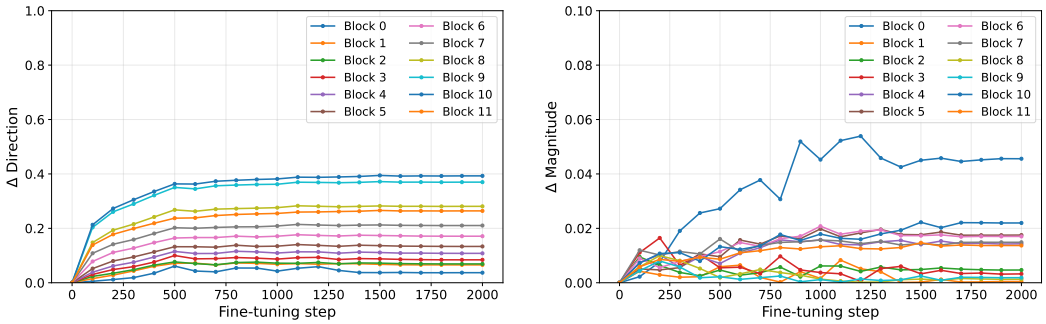

Figure 1: **Consistency of pre-trained activations along the fine-tuning trajectory.** We compare pre-trained activations $\boldsymbol{x}_{\text{pre}}$ with trajectory activations $\boldsymbol{x}_t$ every 100 steps on ViT-B/16, averaged over 8 vision tasks. (Left) directional shift: $1 - \cos(\boldsymbol{x}_t, \boldsymbol{x}_{\text{pre}})$. (Right) relative magnitude change: $|\|\boldsymbol{x}_t\|_2 - \|\boldsymbol{x}_{\text{pre}}\|_2| / \|\boldsymbol{x}_{\text{pre}}\|_2$.

This yields a two-factor transformation of the source task vector, where $\boldsymbol{R}_{\text{in}}^{(l)}$ transports its input-side factor coordinates and $\boldsymbol{R}_{\text{out}}^{(l)}$ transports its output-side factor coordinates. Importantly, our method BIFA specifies what structure is transported—the two factor spaces—and is agnostic to how their cross-model correspondences are estimated. A detailed description of the complete procedure is provided in Appendix A. Finally, the total updated target model $\hat{\boldsymbol{\theta}}_{B,\text{ft}}$ is constructed by adding the aligned task vectors to its original pre-trained weights:

$$\hat{\boldsymbol{\theta}}_{B,\text{ft}} = \boldsymbol{\theta}_{B,\text{pre}} + \hat{\boldsymbol{\tau}}_B. \tag{7}$$

4.3 Estimating the factor correspondences

Equation 5 describes the task vector through activation and gradient factors accumulated over the fine-tuning trajectory. Explicitly collecting these factors throughout fine-tuning, however, would require replaying the entire optimization process. Instead, we use the activation and gradient signals obtained at the pre-trained initialization as practical proxies for estimating the two factor-space alignments, as discussed below. This substitution allows us to approximate both alignments using a single forward–backward pass over a small calibration set.

Input-factor correspondence. For the input side, prior work has observed that layer inputs remain relatively stable during fine-tuning (Cheng et al., 2025). We further verify this behavior by comparing the pre-trained activations with those collected throughout the fine-tuning trajectory. As shown in Figure 1, both their directions and magnitudes remain largely consistent over optimization, supporting the use of pre-trained activations for estimating the input-side alignment.

Output-factor correspondence. For the output side, gradients depend on the evolving predictions and loss landscape, so initialization gradients are not assumed to approximate those throughout the fine-tuning trajectory. Instead, since the source task vector $\boldsymbol{\tau}_A$ already contains the accumulated fine-tuning update, we use the initialization gradient only as a local, task-conditioned signal for estimating the cross-model correspondence between the output-side factor spaces. The remaining question is whether this local backward signal provides an informative correspondence for transporting the trajectory-level task vector. Recent studies have shown that the early phase of fine-tuning dominates the task vector (Zhou et al., 2026; Malladi et al., 2023), making the pre-trained initialization a natural operating point for this estimation. We further verify this choice empirically in Section 5.4: the output-side correspondence estimated at initialization transports the final source task vector more effectively than correspondences estimated at later fine-tuning steps.

Estimating the maps. Given these initialization signals, we estimate the two factor-space maps using the same set of N calibration inputs. Following the sequence matching procedure described in Section 3.2, we extract M aligned calibration tokens from these N inputs. We denote the resulting input activations for models A and B as $\bar{\boldsymbol{X}}_{A,\text{pre}}^{(l)} \in \mathbb{R}^{M \times d_{\text{in},A}}$ and $\bar{\boldsymbol{X}}_{B,\text{pre}}^{(l)} \in \mathbb{R}^{M \times d_{\text{in},B}}$, respectively. Similarly, their corresponding output gradients are denoted as $\bar{\boldsymbol{G}}_{A,\text{pre}}^{(l)} \in \mathbb{R}^{M \times d_{\text{out},A}}$ and

Table 1: **Width scaling (narrow → wide).** Task-vector transfer from a ViT-B/16 model *A* (LAION-2B) to a wider ViT-B/16-plus model *B* (LAION-400M). N denotes the number of calibration samples used to estimate the alignment maps. ΔAcc denotes accuracy gain over model *B* zero-shot baseline.

Model	N	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG (ΔAcc)
θ_B *zero-shot*	–	42.51	49.61	37.77	65.95	55.42	84.50	57.02	68.50	57.66 (+0.00)
θ_B *fine-tune*	–	99.00	99.19	97.80	96.46	82.55	90.11	99.81	78.29	92.90 (+35.24)
θ_A *fine-tune*	–	94.67	99.03	97.89	96.81	83.46	92.10	99.72	79.35	92.88 (+35.22)
$\theta_B + \tau_A^{\text{padded}}$	–	46.41	38.21	27.27	62.57	53.56	82.00	51.70	68.29	53.75 (-3.91)
THESEUS	10	60.25	51.10	44.84	66.46	55.95	84.66	60.24	68.71	61.53 (+3.87)
BIFA-C	10	62.78	51.79	41.80	66.70	57.02	84.84	75.66	68.64	63.65 (+5.99)
BIFA-R	10	64.55	55.24	43.78	66.93	56.27	84.92	75.72	68.52	64.49 (+6.83)
BIFA-P	10	65.07	55.66	46.63	65.63	57.07	84.24	76.39	68.54	**64.90 (+7.24)**
THESEUS	20	63.96	52.66	53.73	67.58	56.11	84.50	56.92	68.67	63.02 (+5.36)
BIFA-C	20	64.63	51.79	53.90	68.84	58.09	84.75	78.55	68.45	66.12 (+8.46)
BIFA-R	20	67.11	57.25	66.68	68.53	56.32	84.81	82.63	68.57	68.99 (+11.33)
BIFA-P	20	67.48	57.06	67.12	67.92	58.24	83.59	84.92	68.80	**69.39 (+11.73)**
THESEUS	50	63.07	54.29	53.46	68.87	56.06	84.67	66.85	68.74	64.50 (+6.84)
BIFA-C	50	70.96	55.24	48.06	68.03	58.40	84.91	85.06	68.51	67.40 (+9.74)
BIFA-R	50	69.92	56.50	57.04	68.90	57.60	85.00	86.24	68.62	68.73 (+11.07)
BIFA-P	50	75.00	60.99	78.14	72.39	59.25	85.23	91.97	68.98	**73.99 (+16.33)**
THESEUS	100	63.22	56.14	50.68	68.68	55.90	84.65	73.07	68.89	65.15 (+7.49)
BIFA-C	100	72.89	56.98	50.10	69.90	61.01	84.89	69.79	68.76	66.79 (+9.13)
BIFA-R	100	70.74	57.42	61.30	69.20	58.98	84.98	86.15	68.76	69.69 (+12.03)
BIFA-P	100	79.70	66.11	78.63	77.49	61.86	85.16	92.93	69.31	**76.40 (+18.74)**

$\bar{\boldsymbol{G}}_{B,\text{pre}}^{(l)} \in \mathbb{R}^{M \times d_{\text{out},B}}$. Finally, with a linear alignment estimator $\mathcal{A}(\cdot,\cdot)$, we estimate the two maps as

$$\begin{aligned} \boldsymbol{R}_{\text{in}}^{(l)} &= \mathcal{A}\Big(\bar{\boldsymbol{X}}_{A,\text{pre}}^{(l)}, \bar{\boldsymbol{X}}_{B,\text{pre}}^{(l)}\Big), \\ \boldsymbol{R}_{\text{out}}^{(l)} &= \mathcal{A}\Big(\bar{\boldsymbol{G}}_{A,\text{pre}}^{(l)}, \bar{\boldsymbol{G}}_{B,\text{pre}}^{(l)}\Big). \end{aligned} \tag{8}$$

Here, $\mathcal{A}$ estimates a source-to-target map $\boldsymbol{R}$ such that $\boldsymbol{Z}_A\boldsymbol{R} \approx \boldsymbol{Z}_B$ for paired signals $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$. We consider a CCA-based mapping, ridge regression, and orthogonal Procrustes, with their detailed formulations provided in Appendix C.

5 Experiments

We evaluate BIFA on the 8-Vision benchmark (Ilharco et al., 2023) and four MergeBench domains (He et al., 2025): instruction following, mathematics, multilingual understanding, and coding. Following THESEUS (Rinaldi et al., 2026b), we consider three transfer settings: (i) different widths, (ii) different widths and depths, and (iii) identical architectures. For (ii), we further evaluate cross-family transfer, where the models additionally differ in tokenization, vocabulary, and architectural configuration. Alignment maps are estimated from N calibration samples per task. Additional results on T5 and GLUE are provided in Appendix G.2.

Baselines. We report θ_B *zero-shot* as the target reference and θ_B *fine-tune* as an oracle. We compare against naïve task-vector transfer and THESEUS (Rinaldi et al., 2026b), the training-free baseline. Naïve transfer directly adds the source task vector when parameter shapes match and uses zero-padding otherwise. For identical architectures, we additionally compare with TransFusion (Rinaldi et al., 2025) and GradFix (Rinaldi et al., 2026a), which require compatible parameter shapes.

We consider three linear estimators for the cross-model correspondence: (i) **BIFA-C** uses a CCA-based mapping, which aligns the source and target spaces through their maximally correlated subspaces; (ii) **BIFA-R** uses ridge regression, which relaxes the orthogonality constraint and allows more flexible linear transformations; and (iii) **BIFA-P** uses orthogonal Procrustes, which constrains

the mapping to preserve the geometry of the source space. Together, these estimators cover complementary mapping regimes, allowing us to evaluate the proposed alignment signals under different linear transformation constraints. BIFA-C and BIFA-R use validation-tuned hyperparameters, while BIFA-P requires no hyperparameter tuning. Full experimental details are provided in Appendix F.

5.1 TRANSFER ACROSS DIFFERENT WIDTHS

Vision. We first evaluate task-vector transfer across models of differing widths in a vision setting. We transfer the task vector τ_A learned by source model *A*, a CLIP ViT-B/16 pre-trained on LAION-2B, to a wider target model *B*, ViT-B/16-plus from OpenCLIP (Cherti et al., 2023) pre-trained on LAION-400M. As shown in Table 1, naïve transfer with zero-padding ($\theta_B + \tau_A^{\text{padded}}$) substantially degrades performance, confirming that task vectors cannot be directly transferred across mismatched parameter spaces. In contrast, all BIFA variants consistently outperform THESEUS across calibration sizes, regardless of the alignment estimator. This suggests that explicitly exploiting the activation–gradient factor structure in BIFA provides benefits beyond the particular linear estimator used to relate the two spaces. Among the three variants, Procrustes-based BIFA-P achieves the strongest overall performance, which we further analyze in Appendix D.3 through preservation of the task-vector singular spectrum.

5.2 TRANSFER ACROSS WIDTH AND DEPTH

Depth matching. For models with different depths, we establish layer correspondence via relative-depth matching. Given source and target models with D_A and D_B layers, each target layer j is matched to source layer $i(j) = \text{round}\left(\frac{j(D_A-1)}{D_B-1}\right)$, where $i(j) \in \{0, \ldots, D_A-1\}$. For each matched pair $(i(j), j)$, we estimate the input- and output-side maps and transfer the source task vector to the target coordinate space. This rule preserves layer ordering and approximates representation-based correspondence without additional matching overhead, as further analyzed in Appendix E.

Cross-Family LLM. To further evaluate generality, we consider cross-family task-vector transfer from Llama-3.2-3B (Grattafiori et al., 2024) to Gemma-2-2B (Gemma Team et al., 2024). The models differ in scale, tokenization, vocabulary, and architecture, introducing substantially greater mismatch than within-family transfer. We handle the layer-count mismatch using relative-depth matching and the tokenizer mismatch using the sequence-matching procedure in Section 3.2. Since the models have different vocabularies and output heads, we tune only the final LM head using the same calibration data and report it as a control. As shown in Table 2, THESEUS falls below the zero-shot target under cross-family mismatch, whereas BIFA consistently yields positive transfer. These results demonstrate that our formulation extends to cross-family transfer under substantial structural mismatch.

Table 2: **Cross-family LLM task-vector transfer.** Source: Llama-3.2-3B, Target: Gemma-2-2B.

Model	N	Instruction following	Math	Multi lingual	Code	AVG (ΔAcc)
θ_B *zero-shot*	–	13.68	27.97	44.12	16.10	25.47 (+0.00)
θ_B *fine-tune*	–	37.70	56.56	45.90	32.38	43.14 (+17.67)
θ_B *head-tuned*	100	13.30	29.11	43.68	17.44	25.88 (+0.41)
THESEUS	100	16.45	18.65	43.52	15.18	23.45 (-2.02)
BIFA-P	100	21.07	29.34	44.24	18.17	**28.21 (+2.74)**
θ_B *head-tuned*	200	12.93	27.90	42.74	17.56	25.28 (-0.19)
THESEUS	200	14.04	19.78	42.58	16.40	23.20 (-2.27)
BIFA-C	200	14.05	28.51	43.20	19.63	26.35 (+0.88)
BIFA-R	200	14.05	28.96	42.80	20.98	26.70 (+1.23)
BIFA-P	200	22.73	29.57	44.65	18.29	**28.81 (+3.34)**

Vision. To evaluate the structural generalization of our approach, we consider a challenging setting where the source and target models differ in both width and depth: ViT-B/16 as the source model *A* and ViT-L/14 as the target model *B*, both pre-trained on DataComp-XL. Additional deeper-to-shallower results are provided in Appendix G.3. As shown in Table 3, BIFA remains effective under simultaneous depth and width mismatch, outperforming the existing transfer baselines. This demonstrates that the proposed alignment remains effective across models with substantial structural differences.

Table 3: **Depth and width scaling.** Task-vector transfer from a ViT-B/16 model to a ViT-L/14 model, both pre-trained on DataComp-XL. N denotes the number of calibration samples used to estimate the alignment maps. ΔAcc denotes accuracy gain over model *B* zero-shot baseline.

Model	N	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG (ΔAcc)
θ_B *zero-shot*	–	64.11	58.91	67.67	72.60	66.96	93.06	86.63	73.77	72.96 (+0.00)
θ_B *fine-tune*	–	99.33	99.29	98.21	97.58	86.11	95.22	99.74	82.40	94.74 (+21.77)
$\theta_B + \tau_A^{\text{padded}}$	–	64.07	58.38	68.14	72.87	66.70	93.11	86.98	73.78	73.00 (+0.04)
THESEUS	100	78.85	65.55	72.96	74.28	67.87	93.16	91.45	74.12	77.28 (+4.32)
BIFA-C	100	75.52	73.54	69.31	76.21	68.51	93.15	96.97	74.20	78.43 (+5.47)
BIFA-R	100	75.03	73.87	74.22	76.31	69.62	93.30	97.03	74.48	79.24 (+6.28)
BIFA-P	100	87.25	78.63	86.10	82.33	71.17	93.80	98.27	74.70	**84.03 (+11.07)**

Table 4: **Identical architecture transfer.** Task-vector transfer between ViT-B/16 models with identical architectures but different pre-training (*A*: DataComp-XL $\rightarrow$ *B*: LAION-2B). K denotes the number of examples per class used to estimate the alignment maps. ΔAcc denotes accuracy gain over model *B* zero-shot baseline.

Model	K	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG (ΔAcc)
θ_B *zero-shot*	–	50.15	48.35	50.05	68.22	55.96	88.56	65.79	70.44	62.19 (+0.00)
θ_B *fine-tune*	–	94.67	99.03	97.89	96.81	83.46	92.10	99.72	79.35	92.88 (+30.69)
$\theta_B + \tau_A$	–	36.07	33.96	20.33	63.16	54.73	86.82	72.09	66.74	54.24 (-7.95)
TransFusion	–	50.30	49.48	52.57	68.27	56.65	88.60	73.51	70.51	63.74 (+1.55)
GradFix	1	72.33	60.32	57.28	73.02	56.91	85.50	87.66	71.34	70.55 (+8.36)
THESEUS	1	56.85	61.03	57.16	69.95	57.12	88.52	83.08	71.15	68.11 (+5.92)
BIFA-P	1	67.00	74.47	74.94	71.98	63.24	87.76	91.48	72.21	**75.39 (+13.20)**
GradFix	2	72.59	62.95	63.42	73.18	57.18	85.50	88.46	71.64	71.87 (+9.68)
THESEUS	2	63.44	61.58	60.76	71.15	57.76	88.54	80.01	71.44	69.34 (+7.15)
BIFA-P	2	73.29	79.50	71.32	78.60	67.07	87.53	91.39	72.44	**77.64 (+15.45)**
GradFix	5	70.22	63.78	65.06	71.85	57.18	82.92	89.48	71.08	71.45 (+9.26)
THESEUS	5	65.48	61.22	60.67	72.12	59.14	88.59	85.36	71.48	70.51 (+8.32)
BIFA-C	5	67.56	69.95	70.07	73.97	61.91	88.88	92.58	71.20	74.52 **(+12.33)**
BIFA-R	5	74.19	69.02	67.41	74.25	61.86	88.71	91.79	71.34	74.82 **(+12.63)**
BIFA-P	5	77.77	81.96	72.69	85.65	68.45	87.68	92.21	73.11	**79.94 (+17.75)**

5.3 TRANSFER BETWEEN IDENTICAL ARCHITECTURES

We further evaluate BIFA in an identical-architecture setting, where both source and target models use ViT-B/16 but are pre-trained on different data distributions. Specifically, we transfer task vectors from a source model *A* pre-trained on DataComp-XL to a target model *B* pre-trained on LAION-2B. For methodological consistency, we follow the evaluation protocol of GradFix (Rinaldi et al., 2026a) and use a standard K-shot setting, where K denotes the number of examples per class used for coordinate alignment. As shown in Table 4, naïvely applying the source task vector to the target model, $\theta_B + \tau_A$, degrades performance below the zero-shot baseline, confirming the need for alignment even when architectures match. In contrast, BIFA consistently outperforms existing transfer baselines across all K and further improves as the calibration size increases.

5.4 ANALYSIS OF ALIGNMENT SIGNALS

Effect of bilinear factor alignment across correspondence estimators. To isolate the effect of the alignment signals, we evaluate all combinations of activation- and gradient-based statistics on the input and output sides using the same three linear estimators. As shown in Table 5, the activation–gradient configuration proposed by BIFA consistently achieves the best performance across all mapping choices. Notably, even its lowest performance across the three estimators exceeds every alternative signal configuration, including Act.–Act. with Procrustes—the strongest alternative

Table 5: **Ablation of alignment signals.** Average accuracy over 8 vision tasks for ViT-B/16 → ViT-B/16-plus with $N = 100$ calibration samples.

Input	Output	CCA-based	Ridge	Procrustes
Grad.	Act.	60.42	59.87	59.89
Grad.	Grad.	59.88	60.78	61.33
Act.	Act.	61.89	64.38	65.15
Act.	Grad.	**66.79**	**69.69**	**76.40**

Table 6: **Effect of task conditioning.** Average accuracy across 8 vision tasks for ViT-B/16 → ViT-B/16-plus for different calibration sizes N.

Task conditioning	$N = 10$	$N = 20$	$N = 50$	$N = 100$
Independent perm.	60.63	60.60	64.08	67.12
Uniform	61.84	64.79	68.24	69.96
Shared perm.	64.15	67.72	72.11	74.18
Ground truth (BIFA-P)	**64.90**	**69.39**	**73.99**	**76.40**

and the alignment used by THESEUS. These results highlight the importance of our proposed input–output alignment configuration for effective task-vector transfer.

Effect of gradient estimation step on output-side alignment. We examine whether the output-side correspondence estimated at the pre-trained initialization is informative for transporting the trajectory-level task vector. Using ViT-B/16 → ViT-B/16-plus with $N = 100$ calibration samples and Procrustes alignment, we estimate $\boldsymbol{R}_{\text{in}}^{t}$ and $\boldsymbol{R}_{\text{out}}^{t}$ at different fine-tuning steps t, while keeping the final source task vector $\boldsymbol{\tau}_A$ fixed. As shown in Figure 2, replacing $\boldsymbol{R}_{\text{out}}^{pre}$ with later-step correspondences causes a substantial drop in transfer performance. In contrast, replacing $\boldsymbol{R}_{\text{in}}^{pre}$ with $\boldsymbol{R}_{\text{in}}^{t}$ has negligible effect when $\boldsymbol{R}_{\text{out}}^{pre}$ is retained, consistent with the activation consistency in Section 4.3. These results support the use of initialization gradients as a local signal for estimating the output-side correspondence that effectively transports the accumulated task vector.

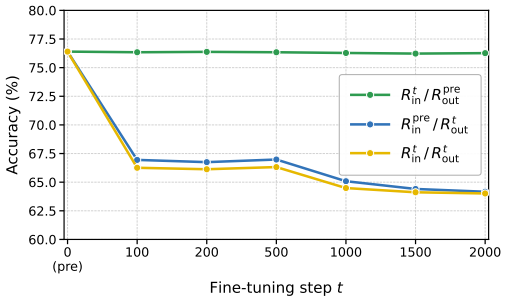

Figure 2: **Transfer performance across alignment estimation steps.** Average accuracy across 8 vision tasks.

Effect of task conditioning on output-side alignment. We further examine how task conditioning affects the output-side correspondence, while keeping the input-side alignment $\boldsymbol{R}_{\text{in}}$ fixed. We compare four variants: the ground-truth labels used by BIFA, a shared permutation applied to both models, independent permutations applied separately to each model, and uniform targets assigning probability $1/C$ to every class. As shown in Table 6, ground-truth labels achieve the best transfer performance, consistent with matching the task objective that produced τ_A. A shared permutation causes only a modest drop despite altering the original label semantics. This suggests that, beyond matching the task objective, maintaining consistent backward conditioning across the source and target models is also important for estimating a meaningful output-side correspondence. In contrast, independent permutations substantially degrade transfer, while uniform targets perform better despite removing class-specific information. These results indicate that effective output-side alignment benefits from both task-relevant conditioning and cross-model consistency of the backward structure.

6 Conclusion

In this work, we present BIFA, a training-free framework for cross-model task-vector transfer by revisiting how parameter updates are formed. By exposing the bilinear interaction between input activations and output gradients underlying task-vector formation, BIFA separately aligns these two factor spaces to transfer the source task vector. Across differences in width, depth, pre-training configuration, and model family, BIFA consistently improves transfer over existing methods. Further analyses show that these gains persist across multiple correspondence estimators and highlight the importance of activation–gradient correspondence and task-conditioned initialization signals. More broadly, our results suggest that exploiting the internal structure of parameter updates may provide a useful direction for developing more general cross-model transfer methods.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *ICLR*, 2023.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. In *NeurIPS*, 2021.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. In *ICML*, 2025.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *CVPR*, 2023.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Adrián Csiszárik, Péter Kőrösi-Szabó, Ákos K. Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. In *NeurIPS*, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. Datacomp: In search of the next generation of multimodal datasets. In *NeurIPS*, 2023.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. In *CVPR*, 2025.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yifei He, Siqi Zeng, Yuzheng Hu, Rui Yang, Tong Zhang, and Han Zhao. Mergebench: A benchmark for merging domain-specialized llms. In *NeurIPS*, 2025.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *NeurIPS*, 2021.

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

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019.

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

Viet Dac Lai, Chien Van Nguyen, Nghia Trung Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In *EMNLP*, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q. Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath-tir. Hugging Face repository, 2024.

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

Zheda Mai, Ke Zhang, Fu-En Wang, Zixiao Ken Wang, Albert Y. C. Chen, Lu Xia, Min Sun, Wei-Lun Chao, and Cheng-Hao Kuo. Revisiting model stitching in the foundation model era. In *CVPR*, 2026.

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

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NeurIPS*, 2017.

Filippo Rinaldi, Giacomo Capitani, Lorenzo Bonicelli, Donato Crisostomi, Federico Bolelli, Elisa Ficarra, Emanuele Rodolà, Simone Calderara, and Angelo Porrello. Update your transformer to the latest release: Re-basin of task vectors. In *ICML*, 2025.

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 *ICLR*, 2026a.

Filippo Rinaldi, Aniello Panariello, Giacomo Salici, Angelo Porrello, and Simone Calderara. Transporting task vectors across different architectures without training. In *ICML*, 2026b.

Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. Natural language understanding with the quora question pairs dataset. *arXiv preprint arXiv:1907.01041*, 2019.

Shivalika Singh, Freddie Vargus, Daniel D'souza, Börje F. Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura O'Mahony, et al. Aya dataset: An open-access collection for multilingual instruction tuning. In *ACL*, 2024.

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

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. In *NeurIPS*, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

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

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

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

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with oss-instruct. In *ICML*, 2024.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, 2018.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo-Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *CVPR*, 2022.

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

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

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *ICML*, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Sujoy Basu, Yi Luan, Denny Zhou, Siddhartha Brahma, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Luca Zhou, Daniele Solombrino, Donato Crisostomi, Maria Sofia Bucarelli, Giuseppe Alessio D'Inverno, Fabrizio Silvestri, and Emanuele Rodolà. On task vectors and gradients. In *Proceedings of UniReps: the Third Edition of the Workshop on Unifying Representations in Neural Models*, 2026.

TABLE OF CONTENTS

We provide the following items in this Appendix:

- (A) Pseudocode of BIFA
- (B) Bilinear structure under practical optimizers
- (C) Alignment mapping formulations
- (D) Additional analysis of factor-space alignment
- (E) Analysis of depth matching strategy
- (F) Experimental details
- (G) Additional experiments
- (H) Computational cost analysis
- (I) Limitations
- (J) AI use statement

A PSEUDOCODE OF BIFA

Algorithm 1 BIFA: Training-Free Task-Vector Transfer

Require: Source model $\boldsymbol{\theta}_{A,\mathrm{pre}}, \boldsymbol{\theta}_{A,\mathrm{ft}}$, target model $\boldsymbol{\theta}_{B,\mathrm{pre}}$, calibration dataset $\mathcal{D}$

1: **for** each layer $l \in \{1, \ldots, L\}$ **do**

2: Compute source task-vector: $\boldsymbol{\tau}_A^{(l)} \leftarrow \boldsymbol{\theta}_{A,\mathrm{ft}}^{(l)} - \boldsymbol{\theta}_{A,\mathrm{pre}}^{(l)}$

3: Obtain input activations $\boldsymbol{X}_{A,\mathrm{pre}}^{(l)}, \boldsymbol{X}_{B,\mathrm{pre}}^{(l)}$ from $\mathcal{D}$

4: Obtain output-side gradients $\boldsymbol{G}_{A,\mathrm{pre}}^{(l)}, \boldsymbol{G}_{B,\mathrm{pre}}^{(l)}$ from $\mathcal{D}$

5: Interpolate sequence lengths if needed and flatten batch/token dimensions.

6: Estimate input-side alignment:

$$\boldsymbol{R}_{\mathrm{in}}^{(l)} \leftarrow \mathcal{A}\Big(\boldsymbol{X}_{A,\mathrm{pre}}^{(l)}, \boldsymbol{X}_{B,\mathrm{pre}}^{(l)}\Big)$$

7: Estimate output-side alignment:

$$\boldsymbol{R}_{\mathrm{out}}^{(l)} \leftarrow \mathcal{A}\Big(\boldsymbol{G}_{A,\mathrm{pre}}^{(l)}, \boldsymbol{G}_{B,\mathrm{pre}}^{(l)}\Big)$$

8: Transfer task-vector as in Equation 6:

$$\hat{\boldsymbol{\tau}}_B^{(l)} \leftarrow (\boldsymbol{R}_{\mathrm{out}}^{(l)})^\top \boldsymbol{\tau}_A^{(l)} \boldsymbol{R}_{\mathrm{in}}^{(l)}$$

9: **end for**

10: **return** $\boldsymbol{\theta}_{B,\mathrm{pre}} + \hat{\boldsymbol{\tau}}_B$

Here, $\mathcal{A}$ denotes a linear alignment method that estimates a map $\boldsymbol{R}$ such that $\boldsymbol{X}_A \boldsymbol{R} \approx \boldsymbol{X}_B$; we consider a CCA-based mapping, ridge regression, and orthogonal Procrustes.

B BILINEAR STRUCTURE UNDER PRACTICAL OPTIMIZERS

Equation 4 gives the exact module-level factorization of the instantaneous gradient,

$$B_t := \nabla_{\boldsymbol{\theta}_t} \mathcal{L}_t = \boldsymbol{g}_t^\top \boldsymbol{x}_t \tag{A}$$

where t denotes the optimization step. This bilinear gradient structure follows directly from the chain rule and is independent of the optimizer used to update the parameters.

Equation 5 then shows how these instantaneous bilinear gradients accumulate into a task vector under vanilla gradient descent:

$$\boldsymbol{\tau} = -\sum_{t=1}^{T} \eta_t B_t. \tag{B}$$

Practical optimizers generally modify this simple accumulation through operations such as temporal aggregation, adaptive rescaling or preconditioning, and regularization. Thus, while the instantaneous factorization in Equation 4 remains exact, Equation 5 should be viewed as the vanilla-gradient-descent form of task-vector formation rather than an exact characterization of an arbitrary optimizer trajectory.

Momentum-based gradient accumulation. Many practical optimizers incorporate momentum-based accumulation of past gradients before applying subsequent optimizer-specific transformations. At optimization step t, this accumulation can be written generally as

$$M_t = \sum_{s=1}^{t} \alpha_{t,s} B_s, \tag{C}$$

where $s \leq t$ indexes past optimization steps and $\alpha_{t,s}$ denotes the corresponding temporal weight.

Substituting $B_s = \boldsymbol{g}_s^\top \boldsymbol{x}_s$ into Equation C gives

$$M_t = \sum_{s=1}^{t} \alpha_{t,s} \boldsymbol{g}_s^\top \boldsymbol{x}_s. \tag{D}$$

Consequently, the resulting task vector becomes

$$\tau = -\sum_{t=1}^{T} \eta_t M_t = -\sum_{s=1}^{T} w_s g_s^\top x_s, \qquad w_s := \sum_{t=s}^{T} \eta_t \alpha_{t,s}. \tag{E}$$

Thus, momentum-based updates preserve the accumulated bilinear structure, while changing only the effective temporal weighting of each gradient contribution.

Adaptive preconditioning. Adaptive optimizers may additionally apply coordinate-dependent transformations to the temporally accumulated gradient. Because the precise form of these transformations depends on the optimizer, we do not claim that Equation 5 provides a universal characterization of the complete update trajectory of arbitrary adaptive optimizers. Instead, we explicitly examine AdamW (Loshchilov & Hutter, 2019), which is used throughout our experiments, to determine whether such optimizer-induced transformations substantially alter the separable structure exploited by BIFA.

The AdamW update of $\boldsymbol{\theta}_t$ can be written as

$$\Delta\boldsymbol{\theta}_t = -\eta_t \left(P_t \odot \hat{M}_t \right) - \eta_t \lambda \boldsymbol{\theta}_t, \qquad \hat{M}_t = \sum_{s=0}^{t} \alpha_{t,s} B_s, \tag{F}$$

where η_t is the learning rate, $\hat{M}_t$ denotes the momentum-aggregated first-moment term, $\alpha_{t,s}$ represents the corresponding temporal weights,

$$P_t = \left(\sqrt{\hat{V}_t} + \epsilon \right)^{-1} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$$

is the element-wise adaptive preconditioner derived from the second-moment estimate, and λ is the decoupled weight-decay coefficient. Accordingly, Equation 5 captures the direct accumulation of the underlying raw bilinear gradients, whereas the actual AdamW task vector additionally incorporates temporal weighting, coordinate-wise adaptive preconditioning, and a separate weight-decay component.

To characterize the effect of adaptive preconditioning on the bilinear structure, consider a rank decomposition of P_t,

$$P_t = \sum_{j=1}^{R_t} r_{t,j} c_{t,j}^\top, \tag{G}$$

where $R_t = \operatorname{rank}(P_t)$, with $r_{t,j} \in \mathbb{R}^{d_{\text{out}}}$ and $c_{t,j} \in \mathbb{R}^{d_{\text{in}}}$. For an individual rank-one component, the following identity holds:

$$\left(rc^\top \right) \odot \left(\boldsymbol{g}^\top \boldsymbol{x} \right) = \left(\boldsymbol{g} \operatorname{Diag}(r) \right)^\top \left(\boldsymbol{x} \operatorname{Diag}(c) \right). \tag{H}$$

Combining Equations F–H, the adaptively preconditioned gradient can therefore be expressed as

$$P_t \odot \hat{M}_t = \sum_{s=0}^{t} \sum_{j=1}^{R_t} \alpha_{t,s} \left(\boldsymbol{g}_s \operatorname{Diag}(r_{t,j})\right)^\top \left(\boldsymbol{x}_s \operatorname{Diag}(c_{t,j})\right). \tag{I}$$

Hence, for AdamW, adaptive preconditioning does not remove the underlying activation–gradient factorization. Rather, it generally expands each temporally accumulated contribution into a sum of separable bilinear components, with independently rescaled backward-gradient- and activation-side factors. The decoupled weight-decay term, in contrast, is a separate contribution and is not represented by this gradient factorization.

Dominant separable component of the AdamW task vector. The AdamW task vector thus comprises the adaptively preconditioned gradient accumulation, which admits the separable decomposition above, and the decoupled weight-decay term, which does not. To assess how much of the resulting task vector is explained by the separable bilinear structure alone, we reconstruct it from the gradient term while further restricting the preconditioner to its dominant rank-one component. Importantly, we do not assume that the actual AdamW preconditioner is low-rank; rather, this deliberately restrictive reconstruction serves as a structural diagnostic.

Specifically, using the optimizer states from the original AdamW trajectory, we retain only the dominant rank-one component of P_t,

$$P_t^{(1)} = r_t c_t^\top. \tag{J}$$

The corresponding preconditioned update becomes

$$P_t^{(1)} \odot \hat{M}_t = \sum_{s=0}^{t} \alpha_{t,s} \left(\boldsymbol{g}_s \operatorname{Diag}(r_t)\right)^\top \left(\boldsymbol{x}_s \operatorname{Diag}(c_t)\right), \tag{K}$$

which retains a two-factor bilinear form with independently rescaled activation- and backward-gradient-side factors.

We then reconstruct a task vector using only these dominant separable components,

$$\tau_{\text{rank1}} = -\sum_t \eta_t \left(P_t^{(1)} \odot \hat{M}_t\right), \tag{L}$$

while excluding both the higher-rank components of the adaptive preconditioner and the decoupled weight-decay term. Thus, this reconstruction is intentionally more restrictive than the actual AdamW trajectory and is used only to assess how much of the resulting task vector can be explained by the dominant separable activation–gradient structure.

Despite this restriction, τ_{rank1} achieves a layer-wise mean cosine similarity of 0.972 with the full task vector produced by AdamW. Moreover, when transferred using the Procrustes variant of BIFA, τ_{rank1} obtains 76.19% average accuracy across eight tasks, closely matching the 76.02% obtained using the full AdamW task vector. These results indicate that the components most relevant to the direction and transfer behavior of the AdamW task vector are already well captured by the dominant separable activation–gradient structure, with the excluded higher-rank and weight-decay contributions playing a minor role in practice.

C ALIGNMENT MAPPING FORMULATIONS

We formulate both input- and output-side alignment as estimating a linear map $\boldsymbol{R}$ between paired source and target signals. Specifically, given $\boldsymbol{Z}_A \in \mathbb{R}^{M \times d_A}$ and $\boldsymbol{Z}_B \in \mathbb{R}^{M \times d_B}$, we seek a mapping $\boldsymbol{R} \in \mathbb{R}^{d_A \times d_B}$ such that

$$\boldsymbol{Z}_A \boldsymbol{R} \approx \boldsymbol{Z}_B. \tag{M}$$

Here, $\boldsymbol{Z}$ denotes either the input activations or the output-side gradients. We consider three linear mapping formulations: a CCA-based mapping, ridge regression, and orthogonal Procrustes.

C.1 CCA-BASED LINEAR MAPPING

Given paired source and target signals $\boldsymbol{Z}_A \in \mathbb{R}^{M \times d_A}$ and $\boldsymbol{Z}_B \in \mathbb{R}^{M \times d_B}$ (e.g., input activations or output-side gradients), we consider a CCA-based linear mapping that emphasizes directions exhibiting strong cross-model correlation. Let $\tilde{\boldsymbol{Z}}_A$ and $\tilde{\boldsymbol{Z}}_B$ denote the centered source and target signals, and define

$$\boldsymbol{C}_{AA} = \tilde{\boldsymbol{Z}}_A^\top \tilde{\boldsymbol{Z}}_A, \qquad \boldsymbol{C}_{BB} = \tilde{\boldsymbol{Z}}_B^\top \tilde{\boldsymbol{Z}}_B, \qquad \boldsymbol{C}_{AB} = \tilde{\boldsymbol{Z}}_A^\top \tilde{\boldsymbol{Z}}_B. \tag{N}$$

CCA whitens the two spaces and performs singular value decomposition on the whitened cross-covariance:

$$\boldsymbol{C}_{AA}^{-1/2} \boldsymbol{C}_{AB} \boldsymbol{C}_{BB}^{-1/2} = \boldsymbol{U} \operatorname{diag}(\boldsymbol{\rho}) \boldsymbol{V}^\top, \tag{O}$$

where the entries of $\boldsymbol{\rho}$ are the canonical correlations.

Retaining the top-k canonical components, we construct the source-to-target mapping as

$$\boldsymbol{R}_{\mathrm{C}} = \boldsymbol{C}_{AA}^{-1/2} \boldsymbol{U}_k \operatorname{diag}(\boldsymbol{\rho}_k) \boldsymbol{V}_k^\top \boldsymbol{C}_{BB}^{1/2}. \tag{P}$$

We retain the top-k canonical components, where k is set to a fixed fraction of $\min(d_A, d_B)$ and capped by the available CCA rank.

The resulting transformation whitens the source representation, transfers it through the dominant correlated canonical directions, and maps it back to the target representation space. For input-side alignment, $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$ correspond to the paired input activations, while for output-side alignment they correspond to the paired output gradients. Unlike orthogonal Procrustes, the CCA-based mapping is not constrained to preserve source-side Euclidean geometry; instead, it prioritizes cross-model directions with high canonical correlation.

C.2 RIDGE-REGULARIZED LINEAR MAPPING

Given paired source and target signals $\boldsymbol{Z}_A \in \mathbb{R}^{M \times d_A}$ and $\boldsymbol{Z}_B \in \mathbb{R}^{M \times d_B}$ (e.g., input activations or output-side gradients), ridge regression estimates a regularized linear mapping by solving

$$\boldsymbol{R}_{\mathrm{R}} = \arg\min_{\boldsymbol{R}} \|\boldsymbol{Z}_A \boldsymbol{R} - \boldsymbol{Z}_B\|_F^2 + \lambda \|\boldsymbol{R}\|_F^2, \tag{Q}$$

where $\lambda > 0$ controls the strength of the ridge regularization. The objective admits the closed-form solution

$$\boldsymbol{R}_{\mathrm{R}} = \left(\boldsymbol{Z}_A^\top \boldsymbol{Z}_A + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{Z}_A^\top \boldsymbol{Z}_B. \tag{R}$$

For input-side alignment, $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$ correspond to the paired input activations $\bar{\boldsymbol{X}}_{A,\mathrm{pre}}^{(l)}$ and $\bar{\boldsymbol{X}}_{B,\mathrm{pre}}^{(l)}$, respectively; for output-side alignment, they correspond to the paired gradients $\bar{\boldsymbol{G}}_{A,\mathrm{pre}}^{(l)}$ and $\bar{\boldsymbol{G}}_{B,\mathrm{pre}}^{(l)}$.

Unlike orthogonal Procrustes, ridge regression does not impose a semi-orthogonality constraint on the mapping. Consequently, it permits non-uniform scaling and general linear mixing across feature dimensions, and therefore does not necessarily preserve source-side norms, inner products, or angles. This additional flexibility allows the mapping to more directly minimize the alignment error between paired source and target signals. The ridge penalty regularizes this flexibility by discouraging large mapping coefficients and improving numerical stability when the source signals are ill-conditioned.

C.3 ORTHOGONAL PROCRUSTES

Given paired source and target signals $\boldsymbol{Z}_A \in \mathbb{R}^{M \times d_A}$ and $\boldsymbol{Z}_B \in \mathbb{R}^{M \times d_B}$ (e.g., input activations or output-side gradients), orthogonal Procrustes estimates a linear correspondence while constraining the transformation to be semi-orthogonal. For $d_A \le d_B$ for notational simplicity, we solve

$$\boldsymbol{R}_{\mathrm{P}} = \arg\min_{\boldsymbol{R}} \|\boldsymbol{Z}_A \boldsymbol{R} - \boldsymbol{Z}_B\|_F^2, \qquad \text{s.t.} \quad \boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{I}_{d_A}, \tag{S}$$

where $\boldsymbol{R}_{\mathrm{P}} \in \mathbb{R}^{d_A \times d_B}$. Here, $\boldsymbol{Z}$ denotes either the input activations or the output-side gradients, yielding $\boldsymbol{R}_{\mathrm{in}}^{(l)}$ and $\boldsymbol{R}_{\mathrm{out}}^{(l)}$, respectively.

The problem admits a closed-form solution. Let the compact singular value decomposition of the cross-covariance be

$$\boldsymbol{Z}_A^\top \boldsymbol{Z}_B = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \tag{T}$$

where $\boldsymbol{U} \in \mathbb{R}^{d_A \times d_A}$ and $\boldsymbol{V} \in \mathbb{R}^{d_B \times d_A}$. Then, the Procrustes mapping is given by

$$\boldsymbol{R}_{\mathrm{P}} = \boldsymbol{U}\boldsymbol{V}^\top. \tag{U}$$

Beyond providing a closed-form estimate, the semi-orthogonality constraint preserves the geometry of the source representation under the coordinate transformation. For any source-side vectors $\boldsymbol{z}, \boldsymbol{z}_i, \boldsymbol{z}_j \in \mathbb{R}^{1\times d_A}$, the mapping $\boldsymbol{z} \mapsto \boldsymbol{z}\boldsymbol{R}_{\mathrm{P}}$ preserves the Euclidean norm:

$$\|\boldsymbol{z}\boldsymbol{R}_{\mathrm{P}}\|_2^2 = \boldsymbol{z}\boldsymbol{R}_{\mathrm{P}}\boldsymbol{R}_{\mathrm{P}}^\top \boldsymbol{z}^\top = \boldsymbol{z}\boldsymbol{z}^\top = \|\boldsymbol{z}\|_2^2, \tag{V}$$

as well as pairwise inner products:

$$\langle \boldsymbol{z}_i\boldsymbol{R}_{\mathrm{P}}, \boldsymbol{z}_j\boldsymbol{R}_{\mathrm{P}} \rangle = \boldsymbol{z}_i\boldsymbol{R}_{\mathrm{P}}\boldsymbol{R}_{\mathrm{P}}^\top \boldsymbol{z}_j^\top = \boldsymbol{z}_i\boldsymbol{z}_j^\top = \langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle. \tag{W}$$

Consequently, pairwise distances and cosine similarities are also preserved. Thus, Procrustes aligns the source signals to the target coordinate space without introducing anisotropic scaling or deformation of their source-side geometry.

D Additional analysis of factor-space alignment

D.1 Linear correspondence in the input-activation space

We analyze the correspondence between the input activation spaces of different pre-trained models using activation cosine similarity and Singular Vector Canonical Correlation Analysis (SVCCA) (Raghu et al., 2017). Cosine similarity directly compares corresponding activation vectors in their original coordinates, whereas SVCCA measures the similarity between their dominant representation subspaces up to linear transformations. Therefore, cosine similarity reflects coordinate-wise alignment, while SVCCA captures linear correspondence between the underlying representation spaces.

We first consider same-width ViT-B/16 models with different pre-training distributions. As shown in Figure A, the models exhibit high SVCCA but extremely low activation cosine similarity before alignment. The high SVCCA indicates that the two activation spaces exhibit strong linear correspondence, despite the lack of direct coordinate-wise alignment reflected by the low cosine similarity. This suggests that corresponding representations can be related through a linear transformation, providing motivation for estimating an explicit input-side alignment map between the two models.

We further examine whether such linear correspondence persists when the source and target models have different hidden widths. Since raw cosine similarity is only directly defined when the feature dimensions match, we use SVCCA for this cross-width analysis. Figure B reports layer-wise SVCCA between a ViT-B/16 source model and a wider ViT-B/16-plus target model. Despite the width mismatch, SVCCA remains consistently high across corresponding layers, indicating substantial shared representational structure that is accessible through linear projections even when their hidden dimensions differ. Together, these SVCCA results support substantial linear correspondence between the input activation spaces across both same-width and different-width models, motivating our use of linear input-side alignment.

D.2 Linear correspondence in the output-gradient space

We further examine whether the output-gradient spaces of the source and target models exhibit a correspondence that can be captured through linear transformations at the pre-trained initialization. Similar to the input-side analysis, we first measure their layer-wise similarity using SVCCA. As shown in Figure C (left), SVCCA remains high across corresponding layers, indicating that the dominant structures of the two gradient spaces can be substantially related through linear projections.

Unlike input activations, however, output gradients are local and sample-dependent signals whose exact values depend on the model predictions and the corresponding loss landscape. Therefore, reproducing individual target gradient vectors point-wise is not the primary objective of the output-side

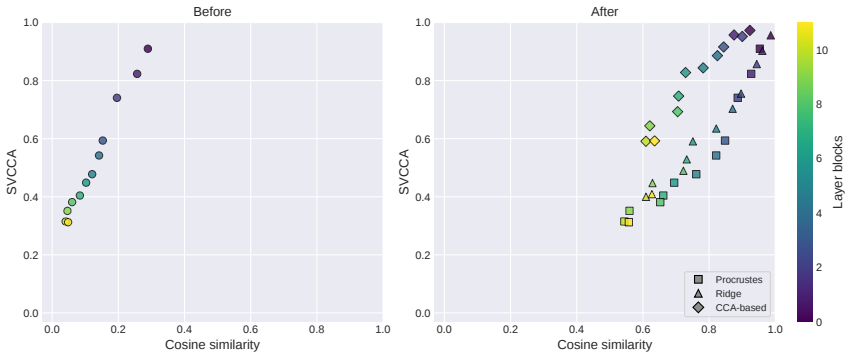

Figure A: **Representational similarity before and after input-side alignment.** We compare SVCCA and cosine similarity between pre-trained activations of ViT-B/16 models with different pre-training distributions (A: DataComp-XL → B: LAION-2B). (Left) before alignment. (Right) After linear alignment using different estimators. Results are averaged across 8 vision tasks and sub-layers within each residual block, using $N = 100$ calibration samples to estimate the alignment map.

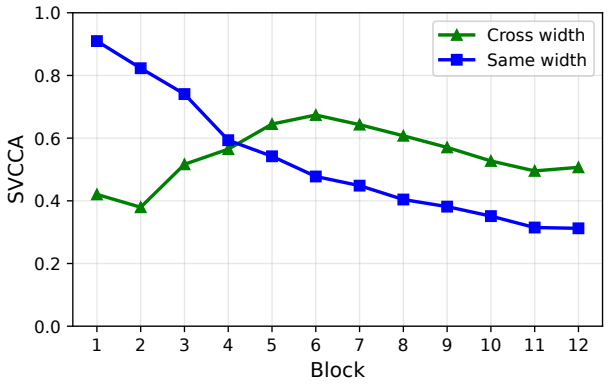

Figure B: **Linear correspondence between pre-trained activation spaces.** Layer-wise SVCCA for same-width ViT-B/16 pairs and a cross-width ViT-B/16 → ViT-B/16-plus pair. Results are averaged across 8 vision tasks and sub-layers within each residual block, using $N = 100$ calibration samples.

alignment. As discussed in Section 4.3, the role of $\boldsymbol{R}_{\text{out}}^{(l)}$ is to estimate the cross-model coordinate correspondence that transports the accumulated source task vector $\tau_A^{(l)}$ into the target output coordinates. We therefore examine whether the estimated mapping improves the correspondence between the dominant gradient subspaces of the two models, rather than requiring point-wise agreement between individual gradients.

Specifically, after applying the estimated output-side mapping, we denote the aligned source gradients as $\tilde{\boldsymbol{G}}_A^{(l)} = \bar{\boldsymbol{G}}_{A,\text{pre}}^{(l)} \boldsymbol{R}_{\text{out}}^{(l)}$. Let $\boldsymbol{V}_{A,k}^{(l)}$ and $\boldsymbol{V}_{B,k}^{(l)}$ denote orthonormal bases for the leading k right-singular subspaces of $\tilde{\boldsymbol{G}}_A^{(l)}$ and $\bar{\boldsymbol{G}}_{B,\text{pre}}^{(l)}$, respectively. We quantify their correspondence using the normalized subspace overlap

$$S_k^{(l)} = \frac{1}{k} \left\| \left(\boldsymbol{V}_{A,k}^{(l)} \right)^\top \boldsymbol{V}_{B,k}^{(l)} \right\|_F^2, \tag{X}$$

which is equivalent to the average squared cosine of the principal angles between the two subspaces and ranges from 0 to 1, with larger values indicating stronger subspace correspondence.

We use the leading 10% of gradient directions for this analysis. Although only a small fraction of the gradient dimensions is retained, these directions account for 89.5% of the total target-gradient energy on average across the eight vision tasks, indicating that they capture the dominant structure of the output-gradient space. As shown in Figure C (right), the dominant-subspace overlap substantially

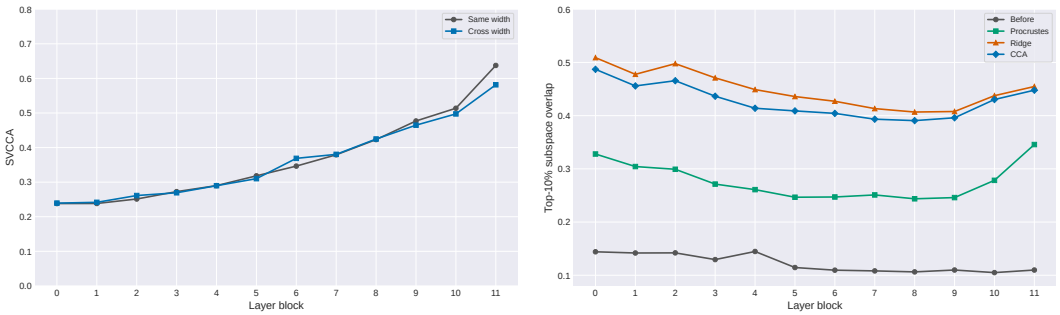

Figure C: **Output-gradient correspondence before and after output-side alignment.** (Left) SVCCA between source and target output gradients at the pre-trained initialization. (Right) Dominant-subspace overlap before and after linear alignment using different estimators. Results are averaged across 8 vision tasks and sub-layers within each residual block, using $N = 100$ calibration samples. The subspace overlap uses the leading 10% gradient directions.

increases after applying the estimated output-side mapping on held-out samples. This improvement shows that the estimated mapping captures non-trivial cross-model correspondence in the dominant gradient subspaces beyond the calibration samples.

D.3 Why does Procrustes transfer better?

Although more flexible linear estimators can achieve stronger local alignment between source and target signals, Procrustes consistently yields the best downstream task-vector transfer. We investigate this difference from the perspective of the geometry of the transferred task vector.

For the Procrustes variant, the input- and output-side maps satisfy the semi-orthogonality constraints

$$\boldsymbol{R}_{\text{in}}\boldsymbol{R}_{\text{in}}^{\top} = \boldsymbol{I}, \qquad \boldsymbol{R}_{\text{out}}\boldsymbol{R}_{\text{out}}^{\top} = \boldsymbol{I}. \tag{Y}$$

Consequently, applying $\hat{\tau}_B = \boldsymbol{R}_{\text{out}}^{\top}\tau_A\boldsymbol{R}_{\text{in}}$ preserves the non-zero singular values of the source task vector. In contrast, ridge regression and the CCA-based mapping impose no such geometric constraint and can therefore rescale different singular directions non-uniformly.

To examine whether this distinction is relevant for transfer, we compare the singular-value spectrum of each transferred task vector with that of the corresponding directly fine-tuned target task vector. For a task vector τ with singular values $\{\sigma_i\}$, we define its normalized singular-energy spectrum as

$$p_i(\tau) = \frac{\sigma_i^2}{\sum_j \sigma_j^2}, \tag{Z}$$

and measure the Jensen–Shannon divergence between the transferred and target spectra.

As shown in Figure D, the task vectors transferred with Procrustes exhibit spectral energy distributions substantially closer to those of the target task vectors, whereas ridge regression and the CCA-based mapping introduce considerably larger spectral distortions. Since Procrustes preserves the source task-vector norm and non-zero singular values by construction, these results suggest that retaining this geometric structure is beneficial for cross-model transfer, providing a possible explanation for its superior downstream performance.

E Analysis of depth matching strategy

Our depth-matching experiments employ an intentionally simple relative-depth rule based on normalized layer positions. To examine whether this rule captures meaningful cross-model correspondence, we compare the resulting layer pairs with those obtained by directly maximizing representational similarity. We use CKA (Kornblith et al., 2019) to compare candidate layer pairs, as it measures whether different layers encode similar representational structure over the same inputs, making it suitable for analyzing cross-model depth correspondence.

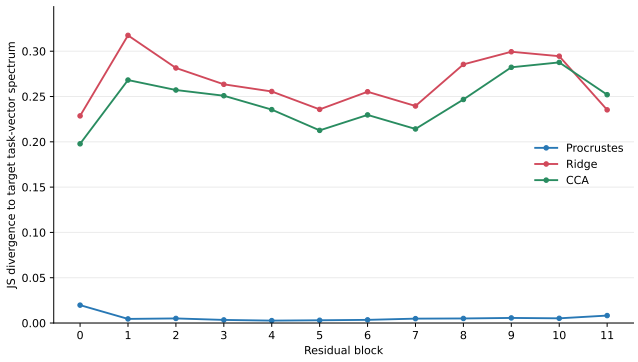

Figure D: **Spectral geometry of transferred task vectors.** Jensen–Shannon divergence between the normalized singular-energy spectra of transferred task vectors and the corresponding target fine-tuned task vectors, averaged across 8 vision tasks. Lower is better.

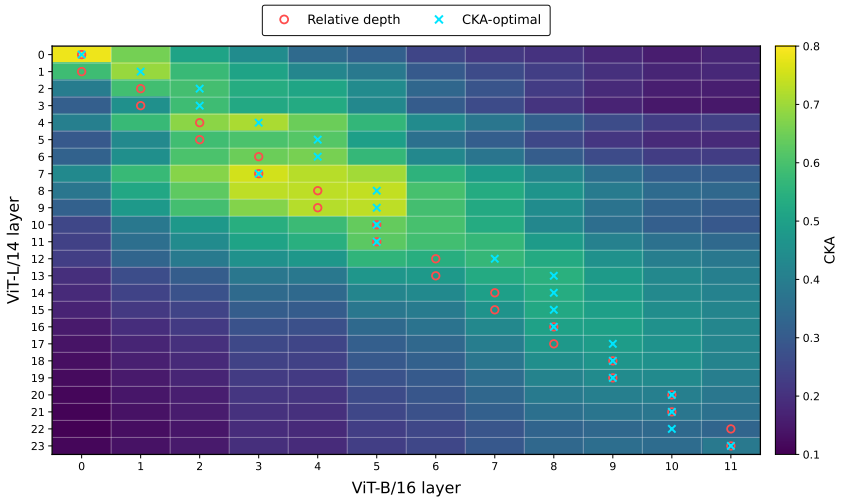

Figure E: **Representation correspondence under relative-depth and CKA-based layer matching.** Layer-wise CKA similarity between ViT-L/14 and ViT-B/16. Each cell shows the CKA similarity between a pair of layers, while red circles indicate the correspondences selected by our relative-depth rule and blue crosses indicate the CKA-optimal matches for each ViT-L/14 layer. CKA values are averaged across eight vision tasks using $N = 100$ calibration samples.

As shown in Figure E, the layer-wise CKA similarity pattern exhibits a clear correspondence across model depth, with high-similarity regions progressively shifting toward deeper ViT-B/16 layers as the ViT-L/14 depth increases. The layer pairs selected by relative-depth matching largely follow these high-similarity regions and frequently coincide with or lie near the CKA-optimal correspondences. This suggests that normalized layer position provides a simple approximation to cross-model representational correspondence.

We further compare relative-depth matching with three adaptive alternatives: (i) *CKA-based* matching, which selects layer pairs with the highest CKA similarity; (ii) *SVCCA-based* matching, which analogously selects pairs according to SVCCA similarity; and (iii) *task-sensitivity-based* matching. For the latter, each layer is represented by a vector of per-sample average gradient magnitudes over the calibration set, and source and target layers are matched according to the similarity between their sensitivity profiles. We use Procrustes alignment for all variants so that only the layer-matching strategy differs.

As shown in Table A, the representation-based CKA and SVCCA strategies provide only modest improvements over relative-depth matching, while task-sensitivity-based matching performs less

Table A: **Comparison of depth-matching strategies.** Task-vector transfer from ViT-B/16 to ViT-L/14, both pre-trained on DataComp-XL. We compare relative-depth matching with adaptive alternatives based on CKA, SVCCA, and task sensitivity. All depth-matching variants use Procrustes alignment. N denotes the number of calibration samples used to estimate the alignment maps.

Model	N	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG
θ_B *zero-shot*	–	64.11	58.91	67.67	72.60	66.96	93.06	86.63	73.77	72.96
THESEUS	100	78.85	65.55	72.96	74.28	67.87	93.16	91.45	74.12	77.28
CKA-based	100	87.33	78.70	86.51	83.22	71.59	93.66	98.57	74.84	84.30
SVCCA-based	100	89.59	79.44	85.91	82.53	71.27	93.69	98.54	74.79	84.47
Task-sensitivity-based	100	89.92	77.32	82.79	79.90	69.89	93.64	98.11	74.75	83.29
Relative-depth (BIFA-P)	100	87.25	78.63	86.10	82.33	71.17	93.80	98.27	74.70	84.03

favorably. Importantly, these adaptive strategies require additional calibration-based computation to evaluate pairwise layer similarities or sensitivity profiles. In contrast, relative-depth matching determines the correspondence directly from layer indices and therefore introduces no additional matching computation, while achieving performance close to the representation-based alternatives. Together with the correspondence analysis in Figure E, these results indicate that normalized layer position provides a simple and effective approximation to representation-based layer correspondence.

F Experimental details

F.1 Training details

Following the fine-tuning protocol established by Task Arithmetic (Ilharco et al., 2023), we obtain each task vector by fine-tuning the pre-trained model for 2,000 training steps with a batch size of 128 and a learning rate of 1×10^{-5}. We use the AdamW optimizer with a weight decay of 0.1, together with cosine annealing and 200 warm-up steps. The dataset has been split into training and validation sets, comprising 90% and 10% of the data, respectively. For LLM experiments, we construct task vectors from the domain-specialized models released by MergeBench (He et al., 2025).

Compute resources. Fine-tuning, performance evaluation, and computational cost measurements for ViT-B/16-scale experiments were conducted on NVIDIA GeForce RTX 3090 GPUs. Larger-scale experiments, including those involving ViT-L/14 and LLMs, were conducted on NVIDIA RTX PRO 6000 Blackwell GPUs.

F.2 Dataset details

We evaluate our method on diverse vision and language-model benchmarks to assess transferability across modalities and task types. The vision datasets cover remote sensing, traffic sign recognition, digit recognition, texture classification, fine-grained recognition, and scene classification. For LLM evaluation, we use four domains from MergeBench (He et al., 2025): instruction following, mathematics, multilingual understanding, and coding. In each domain, the source task vector is obtained from a model specialized on the corresponding post-training dataset, while transfer performance is measured on separate evaluation benchmarks, ensuring that the specialization and evaluation data are distinct. We additionally evaluate T5-based transfer on five GLUE tasks as a conventional NLP setting.

Vision Datasets.

- **EuroSAT (Helber et al., 2019).** A remote sensing dataset based on Sentinel-2 satellite imagery, consisting of approximately 27,000 images across 10 classes.
- **GTSRB (Stallkamp et al., 2011).** The German Traffic Sign Recognition Benchmark, a widely used dataset for traffic sign classification, consisting of 51,839 images across 43 categories.

- **SVHN (Netzer et al., 2011).** A real-world digit classification dataset containing house number digits extracted from Google Street View images, with 73,257 images across 10 classes (0–9).
- **RESISC45 (Cheng et al., 2017).** A scene classification dataset based on satellite and aerial imagery, comprising 31,500 images across 45 classes.
- **DTD (Cimpoi et al., 2014).** The Describable Textures Dataset, designed for recognizing and categorizing visual texture patterns, comprising 5,640 images annotated into 47 distinct classes.
- **Cars (Krause et al., 2013).** A fine-grained image classification dataset for distinguishing car models, manufacturers, and years, consisting of 16,185 images across 196 classes.
- **MNIST (Lecun et al., 1998).** The Modified National Institute of Standards and Technology dataset, a benchmark for handwritten digit recognition, composed of grayscale images of digits from 0 to 9 collected from a variety of writers, totaling 70,000 images across 10 classes.
- **SUN397 (Xiao et al., 2010).** The SUN397 scene recognition dataset, a large-scale benchmark for scene classification, consisting of 108,754 images across 397 classes.

LLM Datasets.

- **Instruction Following.** The source model is specialized on TULU-3 Persona IF (Lambert et al., 2024), consisting of approximately 29.9K instruction-following examples. Transfer performance is evaluated on IFEval (Zhou et al., 2023), which contains 541 prompts designed to assess adherence to verifiable instructions.
- **Mathematics.** For mathematical reasoning, the source model is specialized using DART-Math (Tong et al., 2024) and NuminaMath-TIR (Li et al., 2024), containing approximately 591K and 72.4K examples, respectively. Evaluation is performed on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), comprising 1,320 and 5,000 mathematical reasoning problems, respectively.
- **Multilingual Understanding.** The source model is specialized on the Aya dataset (Singh et al., 2024), using approximately 5.94K human-curated examples spanning multiple languages. We evaluate multilingual capability on M_MMLU, M_ARC, and M_HellaSwag (Lai et al., 2023), which contain approximately 60K, 10.34K, and 37.35K examples, respectively.
- **Coding.** For code generation, the source model is specialized on Magicoder (Wei et al., 2024), containing approximately 110K coding examples generated from real-world code snippets. Evaluation is conducted on HumanEval (Chen et al., 2021), which contains 164 programming problems for assessing functional code generation.

NLP Datasets.

- **MNLI (Williams et al., 2018).** The Multi-Genre Natural Language Inference dataset, a benchmark for natural language inference across diverse text genres, comprising approximately 433,000 training examples of sentence pairs annotated with entailment, contradiction, or neutral relationships.
- **QNLI (Wang et al., 2019).** A natural language inference dataset consisting of 104,743 training examples of question–sentence pairs annotated to determine whether the sentence contains the answer to the question.
- **RTE (Wang et al., 2019).** The Recognizing Textual Entailment dataset, a benchmark for textual inference, comprising approximately 2,500 training examples of sentence pairs annotated with entailment relationships.
- **QQP (Sharma et al., 2019).** The Quora Question Pairs, a paraphrase identification dataset consisting of approximately 364,000 training examples of question pairs annotated as duplicate or non-duplicate.
- **SNLI (Bowman et al., 2015).** The Stanford Natural Language Inference dataset, a large-scale benchmark for natural language inference, comprising approximately 570,000 examples of sentence pairs annotated with entailment, contradiction, or neutral labels.

Table B: **Robustness to random calibration-sampling seeds.** Task-vector transfer from a ViT-B/16 model *A* (LAION-2B) to a ViT-B/16-plus model *B* (LAION-400M). Results are averaged over five random seeds. N denotes the number of calibration samples used to estimate the Procrustes maps. We report mean ± standard deviation for each N.

Model	N	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG
θ_B *zero-shot*	–	42.51	49.61	37.77	65.95	55.42	84.50	57.02	68.50	57.66
BIFA-P	10	50.90 ± 6.54	54.90 ± 1.59	51.76 ± 6.46	66.38 ± 1.86	56.06 ± 0.87	84.43 ±0.34	79.55 ±3.69	68.61 ±0.10	**64.07** ± **2.68**
BIFA-P	20	70.47 ± 4.43	59.64 ± 1.49	60.66 ± 5.71	68.24 ± 0.53	56.66 ± 1.02	84.21 ± 0.59	86.76 ±2.80	68.70 ±0.18	**69.41** ±**2.09**
BIFA-P	50	79.81 ± 1.32	63.84 ± 4.16	69.52 ± 7.24	72.61 ± 0.88	58.57 ± 0.86	84.72 ±0.34	92.61 ±1.14	69.13 ±0.19	**73.85** ±**2.02**
BIFA-P	100	81.74 ± 1.62	67.93 ± 2.93	74.73 ± 4.20	76.81 ± 0.55	61.37 ± 1.36	84.91 ±0.34	93.72 ±1.35	69.41 ±0.16	**76.32** ±**1.56**

F.3 VALIDATION-BASED HYPERPARAMETER TUNING

For validation-tuned baselines, we tune the task-vector scaling coefficient α on the validation split. For THESEUS, we search $\alpha \in \{0.1, 0.2, \ldots, 5.0\}$, while for GradFix and TransFusion we use $\alpha \in \{0.1, 0.2, \ldots, 2.0\}$.

For BIFA-R and BIFA-C, we search $\alpha \in \{0.1, 0.2, \ldots, 2.0\}$ on the validation split. For the ridge mapping, we jointly tune $\lambda \in \{10^{-2}, 10^{-1}, 1, 10, 10^2\}$ and α. For the CCA-based mapping, we jointly select the retained rank from $\{5\%, 10\%, 15\%, 20\%, 25\%\}$ of $\min(d_A, d_B)$ and α. In contrast, BIFA-P requires no validation-based tuning and uses a fixed scaling coefficient of $\alpha = 1$.

G ADDITIONAL EXPERIMENTS

G.1 ROBUSTNESS TO CALIBRATION SAMPLING

Robustness to random calibration seeds. We conduct an ablation study investigating the sensitivity of BIFA to two factors: the calibration size $N \in \{10, 20, 50, 100\}$ and the random sampling seed. Specifically, we consider transfer from a ViT-B/16 model pre-trained on LAION-2B to a ViT-B/16-plus model pre-trained on LAION-400M, using Procrustes alignment throughout. As shown in Table B, BIFA achieves higher average performance than the zero-shot baseline even with only $N = 10$ calibration samples. As the calibration size increases, the average performance improves while the variance across random seeds generally decreases. These results show that BIFA is robust to calibration-sample variation, particularly as more calibration data become available.

Sensitivity to calibration set composition. To isolate the effect of calibration-set composition, we fix the calibration budget to $N = 100$, use Procrustes alignment throughout, and vary only the strategy used to construct the calibration set. In addition to our default *Random* sampling, which uniformly draws examples from the training split, we consider six alternative strategies. (i) *Class-balanced* samples examples as evenly as possible across classes to improve label coverage. (ii) *Half-class* restricts calibration samples to half of the classes. (iii) *Centroid-near* computes class centroids in the pre-trained feature space and selects examples closest to their corresponding centroids, whereas (iv) *Centroid-far* selects those farthest from the centroids. (v) *Gaussian noise* evaluates robustness to noisy calibration inputs, while (vi) *Contrast shift* introduces a low-level appearance perturbation by modifying image contrast.

As shown in Table C, BIFA is broadly robust to the construction and quality of the calibration set. Our default *Random* strategy performs comparably to carefully curated alternatives such as *Class-balanced* and *Centroid-near*, indicating that carefully selected calibration samples are not required for effective coordinate alignment. While some variation is observed when class coverage is reduced (*Half-class*) or when samples are selected farther from class centroids (*Centroid-far*), BIFA

Table C: **Effect of calibration sampling strategy.** Task-vector transfer from a ViT-B/16 model *A* (LAION-2B) to a ViT-B/16-plus model *B* (LAION-400M). We compare different ways of constructing the calibration set used to estimate the Procrustes maps, using $N = 100$ calibration samples for all strategies.

Strategy	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG
θ_B *zero-shot*	42.51	49.61	37.77	65.95	55.42	84.50	57.02	68.50	57.66
THESEUS	63.22	56.14	50.68	68.68	55.90	84.65	73.07	68.89	65.15
Class-balanced	82.48	68.31	71.78	79.60	62.02	85.11	93.97	69.30	76.57
Half-class	63.88	60.91	72.29	73.93	61.32	85.33	75.55	69.30	70.31
Centroid-near	78.55	72.28	71.71	78.92	63.51	85.81	93.79	69.74	76.78
Centroid-far	68.66	60.97	69.60	72.30	59.89	84.80	84.52	69.37	71.26
Gaussian noise	57.77	62.09	60.81	73.50	62.12	84.64	94.27	69.17	70.55
Contrast shift	76.11	68.60	72.71	77.07	61.48	85.05	94.07	69.57	75.58
Random (BIFA-P)	79.70	66.11	78.63	77.49	61.86	85.16	92.93	69.31	76.40

Table D: **Width scaling using the same calibration-data budget.** Task-vector transfer from a ViT-B/16 model A (LAION-2B) to a wider ViT-B/16-plus model B (LAION-400M). K denotes the number of calibration examples per class used to estimate the alignment maps and to train the few-shot fine-tuning baselines. θ_B^{1st} denotes target models fine-tuned on the same K-shot calibration set for 1 step. ΔAcc denotes the accuracy gain over the corresponding θ_B^{1st} baseline under the same K-shot setting.

Model	K	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG (ΔAcc)
θ_B *zero-shot*	–	42.51	49.61	37.77	65.95	55.42	84.50	57.02	68.50	57.66 (-)
θ_B *fine-tune*	–	99.00	99.19	97.80	96.46	82.55	90.11	99.81	78.29	92.90 (-)
θ_A *fine-tune*	–	94.67	99.03	97.89	96.81	83.46	92.10	99.72	79.35	92.88 (-)
$\theta_B + \tau_A^{\text{padded}}$	–	46.41	38.21	27.27	62.57	53.56	82.00	51.70	68.29	53.75 (-)
θ_B^{1st}	1	60.96	56.47	41.09	68.68	56.22	78.67	81.76	69.11	64.12 (+0.00)
THESEUS	1	44.55	53.79	45.87	68.04	56.11	84.46	57.01	69.10	59.86 (-4.26)
BIFA-P	1	66.88	62.26	65.41	73.20	59.20	85.46	85.95	70.40	**71.09 (+6.97)**
θ_B^{1st}	2	62.48	58.77	56.95	71.06	56.28	77.01	83.74	69.17	66.93 (+0.00)
THESEUS	2	44.55	53.99	48.38	68.33	56.27	84.50	55.48	69.08	60.07 (-6.86)
BIFA-P	2	71.33	67.91	73.92	77.84	62.28	86.58	91.87	70.57	**75.28 (+8.35)**
θ_B^{1st}	5	62.52	59.83	55.49	72.83	56.60	75.70	85.03	69.34	67.16 (+0.00)
THESEUS	5	49.66	55.01	50.59	68.88	56.70	84.69	59.25	68.91	61.71 (-5.45)
BIFA-C	5	68.67	60.60	56.45	69.16	54.36	84.57	79.95	68.88	67.83 (+0.67)
BIFA-R	5	70.48	60.60	72.61	69.25	57.98	84.74	88.43	69.03	71.64 (+4.48)
BIFA-P	5	79.25	74.05	77.51	81.85	64.25	86.94	93.50	70.96	**78.53 (+11.37)**

remains effective across these calibration settings. Similarly, the method is relatively robust to input perturbations, with *Contrast shift* yielding performance close to the default setting and *Gaussian noise* showing some additional sensitivity. Notably, across all evaluated calibration conditions, BIFA remains above THESEUS under its standard random-calibration setting. Overall, these results suggest that BIFA does not require carefully curated calibration data and support the use of simple random sampling in our main experiments.

G.2 Extended results on width scaling

Few-shot fine-tuning under the same calibration-data budget. We further compare BIFA with a few-shot target fine-tuning baseline using the same calibration data budget and a matched forward–backward computational budget. Table D reports the results, where K denotes the number of calibration samples per class. θ_B^{1st} denotes the target model fine-tuned on the same K-shot calibration set for one step, matching the forward–backward budget used by BIFA to estimate the alignment

Table E: **Reverse width scaling (wide → narrow).** Task-vector transfer from a ViT-B/16-plus model *A* (LAION-400M) to a narrower ViT-B/16 model *B* (LAION-2B). N denotes the number of calibration samples used to estimate the alignment maps. ΔAcc denotes accuracy gain over model B zero-shot baseline.

Model	N	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG (ΔAcc)
θ_B *zero-shot*	–	50.15	48.35	50.05	68.22	55.96	88.56	65.79	70.44	62.19 (+0.00)
θ_B *fine-tune*	–	94.67	99.03	97.89	96.81	83.46	92.10	99.72	79.35	92.88 (+30.69)
θ_A *fine-tune*	–	99.00	99.19	97.80	96.46	82.55	90.11	99.81	78.29	92.90 (+30.71)
$\theta_B + \tau_A^{\text{crop}}$	–	33.11	36.18	19.73	45.10	55.53	74.13	36.79	57.18	44.72 (-17.47)
THESEUS	10	50.07	54.94	61.70	68.52	55.59	87.60	71.86	70.51	65.10 (+2.91)
BIFA-P	10	39.44	56.74	68.59	68.43	57.61	85.60	88.14	70.07	**66.83 (+4.64)**
THESEUS	20	49.41	57.40	65.25	70.48	56.28	88.05	73.54	70.14	66.32 (+4.13)
BIFA-P	20	59.00	67.07	80.43	71.11	58.24	86.05	70.09	70.50	**70.31 (+8.12)**
THESEUS	50	51.56	59.47	65.95	72.25	56.12	87.97	79.55	70.21	67.89 (+5.70)
BIFA-P	50	77.15	72.60	87.12	74.90	62.02	86.27	87.15	70.80	**77.25 (+15.06)**
THESEUS	100	51.85	60.97	69.22	73.10	57.82	87.86	78.54	70.34	68.71 (+6.52)
BIFA-C	100	68.59	67.35	71.56	71.60	53.94	88.29	88.39	70.09	72.48 (+10.29)
BIFA-R	100	69.22	67.60	84.71	72.83	61.49	88.76	90.09	71.14	75.73 (+13.54)
BIFA-P	100	88.33	78.39	89.10	79.32	65.05	86.79	93.19	71.21	**81.42 (+19.23)**

maps. Compared with θ_B^{1st}, which directly updates the target parameters under the matched budget, BIFA achieves substantially higher performance across different values of K. This result suggests that using the calibration set to estimate bilinear coordinate alignment provides a more effective use of limited supervision than directly applying a small number of gradient updates to the target model. Importantly, BIFA obtains these gains without iterative target-model fine-tuning, highlighting its practical utility as a training-free alternative to few-shot target adaptation.

Transfer from wide to narrow models. We further evaluate whether task-vector transfer remains effective when transferring from a wide model to a narrow model. Specifically, we consider transferring the task vector τ_A obtained from a ViT-B/16-plus model *A* pre-trained on LAION-400M to a ViT-B/16 target model *B* pre-trained on LAION-2B. As shown in Table E, an approach that resolves the dimensional mismatch by simply cropping the task vector ($\theta_B + \tau_A^{\text{crop}}$) results in performance that falls even below the zero-shot baseline of the target model. This indicates that there exists an intrinsic mismatch between the parameter spaces of structurally different models, which cannot be eliminated by straightforward dimensionality reduction, and suggests that reverse-direction transfer is infeasible without explicit alignment. In contrast, BIFA consistently outperforms the zero-shot baseline with only a small number of calibration samples, and steadily approaches the fine-tuned performance as the number of samples N increases. This demonstrates that our proposed method operates robustly regardless of the transfer direction.

Generalization to different pre-training sources. We further analyze task-vector transfer when the source and target models are pre-trained on different data sources, while retaining the narrow-to-wide transfer setting. In this setup, the source model *A* is a ViT-B/16 pre-trained on DataComp-XL, whereas the target model *B* is a ViT-B/16-plus pre-trained on LAION-400M, introducing pre-training mismatch in addition to architectural discrepancy. As shown in Table F, transfer via zero-padding again results in a significant performance drop, indicating that discrepancies in both model width and pre-training data distributions further aggravate the misalignment between parameter spaces. BIFA achieves robust transfer performance even in this challenging scenario. With only a small number of calibration samples, it consistently surpasses the zero-shot baseline, and its performance improves steadily as N increases. Moreover, it significantly outperforms THESEUS across all values of N, demonstrating that our bilinear alignment effectively handles both architectural and data-induced discrepancies.

Table F: **Width scaling (narrow → wide) under different pre-training sources.** Task-vector transfer from a ViT-B/16 model *A* (DataComp-XL) to a wider ViT-B/16-plus model *B* (LAION-400M). N denotes the number of calibration samples used to estimate the alignment maps. ΔAcc denotes accuracy gain over model B zero-shot baseline.

Model	N	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG (ΔAcc)
θ_B *zero-shot*	–	42.51	49.61	37.77	65.95	55.42	84.50	57.02	68.50	57.66 (+0.00)
θ_B *fine-tune*	–	99.00	99.19	97.80	96.46	82.55	90.11	99.81	78.29	92.90 (+35.24)
θ_A *fine-tune*	–	98.96	99.06	98.02	96.71	84.25	92.50	99.77	79.25	93.57 (+35.91)
$\theta_B + \tau_A^{\text{padded}}$	–	38.07	44.17	31.03	60.95	54.79	82.83	15.64	68.00	49.44 (-8.22)
THESEUS	10	42.22	51.72	38.71	67.38	55.53	84.11	56.94	68.55	58.15 (+0.49)
BIFA-P	10	46.48	54.40	31.90	66.08	57.02	84.59	75.08	68.58	**60.52 (+2.86)**
THESEUS	20	44.48	52.51	41.20	68.17	56.17	84.16	57.43	68.44	59.07 (+1.41)
BIFA-P	20	66.93	56.67	46.88	68.03	58.35	84.60	81.10	68.81	**66.42 (+8.76)**
THESEUS	50	47.37	52.82	41.80	68.75	55.74	84.40	61.93	68.73	60.19 (+2.53)
BIFA-P	50	80.22	58.59	66.69	73.46	60.43	84.90	90.21	69.09	**72.95 (+15.29)**
THESEUS	100	48.15	52.83	43.42	69.11	55.85	84.39	61.67	68.83	60.53 (+2.87)
BIFA-C	100	63.15	55.99	53.90	67.95	51.81	83.92	77.28	68.58	65.32 (+7.66)
BIFA-R	100	65.44	54.16	53.57	69.68	58.51	84.69	74.13	68.74	66.12 (+8.46)
BIFA-P	100	83.41	65.92	73.28	78.05	62.66	85.23	90.16	69.46	**76.02 (+18.36)**

Table G: **Performance of linear probing for task-vector transfer.** Source: T5-3B, Target: T5-Large.

Encoder	N	MNLI	QNLI	RTE	QQP	SNLI	AVG (ΔAcc)
θ_A *fine-tune*	All	91.22	95.32	84.83	91.05	91.29	90.74 (-)
θ_B *fine-tune*	All	85.28	93.59	77.25	89.57	88.51	86.84 (-)
θ_A	50	39.34	75.58	58.12	68.62	53.21	58.97 (+4.27)
θ_B	50	38.74	69.50	57.40	62.56	45.28	54.70 (+0.00)
THESEUS	50	46.12	72.65	66.42	63.18	54.61	60.59 (+5.89)
BIFA-P	50	46.32	76.71	67.87	83.40	60.80	**67.02 (+12.32)**
θ_A	100	60.13	78.46	59.93	68.12	58.32	64.99 (+2.69)
θ_B	100	58.48	75.43	61.37	62.44	53.78	62.30 (+0.00)
THESEUS	100	60.93	78.34	66.78	63.19	56.38	65.12 (+2.82)
BIFA-C	100	65.87	81.97	64.62	63.18	57.59	66.65 (+4.35)
BIFA-R	100	66.86	79.83	64.98	63.18	58.49	66.67 (+4.37)
BIFA-P	100	61.94	79.33	67.50	84.31	58.43	**70.30 (+8.00)**

Additional NLP Results. To assess generalization to language models, we evaluate BIFA using T5 architectures (Raffel et al., 2020), with T5-3B as the source model *A*, and T5-Large as the target model *B*. Following the protocol of THESEUS (Rinaldi et al., 2026b), we transfer the encoder task vector τ_A and evaluate the transferred encoder via linear probing (Alain & Bengio, 2016). We vary the number of calibration samples N used to estimate the alignment maps and to train the probe, ensuring an identical data budget for all compared methods. As shown in Table G, BIFA demonstrates robust improvements in linear probing accuracy across all tested benchmarks and data regimes. We observe that the overall performance scales up as the number of samples N increases, highlighting that the effectiveness of BIFA successfully extends beyond the computer vision domain to NLP tasks.

G.3 Extended results on width and depth scaling

Transfer from deeper to shallower models. We further examine a more challenging transfer scenario in which the source model is deeper and wider than the target model. Specifically, we transfer task vectors from ViT-L/14 to ViT-B/16, both pre-trained on the DataComp-XL distribution, thereby introducing mismatches in both depth and hidden width. As shown in Table H, despite this structural discrepancy, BIFA maintains strong transfer performance, indicating that the learned task-specific updates can be effectively translated into the coordinate system of a smaller target model. This result highlights the effectiveness of our depth-matching strategy, which provides reliable layer correspondences when transferring across models with different numbers of transformer blocks.

Table H: **Deeper to shallower model.** Task-vector transfer from a ViT-L/14 model to ViT-B/16 model, both pretrained on DataComp-XL. N denotes the number of calibration samples used to estimate the alignment maps. ΔAcc denotes accuracy gain over model B zero-shot baseline.

Model	N	EuroSAT	GTSRB	SVHN	RESISC45	DTD	Cars	MNIST	SUN397	AVG (ΔAcc)
θ_B *zero-shot*	–	45.81	55.44	62.71	68.60	58.03	88.77	76.43	70.03	65.72 (+0.00)
θ_B *fine-tune*	–	98.96	99.06	98.02	96.71	84.25	92.50	99.77	79.25	93.57 (+27.85)
θ_A *fine-tune*	–	99.33	99.29	98.21	97.58	86.11	95.22	99.74	82.40	94.74 (+29.02)
$\theta_B + \tau_A^{\text{crop}}$	–	43.33	54.71	63.63	68.44	57.97	88.83	75.91	69.71	65.31 (-0.41)
THESEUS	100	52.66	62.55	71.05	74.33	67.81	88.58	82.96	69.97	71.23 (+5.51)
BIFA-C	100	77.63	73.38	66.26	72.62	60.11	87.48	90.24	69.93	74.71 (+8.99)
BIFA-R	100	53.11	72.43	76.55	71.54	60.11	89.09	93.53	69.34	73.21 (+7.49)
BIFA-P	100	83.85	79.80	79.85	76.55	63.29	88.33	97.44	70.12	**79.90 (+14.18)**

Table I: **Runtime and memory comparison.** Practical preparation cost and average transfer accuracy in the identical-architecture ViT-B/16 setting. All methods use $N = 100$ randomly sampled calibration examples with a batch size of 32.

Method	Statistics extraction	End-to-end preparation	Peak GPU memory	Avg. Acc.
GradFix	–	22.08s	5.92G	71.78
THESEUS	21.20s	75.24s	2.28G	69.03
BIFA-C	67.47s	165.71s	6.02G	72.68
BIFA-R	32.88s	41.49s	5.73G	73.11
BIFA-P	24.28s	78.41s	5.73G	79.66

H COMPUTATIONAL COST ANALYSIS

We measure the practical preparation cost of BIFA and the training-free transfer baselines in the identical-architecture ViT-B/16 setting. For all methods, we use $N = 100$ randomly sampled calibration examples with a batch size of 32 and report statistics-extraction time, end-to-end preparation time, peak GPU memory, and average transfer accuracy.

As shown in Table I, THESEUS and BIFA-P both estimate alignment maps from calibration statistics, while BIFA-P additionally requires backward passes to collect task-conditioned output-side gradients. This results in higher one-time statistics-extraction cost than THESEUS, while the overall preparation time remains of the same order. BIFA-R requires additional second-order statistics for ridge estimation, e.g., the input-side Gram matrix in addition to the cross-covariance, and therefore spends more time on statistics extraction. However, its subsequent ridge-map estimation is substantially cheaper in our implementation than the Procrustes solve, reducing the total preparation time. BIFA-C incurs the largest cost because CCA requires estimating covariance statistics for both sides together with their cross-covariance, followed by whitening and decomposition, leading to higher costs in both statistics extraction and overall preparation. In contrast, GradFix avoids alignment-map estimation and therefore incurs substantially lower preparation time, but uses comparable peak GPU memory to BIFA and achieves lower transfer accuracy. Overall, these results characterize the practical trade-off of BIFA: the additional backward signal introduces moderate one-time preparation overhead while providing substantially stronger transfer performance.

I LIMITATIONS

BIFA assumes that the source and target models have sufficiently compatible internal structures such that their layer-wise activation and gradient spaces can be related through linear coordinate maps. This assumption covers many practical model-update scenarios, where models preserve a common high-level backbone while differing in width, depth, pre-training, or training recipes. Accordingly, our experiments span such mismatches within Transformer-based architectures, including cross-

family LLM transfer with different tokenizers and architectural configurations. However, transfer across fundamentally different computational paradigms, such as Transformers and CNNs, or across substantially different modality-specific pipelines, may require more general alignment mechanisms. Moreover, the correspondence between model spaces may not always be fully captured by linear transformations. Consistent with this limitation, although BIFA substantially narrows the gap to direct target fine-tuning, a residual gap remains, suggesting that linear coordinate alignment may not capture all task-specific information induced by optimization. Extending task-vector transfer to nonlinear or structurally heterogeneous alignment is therefore an important direction for future work.

BIFA also requires a small task-relevant calibration set to estimate the alignment maps and compute output-side gradients. Although calibration quality and representativeness can affect transfer performance, this requirement is substantially weaker than target-model fine-tuning because no target parameters are updated. Our experiments further show that simple random sampling performs comparably to more carefully curated strategies and that even small calibration budgets yield consistent improvements over zero-shot target models and existing transfer baselines. Nevertheless, BIFA still requires access to task-relevant data; developing data-free or synthetic alternatives for estimating alignment maps remains an interesting direction for future work.

J AI USE STATEMENT

In accordance with the ICLR 2027 policy on the use of Large Language Models (LLMs), we disclose that LLMs were used to aid or polish the writing of this paper. Specifically, LLMs were used for minor grammar corrections, sentence restructuring, and improvements in clarity, readability, and conciseness. LLMs were not used to generate the core scientific ideas, methodology, experimental design, results, analyses, or conclusions. All scientific and technical contributions presented in this work were developed and verified by the human authors.