FROM PARAMETER REBASIN TO FUNCTIONAL TRANSFER ACROSS HETEROGENEOUS TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting pretrained Transformers to downstream tasks produces task-specific updates that are expensive to reproduce for every model variant. Reusing these updates across different depths and widths is challenging because their parameters have incompatible shapes and do not share a canonical coordinate system. We introduce ARIADNE, a training-free method that transfers task-specific behavior across heterogeneous Transformers without resizing the source model or transporting its parameters. Rather than treating the source task vector, *i.e.*, the difference in parameter space between the fine-tuned and the base models, as the object to move, ARIADNE characterizes fine-tuning by the activation changes it induces at matched blocks. First, it aligns these effects to a target base through local Procrustes maps. Then, it directly synthesizes a native target update by closed-form ridge regression. Across Transformer expansion, reduction and language-model transfer, ARIADNE improves target-model performance while substantially reducing the time and memory required to construct an update relative to previous cross-architecture rebasin approaches. These results show that local functional effects provide an efficient alternative to parameter transport.

1 INTRODUCTION

Foundation models (Radford et al., 2021; Yang et al., 2025) are increasingly reused rather than trained from scratch. A single pretrained checkpoint can give rise to many fine-tuned task variants, represented as task vectors (Ilharco et al., 2023), each defined in the parameterization of its source architecture. When a deployment requires a different backbone, the learned task-specific behavior cannot be directly transferred because source and target models inhabit different parameter spaces. Valuable model families consequently remain tied to the architecture on which they were trained.

This coupling is limiting in practice. For example, a practitioner may have an instruction-tuned assistant at one Transformer depth and width, but later need to deploy the same instruction-following behavior in a smaller model under a tighter serving budget, or in a newer and larger backbone. Similarly, reducing depth can lower latency and memory use, while expansion can make an existing model family compatible with a larger target architecture. Retraining every task-specific model in the new architecture may be unavailable or prohibitively expensive; it also repeats substantial training compute and its associated energy use. The practical goal is therefore to reuse task-specific behavior across heterogeneous Transformer architectures without repeating fine-tuning.

Existing work tackles this challenge by either adapting pretrained architectures or transporting task-specific updates across models. Function-preserving transformations such as Net2Net (Chen et al., 2015) and Network Morphism (Wei et al., 2016) expand trained networks, while Transformer-specific growth and compression methods often rely on parameter transformations or continued training (Gesmundo & Maile, 2023; Ding et al., 2023; Chen et al., 2022; Fan et al., 2020; Sajjad et al., 2023; Xu et al., 2020). A natural alternative is to align source and target parameterizations and then transport the source task vector, as in cross-architecture rebasin methods such as THESEUS (Rinaldi et al., 2026b) and BiCo (Son et al., 2026). This formulation treats the source task vector, namely the parameter change caused by fine-tuning, as the object to move.

With reference to these transport-based methods (Rinaldi et al., 2026b; Son et al., 2026), a sufficiently accurate correspondence in parameter space or at the global functional level is therefore

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.

required. This typically entails assumptions on the backbone and relative depth of the models, together with architectural resampling operations such as block copying, interpolation, or deletion to match their depths. These choices are inherently heuristic and do not guarantee that the transported task vector remains informative. More elaborate resampling procedures can mitigate this issue (Kangaslahti et al., 2026), but require gradient access. In this work, we therefore seek transfer techniques that are simple, efficient, and robust to mismatches between model parameterizations.

To reach these goals, ARIADNE adopts a different transfer paradigm. Rather than transporting the source task vector, it transfers the observable effect of source fine-tuning, that is the change in the transformer block activation induced by fine-tuning. For each matched pair of Transformer blocks, ARIADNE aligns source and target base residual-stream states with a local Procrustes map and uses it to express the source fine-tuning activation displacement in target coordinates. Then, a closed-form ridge regression computes an additive update to one residual-writing target component that matches this desired effect. The base-model relation is therefore only a local coordinate bridge, not the optimization target. The resulting update is native to the target backbone and requires neither a target fine-tuned checkpoint, source resizing, parameter transport, nor inference-time interventions. The name of the method recalls Ariadne's thread: the activation displacement guides construction of an update through a differently structured target model. Our contributions are:

- We formulate transfer across Transformer depths and widths as direct reproduction of fine-tuning-induced functional effects, rather than parameter rebasin.
- We introduce a training-free local Procrustes and ridge construction that synthesizes target-space task vectors from a source base and fine-tuned counterpart.
- We evaluate direct functional transfer across Vision Transformer expansion and reduction, and across language-model variants.

2 RELATED WORK

Rebasin and cross-architecture transfer. Rebasin methods align parameterizations through permutation symmetries, enabling task-vector reuse across pretrained checkpoints (Ainsworth et al., 2023; Rinaldi et al., 2025). GradFix (Rinaldi et al., 2026a) transfers updates between models with the same architecture by masking source task vectors using target gradient signs from a small labeled calibration set. THESEUS (Rinaldi et al., 2026b) and BiCo (Son et al., 2026) combine architectural alignment with Procrustes-based task-vector transport across different widths and depths. THESEUS aligns input and output activations, while BiCo replaces output activations with output-side gradients. ARIADNE instead fits a native target update to the aligned activation difference induced by source fine-tuning, without transporting source tensors.

Representation alignment and activation matching. Model stitching connects subnetworks through mappings between intermediate representations (Bansal et al., 2021). Relative representations use similarities to shared anchors (Moschella et al., 2023), while latent-space translation estimates algebraic maps between pretrained feature spaces (Maiorca et al., 2023). Representation metrics further formalize comparisons under transformations of hidden coordinates (Williams et al., 2021). Activation matching also provides an objective for constructing model weights. RegMean fits merged linear layers by minimizing output discrepancies with the original models (Jin et al., 2023); PLeaS combines partial permutation matching with layer-wise least squares to reconstruct features at a chosen model size (Nasery et al., 2025).

Distillation and network transformation. Knowledge distillation transfers knowledge from a teacher model to a student by training the latter to reproduce the teacher's predictions (Hinton et al., 2015). This allows learned behavior to be reused across models of different sizes without requiring a correspondence between their parameters. TinyBERT extends this supervision to Transformer representations at both general and task-specific stages (Jiao et al., 2020). Architectural transformations offer another route to model reuse: Net2Net (Chen et al., 2015) and Network Morphism (Wei et al., 2016) preserve network functions during architectural changes, while Transformer growth and compression methods often use continued training or layer removal (Gesmundo & Maile, 2023; Ding et al., 2023; Chen et al., 2022; Fan et al., 2020; Sajjad et al., 2023; Xu et al., 2020). Boomerang Distillation reinserts teacher blocks into a distilled student to obtain intermediate model sizes (Kangaslahti et al., 2026).

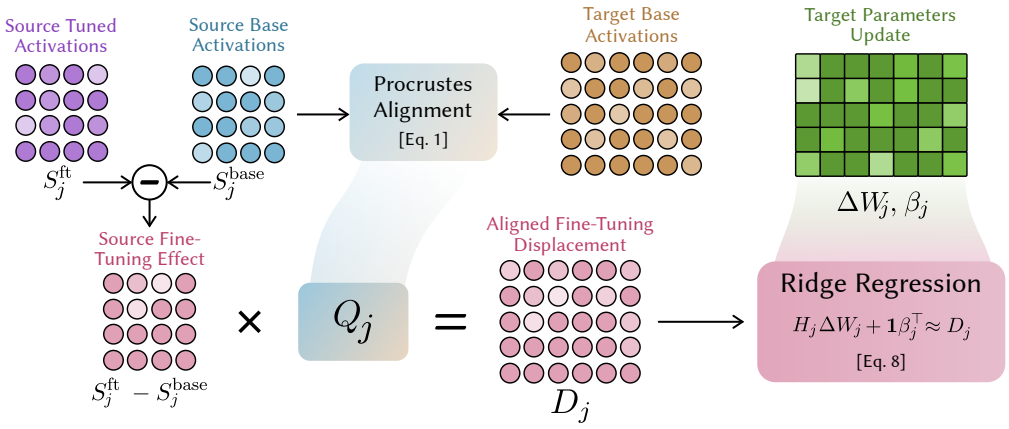

Figure 1: **ARIADNE transfers the functional effect of source fine-tuning to the target model in activation space.** Source ● and target ● base activations are aligned using a component-wise Procrustes transformation Q_j. The source fine-tuning effect ● is mapped into target coordinates to obtain the desired activation shift D_j, giving the aligned fine-tuning displacement. Ridge regression converts this desired shift into target-native parameter updates ■.

3 METHOD

ARIADNE, depicted in Fig. 1, transfers task-specific behavior directly to a native target backbone. Let $\theta_{\text{s}}^{\text{base}}$ and $\theta_{\text{s}}^{\text{ft}}$ be a source base checkpoint and its fine-tuned counterpart, with source task vector $\tau_{\text{s}} = \theta_{\text{s}}^{\text{ft}} - \theta_{\text{s}}^{\text{base}}$. Given an untouched target base $\theta_{\text{t}}^{\text{base}}$, possibly with different depth, width, or block design, we construct a sparse target update τ_{t} whose selected residual-stream effects match the source fine-tuning effects on a small calibration set. Crucially, τ_{t} is not obtained by transporting τ_{s} into the target parameter space. Instead, ARIADNE synthesizes τ_{t} directly in the target parameterization to reproduce the functional change induced by moving from $\theta_{\text{s}}^{\text{base}}$ to $\theta_{\text{s}}^{\text{ft}}$. The method neither resizes source checkpoints nor seeks functional equivalence of the source and target base models.

3.1 BLOCK-LEVEL FUNCTIONAL CORRESPONDENCE

Prior rebasin methods transfer task vectors in parameter space: given a source task vector, they transform its parameter update and inject it into the target model. ARIADNE moves this problem to activation space. Rather than asking how to transform τ_{s}, we characterize the activation change induced by fine-tuning and express that change in target coordinates.

We use the term *block* to denote a recurring computational unit of the backbone, typically comprising an aggregation of operations such as self-attention and feed-forward transformations, as commonly found in Transformer architectures. When transferring information between source and target architectures, we therefore require a correspondence between their blocks, specifying which source blocks provide the information to be transferred to which target blocks. Let π be a user-specified, possibly partial map from target to source blocks, where $\pi(j)$ identifies the source block whose fine-tuning effect supervises target block j. We use relative-depth pairing, $\pi(j) = \text{round}(j(D_{\text{s}} - 1)/(D_{\text{t}} - 1))$, although structural ancestry or domain knowledge may also define the map. The map π does not need to be one-to-one; indeed, when expansion may reuse source blocks, while reduction may omit some.

For every $j \in \text{dom}(\pi)$, we extract the output of target block j from the pre-trained target model, denoted by T_j^{base}, and the output of the mapped source block $\pi(j)$ from both the pre-trained and fine-tuned source models, denoted by $S_{\pi(j)}^{\text{base}}$ and $S_{\pi(j)}^{\text{ft}}$. We use a small calibration set of n examples to compute these activation banks, which are centered around zero by subtracting their mean. When source and target token grids differ, we additionally interpolate the source activations to the target grid before row-wise comparison, following Rinaldi et al. (2026b).

Algorithm 1 ARIADNE

Require: Source models $\theta_{\mathrm{s}}^{\mathrm{base}}$ and $\theta_{\mathrm{s}}^{\mathrm{ft}}$, target base $\theta_{\mathrm{t}}^{\mathrm{base}}$, calibration set, block map π
Ensure: Target task vector τ_{t}
1: Run the source base, source fine-tuned, and target base models on the calibration set.
2: Collect $S_{\pi(j)}^{\mathrm{base}}$, $S_{\pi(j)}^{\mathrm{ft}}$, T_j^{base}, and H_j for all $j \in \mathrm{dom}(\pi)$.
3: Initialize $\tau_{\mathrm{t}} \leftarrow 0$.
4: **for** each $j \in \mathrm{dom}(\pi)$ **do**
5: Compute the source-to-target alignment Q_j using Eq. (1).
6: Transport the source fine-tuning effect: $D_j \leftarrow (S_{\pi(j)}^{\mathrm{ft}} - S_{\pi(j)}^{\mathrm{base}})Q_j$.
7: Compute the ridge statistics from Eq. (8) and select λ_j using Eq. (10).
8: Solve Eq. (9) for $(\Delta W_j, \beta_j)$ and insert the update into τ_{t}.
9: **end for**
10: **return** τ_{t}

Next, we align the two base activations spaces by solving the rectangular Procrustes problem:

$$Q_j = \arg\min_{R \in \mathcal{P}_{d_{\mathrm{s}}, d_{\mathrm{t}}}} \left\| \widetilde{S}_{\pi(j)}^{\mathrm{base}} R - \widetilde{T}_j^{\mathrm{base}} \right\|_F^2, \qquad Q_j \in \mathbb{R}^{d_{\mathrm{s}} \times d_{\mathrm{t}}}, \tag{1}$$

where $\mathcal{P}_{d_{\mathrm{s}}, d_{\mathrm{t}}}$ denotes the appropriate rectangular Stiefel manifold, *i.e.*, matrices with orthonormal columns when $d_{\mathrm{s}} \geq d_{\mathrm{t}}$ and orthonormal rows otherwise. With the thin SVD

$$Q_j = U_j V_j^\top \qquad \text{where} \qquad \left(\widetilde{S}_{\pi(j)}^{\mathrm{base}}\right)^\top \widetilde{T}_j^{\mathrm{base}} = U_j \Sigma_j V_j^\top, \qquad . \tag{2}$$

We then apply the learned map Q_j to the source fine-tuning effect, defined as the activation displacement $S_{\pi(j)}^{\mathrm{ft}} - S_{\pi(j)}^{\mathrm{base}}$. This expresses the source fine-tuning effect in the target coordinates:

$$D_j = \left(S_{\pi(j)}^{\mathrm{ft}} - S_{\pi(j)}^{\mathrm{base}}\right) Q_j, \qquad D_j \in \mathbb{R}^{n \times d_{\mathrm{t}}}. \tag{3}$$

The quantity D_j is the desired residual-stream displacement after target block j. It plays the role that a transported task vector would play in parameter rebasin, but in activation space: it specifies the effect that the target component must realize in the residual stream. The regression in the next subsection turns this effect into the additive target parameters ΔW_j and β_j that form τ_{t}.

Invariance to source activation coordinates. The columns of a collected source residual-stream matrix are hidden feature coordinates, whose orientation is not intrinsic. Consider an equivalent source representation obtained by applying an orthogonal change of basis $O \in \mathbb{R}^{d_{\mathrm{s}} \times d_{\mathrm{s}}}$ to every source residual vector. The collected base and fine-tuned state matrices then become $S_{\pi(j)}^{\mathrm{base}} O$ and $S_{\pi(j)}^{\mathrm{ft}} O$. If Q_j aligns the original source-base states to the target-base states, a corresponding optimal Procrustes bridge after this change of coordinates is $Q_j' = O^\top Q_j$, since

$$\left(\widetilde{S}_{\pi(j)}^{\mathrm{base}} O\right) Q_j' = \widetilde{S}_{\pi(j)}^{\mathrm{base}} Q_j. \tag{4}$$

The aligned fine-tuning displacement is therefore also unchanged:

$$D_j' = \left(S_{\pi(j)}^{\mathrm{ft}} O - S_{\pi(j)}^{\mathrm{base}} O\right) Q_j' = \left(S_{\pi(j)}^{\mathrm{ft}} - S_{\pi(j)}^{\mathrm{base}}\right) Q_j = D_j. \tag{5}$$

Thus, the desired target effect is independent of the particular orthogonal coordinates used to represent the source hidden states. In contrast, a parameter update changes under such a reparameterization, because the affected source weights must change with the hidden coordinates. Recovering these parameter-coordinate transformations across models with different widths, depths, or block structures is precisely the difficult part of rebasin. ARIADNE instead transfers a local activation effect whose target-space specification is unchanged by this coordinate choice.

Alternative alignment signals: ARIADNE$_\nabla$. Our default constructs Q_j from source and target base residual-stream states. More generally, the alignment can use any paired source–target signal whose source coordinates transform as $X_{\mathrm{s}} \mapsto X_{\mathrm{s}} O$ under an orthogonal change of source basis. Following BiCo, we devise an augmented variant, referred to as ARIADNE$_\nabla$, which constructs Q_j from gradients of the task loss with respect to each block output. These row-vector gradients obey the same transformation rule, so the invariance argument above and the resulting activation displacement D_j remain unchanged. Unlike activation alignment, this label-dependent variant requires backward passes, we therefore treat it as an alternative approach rather than the default method.

3.2 DIRECT COMPONENT REGRESSION

The desired residual-stream displacement D_j specifies how the target representation should change on the calibration inputs, but is not itself deployable: it is an input-dependent activation matrix rather than a target-model parameter update. We instead seek a task vector that can be added once to the target checkpoint and used on new inputs without activation interventions. To this end, we will absorb the effect encoded in D_j into the parameters of a selected component of a block. In our experiments, this component corresponds to `c_proj` for CLIP-based Vision Transformers and `down_proj` for LLMs in the Qwen family, *i.e.*, the final MLP projection before the residual addition.

For each $j \in \mathrm{dom}(\pi)$, let $H_j \in \mathbb{R}^{n \times d_j^{\mathrm{in}}}$ denote the activation matrix entering the selected target component in block j with n samples. Let its original output be $H_j W_j + \mathbf{1} b_j^\top$. We seek an additive update to its existing parameters that compiles the effect encoded in D_j:

$$W_j' = W_j + \Delta W_j, \qquad b_j' = b_j + \beta_j. \tag{6}$$

The resulting change in the component output is therefore

$$\left(H_j W_j' + \mathbf{1} b_j'^{\top}\right) - \left(H_j W_j + \mathbf{1} b_j^\top\right) = H_j \Delta W_j + \mathbf{1} \beta_j^\top. \tag{7}$$

We choose the additive update so that this change matches D_j:

$$\min_{\Delta W_j, \beta_j} \left\| H_j \Delta W_j + \mathbf{1} \beta_j^\top - D_j \right\|_F^2 + \lambda_j \left\| \Delta W_j \right\|_F^2. \tag{8}$$

Centering the input and target banks removes the intercept from the weight solve. Let $\widetilde{H}_j = H_j - \mathbf{1}\mu_{H_j}^\top$ and $\widetilde{D}_j = D_j - \mathbf{1}\mu_{D_j}^\top$, where the means are column vectors. The closed-form ridge solution is

$$\Delta W_j = \left(\widetilde{H}_j^\top \widetilde{H}_j + \lambda_j I\right)^{-1} \widetilde{H}_j^\top \widetilde{D}_j, \qquad \beta_j = \mu_{D_j} - \Delta W_j^\top \mu_{H_j}. \tag{9}$$

Here, H_j is collected from the target base model. Conditional on this fixed input bank, the additive update changes the selected linear component output by exactly $H_j \Delta W_j + \mathbf{1}\beta_j^\top$.

Automatic Ridge Scaling. Considering Eq. (8), rather than tuning λ_j, we set it from the calibration statistics using the empirical-Bayes ridge scale motivated by Srivastava & Kubokawa (2007):

$$\lambda_j = \frac{\mathrm{tr}\left(\widetilde{H}_j^\top \widetilde{H}_j\right)}{n-1}. \tag{10}$$

This scales the regularization with the empirical magnitude of the component inputs while shrinking directions weakly supported by the calibration data. For a component with bias, $\beta_j = \mu_{D_j} - \Delta W_j^\top \mu_{H_j}$. For a bias-free component, we set $\beta_j = 0$ and solve using uncentered H_j and D_j.

Returned task vector. For N selected blocks, we compile one update per block using Eq. (8). These regressions are independent: each H_j is collected from the target base model, no fitted update is applied during optimization, and all updates are assembled only after every regression has been solved. This prevents the target of one regression from depending on the fitted parameters of another.

The resulting updates define a **sparse, compiled task vector** in the native target parameterization, with non-zero entries only for the selected block components introduced above. We denote this compiled task vector by τ_{t}, which can be directly added to $\theta_{\mathrm{t}}^{\mathrm{base}}$.

4 EXPERIMENTS

Models and tasks. We evaluate ARIADNE across vision and language models with heterogeneous architectures and scales. In vision, we transfer task updates between `DataComp-XL` OpenCLIP ViT-B/16 and ViT-L/14 checkpoints in both extension and reduction settings, evaluating performance on the Vision8 benchmark (Ilharco et al., 2023). In language, we consider `Qwen2.5` checkpoints (Yang et al., 2024) at the 0.5B, 1.5B, and 3B parameter scales. We evaluate instruction-following transfer between the 0.5B and 1.5B models in both directions on IFEval (Zhou et al., 2023), mathematical-reasoning transfer from `Qwen2.5-Math` 1.5B to the 0.5B and 3B base models on GSM8K (Cobbe et al., 2021), and refusal-removal transfer between `Qwen2.5-Instruct` 0.5B and 1.5B, evaluated on HarmBench with ARC-Challenge as a capability control.

Table 1: **Task-vector transfer across transformer depth changes.** We report ViT-B/16 → ViT-L/14 and ViT-L/14 → ViT-B/16 for $\mathcal{B} \in \{1, 10\}$ calibration batches. Time is the median phase-total runtime per task, summing activation collection, transformation, and transport.

	No Grad.	$\mathcal{B}$	ESAT	SVHN	GTSRB	RESISC	DTD	Cars	MNIST	SUN	Avg (ΔAcc)	Time [s]	Rel. Time
ViT-B/16 → ViT-L/14 (Extension)													
Reference checkpoints													
Source base	–	–	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95	–	–
Source fine-tuned	–	–	98.68	97.91	99.04	96.12	82.27	93.24	99.73	78.55	93.19	–	–
Target base	–	–	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87 (+0.00)	–	–
THESEUS	✓	1	73.58	68.98	63.55	72.57	66.37	93.02	90.22	73.61	75.24 (+1.37)	1023	76.7
BiCo	✗	1	72.10	70.57	61.67	72.54	66.37	93.02	93.73	73.55	75.44 (+1.58)	933	69.9
ARIADNE	✓	1	82.85	81.59	71.65	74.02	66.63	93.02	96.05	73.58	79.92 (+6.06)	**13**	**1**
ARIADNE$_\nabla$	✗	1	83.58	81.63	70.89	76.17	66.55	93.09	94.83	73.76	**80.06 (+6.20)**	**20**	**1.5**
THESEUS	✓	10	73.13	72.17	62.40	72.93	66.37	93.02	90.98	73.73	75.59 (+1.72)	1919	36.2
BiCo	✗	10	74.44	71.55	67.90	73.35	66.37	93.02	96.11	73.63	77.05 (+3.18)	1998	37.7
ARIADNE	✓	10	87.50	90.93	84.17	79.01	69.48	93.02	98.47	73.77	84.54 (+10.68)	**53**	**1**
ARIADNE$_\nabla$	✗	10	90.78	90.22	87.60	84.96	72.40	93.62	98.24	74.19	**86.50 (+12.63)**	**80**	**1.5**
ViT-L/14 → ViT-B/16 (Reduction)													
Reference checkpoints													
Source base	–	–	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87	–	–
Source fine-tuned	–	–	99.14	98.30	99.20	96.98	85.70	94.93	99.76	82.62	94.58	–	–
Target base	–	–	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95 (+0.00)	–	–
THESEUS	✓	1	55.80	63.49	55.24	68.43	57.45	88.89	76.49	70.01	66.98 (+0.02)	426	90.5
BiCo	✗	1	56.75	62.60	55.89	68.92	57.27	88.89	79.43	69.99	67.47 (+0.52)	423	90.0
ARIADNE	✓	1	69.56	77.98	67.98	69.92	57.45	88.89	94.07	69.99	74.48 (+7.53)	**4**	**1**
ARIADNE$_\nabla$	✗	1	82.67	75.03	64.50	71.62	58.27	88.84	93.82	70.01	**75.60 (+8.64)**	**8**	**1.8**
THESEUS	✓	10	55.84	62.57	56.55	67.81	57.45	88.89	76.49	69.94	66.94 (−0.01)	826	38.8
BiCo	✗	10	59.38	62.54	56.43	70.34	57.51	88.89	79.39	70.25	68.09 (+1.14)	825	38.8
ARIADNE	✓	10	79.05	87.60	76.92	72.08	59.22	88.89	97.83	70.06	78.96 (+12.00)	**21**	**1**
ARIADNE$_\nabla$	✗	10	90.62	86.35	84.45	78.43	61.64	88.89	97.68	70.27	**82.29 (+15.34)**	**38**	**1.8**

Baselines and protocol. We compare against the parameter-transport methods THESEUS and BiCo in both directions. Our default ARIADNE uses activation-based Procrustes alignment and therefore requires neither labels nor gradients; ARIADNE$_\nabla$ replaces this alignment signal with task-loss gradients for a supervision-matched comparison with BiCo. We denote by B the calibration budget, measured as the number of batches used to estimate the alignment and fit the target update, and use the same budget across methods. Task-vector and merge multipliers are selected on validation data, with test data reserved for final evaluation. The supplementary material reports additional results across calibration budgets, ridge choices, and resource measurements.

4.1 SINGLE-TASK VISION TRANSFER

Functional transfer succeeds in both directions. The central result in Tab. 1 is a simultaneous improvement in transfer accuracy and construction efficiency. Remarkably, with only $B = 1$ calibration batch, ARIADNE already surpasses the best baseline using $B = 10$ batches in both extension and reduction. Increasing the calibration budget further strengthens transfer, with ARIADNE improving over the target base by 10.68 and 12.00 points in extension and reduction, respectively. These gains are obtained while reducing fitting time by well over an order of magnitude: at the larger calibration budget, ARIADNE requires only 53 s for extension and 21 s for reduction, compared with roughly 32 min and 14 min for the parameter-transport baselines. The reported timing includes activation collection, transformation, and transport, and therefore covers the full measured fitting pipeline. As shown in Tab. 10, most of the runtime gap arises from the transformation stage, while Tab. 9 shows that ARIADNE also requires substantially less peak host memory. The larger accuracy margin under reduction is particularly informative: ARIADNE cannot rely on additional target capacity or duplicated source computation, yet still recovers a substantial part of the source adaptation in a model with half as many blocks.

Alignment signal and calibration budget. The choice of alignment signal exposes a practical trade-off between supervision and transfer quality. At very small calibration budgets, gradient alignment provides a marginal advantage over activation alignment, but its benefit grows with additional labeled calibration data, as shown by the full calibration-budget sweep in Tab. 8. Thus, when labels are available and the extra cost of backward passes can be accommodated, ARIADNE$_\nabla$ provides the stronger option, whereas activation-based ARIADNE remains competitive while requiring

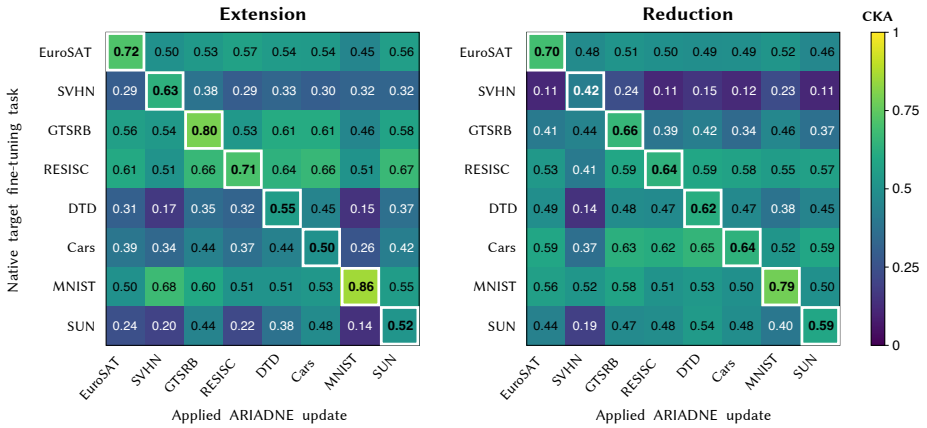

Figure 2: **Task specificity of the transferred effect.** Debiased linear CKA between final-block CLS changes induced by native target fine-tuning (rows) and transported source fine-tuning (columns), for extension (left) and reduction (right). Updates use $\mathcal{B} = 10$.

neither labels nor gradients, making it a lighter default for unlabeled or resource-constrained calibration. More calibration generally improves both variants, though with diminishing returns. As detailed in Tabs. 5 and 6, increasing the calibration budget from one to ten batches yields substantial gains for activation-based ARIADNE, whereas further increases provide only modest improvements. Thus, most of the benefit is already captured with a modest calibration budget, making it a practical compromise between construction cost and the reliability of the resulting update.

Transferred updates produce task-specific representation changes. Accuracy alone does not show whether the transferred update preserves task-specific structure. We therefore measure each update's effect as the change in the final-block CLS representation relative to the target base. Using CKA, we compare these changes with those produced by native target fine-tuning on each of the eight tasks. In Fig. 2, diagonal entries correspond to matching tasks, whereas off-diagonal entries compare different tasks. Each native fine-tuning effect is generally most similar to the update transferred from the same task, indicating that ARIADNE preserves task-specific representational changes rather than inducing a generic perturbation.

What makes transfer work? Table 2 changes one part of the default configuration at a time. *Random* Q_j replaces the learned Procrustes bridge with a random map, reducing averages essentially to the corresponding target-base performance. *Ridge-regression* Q_j instead learns an unconstrained ridge map between the same base activations. It recovers most of the loss, but remains behind Procrustes, indicating that the orthogonality constraint is useful rather than merely convenient. *Endpoint target* asks the target component to reproduce the aligned absolute source fine-tuned activation instead of only the fine-tuning-induced displacement (Tab. 11). Its weaker performance supports transferring what fine-tuning *changes*, while letting the target realize the displacement using the representation it already has, rather than directly reproducing source fine-tuned activations. Finally, we set the target-update regularizer to zero, but the resulting system is ill-conditioned: limited calibration and correlated features can leave the feature Gram matrix with small or zero eigenvalues, making the unregularized solve unstable. The empirical-Bayes scale avoids this failure without a sweep and remains a good heuristic relative to the fixed coefficients tested in either direction, as detailed in Tab. 12. The remaining rows alter only the block map π. *Reversed pairing* flips the relative-depth order, matching shallow target blocks to deep source blocks and vice versa. *Pairing* move every default match to the following or preceding source block, respectively. Both yield a drop in accuracy, with the reversed pairing being sensibly worse. The correspondence prior can therefore be coarse and locally shifted, but it must respect the network's overall computational order.

4.2 LANGUAGE-MODEL TRANSFER

Table 3 reports transfer results, covering instruction following, mathematical reasoning, and safety alignment. In the *Extension* setting, ARIADNE outperforms both baselines across all tasks. Notably, on IFEval, the transferred model even surpasses the source fine-tuned model, indicating that the

Table 2: **Component ablations.** ARIADNE ablations at $\mathcal{B} = 10$.

Variant	ESAT	SVHN	GTSRB	RESISC	DTD	Cars	MNIST	SUN	Avg (ΔAcc)
ViT-B/16 → ViT-L/14 (Extension)									
ARIADNE	87.98	91.02	83.26	78.87	68.68	93.02	98.50	73.73	**84.38** (+0.00)
Alignment and update components									
Random Q_j	72.35	67.73	59.43	72.56	66.08	93.13	86.66	73.55	73.93 (−10.45)
Ridge-regression Q_j	87.61	89.10	80.54	77.04	68.62	93.02	95.70	73.57	83.15 (−1.23)
Endpoint target	77.65	74.80	61.20	72.54	66.37	93.02	90.77	73.55	76.24 (−8.14)
Block-pairing policy π									
Reversed pairing	82.96	71.45	64.73	73.76	67.38	92.99	91.88	73.53	77.34 (−7.05)
Pairing $+1$	87.01	90.31	82.80	78.87	68.50	93.02	98.36	73.72	84.07 (−0.31)
Pairing -1	83.62	88.53	77.56	75.89	67.55	93.02	97.46	73.60	82.15 (−2.23)
ViT-L/14 → ViT-B/16 (Reduction)									
ARIADNE	79.30	88.76	75.92	72.36	60.22	88.89	98.23	70.01	**79.21** (+0.00)
Alignment and update components									
Random Q_j	55.84	62.57	55.02	68.62	57.45	88.89	76.49	69.37	66.78 (−12.43)
Ridge-regression Q_j	77.24	83.67	75.37	72.10	59.34	89.04	85.21	70.18	76.52 (−2.69)
Endpoint target	63.54	84.68	75.98	70.49	57.45	88.89	93.34	70.01	75.55 (−3.66)
Block-pairing policy π									
Reversed pairing	65.80	73.71	58.79	69.52	58.04	88.89	86.88	70.01	71.46 (−7.76)
Pairing $+1$	79.34	88.12	75.46	72.54	59.75	88.88	98.20	70.04	79.04 (−0.17)
Pairing -1	74.32	88.71	75.18	71.06	60.11	88.89	98.18	70.02	78.31 (−0.91)

Table 3: **ARIADNE for instruction tuning, mathematical reasoning and safety.** We evaluate transfer on IFEval ($\mathcal{B} = 100$), GSM8K ($\mathcal{B} = 140$) and safety HarmBench ($\mathcal{B} = 100$).

Method	IFEval			GSM8K			HarmBench		Time [s]	Rel. Time
	Inst. Strict	Prompt Strict	Avg.	Strict	Flex.	Avg.	Non-refusal	Cap. Check		
Extension										
	0.5B→1.5B			**1.5B→3B**			**0.5B Inst→1.5B Inst**			
Source base	23.41	15.62	19.52	61.40	62.02	61.70	54.17	–	–	–
Source fine-tuned	37.72	26.01	31.87	76.24	76.41	76.33	99.58	–	–	–
Target base	29.51	21.23	25.37	71.42	76.38	73.90	27.92	38.65	–	–
THESEUS	31.48	33.80	32.64 (+7.27)	72.24	73.61	72.93 (−0.97)	27.50 (−0.42)	38.05	8702	6.65
BiCo	36.41	27.93	32.17 (+6.80)	70.80	75.81	73.31 (−0.59)	35.00 (+7.08)	**38.65**	9060	6.93
ARIADNE	**42.59**	**44.44**	**43.52** (+18.15)	**73.30**	**76.80**	**75.05** (+1.15)	**67.92** (+40.00)	35.92	**1308**	**1**
Reduction										
	1.5B→0.5B			**1.5B→0.5B**			**1.5B Inst→0.5B Inst**			
Source base	29.51	21.23	25.37	61.40	62.02	61.70	27.92	–	–	–
Source fine-tuned	40.56	47.09	43.83	76.24	76.41	76.33	93.33	–	–	–
Target base	23.41	15.62	19.52	32.83	33.61	33.22	54.17	28.58	–	–
THESEUS	24.07	26.85	25.46 (+5.94)	32.00	32.40	32.20 (−1.02)	52.08 (−2.09)	**27.99**	3746	7.14
BiCo	25.00	28.70	26.85 (+7.33)	**33.80**	34.21	34.01 (+0.79)	54.58 (+0.41)	27.98	3744	7.14
ARIADNE	**30.09**	**31.02**	**30.56** (+11.04)	33.70	**34.60**	**34.15** (+0.93)	**67.50** (+13.33)	27.30	**525**	**1**

transferred instruction-following behavior can combine with the stronger pre-trained capabilities of the target model. On the IFEval average, ARIADNE improves over the target base by $+18.15$ points, more than twice the gains achieved by other approaches. On GSM8K, ARIADNE is the only method that improves over the target base ($+1.15$), whereas both baselines slightly degrade performance. ARIADNE raises the HarmBench non-refusal rate from 27.92% to 67.92%, while THESEUS fails to transfer the behavior and BiCo achieves only a modest gain ($+7.08$). This improvement comes at a small cost on the capability check (-2.73), suggesting that general capabilities are largely preserved. In the *Reduction* setting, the results are less pronounced, but ARIADNE again achieves the best performance across all settings. While the gain is more noticeable on IFEval, on GSM8K all methods remain within approximately one point of the target base. We attribute this to the smaller target model's limited capacity: its base accuracy is substantially lower than that of the source fine-tuned model, leaving limited room for the transferred reasoning skills to manifest. On HarmBench, ARIADNE is the only method that successfully transfers the abliteration to the smaller model, whereas THESEUS and BiCo leave the target's behavior essentially unchanged.

Table 4: **Merge-then-transport task-vector transfer.** We first merge source task vectors and then transport the resulting update to the target model using Vision8 calibration. All methods use $\mathcal{B} = 10$.

Method	Merge	EuroSAT	SVHN	GTSRB	RESISC	DTD	Cars	MNIST	SUN	Avg (ΔAcc)
ViT-B/16 → ViT-L/14 (Extension)										
Reference checkpoint										
Target zero-shot	–	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87 (+0.00)
THESEUS	TA	76.17	68.39	60.75	71.60	66.78	92.46	89.09	73.56	74.85 (+0.98)
BiCo	TA	70.08	70.46	58.56	72.63	66.73	93.02	90.56	73.66	74.46 (+0.59)
ARIADNE	TA	75.27	73.14	63.46	72.45	64.95	92.75	90.01	73.43	75.68 (+1.82)
THESEUS	TSV	72.92	67.78	58.57	72.56	66.25	93.02	86.60	73.57	73.91 (+0.04)
BiCo	TSV	71.36	69.41	58.85	72.77	66.67	93.02	89.73	73.56	74.42 (+0.55)
ARIADNE	TSV	82.67	78.04	69.86	73.44	66.02	92.83	96.66	73.34	**79.11 (+5.24)**
ViT-L/14 → ViT-B/16 (Reduction)										
Reference checkpoint										
Target zero-shot	–	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95 (+0.00)
THESEUS	TA	54.16	63.79	55.27	69.15	57.39	88.78	75.94	70.04	66.82 (−0.13)
BiCo	TA	54.65	62.76	55.21	68.99	57.51	88.99	76.62	70.11	66.86 (−0.09)
ARIADNE	TA	54.77	66.71	59.09	69.31	57.45	88.70	79.53	70.09	68.21 (+1.25)
THESEUS	TSV	55.23	63.00	55.15	69.07	57.74	88.75	75.46	69.83	66.78 (−0.17)
BiCo	TSV	55.76	62.47	55.58	68.85	57.92	88.96	76.61	70.11	67.03 (+0.08)
ARIADNE	TSV	58.93	75.50	64.96	68.78	57.51	88.50	89.28	69.50	**71.62 (+4.67)**

4.3 MULTI-TASK TRANSFER

We finally ask whether one fit can transfer a source checkpoint that already combines all eight task vectors. We merge in source space with Task Arithmetic (TA) (Ilharco et al., 2023) or Task Singular Vectors (TSV) (Gargiulo et al., 2025), apply the merged vector to the source base, and fit ARIADNE to its joint functional effect. This *merge-then-transfer* strategy requires one fit, whereas transferring each vector before merging requires eight.

Table 4 shows that ARIADNE is the only approach that improves the target average for both merge rules and both directions. The choice of source merge is consequential: TSV raises ARIADNE from 75.68% to 79.11% in extension and from 68.21% to 71.62% in reduction. In contrast, changing from TA to TSV does not consistently help the transport baselines. Direct functional transfer can preserve a good merged effect, but it cannot undo interference already introduced by a poorer source merge. The comparison with transfer-then-merge in Tab. 7 makes the efficiency trade-off explicit. Fitting vectors separately is better for TA by 2.73 points in extension and 4.15 points in reduction, and better for TSV reduction by 1.01 points. With TSV extension, however, merging first is 0.17 points better despite using one eighth as many fits. Merge-then-transfer is therefore not uniformly accuracy-equivalent but it is most attractive when the source-side merger already produces a coherent joint effect. Moreover, some datasets remain at or slightly below the target base even when the average improves, showing that multi-task interference is reduced rather than eliminated.

5 CONCLUSIONS AND LIMITATIONS

We introduced ARIADNE, a training-free method for transferring fine-tuning effects across Transformers with different depths and widths. By aligning source activation changes and fitting target-native updates in closed form, ARIADNE avoids parameter transport while improving vision, language, and multi-task transfer at substantially lower construction cost. Our results support local functional change as a practical object for cross-architecture transfer.

Our evaluation covers related vision and language backbones within two model families, so it does not establish transfer across arbitrary architectures or modalities. The method also requires representative paired calibration inputs and a block correspondences. These limitations motivate more adaptive correspondence and update selection, as well as broader cross-family evaluation.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the grammar, wording, and structure of the manuscript, as well as with writing and debugging code. All AI-assisted text was reviewed and revised by the authors, and all AI-assisted code was inspected and validated through the relevant experiments and checks. Generative AI was not used to produce experimental data or to make final scientific decisions. We take responsibility for the complete content of this work, including all text, claims, code, and results produced with the aid of generative AI.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023.

Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 2021.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. bert2bert: Towards reusable pretrained language models. In *Proceedings of the Association for Computational Linguistics*, 2022.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2015.

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.

Ning Ding, Yehui Tang, Kai Han, Chao Xu, and Yunhe Wang. Network expansion for practical training acceleration. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2023.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2025.

Andrea Gesmundo and Kaitlin Maile. Composable function-preserving expansions for transformer architectures. *arXiv preprint arXiv:2308.06103*, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *Neural Information Processing Systems Workshops*, 2015.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *Findings of the association for computational linguistics: EMNLP 2020*, pp. 4163–4174, 2020.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *International Conference on Learning Representations*, 2023.

Sara Kangaslahti, Nihal V Nayak, Jonathan Geuter, Marco Fumero, Francesco Locatello, and David Alvarez-Melis. Boomerang distillation enables zero-shot model size interpolation. In *International Conference on Learning Representations*, volume 2026, pp. 73351–73389, 2026.

Valentino Maiorca, Luca Moschella, Antonio Norelli, Marco Fumero, Francesco Locatello, and Emanuele Rodolà. Latent space translation via semantic alignment. In *Advances in Neural Information Processing Systems*, 2023.

Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative representations enable zero-shot latent space communication. In *International Conference on Learning Representations*, 2023.

Anshul Nasery, Jonathan Hayase, Pang Wei Koh, and Sewoong Oh. Pleas-merging models with permutations and least squares. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

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. *International Conference on Machine Learning*, 2025.

Filippo Rinaldi, Aniello Panariello, Giacomo Salici, Fengyuang Liu, Marco Ciccone, Angelo Porrello, and Simone Calderara. Gradient-sign masking for task vector transport across pre-trained models. In *International Conference on Learning Representations*, 2026a.

Filippo Rinaldi, Aniello Panariello, Giacomo Salici, Angelo Porrello, and Simone Calderara. Transporting task vectors across different architectures without training. *International Conference on Machine Learning*, 2026b.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 2023.

Jungyong Son, Jinwook Jung, Minhee Park, and Sungyong Baik. Bilinear coordinate alignment for training-free task-vector transfer. *arXiv preprint arXiv:2605.28444*, 2026.

Muni S Srivastava and Tatsuya Kubokawa. Comparison of discrimination methods for high dimensional data. *Journal of the Japan Statistical Society*, 2007.

Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International Conference on Machine Learning*, 2016.

Alex H Williams, Erin Kunz, Simon Kornblith, and Scott Linderman. Generalized shape metrics on neural representations. In *Advances in Neural Information Processing Systems*, 2021.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. Bert-of-theseus: Compressing bert by progressive module replacing. In *Empirical Methods in Natural Language Processing*, 2020.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL `https://arxiv.org/abs/2311.07911`.

Table 5: **Single-task transfer across calibration budgets: Extension.** We report Vision8 test accuracy (%) for ARIADNE across calibration budgets $\mathcal{B} \in \{1, 5, 10, 20, 50\}$. Results are mean $\pm$ standard deviation over three calibration runs. ΔAcc denotes the improvement over the target zero-shot checkpoint.

Method	$\mathcal{B}$	EuroSAT	SVHN	GTSRB	RESISC	DTD	Cars	MNIST	SUN	**Avg (ΔAcc)**
Reference checkpoint										
Target zero-shot	–	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87 (+0.00)
ARIADNE	1	82.85±0.82	81.59±2.02	71.65±1.47	74.02±0.41	66.63±0.24	93.02±0.00	96.05±0.54	73.58±0.18	**79.92±0.32 (+6.06)**
ARIADNE	5	86.15±1.84	89.84±0.12	81.64±1.45	77.56±0.62	68.40±0.44	93.02±0.00	98.03±0.12	73.67±0.10	**83.54±0.11 (+9.67)**
ARIADNE	10	87.50±0.44	90.93±0.24	84.17±0.96	79.01±0.20	69.48±0.70	93.02±0.00	98.47±0.08	73.77±0.04	**84.54±0.19 (+10.68)**
ARIADNE	20	87.76±1.22	91.82±0.26	86.71±0.49	79.13±0.06	70.57±0.53	93.02±0.00	98.66±0.09	73.70±0.03	**85.17±0.07 (+11.31)**
ARIADNE	50	89.38±1.28	92.61±0.41	87.25±1.13	79.79±0.07	70.78±1.29	93.02±0.00	98.84±0.07	73.80±0.04	**85.68±0.19 (+11.82)**

A APPENDIX

A.1 CALIBRATION BUDGET AND STABILITY

Tables 5 and 6 expand the main comparison across five calibration budgets and three calibration runs. Accuracy improves rapidly at small budgets: moving from one to five batches raises the mean from 79.92% to 83.54% in extension and from 74.48% to 77.97% in reduction. The subsequent gains are smaller, reaching 85.68% and 80.20% at 50 batches. Additional observations therefore refine the fitted effect rather than changing the qualitative outcome. They also improve stability in the harder reduction setting: its average standard deviation falls from 0.98 at one batch to 0.24 at 50, with the especially variable EuroSAT and GTSRB fits becoming more consistent. The benefit is task-dependent, however. EuroSAT, SVHN, GTSRB, and MNIST continue to improve with calibration, whereas Cars and SUN397 remain near their target-base values. More calibration reduces estimation error but cannot make every source effect useful to the target.

A.2 TRANSPORT THEN MERGE

Table 7 reports the more expensive alternative to the merge-then-transfer strategy in the main paper: each of the eight task vectors is first transferred independently, after which the target vectors are merged. ARIADNE improves on the target base for both merge rules and directions, gaining 4.54–5.08 points in extension and 5.40–5.68 points in reduction. Once tasks have been transferred separately, TA and TSV differ by at most 0.53 points, much less than when the source vectors are merged before transfer. This suggests that separate functional fits reduce the source-side interference that makes the merge rule consequential in Tab. 4. The improvement comes at the cost of eight fits rather than one, and it is not uniform across datasets: gains on EuroSAT, SVHN, GTSRB, and MNIST coexist with small losses on some of DTD, Cars, RESISC45, and SUN397. The average therefore captures useful multi-task transfer, not Pareto improvement on every task.

A.3 ACTIVATION AND GRADIENT ALIGNMENT

Table 8 separates the alignment signal from the direct-regression objective. With one calibration batch, activation- and gradient-derived alignments are essentially tied in both directions. Their gap grows with the budget: at 20 batches, gradient alignment is 2.41 points better in extension and 4.20 points better in reduction. Gradients are especially helpful for EuroSAT, GTSRB, RESISC45, and DTD, but do not dominate on every task; activation alignment remains competitive on SVHN and MNIST at several budgets. The gradient variant is thus useful when labels and backward passes are available, while activation alignment retains the practical advantage of label-free calibration.

A.4 RESOURCE SCALING

Table 9 shows that ARIADNE's advantage is largest in the low-data regime: at one batch it is about 70–77$\times$ faster for extension and 90$\times$ faster for reduction. The relative gap narrows as calibration grows, but remains 32–35$\times$ at 20 batches. Absolute ARIADNE runtime rises from 13 to 234 seconds in extension and from 5 to 95 seconds in reduction between one and 50 batches. Host-memory

Table 6: **Single-task transfer across calibration budgets: Reduction.** We report Vision8 test accuracy (%) for ARIADNE across calibration budgets $\mathcal{B} \in \{1, 5, 10, 20, 50\}$. Results are mean $\pm$ standard deviation over three calibration runs. ΔAcc denotes the improvement over the target zero-shot checkpoint.

Method	$\mathcal{B}$	EuroSAT	SVHN	GTSRB	RESISC	DTD	Cars	MNIST	SUN	Avg (ΔAcc)
Reference checkpoint										
Target zero-shot	–	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95 (+0.00)
ARIADNE	1	69.56±5.72	77.98±1.38	67.98±0.67	69.92±0.62	57.45±0.00	88.89±0.00	94.07±0.91	69.99±0.11	**74.48±0.98 (+7.53)**
ARIADNE	5	76.47±0.58	85.67±1.68	74.47±1.85	72.00±0.50	58.59±1.02	88.93±0.06	97.64±0.14	70.01±0.01	**77.97±0.55 (+11.02)**
ARIADNE	10	79.05±0.77	87.60±1.41	76.92±3.01	72.08±0.50	59.22±1.54	88.89±0.00	97.83±0.35	70.06±0.07	**78.96±0.73 (+12.00)**
ARIADNE	20	80.70±1.55	88.47±0.78	77.95±2.71	72.56±0.26	58.55±1.91	88.89±0.00	98.10±0.09	70.09±0.03	**79.41±0.41 (+12.46)**
ARIADNE	50	81.44±0.53	88.73±0.50	79.56±0.73	73.83±0.33	60.84±0.54	88.89±0.00	98.25±0.03	70.05±0.06	**80.20±0.24 (+13.24)**

Table 7: **Transport-then-merge task-vector transfer.** We first transport individual task vectors to the target model using ARIADNE, THESEUS, or BiCo and then merge them using Task Arithmetic (TA) or TSV. All methods use $\mathcal{B} = 10$ calibration batches and a single shared validation-selected merge coefficient. ΔAcc denotes the improvement over the target zero-shot checkpoint.

Method	Merge	EuroSAT	SVHN	GTSRB	RESISC	DTD	Cars	MNIST	SUN	Avg (ΔAcc)
ViT-B/16 → ViT-L/14 (Extension)										
Reference checkpoint										
Target zero-shot	–	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87 (+0.00)
THESEUS	TA	73.95	70.19	60.51	73.00	66.31	92.98	88.56	73.73	74.90 (+1.03)
BiCo	TA	71.60	71.36	63.83	72.73	68.09	92.61	94.66	73.60	76.06 (+2.19)
ARIADNE	TA	79.59	78.91	67.92	73.67	65.66	92.44	95.83	73.26	**78.41 (+4.54)**
THESEUS	TSV	72.59	67.76	58.56	72.54	66.37	93.02	86.69	73.54	73.88 (+0.02)
BiCo	TSV	74.69	72.71	67.63	71.31	66.43	92.69	96.80	73.33	76.95 (+3.08)
ARIADNE	TSV	84.86	80.10	67.89	71.69	66.31	91.29	96.98	72.42	**78.94 (+5.08)**
ViT-L/14 → ViT-B/16 (Reduction)										
Reference checkpoint										
Target zero-shot	–	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95 (+0.00)
THESEUS	TA	55.84	62.58	55.25	69.17	57.57	88.88	76.38	70.01	66.96 (+0.01)
BiCo	TA	56.05	61.81	55.39	69.58	57.45	88.84	77.81	70.11	67.13 (+0.18)
ARIADNE	TA	62.88	74.80	67.01	69.12	57.86	87.76	90.61	68.82	**72.36 (+5.40)**
THESEUS	TSV	56.09	62.66	55.25	69.12	57.39	88.89	76.31	70.01	66.96 (+0.01)
BiCo	TSV	56.05	62.17	55.29	69.66	57.80	88.89	77.72	70.00	67.20 (+0.24)
ARIADNE	TSV	65.14	76.18	65.80	67.23	59.10	84.87	94.17	68.56	**72.63 (+5.68)**

growth is much slower, increasing by only about 20–23% over the same range. At every jointly measured budget, ARIADNE also uses substantially less host memory than either baseline; the missing 50-batch baseline entries are therefore not used for a relative comparison.

The decomposition in Tab. 10 locates the computational difference. For extension at $\mathcal{B} = 10$, transformations account for 45 of ARIADNE's 53 seconds, compared with roughly 1,850–1,880 seconds for the baselines. In reduction, ARIADNE divides its 21 seconds almost evenly between collection and transformation, while baseline transformations still require about 13 minutes. Applying the fitted update is negligible for ARIADNE in both directions. The measured speedup is therefore concentrated in the transformation stage rather than data collection or update application.

A.5 TARGET FORMULATION AND REGULARIZATION

Table 11 gives the per-task version of the target ablation. Regressing toward the absolute source fine-tuned endpoint remains better than the target base on average, particularly in reduction, so the endpoint contains some transferable information. It is nevertheless consistently weaker than matching the source-induced change: the gap is 8.14 points in extension and 3.66 points in reduction. The larger extension penalty is driven by broad losses on EuroSAT, SVHN, GTSRB, and RESISC45. These results support treating the target base as the reference state and importing only the task-

Table 8: **Activation- versus gradient-based alignment.** We compare activation-derived Q_{act} and gradient-derived Q_{grad} within ARIADNE across calibration budgets $\mathcal{B} \in \{1, 2, 5, 10, 20\}$. Entries are Vision8 test accuracy (%). ΔAcc denotes the improvement over the target zero-shot checkpoint.

	$\mathcal{B}$	EuroSAT	SVHN	GTSRB	RESISC	DTD	Cars	MNIST	SUN	**Avg (ΔAcc)**
ViT-B/16 → ViT-L/14 (Extension)										
Reference checkpoints										
Source zero-shot	–	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95
Source fine-tuned	–	98.68	97.91	99.04	96.12	82.27	93.24	99.73	78.55	93.19
Target zero-shot	–	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87 (+0.00)
Q_{act}	1	82.22	83.41	70.00	73.97	66.37	93.02	95.54	73.71	**79.78 (+5.91)**
Q_{grad}	1	83.58	81.63	68.62	76.17	66.55	93.09	94.83	73.76	**79.78 (+5.91)**
Q_{act}	2	86.46	87.86	76.29	76.38	68.62	93.02	96.47	73.53	82.33 (+8.46)
Q_{grad}	2	89.42	84.30	81.84	78.31	67.43	93.24	96.24	73.89	**83.09 (+9.22)**
Q_{act}	5	88.23	89.73	79.99	78.15	68.74	93.02	97.91	73.55	83.67 (+9.80)
Q_{grad}	5	89.59	87.42	81.83	82.40	72.70	93.37	97.72	74.12	**84.89 (+11.02)**
Q_{act}	10	87.98	91.02	83.26	78.87	68.68	93.02	98.50	73.73	84.38 (+10.51)
Q_{grad}	10	90.78	90.22	87.60	84.96	72.40	93.62	98.24	74.19	**86.50 (+12.63)**
Q_{act}	20	88.93	92.10	86.15	79.17	70.04	93.02	98.77	73.73	85.24 (+11.37)
Q_{grad}	20	92.14	92.50	91.00	85.49	73.64	93.62	98.66	74.17	**87.65 (+13.78)**
ViT-L/14 → ViT-B/16 (Reduction)										
Reference checkpoints										
Source zero-shot	–	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87
Source fine-tuned	–	99.14	98.30	99.20	96.98	85.70	94.93	99.76	82.62	94.58
Target zero-shot	–	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95 (+0.00)
Q_{act}	1	76.01	79.36	68.74	69.21	57.45	88.89	94.97	70.01	75.58 (+8.63)
Q_{grad}	1	82.67	75.03	64.50	71.62	58.27	88.84	93.82	70.01	**75.60 (+8.65)**
Q_{act}	2	76.63	84.44	70.53	69.98	57.45	88.89	95.89	70.01	76.73 (+9.78)
Q_{grad}	2	90.16	82.69	77.79	72.10	58.16	88.89	95.59	70.01	**79.42 (+12.47)**
Q_{act}	5	76.50	86.70	74.06	71.50	58.92	89.00	97.60	70.01	78.04 (+11.09)
Q_{grad}	5	88.93	84.07	78.84	76.93	62.53	89.01	96.37	69.97	**80.83 (+13.88)**
Q_{act}	10	79.30	88.76	75.92	72.36	60.22	88.89	98.23	70.01	79.21 (+12.26)
Q_{grad}	10	90.62	86.35	84.45	78.43	61.64	88.89	97.68	70.27	**82.29 (+15.34)**
Q_{act}	20	80.58	89.29	75.81	72.40	57.45	88.89	98.12	70.10	79.08 (+12.13)
Q_{grad}	20	91.23	90.23	86.20	78.94	61.94	89.03	98.22	70.46	**83.28 (+16.33)**

specific displacement, rather than asking a heterogeneous target to reproduce source features that are unrelated to fine-tuning.

Table 12 shows a broad useful range for the target-update regularizer. Fixed trace-scaled coefficients from 0.001 to 0.1 produce similar averages, while excessive shrinkage at $\rho = 10$ removes 3.04 points in extension and 3.82 points in reduction relative to the best fixed choice. The empirical-Bayes rule is within 0.12 points of the best fixed coefficient in extension and 0.51 in reduction without validation-time tuning. These results support the automatic ridge scale as a practical default, with little loss relative to the best tested fixed coefficients.

Table 9: **Runtime and host-memory scaling with calibration budget.** We compare ARIADNE, THESEUS, and BiCo across calibration budgets $\mathcal{B}$. Fit time is the median phase-total runtime per Vision8 task, including activation collection, transformation, and transport. Host memory is the maximum peak RSS across the eight Vision8 tasks and is reported in decimal GB. Relative time and relative host memory are normalized to ARIADNE at the corresponding direction and calibration budget. Baseline measurements at $\mathcal{B} = 50$ are unavailable.

Method	$\mathcal{B}$	Fit time (s)	Rel. time	Host peak (GB)	Rel. host
ViT-B/16 → ViT-L/14 (Extension)					
THESEUS	1	1023.53	76.7×	73.06	4.74×
BiCo	1	933.46	69.9×	89.42	5.80×
ARIADNE	1	**13.35**	**1**×	**15.41**	**1**×
THESEUS	2	1075.37	60.5×	71.42	4.44×
BiCo	2	1140.61	64.2×	89.33	5.55×
ARIADNE	2	**17.77**	**1**×	**16.10**	**1**×
THESEUS	5	1366.84	44.7×	73.33	4.60×
BiCo	5	1323.15	43.2×	90.43	5.67×
ARIADNE	5	**30.61**	**1**×	**15.95**	**1**×
THESEUS	10	1919.15	36.2×	73.19	4.53×
BiCo	10	1998.17	37.7×	91.16	5.64×
ARIADNE	10	**53.07**	**1**×	**16.16**	**1**×
THESEUS	20	3024.66	31.8×	73.64	4.31×
BiCo	20	3205.04	33.7×	90.84	5.31×
ARIADNE	20	**95.15**	**1**×	**17.10**	**1**×
ARIADNE	50	**234.02**	**1**×	**18.50**	**1**×
ViT-L/14 → ViT-B/16 (Reduction)					
THESEUS	1	426.41	90.5×	38.83	2.02×
BiCo	1	423.76	90.0×	48.00	2.49×
ARIADNE	1	**4.71**	**1**×	**19.24**	**1**×
THESEUS	2	471.62	71.8×	39.92	2.09×
BiCo	2	540.56	82.3×	48.76	2.55×
ARIADNE	2	**6.57**	**1**×	**19.14**	**1**×
THESEUS	5	600.22	49.6×	39.87	2.02×
BiCo	5	619.89	51.2×	50.16	2.54×
ARIADNE	5	**12.11**	**1**×	**19.74**	**1**×
THESEUS	10	826.09	38.8×	40.38	2.00×
BiCo	10	825.73	38.8×	50.51	2.50×
ARIADNE	10	**21.29**	**1**×	**20.22**	**1**×
THESEUS	20	1285.89	32.3×	41.34	1.99×
BiCo	20	1370.81	34.5×	50.06	2.41×
ARIADNE	20	**39.75**	**1**×	**20.75**	**1**×
ARIADNE	50	**95.04**	**1**×	**23.60**	**1**×

Table 10: **Phase-level runtime decomposition.** We decompose fit time into activation collection, transformation, and transport. The upper block reports ARIADNE across calibration budgets, while the lower block compares ARIADNE, THESEUS, and BiCo at $\mathcal{B} = 10$. Values are median seconds per Vision8 task; evaluation and merge-coefficient search are excluded.

Direction	$\mathcal{B}$	Collection (s)	Transformation (s)	Transport (s)	Total fit (s)
ARIADNE runtime scaling					
Extension	1	0.80	12.53	0.03	13.35
	2	1.57	16.16	0.03	17.77
	5	3.89	26.69	0.03	30.61
	10	7.75	45.29	0.03	53.07
	20	15.48	79.63	0.03	95.15
	50	38.67	195.36	0.03	234.02
Reduction	1	0.99	3.71	0.01	4.71
	2	1.97	4.60	0.01	6.57
	5	4.90	7.20	0.01	12.11
	10	9.79	11.49	0.01	21.29
	20	19.59	20.14	0.01	39.75
	50	49.08	45.92	0.01	95.04
Method comparison at $\mathcal{B} = 10$					
Extension	THESEUS	68.54	1845.84	5.00	1919.15
	BiCo	110.58	1881.09	4.96	1998.17
	ARIADNE	**7.75**	**45.29**	**0.03**	**53.07**
Reduction	THESEUS	34.25	790.11	2.48	826.09
	BiCo	53.56	767.37	2.32	825.73
	ARIADNE	**9.79**	**11.49**	**0.01**	**21.29**

Table 11: **Target formulation ablation.** We compare the default change-target formulation of ARIADNE with an endpoint-target variant at $\mathcal{B} = 10$ calibration batches. The default formulation transfers the source fine-tuning effect by regressing the target update toward the fine-tuning-induced activation change, $h_s^1 - h_s^0$, whereas the endpoint variant instead regresses the updated target directly toward the absolute source fine-tuned activation h_s^1. Thus, the endpoint variant attempts to reproduce the source fine-tuned state rather than transfer its functional change.

Variant	EuroSAT	SVHN	GTSRB	RESISC	DTD	Cars	MNIST	SUN	Avg (ΔAcc)
ViT-B/16 → ViT-L/14 (Extension)									
Reference checkpoints									
Source zero-shot	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95
Source fine-tuned	98.68	97.91	99.04	96.12	82.27	93.24	99.73	78.55	93.19
Target zero-shot	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87 (+0.00)
ARIADNE (change target)	87.98	91.02	83.26	78.87	68.68	93.02	98.50	73.73	**84.38 (+10.52)**
ARIADNE (endpoint target)	77.65	74.80	61.20	72.54	66.37	93.02	90.77	73.55	76.24 (+2.37)
ViT-L/14 → ViT-B/16 (Reduction)									
Reference checkpoints									
Source zero-shot	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87
Source fine-tuned	99.14	98.30	99.20	96.98	85.70	94.93	99.76	82.62	94.58
Target zero-shot	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95 (+0.00)
ARIADNE (change target)	79.30	88.76	75.92	72.36	60.22	88.89	98.23	70.01	**79.21 (+12.26)**
ARIADNE (endpoint target)	63.54	84.68	75.98	70.49	57.45	88.89	93.34	70.01	75.55 (+8.60)

Table 12: **Ridge selection.** We compare the empirical-Bayes ridge used by ARIADNE with fixed trace-scaled coefficients $\rho \in \{0.001, 0.01, 0.1, 1, 10\}$. Entries are Vision8 test accuracy (%), with α selected per task on validation. Bold averages indicate the best ridge choice within each transfer direction.

Ridge	EuroSAT	SVHN	GTSRB	RESISC	DTD	Cars	MNIST	SUN	**Avg**
ViT-B/16 → ViT-L/14 (Extension)									
Reference checkpoint									
Target zero-shot	72.51	67.73	58.55	72.54	66.37	93.02	86.66	73.55	73.87
Fixed $\rho = 0.001$	88.23	91.47	82.86	78.59	69.39	93.02	98.51	73.73	84.47
Fixed $\rho = 0.01$	88.15	91.41	83.13	78.84	69.80	93.02	98.50	73.71	**84.57**
Fixed $\rho = 0.1$	88.27	90.71	82.98	78.57	67.49	93.02	98.51	73.71	84.16
Fixed $\rho = 1$	87.16	89.01	80.24	77.51	69.33	93.02	98.11	73.61	83.50
Fixed $\rho = 10$	85.68	85.09	74.80	75.84	67.26	93.02	96.96	73.62	81.53
Empirical Bayes	88.27	91.12	83.43	78.87	68.68	93.02	98.50	73.73	84.45
ViT-L/14 → ViT-B/16 (Reduction)									
Reference checkpoint									
Target zero-shot	55.84	62.57	55.20	69.17	57.45	88.89	76.49	70.01	66.95
Fixed $\rho = 0.001$	79.79	88.71	75.77	71.38	59.81	88.97	98.37	70.01	79.10
Fixed $\rho = 0.01$	79.92	88.84	75.97	71.39	59.63	89.01	98.09	70.01	**79.11**
Fixed $\rho = 0.1$	79.18	88.62	75.68	72.17	57.92	88.89	98.01	70.06	78.82
Fixed $\rho = 1$	78.07	87.28	73.06	71.16	57.45	89.01	97.16	70.01	77.90
Fixed $\rho = 10$	75.10	79.66	69.83	70.19	57.45	88.90	91.16	70.01	75.29
Empirical Bayes	78.52	88.28	75.92	71.48	57.45	88.89	98.23	70.01	78.60