Spectral Re-Basin for Aligning Neural Weights

Ya-Wei Eileen Lin[1] **Thomas Dagès**[1] **Daniel Herbst**[1]
Daniel Cremers[1] **Stefanie Jegelka**[1,2]

[1]Technical University of Munich; School of CIT, MCML, MDSI
[2]MIT; Department of EECS, CSAIL

Abstract

Linear mode connectivity is often analysed through exact parameter symmetries, especially hidden-unit permutations that leave the network function unchanged. Yet, many relationships between trained networks are not captured by bijective coordinate relabelings: widths may differ, neurons may split or clone, and explicit parameter symmetries may be broken. In this paper, we introduce *spectral re-basin*, taking an operator perspective on hidden-unit geometry to align neural weights through the functional geometry of hidden units. Specifically, we represent each hidden layer as a graph of *neurons* built from layerwise descriptors (weights or activations), associate it with a graph shift operator, and study two coupled objects: layerwise graph functional symmetries, which preserve the relational geometry, and intertwining correspondences across layers from two models, which align their graph structures. We show that permutations arise as a special case of this viewpoint and derive results for non-bijective constructions. Our analysis connects correspondence quality to preserved spectral modes, functional exchangeability of neurons, and bounds on endpoint distortion and linear path barriers. These findings suggest that neural weight alignment with low interpolation barriers need not rely on exact parameter symmetries but can be obtained from broader spectral operator compatibility in hidden-unit geometry.

1 Introduction

Linear mode connectivity (LMC) has become a central lens for studying the geometry of neural network loss landscapes [1–7]. Its significance lies in linking a geometric question about the structure of trained solutions to practical questions about whether independently trained models learn compatible representations [8–13] and whether their weights can be *aligned* and then averaged or merged without substantial loss degradation [5, 14–19]. Empirically, independently trained models can often be connected by low-loss curves, and in some settings even by low-loss line segments, despite being far apart in parameter space [1–3, 20, 21]. A prominent explanation for apparent barriers is the presence of parameter symmetries [22–25, 4]. In many neural networks, hidden units can be permuted without changing the function represented by the network. This observation motivates *weight space alignment* methods that search for a hidden-unit relabelling before linearly interpolating between models, turning apparent barriers in parameter space into artefacts of coordinate choice [26–33].

However, the hidden-unit permutation symmetries most commonly used for alignment are bijective and act at the coordinate level [34–38]. When leveraged for aligning two networks, this imposes the strong assumption that the networks can be approximately aligned with a one-to-one correspondence. Yet, many meaningful correspondences between neural networks do not satisfy this [39–42]: networks may have different widths, so one neuron in a smaller model may correspond to a group of neurons

Code: https://github.com/ywelld/spectralrebasin.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

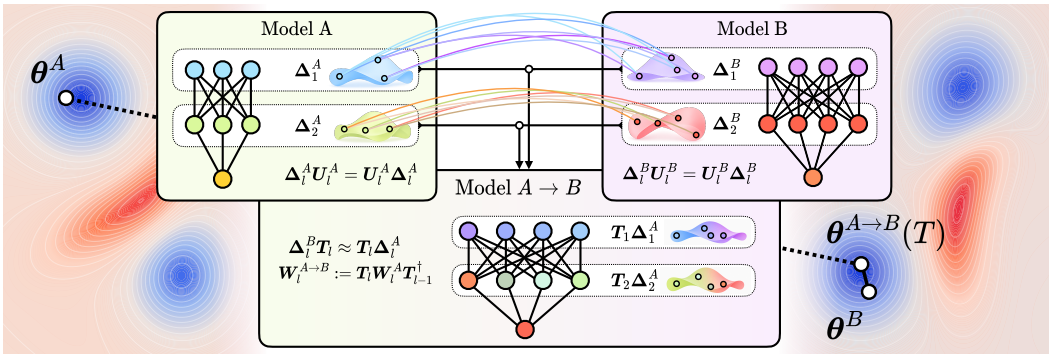


Figure 1: Illustration of *spectral re-basin*: teleporting from a mode $\boldsymbol{\theta}^A$ in the loss landscape of model A to a corresponding mode $\boldsymbol{\theta}^B$ in the loss landscape of model B. Our method constructs layerwise spectral descriptors stemming from graph functional symmetries and uses them to build transformation maps $\boldsymbol{T}_l$ between corresponding layers of models A and B s.t. $\boldsymbol{\Delta}_l^B \boldsymbol{T}_l \approx \boldsymbol{T}_l \boldsymbol{\Delta}_l^A$. This yields a graph functional correspondence between networks, even when their widths are different.

in a larger model [43, 44]. Neurons may split, clone, or become redundantly represented across multiple units [45–51]. In addition, learned features can undergo other deformations, e.g., rotations or shrinking [52–54, 9–11, 41]. These observations suggest that permutations are only one instance of a broader correspondence structure [24, 55, 56].

In this paper, we study weight alignment at a graph functional level. Our observation is that, at each layer, neurons carry a relational geometry that can be represented as a graph built from weights or activations. We therefore associate each layer with a graph shift operator and study transformations that align the corresponding graph structures. For a single layer, this leads to *graph functional symmetries*, defined as orthogonal transformations that commute with the layerwise graph operator. These symmetries contain graph automorphisms and hence hidden-unit permutations as a special case. For layers from two models, this leads to layerwise *intertwining correspondences* between neuron graph function spaces, which align graph operators across models and need not be bijective.

Based on this view, we introduce *spectral re-basin*, a graph-based framework for aligning neural networks in the spectral domain (see illustration in Fig. 1). Our method matches models layer by layer using spectral representations of their weight or activation graphs. The resulting formulation generalises permutation matching to flexible spectral correspondences and naturally applies to models with different widths. We further show that the layerwise residual minimised by our spectral re-basin controls the distance between the rebased and target models and yields an upper bound on the loss barrier. Empirically, we demonstrate the utility of our method for matching models with mismatched widths and matching asymmetric networks of different widths where parameter symmetries are broken. Across these regimes, our approach outperforms competing network matching baselines.

We make the following contributions:

- *(i)* We introduce graph functional symmetries as an operator analogue of hidden-unit permutation symmetry, extending the viewpoint from coordinates to layerwise function spaces.
- *(ii)* We propose *spectral re-basin*, which uses graph functional alignments to match neural weights in the spectral domain with flexible and non-bijective correspondences.
- *(iii)* We relate the resulting graph functional alignment objective to loss barriers by showing that the layerwise alignment residual controls the deviation between the rebased and target models.
- *(iv)* We empirically study our method across identical architectures, architectures with different widths, and mismatched-width asymmetric networks when parameter symmetries are broken.

2 Related work

We briefly review linear mode connectivity for analysing loss barriers along linear paths in parameter space, and neural weight alignment for finding correspondence between neural networks. A more detailed discussion of extensive related work is provided in §B.

Linear mode connectivity. A central empirical observation in modern deep learning is that independently trained networks are often not isolated minima, but belong to broader connected regions of low loss [2, 1, 57–59]. Given two trained solutions, linear mode connectivity (LMC) asks whether the straight-line interpolation between their parameters remains in a low-loss region [3, 21, 60, 38]. When this holds, the two solutions are said to be linearly mode-connected. This question is important for practical goals such as model merging, weight space ensembling, and the broader study of neural network loss landscapes [1, 61, 14–17, 5]. In addition, recent work argues that connectivity should be studied at the layer level [62, 13, 63]. However, testing LMC directly in raw parameter space can be misleading, because neural networks admit many distinct parameterisations that implement essentially the same computation, e.g., hidden-unit permutations, and more general reparameterisations can also induce a strong mismatch between models that are functionally similar [4, 5, 18, 25, 64, 52, 65]. As such, two networks may be close in function space while still appearing far apart in weight space. This motivates alignment-based approaches such as ours: before asking whether two models are linearly connected, one should first place them in a common layerwise coordinate system where interpolation reflects meaningful structural correspondence.

Neural weight alignment. Weight alignment methods address the gap between functional similarity and raw parameter similarity by explicitly seeking a correspondence between the internal features of two trained networks before any interpolation or merging, so that the comparison is carried out in coordinates that respect the symmetries of the architecture. In the classical same-width setting, this correspondence is usually implemented as a layerwise hidden-unit permutation [8, 18, 4, 5, 66], and these approaches have shown that a substantial part of the apparent interpolation barrier often comes from incompatible coordinates [67, 37, 38]. More recent work has broadened this view beyond strict one-to-one permutations: some approaches learn alignment maps directly in weight space [31, 32, 68–72], while others combine matching with representation repair or more general reparameterization schemes to support interpolation and merging across richer symmetry classes and more complex architectures [19, 44, 73–79]. Related methods such as OT fusion [39, 65, 44] and CCA merge [40] also move beyond the permutation picture, the former by formulating alignment as a soft transport problem between hidden units and the latter by seeking correlated latent subspaces based on covariance matrices between neurons. These works suggest that meaningful alignment may be soft, dense, or non-bijective. Our method builds on this broader perspective, but introduces a different structural prior by treating each layer as a graph of hidden units and seeking a correspondence that both aligns weights or activations and remains compatible with the layerwise graph operator.

3 Motivating example: when permutations are too rigid

A natural way to compare two neural networks is to first relabel their hidden units. This approach is effective when the two networks differ by a reordering of functionally equivalent units. Yet, this one-to-one view is not always sufficient. In overparameterised models, the same functional role may be distributed across several neurons, compressed into fewer neurons, or represented in different coordinates within a redundant subspace. Hence, interpolation barriers can persist even when the two endpoint networks compute the same function.

Example 1. Let $h(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$, where σ is ReLU and $\|\boldsymbol{w}\|_2 = 1$. Assume h is not identically zero on the data. Write $[r_1; r_2]$ for the matrix with rows r_1 and r_2. For $\varphi \in [0, \pi/2]$, consider a one-hidden-layer neural network of width two with weights $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{x})$, where $\boldsymbol{\theta} := (\boldsymbol{W}_2, \boldsymbol{W}_1)$, $\boldsymbol{W}_1 := [\cos(\varphi)\boldsymbol{w}^\top; \sin(\varphi)\boldsymbol{w}^\top]$, and $\boldsymbol{W}_2 := [\cos(\varphi); \sin(\varphi)]$. By positive homogeneity (i.e., $\sigma(\lambda x) = \lambda\sigma(x)$ for $\lambda \geq 0$), any network with such weights $f_{\boldsymbol{\theta}}$ computes h.[1]

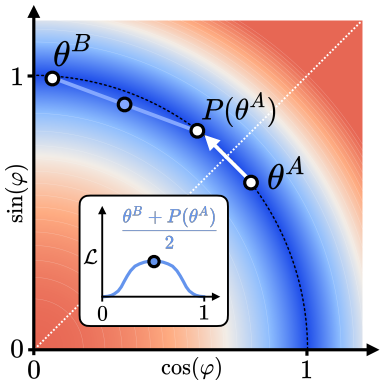


Figure 2: Neural networks realising the same function need not be related by a hidden-unit permutation.

[1] Note that among all parameterisations of h of the form $\boldsymbol{W}_1 = [\alpha_1\boldsymbol{w}^\top; \beta_1\boldsymbol{w}^\top]$, $\boldsymbol{W}_2 = [\alpha_2; \beta_2]$ with $\alpha_1, \alpha_2, \beta_1, \beta_2 \geq 0$ and $\alpha_1\alpha_2 + \beta_1\beta_2 = 1$, setting $\cos(\varphi) := \alpha_1 = \alpha_2$, $\sin(\varphi) := \beta_1 = \beta_2$ yields precisely the ones of minimum squared weight norm $\|\boldsymbol{W}_1\|_F^2 + \|\boldsymbol{W}_2\|_F^2$.

In the cases $\varphi \in \{0, \pi/2\}$, the feature $\boldsymbol{w}$ is represented by a model with one active and one inactive neuron, respectively. In the other extreme $\varphi = \pi/4$, the feature is implemented by two duplicate neurons. Permutations acting on the hidden units cannot match all of these equivalent factorisations, as illustrated in Fig. 2. The only nontrivial permutation of hidden units here is a transposition $\boldsymbol{P}$, which acts by reflection across the diagonal (i.e., $\varphi \mapsto \pi/2 - \varphi$). E.g., starting from $\varphi = 0$, this permutation can move the weights to $\pi/2$, i.e., move the active neuron from one coordinate to another, but it cannot split or merge the neurons to obtain any parameterisation in between with $\varphi \in (0, \pi/2)$.

In other words, given two parameter vectors $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^B$ with associated angles φ^A, φ^B, matching can only choose the closer among $\boldsymbol{\theta}^A$ and $\boldsymbol{P}(\boldsymbol{\theta}^A)$ to $\boldsymbol{\theta}^B$, but in general cannot identify the two weight vectors perfectly. Further, interpolation in weight space between the matched models will generically leave the solution manifold and can thus incur a loss barrier (e.g., under squared loss against h).

4 Spectral re-basin: weight alignment via a graph functional perspective

The previous example suggests that comparing networks through one-to-one neuron relabellings can be too restrictive, as the same functional structure can be represented in different neuron-level coordinates. This suggests comparing networks in a coordinate system that reflects the roles played by *neurons* rather than only their indices. To this end, we develop this idea through a layerwise graph perspective. For clarity, we describe the method for MLPs without biases, i.e., $\boldsymbol{z}_0(\boldsymbol{x}) := \boldsymbol{x}$, $\boldsymbol{z}_l(\boldsymbol{x}) := \sigma(\boldsymbol{W}_l \boldsymbol{z}_{l-1}(\boldsymbol{x}))$ with $\boldsymbol{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$ for $l = 1, \dots, L-1$, and output $\boldsymbol{z}_l(\boldsymbol{x}) = \boldsymbol{W}_L \boldsymbol{z}_{L-1}(\boldsymbol{x})$. However, the same construction applies more generally to channels in ResNets or attention heads. For each layer l, we construct an undirected weighted graph $G_l = ([d_l], E_l, \boldsymbol{A}_l, \boldsymbol{F}_l)$ from a feature matrix $\boldsymbol{F}_l \in \mathbb{R}^{d_l \times N_l}$, chosen to be either the weight vectors $\boldsymbol{W}_l$ or activation vectors $\boldsymbol{Z}_l$ of the layer. Each node i corresponds to a row of $\boldsymbol{F}_l$ and carries feature vector $(\boldsymbol{F}_l)_{i,:} \in \mathbb{R}^{N_l}$. The edge set E_l is built from local proximity $\|(\boldsymbol{F}_l)_{j,:} - (\boldsymbol{F}_l)_{i,:}\|_2$, e.g., a (symmetrised) k-nearest-neighbour graph, and the adjacency matrix $\boldsymbol{A}_l$ is the Gaussian kernel $(a_l)_{ij} = \exp(-\|(\boldsymbol{F}_l)_{j,:} - (\boldsymbol{F}_l)_{i,:}\|_2^2 / \sigma_l^2)$, where $\sigma_l > 0$ is a scale parameter, often chosen as the median pairwise distance scaled by a constant factor [80–82]. Further details on graph construction and graph signal processing are provided in §A.

We call G_l the *neuron graph*, distinct from the computation graph of the network [83, 70]. Essentially, its vertices are the neurons in that layer, its node descriptors are either weight vectors or activation vectors, and its edges encode similarity between these descriptors. Thus, the graph G_l captures the neuron geometry at that layer: nearby vertices correspond to neurons with similar roles, while the graph operator describes how signals vary across this neuron-level geometry.

4.1 Graph functional symmetries

We first consider a single neural network. As hidden-unit permutation symmetries alone do not account for neuron *functionality*, i.e., their roles, we build the neuron graph G_l, whose edges capture descriptor similarity. Thus, when a graph is computed at layer l, its symmetries describe transformations that preserve the *neuron-level functional structure*[2]. In standard one-to-one neuron matching, the graph G_l is not considered, so network symmetries are searched among the whole permutation space. However, to preserve graph structure, permutations $\boldsymbol{P}_l$ should be restricted to graph automorphisms, i.e., satisfy $\boldsymbol{P}_l \boldsymbol{A}_l = \boldsymbol{A}_l \boldsymbol{P}_l$ for the graph adjacency $\boldsymbol{A}_l$. We therefore consider a broader class of graph-preserving symmetries beyond automorphisms, namely *graph functional shifts*. Let $\boldsymbol{\Delta}_l$ be a graph shift operator (GSO) at layer l, defined as an operator that encodes the graph's structure, e.g., the graph Laplacian or the normalised Laplacian [84–87]. We then define the corresponding symmetry group as follows.

Definition 2. Given a graph shift operator $\boldsymbol{\Delta}_l$, the group of graph functional shifts consists of all orthogonal transformations $\boldsymbol{U}_l$ satisfying $\boldsymbol{\Delta}_l \boldsymbol{U}_l = \boldsymbol{U}_l \boldsymbol{\Delta}_l$.

For the standard choices of $\boldsymbol{\Delta}_l$ as above, every graph automorphism $\boldsymbol{P}_l$ satisfies $\boldsymbol{\Delta}_l \boldsymbol{P}_l = \boldsymbol{P}_l \boldsymbol{\Delta}_l$, and is thus a graph functional shift. The advantages of graph functional shifts are twofold. First, once the neuron graph is constructed, the admissible transformations are described by the commutant of

[2]The term "*functional*" arises in two meanings in this work: *(i)* the graph-theoretic and operator-theoretic perspective of graph signal processing, which accounts for functions and operators defined on neuron graphs and their associated signal spaces; and *(ii)* the neural-network perspective, which accounts for the input-output functions realised by different parameterisations of a network. To distinguish these notions, we consistently use "graph-functional" when referring to functions on neuron graphs, graph-spectral operators, or mappings between graph-signal spaces. We elaborate on this distinction in §A.3.

its GSO. In this sense, alignment moves from raw neuron indices to the spectral domain induced by the layer descriptors. Second, while permutations exchange neurons directly, graph functional shifts more generally account for linear recombinations between neurons that preserve the same global functional behaviour of the layer. In fact, graph functional shifts are more related to the functional map perspective [88] developed in shape matching and are not permutations in general.

The commutativity condition of a transformation $\boldsymbol{T}_l$ with $\boldsymbol{\Delta}_l$ implies that $\boldsymbol{T}_l$ does not mix components from different eigenspaces of $\boldsymbol{\Delta}_l$ (see §G.1). Restricting such commutant transformations to orthogonal transforms $\boldsymbol{U}_l$ further implies that graph functional shifts perform rotations within the GSO's eigenspaces. Note that orthogonal transforms are invertible, thus they do not lose information: $\boldsymbol{\Delta}_l = \boldsymbol{U}_l \boldsymbol{\Delta}_l \boldsymbol{U}_l^\top$ compared to $\boldsymbol{\Delta}_l \approx \boldsymbol{T}_l \boldsymbol{\Delta}_l \boldsymbol{T}_l^\dagger$, where $\boldsymbol{T}_l^\dagger$ is the pseudoinverse of $\boldsymbol{T}_l$, and they are isometries preserving distances, i.e., they cannot deform information by shrinking or dilating it. Graph functional shifts thus provide a natural class of graph-preserving symmetries that generalise permutations: in a neural network, they preserve the operator-level organisation of neurons while allowing orthogonal linear recombinations of weights within the layer's GSO eigenspaces.

4.2 Intertwining correspondence

Inspired by the generalisation of permutations to graph functional shifts in graph symmetries (one network), we next extend this to graph alignment (two networks). For clarity, we first focus on aligning weights $\boldsymbol{W}_l^A$ and $\boldsymbol{W}_l^B$ at layer l of networks A and B. By adopting a graph perspective to network alignment and working in the graph functional domain, this suggests to study orthogonal alignments $\boldsymbol{U}_l$ satisfying the intertwining equation $\boldsymbol{\Delta}_l^B \boldsymbol{U}_l = \boldsymbol{U}_l \boldsymbol{\Delta}_l^A$, with GSOs $\boldsymbol{\Delta}_l^X$ computed on the neuron graph G_l^X for $X \in \{A, B\}$ with features $\boldsymbol{W}_l^X$. Yet, while satisfactory for symmetries (single graph $A = B$), when aligning two different networks ($A \neq B$) the orthogonal assumption becomes too restrictive. An orthogonal $\boldsymbol{U}_l$ cannot align networks of different widths as $\boldsymbol{\Delta}_l^A$ and $\boldsymbol{\Delta}_l^B$ have different shapes, nor account for scaling deformations. Thus, we relax the orthogonal assumption in the alignment setting, leading to the following space of graph functional alignments.

Definition 3. Given graph shift operators $\boldsymbol{\Delta}_l^A \in \mathbb{R}^{d_l^A \times d_l^A}$ and $\boldsymbol{\Delta}_l^B \in \mathbb{R}^{d_l^B \times d_l^B}$, the space of graph functional alignments is the space of transformations $\boldsymbol{T}_l : \mathbb{R}^{d_l^A} \to \mathbb{R}^{d_l^B}$ such that $\boldsymbol{\Delta}_l^B \boldsymbol{T}_l = \boldsymbol{T}_l \boldsymbol{\Delta}_l^A$.

When $\boldsymbol{\Delta}_l^A = \boldsymbol{\Delta}_l^B = \boldsymbol{\Delta}_l$, Def. 3 yields the commutant $\{\boldsymbol{T}_l : \boldsymbol{\Delta}_l \boldsymbol{T}_l = \boldsymbol{T}_l \boldsymbol{\Delta}_l\}$. Restricting this set to square orthogonal transformations recovers the graph functional shifts of Def. 2. When $A \neq B$, Def. 3 defines a correspondence between the function spaces of two different neuron graphs. A useful property of GSOs is that their eigendecomposition $\boldsymbol{\Delta}_l = \boldsymbol{\Phi}_l \boldsymbol{\Lambda}_l \boldsymbol{\Phi}_l^\top$, with eigenvalues ordered increasingly and interpreted as graph frequencies $\lambda_{l,i}$, induces a spectral representation of graph signals. In this representation, the columns of $\boldsymbol{\Phi}_l$ form the graph Fourier basis [87, 89, 90]. This basis reflects the underlying graph structure. We then have the following result in the spectral domain.

Proposition 4. *Let $\boldsymbol{\Delta}_l^X = \boldsymbol{\Phi}_l^X \boldsymbol{\Lambda}_l^X {\boldsymbol{\Phi}_l^X}^\top$ with $X \in \{A, B\}$. Graph functional alignments are the transformations $\boldsymbol{T}_l$ that satisfy in the spectral domain*

$$\boldsymbol{\Lambda}_l^B \boldsymbol{C}_l = \boldsymbol{C}_l \boldsymbol{\Lambda}_l^A, \tag{1}$$

where $\boldsymbol{C}_l = {\boldsymbol{\Phi}_l^B}^\top \boldsymbol{T}_l \boldsymbol{\Phi}_l^A$ is the spectral representation of $\boldsymbol{T}_l$ in the GSO eigenbasis. In addition, $\|\boldsymbol{\Delta}_l^B \boldsymbol{T}_l - \boldsymbol{T}_l \boldsymbol{\Delta}_l^A\|_F^2 = \sum_{i,j} (\lambda_{l,i}^B - \lambda_{l,j}^A)^2 (\boldsymbol{C}_l)_{ij}^2$.

The proof is in §E.1. Prop. 4 shows that graph functional alignment in the neuron space is equivalent to a spectral constraint. The penalty is large when $\boldsymbol{C}_l$ mixes eigenmodes whose eigenvalues differ, and zero when it only couples compatible spectral components, encouraging the correspondence to respect the intrinsic geometry encoded by the graph spectrum. When the bases $\boldsymbol{\Phi}_l^X$ are truncated to low frequencies, Eq. (1) is enforced only on the retained subspace, so the full intertwining relation is thus only approximated. The spectral representation $\boldsymbol{C}_l$ is also called *functional map* [88].

4.3 Weight-based spectral re-basin

We now use graph functional alignments to map the weights of network A into the coordinate system of network B. The layerwise maps $\boldsymbol{T}_l$ should satisfy two requirements. First, they should align the layerwise weight maps. Second, they should preserve the graph geometry of the layers. The former will yield a feature alignment loss for the matching and the latter will give a regularisation objective

based on graph functional alignment. A graph functional alignment $\boldsymbol{T}_l$ aligns the weights following

$$\boldsymbol{W}_l^B \boldsymbol{T}_{l-1} = \boldsymbol{T}_l \boldsymbol{W}_l^A, \tag{2}$$

with boundary conditions $\boldsymbol{T}_0 = \boldsymbol{I}$ (same input data) and $\boldsymbol{T}_L = \boldsymbol{I}$ (same output prediction). We solve the alignment sequentially across layers. When optimising the map at layer l, the previous map $\boldsymbol{T}_{l-1}$ is treated as fixed and carried through the incoming weights. This ensures that the representation produced by the rebased previous layers is expressed in the coordinate system used to align layer l. To find the weight alignment, we propose to minimise the feature alignment objective

$$\mathcal{L}_{\text{feat-align}}(\boldsymbol{T}_l) = \|\boldsymbol{W}_l^B \boldsymbol{T}_{l-1} - \boldsymbol{T}_l \boldsymbol{W}_l^A\|_F^2, \tag{3}$$

which becomes in the (truncated) spectral domain

$$\mathcal{L}_{\text{spec-feat-align}}(\boldsymbol{C}_l) = \|\widehat{\boldsymbol{W}}_l^B \boldsymbol{C}_{l-1} - \boldsymbol{C}_l \widehat{\boldsymbol{W}}_l^A\|_F^2, \tag{4}$$

where $\widehat{\boldsymbol{W}}_l^X = {\boldsymbol{\Phi}_l^X}^\top \boldsymbol{W}_l^X \boldsymbol{\Phi}_{l-1}^X$ is the spectral representation of $\boldsymbol{W}_l^X$. To regularise $\boldsymbol{T}_l$ to approximately satisfy the graph functional alignment, we also minimise the functional alignment objective

$$\mathcal{L}_{\text{func-align}}(\boldsymbol{T}_l) = \|\boldsymbol{\Delta}_l^B \boldsymbol{T}_l - \boldsymbol{T}_l \boldsymbol{\Delta}_l^A\|_F^2. \tag{5}$$

In the spectral domain, $\boldsymbol{T}_l = \boldsymbol{\Phi}_l^B \boldsymbol{C}_l {\boldsymbol{\Phi}_l^A}^\top$, this becomes

$$\mathcal{L}_{\text{spec-func-align}}(\boldsymbol{C}_l) = \|\boldsymbol{\Lambda}_l^B \boldsymbol{C}_l - \boldsymbol{C}_l \boldsymbol{\Lambda}_l^A\|_F^2. \tag{6}$$

The spectral formulations are equivalent to their coordinate-space counterparts when the full spectral basis is used, and they differ only when the spectral domain is truncated. Formulating $\mathcal{L}_{\text{spec-feat-align}}$ and $\mathcal{L}_{\text{spec-func-align}}$ in the spectral domain offers several advantages. First, it treats alignment from a global perspective rather than relying on local correspondences, which can be highly sensitive to perturbations such as noise. Moreover, by operating on the low-frequency components of the spectrum, it also implicitly performs low-pass filtering, thereby suppressing noise that would otherwise affect local alignment. Finally, the formulation is width-agnostic, making it applicable to layers of different widths. In practice, we use only the spectral formulations. Thus, our final objective for layer l, given $\boldsymbol{C}_{l-1}$, consists of the spectral feature-alignment and spectral functional-alignment terms:

$$\mathcal{L}_{\text{spec-align}}(\boldsymbol{C}_l) = \mathcal{L}_{\text{spec-feat-align}}(\boldsymbol{C}_l) + \mu \mathcal{L}_{\text{spec-func-align}}(\boldsymbol{C}_l). \tag{7}$$

For fixed $\boldsymbol{C}_{l-1}$, this is an unconstrained convex quadratic problem in $\boldsymbol{C}_l$ solvable in closed form (see §G.2) or with a standard least-squares solver. Thus, while permutation-based matching typically involves solving linear assignment subproblems, here, each hidden layer update reduces to a simple continuous least squares problem in the spectral coefficients.

We can then compute the graph functional transformation as $\boldsymbol{T}_l^* = \boldsymbol{\Phi}_l^B \boldsymbol{C}_l^* {\boldsymbol{\Phi}_l^A}^\top$ and align the weights $\boldsymbol{W}_l^A$ to those of $\boldsymbol{W}_l^B$ by computing the aligned weights $\boldsymbol{W}_l^{A\to B} \in \mathbb{R}^{d_l^B \times d_{l-1}^B}$ as

$$\boldsymbol{W}_l^{A\to B} = \boldsymbol{T}_l^* \boldsymbol{W}_l^A (\boldsymbol{T}_{l-1}^*)^\dagger = \boldsymbol{\Phi}_l^B \boldsymbol{C}_l^* {\boldsymbol{\Phi}_l^A}^\top \boldsymbol{W}_l^A (\boldsymbol{\Phi}_{l-1}^B \boldsymbol{C}_{l-1}^* {\boldsymbol{\Phi}_{l-1}^A}^\top)^\dagger, \tag{8}$$

where $(\cdot)^\dagger$ is the Moore-Penrose pseudoinverse. In §G.3, we show how we could conceptually alternatively map both weights to a mixed space for a mixed architecture between A and B.

The following result connects the residual minimised by spectral re-basin to the actual layerwise distance between the rebased model and network B.

Proposition 5. *Let* $\boldsymbol{E}_l(\boldsymbol{T}) = \boldsymbol{W}_l^B \boldsymbol{T}_{l-1} - \boldsymbol{T}_l \boldsymbol{W}_l^A$ *and* $\boldsymbol{W}_l^{A\to B} = \boldsymbol{T}_l \boldsymbol{W}_l^A \boldsymbol{T}_{l-1}^\dagger$. *Then,*

$$\|\boldsymbol{W}_l^B - \boldsymbol{W}_l^{A\to B}\|_F \le \|\boldsymbol{W}_l^B (\boldsymbol{I} - \boldsymbol{T}_{l-1} \boldsymbol{T}_{l-1}^\dagger)\|_F + \|\boldsymbol{E}_l(\boldsymbol{T})\|_F \|\boldsymbol{T}_{l-1}^\dagger\|_2. \tag{9}$$

The proof is in §E.2. Prop. 5 shows that the residual minimised by the spectral feature-alignment term also bounds the layerwise distance between the rebased endpoint and network B, up to a projection term and a conditioning factor. Note that if $\boldsymbol{T}_{l-1}$ is full row rank, then $\|\boldsymbol{W}_l^B - \boldsymbol{W}_l^{A\to B}\|_F \le \|\boldsymbol{E}_l(\boldsymbol{T})\|_F \|\boldsymbol{T}_{l-1}^\dagger\|_2$. If $\boldsymbol{T}_{l-1}$ is orthogonal, it reduces to $\|\boldsymbol{W}_l^B - \boldsymbol{W}_l^{A\to B}\|_F \le \|\boldsymbol{E}_l(\boldsymbol{T})\|_F$.

4.4 Activation-based spectral re-basin

The same framework can be applied to activations. Let $\boldsymbol{Z}_l^X \in \mathbb{R}^{d_l^X \times N}$ denote the activation at layer l for N training data points, with neurons along rows and examples along columns. We build the graph G_l^X from the rows of $\boldsymbol{Z}_l^X$, compute the graph shift operator $\boldsymbol{\Delta}_l^X$, and use its spectral basis $\boldsymbol{\Phi}_l^X$.

The key difference from weight alignment is that activation matching is local to the layer. For weights, the map at layer l must also be compatible with the map at layer $l-1$, since weights connect consecutive layers. For activations, by contrast, we directly align the representations $\boldsymbol{Z}_l^B = \boldsymbol{T}_l \boldsymbol{Z}_l^A$. Hence no explicit transport by $\boldsymbol{T}_{l-1}$ is needed in the activation feature-matching term. As we are based on the same intertwined equation from Def. 3, spectral activation alignment is given by Eq. (1).

To perform activation-based spectral re-basin, we thus need not change the functional alignment term and must only change the feature alignment ones to $\mathcal{L}_{\text{feat-align}}(\boldsymbol{T}_l) = \|\boldsymbol{Z}_l^B - \boldsymbol{T}_l \boldsymbol{Z}_l^A\|_F^2$ and spectrally

$$\mathcal{L}_{\text{spec-feat-align}}(\boldsymbol{C}_l) = \|\widehat{\boldsymbol{Z}}_l^B - \boldsymbol{C}_l \widehat{\boldsymbol{Z}}_l^A\|_F^2, \tag{10}$$

where $\widehat{\boldsymbol{Z}}_l^X$ is the spectral representation of $\boldsymbol{Z}_l$ for GSO $\boldsymbol{\Delta}_l$. The objective is then given by both feature and graph functional alignment as in Eq. (7), which can be solved explicitly (see §G.2). Unlike the weight-based formulation, this problem decouples across layers.

Given the spectral solution $\boldsymbol{C}_l^*$, the transformation is given by $\boldsymbol{T}_l^* = \boldsymbol{\Phi}_l^B \boldsymbol{C}_l^* {\boldsymbol{\Phi}_l^A}^\top$ and the aligned activation $\boldsymbol{Z}_l^{A\to B}$, that are mapped activations of A to the basin of those of B, is given by

$$\boldsymbol{Z}_l^{A\to B} = \boldsymbol{T}_l^* \boldsymbol{Z}_l^A = \boldsymbol{\Phi}_l^B \boldsymbol{C}_l^* {\boldsymbol{\Phi}_l^A}^\top \boldsymbol{Z}_l^A. \tag{11}$$

Accordingly, we define the transformed weight as

$$\boldsymbol{W}_l^{A\to B} = \boldsymbol{T}_l^* \boldsymbol{W}_l^A {\boldsymbol{T}_{l-1}^*}^\dagger. \tag{12}$$

The activation-based variant isolates the representational aspect of alignment. Rather than matching neurons through their incoming parameters, it matches the functions they realise on data. This can be advantageous when two networks implement similar intermediate representations even if their raw weights are less directly comparable. At the same time, the resulting maps $\boldsymbol{T}_l$ can be used to induce transported weights, keeping the activation-based method compatible with the weight-space view of re-basin and with the interpolation barrier bound developed in the next subsection.

4.5 Connection to linear path loss barriers

We now link spectral re-basin to linear path loss barriers. After finding the maps $\boldsymbol{T} = \{\boldsymbol{T}_l^*\}_{l=1}^L$, we express network A in the coordinate system of network B, yielding the rebased parameters $\boldsymbol{\theta}^{A\to B}(\boldsymbol{T}) = \{\boldsymbol{W}_l^{A\to B}\}_{l=1}^L$. We then consider the linear interpolation

$$\boldsymbol{W}_l(t) = (1-t)\boldsymbol{W}_l^{A\to B} + t\boldsymbol{W}_l^B \tag{13}$$

for $t \in [0,1]$, and let $\boldsymbol{\theta}(t)$ denote the parameter vector along this path. To analyse when this interpolation induces a small loss barrier, we bound the parameter distance between the rebased model and network B in terms of the layerwise alignment residuals. We then introduce a layerwise deviation as in Zhou et al. (2023) [13], and relate it to an upper bound on the overall loss barrier.

Let $R_l(\boldsymbol{T}) = \|\boldsymbol{W}_l^B \boldsymbol{T}_{l-1} - \boldsymbol{T}_l \boldsymbol{W}_l^A\|_F$ be the layerwise alignment residual and let $\Pi_l(\boldsymbol{T}) = \|\boldsymbol{W}_l^B(\boldsymbol{I} - \boldsymbol{T}_{l-1}\boldsymbol{T}_{l-1}^\dagger)\|_F$ be the projection error. The layerwise deviation is given by $\alpha_l(\boldsymbol{T}) = \Pi_l(\boldsymbol{T}) + R_l(\boldsymbol{T})\|\boldsymbol{T}_{l-1}^\dagger\|_2$. By Prop. 5, we have $\|\boldsymbol{W}_l^{A\to B} - \boldsymbol{W}_l^B\|_F \le \alpha_l(\boldsymbol{T})$. Let $\mathcal{L}(\boldsymbol{\theta})$ denote the value of the loss function for the model $\boldsymbol{\theta}$ and

$$\mathcal{B}(\boldsymbol{\theta}(t)) = \max_{t\in[0,1]} \mathcal{L}(\boldsymbol{\theta}(t)) - \max\{\mathcal{L}(\boldsymbol{\theta}^{A\to B}(\boldsymbol{T})), \mathcal{L}(\boldsymbol{\theta}^B)\} \tag{14}$$

be the loss barrier. We have the following result.

Proposition 6. *If $\mathcal{L}$ is twice differentiable with $\|\nabla^2 \mathcal{L}(\boldsymbol{\theta}(t))\|_2 \le \beta$ for $t \in [0,1]$, then the loss barrier $\mathcal{B}(\boldsymbol{\theta}(t))$ is bounded by $\mathcal{B}(\boldsymbol{\theta}(t)) \le \frac{\beta}{8}\sum_{l=1}^L \alpha_l^2(\boldsymbol{T})$.*

The proof is in §E.3. Prop. 6 shows that the linear interpolation barrier is controlled by the layerwise quantities minimised by spectral re-basin. Specifically, the feature term in the spectral objective is both the matching loss in Eq. (4) and the spectral counterpart in the barrier upper bound. We remark that since our method can learn rectangular, non-bijective correspondences, matching is directional (see § G.4 for further discussion).

We note that Prop. 5 and Prop. 6 are established using the full graph spectrum to provide a complete theoretical characterisation of our framework. In our experimental studies in the next section, we employ a truncated spectrum for computational efficiency. To bridge this gap between theory and

practice, we extend both results to the truncated spectral setting, which accounts for the effects of spectral truncation. The corresponding theoretical analysis are provided in §E.4.

5 Experimental results

We evaluate our methods on matching networks with both identical widths (§5.1) and different widths (§5.2), using accuracy or loss along the linear interpolation between matched models as the evaluation criterion (as is standard for weight alignment methods). In addition, we examine matching of asymmetric networks across varying widths (§5.3).

Detailed implementation settings, including experimental setup and hyperparameters, are reported in §J. Additional experiments, including an ablation study, width reduction, end-point preserving, effect of alignment direction, and rank ablation of the graph-spectral contribution, are presented in §K.

5.1 Matching on identical architectures

We first evaluate our spectral re-basin method where models share the same architecture and the same width, but differ in their random initialisation. Specifically, we consider VGG11 models [91] trained on CIFAR-10 [92] and ResNet20 models [93] trained on CIFAR-100. Following [94, 40], we train models at two widths: the original width ($\times 1$) and an expanded width ($\times 8$); all CIFAR models are trained using a CLIP-style loss [95]. For each architecture and task, we consider eight pairs of independently trained models. We evaluate the test accuracy at the midpoint of the interpolated parameters, $\frac{1}{2}\boldsymbol{\theta}^{A\to B} + \frac{1}{2}\boldsymbol{\theta}^{B}$, and report the mean and standard deviation across pairs. We compare our methods against both weight-based and activation-based competing methods. The weight-based baselines include *(i)* direct weight merging without alignment, *(ii)* weight matching, which accounts for neuron permutations [5], and *(iii)* our spectral re-basin method based on weights. The activation-based baselines include *(i)* activation matching, which finds permutations based on activations [5], *(ii)* OT fusion, which uses optimal transport (OT) [96] to align neurons [39], *(iii)* CCA merge [40], which aligns models by correlated subspaces from covariance matrices, and *(iv)* our spectral re-basin method based on activations. In addition, we report the average accuracy of the base models and the accuracy of their ensemble as references.

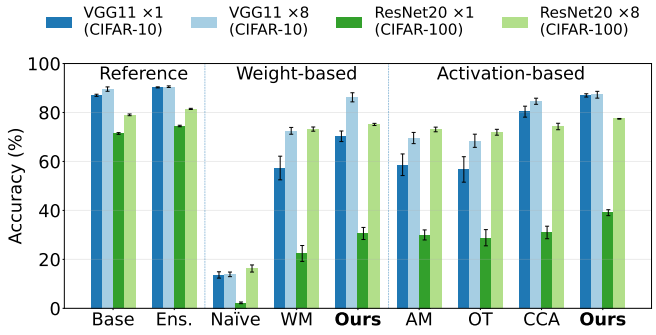


Figure 3: Identical architecture matching. We merge pairs of VGG11 models on CIFAR-10 and ResNet20 models on CIFAR-100, each trained at widths $\times 1$ and $\times 8$.

Fig. 3 reports classification performance of the interpolated model across different random seeds. We observe that our spectral re-basin based on activations achieves the strongest merged performance across all settings, closely approaching the accuracy of the base models in the settings of VGG11 $\times$ 1, VGG11 $\times$ 8, and ResNet20 $\times$ 8. Naïve merging performs poorly, while existing weight-based and activation-based matching methods recover only part of the performance of the original models. Our spectral re-basin variant based on weights is also competitive, outperforming standard weight matching. This indicates that the spectral re-basin provides a robust alignment mechanism when considering the neurons' graph functional symmetries and intertwining correspondence.

5.2 Matching across mismatched widths

We next consider a more challenging setting where the two models have mismatched widths, allowing us to evaluate the utility of our method for aligning models of different widths after mapping them into a common weight space. We study pairs of models from the same architecture family but with different widths. Specifically, we consider matching a standard-width model to a wider model, as well as matching two progressively wider models, i.e., VGG11 $\times 1 \to \times 4$, VGG11 $\times 4 \to \times 8$, ResNet20 $\times 1 \to \times 4$, and ResNet20 $\times 4 \to \times 8$. We compare against the same baselines as in §5.1. For baselines

that are not designed for cross-width matching, including naïve merging, weight matching, activation matching, and CCA merge, we use zero-padding to construct function-preserving networks[3] with matched dimensionality and denote these variants with a superscript *. In addition, as CCA merge operates by aligning subspaces and covariance matrices with different ranks do not belong to the same manifold, we also evaluate a regularised variant that adds $\varepsilon \boldsymbol{I}$ and mark this variant with †.

Fig. 4 reports midpoint interpolation accuracy after cross-width matching. The spectral re-basin approaches remain highly effective. Our activation-based variant outperforms baselines in all cross-width settings, demonstrating that the proposed alignment strategy transfers well beyond the same-width case. In contrast, competing methods recover substantially less performance, indicating that matching dimensionality is not sufficient for robust cross-width merging. Note that, although OT fusion can be applied to models with different widths, the resulting transformed networks are not function-preserving (see §K.3 for further details). These results suggest that our method is effective for aligning and merging same-family models with different widths.

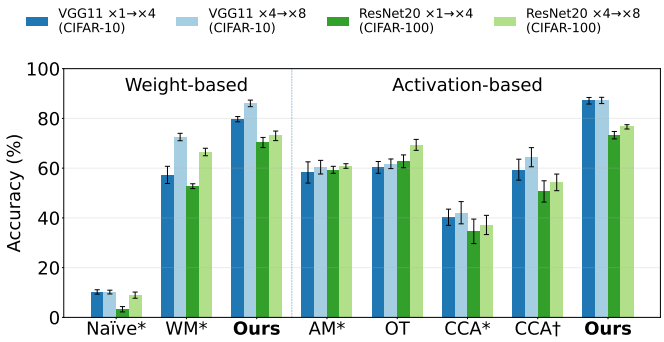


Figure 4: Matching across mismatched widths. We evaluate cross-width alignment for VGG11 on CIFAR-10 and ResNet20 on CIFAR-100 under width transitions $\times 1 \rightarrow \times 4$ and $\times 4 \rightarrow \times 8$. Methods marked with * use zero-padding to construct function-preserving networks with matched dimensionality, while methods marked with † use a regularised covariance matrix to support CCA-based alignment across mismatched widths.

5.3 Matching asymmetric networks of different widths

Finally, we consider matching asymmetric networks of different widths, where the usual parameter symmetries are explicitly broken [37, 38, 98]. Prior work has shown that asymmetric networks exhibit lower interpolation barriers than standard symmetric networks and can often be merged directly without additional post-processing or explicit alignment. This suggests that breaking parameter symmetries can simplify model merging, improve optimisation, and provide a mechanism for controlling model expressivity. Yet, existing studies have primarily focused on asymmetric networks with the same parameterisation and the same width, sharing the same symmetry-breaking mechanism. We consider a harder setting: the two endpoints have different widths and are trained with independently sampled symmetry-breaking masks and constants. Hence, the networks differ in both *(i)* width and random initialisation and *(ii)* the specific mechanism by which parameter symmetries are broken. In this subsection, we examine whether low linear path loss barriers can still be achieved in this setting.

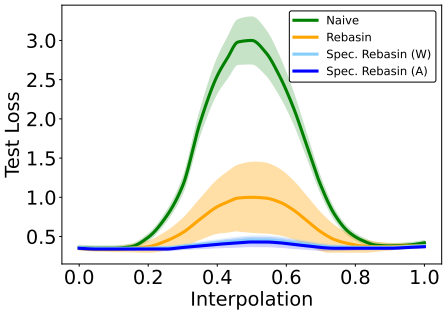


Figure 5: Interpolation barriers for asymmetric ResNet20 cross-width matching on CIFAR-10, from width $\times 4$ to width $\times 8$.

We consider the $\boldsymbol{W}$-Asym. construction from Lim et al. (2024) [37]. Following Lim et al. (2024) [37], we evaluate MLPs of widths 128 and 256 on the MNIST dataset [99], ResNet20 with width multipliers $\times 4$ and $\times 8$ on CIFAR-10, and GIN [100] of widths 128 and 256 on the ogbn-arXiv dataset [101]. For each architecture and task, we measure the midpoint test loss barrier, $\mathcal{L}(\frac{1}{2}\boldsymbol{\theta}^{A\to B} + \frac{1}{2}\boldsymbol{\theta}^{B}) - \frac{1}{2}(\mathcal{L}(\boldsymbol{\theta}^{A\to B}) + \mathcal{L}(\boldsymbol{\theta}^{B}))$. As in §5.2, we use zero-padding to construct function-preserving networks with matched parameter dimensionality before interpolation. We compare our method against the

[3] By a *function-preserving expansion*, we mean an embedding of the narrower endpoint into the wider architecture whose input-output function is unchanged before matching or interpolation. This describes preservation of the padded endpoint itself. Other exact widening constructions are possible, e.g., [97, 48]. We chose zero padding where it is exact as it is deterministic, training-free, and introduces no arbitrary clone allocation or outgoing-weight split.

same alignment baselines used in §5.1 and §5.2. We exclude weight matching, since the permutation symmetries exploited by weight matching are explicitly broken in asymmetric networks.

Fig. 5 shows the test loss along the linear interpolation path for asymmetric ResNet20 $\times 4 \rightarrow \times 8$ on CIFAR-10. Fig. 6 shows that width mismatch remains a nontrivial challenge even for asymmetric networks. Our methods substantially reduce the barrier. The activation-based variant achieves the lowest barriers on the more challenging CIFAR-10 and ogbn-arXiv benchmarks while remaining competitive on MNIST. Our weight-based variant is also consistently strong. By comparison, activation matching and OT fusion recover only part of the connectivity (on MNIST), while CCA-based variants remain less effective even with regularisation. These results suggest that spectral re-basin can achieve low linear path loss barriers between asymmetric networks when their widths differ.

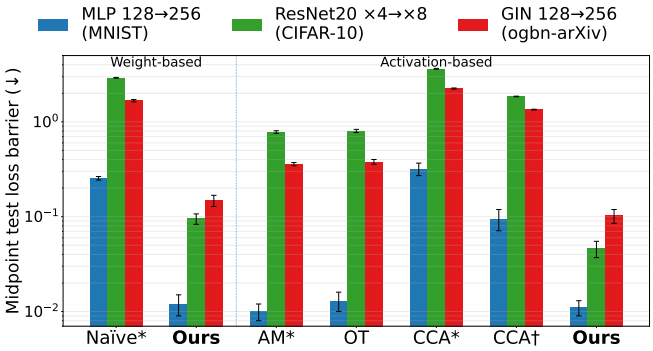


Figure 6: Matching asymmetric networks of different widths. The midpoint test loss barrier (↓) is reported. Methods marked with * use zero-padding to construct function-preserving networks with matched parameter dimensionality before interpolation, while methods marked with † use a regularised covariance matrix for CCA-based alignment across different widths.

6 Conclusions

We introduced *spectral re-basin*, an operator-theoretic framework for aligning neural weights. We represent each layer as a graph of hidden units, and study the geometry induced by the corresponding graph shift operator. This construction leads to graph functional symmetries within a single model and intertwining correspondences between different models. Our analysis shows that the spectral alignment residual controls the discrepancy between the rebased and target weights, and further yields an upper bound on the linear interpolation barrier. Empirically, spectral re-basin improves model matching across identical architectures, mismatched-width architectures, and asymmetric networks where explicit parameter symmetries are broken. These results suggest that weight alignment yielding low linear interpolation barriers need not rely on coordinate-level parameter symmetries alone, but can exploit broader compatibility between the spectral properties of hidden-unit representations.

One limitation of the current framework is that it does not directly address depth mismatch, or cases where one layer in one model should correspond to a composition of several layers in another model. In addition, while the framework provides flexible operator-level correspondences, general dense or rectangular transformations are *not* exact parameter symmetries of nonlinear networks in the same sense as permutations. The activation-based variant is *data dependent*: correspondences estimated on a finite alignment distribution may deteriorate on unseen or distribution-shifted inputs. Hence, the rebased model may only approximately preserve the original function. A promising future direction is to extend spectral re-basin beyond fixed layerwise correspondences by learning alignments jointly over layers, blocks, or computational subgraphs. One could also integrate spectral neuron-level correspondences with architecture-level matching, allowing one-to-many or many-to-one correspondences. Another promising future direction is to extend the spectral re-basin framework to task-vector merging, where task-specific parameter updates are combined to integrate capabilities from multiple fine-tuned models.

Acknowledgments

We thank Leibniz Supercomputing Centre (LRZ) for providing computational resources and the reviewing team for their valuable feedback to improve our work. YEL acknowledges support from the Schmidt Futures Israeli Women's Postdoctoral Award and the Viterbi Fellowship from the Faculty of Electrical and Computer Engineering at the Technion. TD acknowledges support from the Munich Center for Machine Learning (MCML) via the Thomas Bayes Postdoctoral Fellowship.

DH acknowledges support from the Munich Data Science Institute (MDSI) via the MDSI Doctoral Fellowship. This research was supported by an Alexander von Humboldt Professorship.

References

[1] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. *Advances in Neural Information Processing Systems*, 31, 2018.

[2] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[3] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International conference on machine learning*, pages 3259–3269. PMLR, 2020.

[4] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022.

[5] Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023.

[6] Gul Sena Altintas, Gregor Bachmann, Lorenzo Noci, and Thomas Hofmann. Disentangling linear mode-connectivity. *arXiv preprint arXiv:2312.09832*, 2023.

[7] Ekdeep Singh Lubana, Eric J Bigelow, Robert P Dick, David Krueger, and Hidenori Tanaka. Mechanistic mode connectivity. In *International Conference on Machine Learning*, pages 22965–23004. PMLR, 2023.

[8] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.

[9] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in Neural Information Processing Systems*, 30, 2017.

[10] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31, 2018.

[11] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

[12] Liwei Wang, Lunjia Hu, Jiayuan Gu, Zhiqiang Hu, Yue Wu, Kun He, and John Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. *Advances in Neural Information Processing Systems*, 31, 2018.

[13] Zhanpeng Zhou, Yongyi Yang, Xiaojiang Yang, Junchi Yan, and Wei Hu. Going beyond linear mode connectivity: The layerwise linear feature connectivity. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[14] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[15] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

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

[17] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

[18] Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. *Advances in Neural Information Processing Systems*, 33:15300–15311, 2020.

[19] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: REnormalizing permuted activations for interpolation repair. In *The Eleventh International Conference on Learning Representations*, 2023.

[20] Gregory Benton, Wesley Maddox, Sanae Lotfi, and Andrew Gordon Gordon Wilson. Loss surface simplexes for mode connecting volumes and fast ensembling. In *International Conference on Machine Learning*, pages 769–779. PMLR, 2021.

[21] David Yunis, Kumar Kshitij Patel, Pedro Henrique Pamplona Savarese, Gal Vardi, Jonathan Frankle, Matthew Walter, Karen Livescu, and Michael Maire. On convexity and linear mode connectivity in neural networks. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022.

[22] Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pages 129–135. Elsevier, 1990.

[23] An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

[24] Bo Zhao, Robin Walters, and Rose Yu. Symmetry in neural network parameter spaces. *Transactions on Machine Learning Research*, 2026.

[25] Johanni Brea, Berfin Simsek, Bernd Illing, and Wulfram Gerstner. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019.

[26] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, 29, 2016.

[27] Gabriel Eilertsen, Daniel Jönsson, Timo Ropinski, Jonas Unger, and Anders Ynnerman. Classifying the classifier: dissecting the weight space of neural networks. *arXiv preprint arXiv:2002.05688*, 2020.

[28] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020.

[29] Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Hyper-representations as generative models: Sampling unseen neural network weights. *Advances in Neural Information Processing Systems*, 35:27906–27920, 2022.

[30] Jonathan Kahana, Eliahu Horwitz, Imri Shuval, and Yedid Hoshen. Deep linear probe generators for weight space learning. *arXiv preprint arXiv:2410.10811*, 2024.

[31] Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning*, pages 25790–25816. PMLR, 2023.

[32] Aviv Shamsian, Aviv Navon, David W. Zhang, Yan Zhang, Ethan Fetaya, Gal Chechik, and Haggai Maron. Improved generalization of weight space networks via augmentations. In *Forty-first International Conference on Machine Learning*, 2024.

[33] Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Neural functional transformers. *Advances in Neural Information Processing Systems*, 36:77485–77502, 2023.

[34] Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clément Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning*, pages 9722–9732. PMLR, 2021.

[35] Bo Zhao, Nima Dehmamy, Robin Walters, and Rose Yu. Symmetry teleportation for accelerated optimization. *Advances in Neural Information Processing Systems*, 35:16679–16690, 2022.

[36] Bo Zhao, Robert M. Gower, Robin Walters, and Rose Yu. Improving convergence and generalization using parameter symmetries. In *The Twelfth International Conference on Learning Representations*, 2024.

[37] Derek Lim, Theo Putterman, Robin Walters, Haggai Maron, and Stefanie Jegelka. The empirical impact of neural parameter symmetries, or lack thereof. *Advances in Neural Information Processing Systems*, 37:28322–28358, 2024.

[38] Vincent Bürgin, Daniel Herbst, Ya-Wei Eileen Lin, and Stefanie Jegelka. Beyond structural symmetries: Linear mode connectivity via neuron identifiability. In *Forty-third International Conference on Machine Learning*, 2026.

[39] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

[40] Stefan Horoi, Albert Manuel Orozco Camacho, Eugene Belilovsky, and Guy Wolf. Harmony in diversity: Merging neural networks with canonical correlation analysis. In *Forty-first International Conference on Machine Learning*, 2024.

[41] Adrián Csiszárik, Péter Kőrösi-Szabó, Akos Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. *Advances in Neural Information Processing Systems*, 34:5656–5668, 2021.

[42] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[43] Dang Nguyen, Trang Nguyen, Khai Nguyen, Dinh Phung, Hung Bui, and Nhat Ho. On cross-layer alignment for model fusion of heterogeneous neural networks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[44] Moritz Imfeld, Jacopo Graldi, Marco Giordano, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. Transformer fusion with optimal transport. In *The Twelfth International Conference on Learning Representations*, 2024.

[45] Kenji Fukumizu and Shun-ichi Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural networks*, 13(3):317–327, 2000.

[46] Lemeng Wu, Dilin Wang, and Qiang Liu. Splitting steepest descent for growing neural architectures. *Advances in Neural Information Processing Systems*, 32, 2019.

[47] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. 2016.

[48] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International conference on machine learning*, pages 564–572. PMLR, 2016.

[49] Stephen Casper, Xavier Boix, Vanessa D'Amario, Ling Guo, Martin Schrimpf, Kasper Vinken, and Gabriel Kreiman. Frivolous units: Wider networks are not really that wide. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6921–6929, 2021.

[50] Diego Doimo, Aldo Glielmo, Sebastian Goldt, and Alessandro Laio. Redundant representations help generalization in wide neural networks. *Advances in Neural Information Processing Systems*, 35:19659–19672, 2022.

[51] Matthew Farrugia-Roberts. Functional equivalence and path connectivity of reducible hyperbolic tangent networks. *Advances in Neural Information Processing Systems*, 36:79502–79517, 2023.

[52] Elisenda Grigsby, Kathryn Lindsey, and David Rolnick. Hidden symmetries of ReLU networks. In *International Conference on Machine Learning*, pages 11734–11760. PMLR, 2023.

[53] Joachim Bona-Pellissier, François Bachoc, and François Malgouyres. Parameter identifiability of a deep feedforward ReLU neural network. *Machine learning*, 112(11):4431–4493, 2023.

[54] Olivier Laurent, Emanuel Aldea, and Gianni Franchi. A symmetry-aware exploration of bayesian neural network posteriors. In *The Twelfth International Conference on Learning Representations*, 2024.

[55] Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. *ACM Computing Surveys*, 57(9):1–52, 2025.

[56] Allan Zhou, Chelsea Finn, and James Harrison. Universal neural functionals. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[57] C. Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. In *International Conference on Learning Representations*, 2017.

[58] Rohith Kuditipudi, Xiang Wang, Holden Lee, Yi Zhang, Zhiyuan Li, Wei Hu, Rong Ge, and Sanjeev Arora. Explaining landscape connectivity of low-cost solutions for multilayer nets. *Advances in Neural Information Processing Systems*, 32, 2019.

[59] Stanislav Fort and Stanislaw Jastrzebski. Large scale structure of neural network loss landscapes. *Advances in Neural Information Processing Systems*, 32, 2019.

[60] Yujia Qin, Cheng Qian, Jing Yi, Weize Chen, Yankai Lin, Xu Han, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Exploring mode connectivity for pre-trained language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6726–6746, 2022.

[61] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[62] Linara Adilova, Maksym Andriushchenko, Michael Kamp, Asja Fischer, and Martin Jaggi. Layer-wise linear mode connectivity. In *The Twelfth International Conference on Learning Representations*, 2024.

[63] Sidak Pal Singh, Linara Adilova, Michael Kamp, Asja Fischer, Bernhard Schölkopf, and Thomas Hofmann. Landscaping linear mode connectivity. *arXiv preprint arXiv:2406.16300*, 2024.

[64] Mary Phuong and Christoph H. Lampert. Functional vs. parametric equivalence of re{lu} networks. In *International Conference on Learning Representations*, 2020.

[65] Damien Ferbach, Baptiste Goujaud, Gauthier Gidel, and Aymeric Dieuleveut. Proving linear mode connectivity of neural networks via optimal transport. In *International Conference on Artificial Intelligence and Statistics*, pages 3853–3861. PMLR, 2024.

[66] Fidel A Guerrero Peña, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, and Marco Pedersoli. Re-basin via implicit Sinkhorn differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20237–20246, 2023.

[67] Akira Ito, Masanori Yamada, and Atsutoshi Kumagai. Analysis of linear mode connectivity via permutation-based weight matching: With insights into other permutation search methods. In *The Thirteenth International Conference on Learning Representations*, 2025.

[68] Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant deep weight space alignment. In *Forty-first International Conference on Machine Learning*, 2024.

[69] Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals. *Advances in Neural Information Processing Systems*, 36:24966–24992, 2023.

[70] Derek Lim, Haggai Maron, Marc T. Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. In *The Twelfth International Conference on Learning Representations*, 2024.

[71] Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[72] Arkadi Piven, Yam Eitan, Guy Bar-Shalom, Fabrizio Frasca, Daniel Cremers, Thomas Dagès, Ron Kimmel, and Haggai Maron. Permutation-equivariant flow matching for alignment-free neural weight generation. *arXiv preprint arXiv:2609.32833*, 2026.

[73] Neha Verma and Maha Elbayad. Merging text transformer models from different initializations. *Transactions on Machine Learning Research*, 2024.

[74] Aditya Kumar Akash, Sixu Li, and Nicolás García Trillos. Wasserstein barycenter-based model fusion and linear mode connectivity of neural networks. *arXiv preprint arXiv:2210.06671*, 2022.

[75] Xuhong Li, Yves Grandvalet, Rémi Flamary, Nicolas Courty, and Dejing Dou. Representation transfer by optimal transport. *arXiv preprint arXiv:2007.06737*, 2020.

[76] Alexander Theus, Alessandro Cabodi, Sotiris Anagnostidis, Antonio Orvieto, Sidak Pal Singh, and Valentina Boeva. Generalized linear mode connectivity for transformers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[77] Akansh Maurya, Ya-Wei Eileen Lin, Stefanie Jegelka, Sebastian U Stich, and Rotem Mulayoff. Training-free transformer merging via sequential local operator alignment. *arXiv preprint arXiv:2610.06415*, 2026.

[78] Viet-Hoang Tran, Van-Hoan Trinh, Khanh Vinh Bui, and Tan Minh Nguyen. On linear mode connectivity of mixture-of-experts architectures. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[79] Donato Crisostomi, Marco Fumero, Daniele Baieri, Florian Bernard, and Emanuele Rodolà. C^2M^3: Cycle-consistent multi-model merging. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[80] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

[81] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. *Advances in neural information processing systems*, 17, 2004.

[82] Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

[83] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153):1–43, 2018.

[84] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[85] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[86] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[87] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[88] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (ToG)*, 31(4):1–11, 2012.

[89] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656, 2013.

[90] Nathanaël Perraudin and Pierre Vandergheynst. Stationary signal processing on graphs. *IEEE Transactions on Signal Processing*, 65(13):3462–3477, 2017.

[91] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[92] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[93] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[94] George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. ZipIt! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[96] Gabriel Peyré and Marco Cuturi. *Computational optimal transport: With applications to data science*. Now Foundations and Trends, 2019.

[97] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[98] Liu Ziyin, Yizhou Xu, and Isaac L. Chuang. Remove symmetries to control model expressivity. In *The Thirteenth International Conference on Learning Representations*, 2025.

[99] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[100] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[101] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[102] Vera Kůrková and Paul C. Kainen. Functionally equivalent feedforward neural networks. *Neural Computation*, 6(3):543–558, 1994.

[103] Charles Godfrey, Davis Brown, Tegan Emerson, and Henry Kvinge. On the symmetries of deep learning models and their internal representations. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[104] Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J. Burghouts, Efstratios Gavves, Cees G. M. Snoek, and David W. Zhang. Graph neural networks for learning equivariant representations of neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[105] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-hill New York, 1986.

[106] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*. Curran Associates Inc., 2016.

[107] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

[108] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[109] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[110] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional ARMA filters. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3496–3507, 2021.

[111] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[112] Mikhail Belkin and Partha Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

[113] Amit Singer. From graph to manifold Laplacian: The convergence rate. *Applied and Computational Harmonic Analysis*, 21(1):128–134, 2006.

[114] Matthias Hein, Jean-Yves Audibert, and Ulrike Von Luxburg. From graphs to manifolds–weak and strong pointwise consistency of graph Laplacians. In *International Conference on Computational Learning Theory*, pages 470–485. Springer, 2005.

[115] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[116] Daniel Ting, Ling Huang, and Michael I Jordan. An analysis of the convergence of graph Laplacians. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 1079–1086, 2010.

[117] Nicolás García Trillos, Moritz Gerlach, Matthias Hein, and Dejan Slepčev. Error estimates for spectral convergence of the graph Laplacian on random geometric graphs toward the laplace–beltrami operator. *Foundations of Computational Mathematics*, 20(4):827–887, 2020.

[118] Ulrike Von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, pages 555–586, 2008.

[119] Bastian Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. In *International Conference on Learning Representations*, 2019.

[120] Leander Girrbach, Anders Christensen, Ole Winther, Zeynep Akata, and A. Sophia Koepke. Addressing caveats of neural persistence with deep graph persistence. *Transactions on Machine Learning Research*, 2023.

[121] Thomas Gebhart, Paul Schrater, and Alan Hylton. Characterizing the shape of activation space in deep neural networks. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1537–1542. IEEE, 2019.

[122] Vitor AC Horta and Alessandra Mileo. Towards explaining deep neural networks through graph analysis. In *International Conference on Database and Expert Systems Applications*, pages 155–165. Springer, 2019.

[123] Vitor AC Horta, Ilaria Tiddi, Suzanne Little, and Alessandra Mileo. Extracting knowledge from deep neural networks through graph analysis. *Future Generation Computer Systems*, 120:109–118, 2021.

[124] Théo Lacombe, Yuichi Ike, Mathieu Carriere, Frédéric Chazal, Marc Glisse, and Yuhei Umeda. Topological uncertainty: Monitoring trained neural networks through persistence of activation graphs. *arXiv preprint arXiv:2105.04404*, 2021.

[125] Emilie Purvine, Davis Brown, Brett Jefferson, Cliff Joslyn, Brenda Praggastis, Archit Rathore, Madelyn Shapiro, Bei Wang, and Youjia Zhou. Experimental observations of the topology of convolutional neural network activations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9470–9479, 2023.

[126] Youjia Zhou, Yi Zhou, Jie Ding, and Bei Wang. Visualizing and analyzing the topology of neuron activations in deep adversarial training. 2023.

[127] Alexander Tong, David van Dijk, Jay Stanley, Guy Wolf, and Smita Krishnaswamy. Graph spectral regularization for neural network interpretability. 2019.

[128] Roy Velich, Arkadi Piven, David Bensaïd, Daniel Cremers, Thomas Dagès, and Ron Kimmel. Learning eigenstructures of unstructured data manifolds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 36201–36214, 2026.

[129] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and computational harmonic analysis*, 30(2):129–150, 2011.

[130] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[131] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[132] Ya-Wei Eileen Lin, Ronen Talmon, and Ron Levie. Equivariant machine learning on graphs with nonlinear spectral filters. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[133] Ya-Wei Eileen Lin and Ron Levie. Adaptive canonicalization with application to invariant anisotropic geometric networks. In *The Fourteenth International Conference on Learning Representations*, 2026.

[134] Edward Pearce-Crump and William Knottenbelt. Graph automorphism group equivariant neural networks. In *Forty-first International Conference on Machine Learning*, 2024.

[135] Dorian Nogneng and Maks Ovsjanikov. Informative descriptor preservation via commutativity for shape matching. In *Computer Graphics Forum*, volume 36, pages 259–267. Wiley Online Library, 2017.

[136] Jing Ren, Mikhail Panine, Peter Wonka, and Maks Ovsjanikov. Structured regularization of functional map computations. In *Computer Graphics Forum*, volume 38, pages 39–53. Wiley Online Library, 2019.

[137] Shinji Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE transactions on pattern analysis and machine intelligence*, 10(5):695–703, 2002.

[138] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[139] Huaijun Qiu and Edwin R Hancock. Graph matching and clustering using spectral partitions. *Pattern Recognition*, 39(1):22–34, 2006.

[140] Justin Solomon, Leonidas Guibas, and Adrian Butscher. Dirichlet energy for analysis and synthesis of soft maps. In *Computer Graphics Forum*, volume 32, pages 197–206. Wiley Online Library, 2013.

[141] Emanuele Rodolà, Luca Cosmo, Michael M Bronstein, Andrea Torsello, and Daniel Cremers. Partial functional correspondence. In *Computer graphics forum*, volume 36, pages 222–236. Wiley Online Library, 2017.

[142] Davide Eynard, Emanuele Rodola, Klaus Glashoff, and Michael M Bronstein. Coupled functional maps. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 399–407. IEEE, 2016.

[143] Judith Hermanns, Anton Tsitsulin, Marina Munkhoeva, Alex Bronstein, Davide Mottin, and Panagiotis Karras. GRASP: Graph alignment through spectral signatures. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 44–52. Springer, 2021.

[144] Judith Hermanns, Konstantinos Skitsas, Anton Tsitsulin, Marina Munkhoeva, Alexander Kyster, Simon Nielsen, Alexander M Bronstein, Davide Mottin, and Panagiotis Karras. Grasp: Scalable graph alignment by spectral corresponding functions. *ACM Transactions on Knowledge Discovery from Data*, 17(4):1–26, 2023.

[145] Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of the IEEE international conference on computer vision*, pages 5659–5667, 2017.

[146] Marco Fumero, Marco Pegoraro, Valentino Maiorca, Francesco Locatello, and Emanuele Rodolà. Latent functional maps: a spectral framework for representation alignment. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[147] Xinle Cheng, Congyue Deng, Adam W Harley, Yixin Zhu, and Leonidas Guibas. Zero-shot image feature consensus with deep functional maps. In *European Conference on Computer Vision*, pages 277–293. Springer, 2024.

[148] Akira Ito, Masanori Yamada, and Atsutoshi Kumagai. Linear mode connectivity between multiple models modulo permutation symmetries. In *Forty-second International Conference on Machine Learning*, 2025.

[149] Akira Ito, Masanori Yamada, Daiki Chijiwa, and Atsutoshi Kumagai. Do we really need permutations? impact of model width on linear mode connectivity. In *The Fourteenth International Conference on Learning Representations*, 2026.

[150] Harold Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics: methodology and distribution*, pages 162–190. Springer, 1992.

[151] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

[152] Viivi Uurtio, João M Monteiro, Jaz Kandola, John Shawe-Taylor, Delmiro Fernandez-Reyes, and Juho Rousu. A tutorial on canonical correlation methods. *ACM Computing Surveys (CSUR)*, 50(6):1–33, 2017.

[153] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

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

[155] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

Supplementary Material

A Preliminaries **23**

A.1 Permutation symmetries 23

A.2 Graph signal processing 23

A.3 Graph-functional versus neural-functional 23

B Extended related work **24**

B.1 Layers as graphs built from weights or activations 24

B.2 Graph operators, graph Laplacians, and graph signal processing 24

B.3 Graph functional symmetries and operator-level transformations 25

B.4 Graph alignment, spectral matching, and operator-level correspondences 26

B.5 Recent findings on permutations, multiple models, and width for linear path loss barriers 27

C Additional background **27**

C.1 Parameter symmetries 27

C.2 Functional calculus 29

D Technical comparison to OT fusion, CCA merge, and latent functional maps **30**

D.1 Comparison to OT fusion 30

D.2 Comparison to CCA merge 31

D.3 Comparison to latent functional maps 31

E Proofs **32**

E.1 Proof of Proposition 4 32

E.2 Proof of Proposition 5 34

E.3 Proof of Proposition 6 34

E.4 The effects of spectral truncation on Prop. 5 and Prop. 6 36

F Layerwise graph spectral symmetry and neural function-realisability **36**

G Additional remarks on spectral re-basin **40**

G.1 Commutation with the graph shift operator 40

G.2 Explicit solution to spectral re-basin 40

G.3 Mixed weight space spectral re-basin 41

G.4 Effect of matching direction 41

H Complexity comparison to weight matching and activation matching **43**

I Matching multiple networks **43**

I.1 Set up 43

I.2 A shared latent objective 44

I.3 Consistency of induced pairwise maps 44
I.4 Multi-network re-basin and pairwise barrier 44

J Experimental details **45**
J.1 Evaluation protocol 46
J.2 Datasets and base architectures 46
J.3 Construction of width-mismatched padded baselines 47
J.4 Spectral re-basin implementation 47
J.5 Baselines 49

K Additional experiments **50**
K.1 Ablation studies 50
K.2 Matching on identical architectures with different width expansions 52
K.3 End-point preserving after spectral re-basin 53
K.4 Effect of alignment direction 54
K.5 Matched-rank ablation of the graph-spectral contribution 54

A Preliminaries

A.1 Permutation symmetries

Let a neural network be parameterised by $\boldsymbol{\theta} \in \Theta$, where Θ denotes the parameter space. We write $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ for the function realized by the network with parameters $\boldsymbol{\theta}$, mapping inputs in $\mathcal{X}$ to outputs in $\mathcal{Y}$. A parameter symmetry is a transformation $\varphi : \Theta \to \Theta$ s.t. $f_{\varphi(\boldsymbol{\theta})} = f_{\boldsymbol{\theta}}$ for all $\boldsymbol{\theta} \in \Theta$ [22, 102, 103, 52, 24]. Consider a two-layer multi-layer perceptron (MLP) with parameters $\boldsymbol{\theta} = (\boldsymbol{W}_2, \boldsymbol{W}_1) \in \Theta := \mathbb{R}^{d_{\text{out}} \times m} \times \mathbb{R}^{m \times d_{\text{in}}}$, and output

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) := \boldsymbol{W}_2\, \sigma(\boldsymbol{W}_1 \boldsymbol{x}), \tag{15}$$

where σ is a nonlinear activation applied elementwise, and activations are defined by $\boldsymbol{z}_1 = \boldsymbol{x}$ and $\boldsymbol{z}_2 = \sigma(\boldsymbol{W}_1 \boldsymbol{z}_1)$. For any permutation $\pi \in S_m$ of the m hidden units, let $\boldsymbol{P}_\pi \in \{0,1\}^{m \times m}$ denote the associated permutation matrix. Then the transformation

$$(\boldsymbol{W}_2, \boldsymbol{W}_1) \mapsto (\boldsymbol{W}_2 \boldsymbol{P}_\pi^\top, \boldsymbol{P}_\pi \boldsymbol{W}_1) \tag{16}$$

is a parameter symmetry. Thus, permuting the hidden units and applying the inverse permutation in the adjacent layer leaves the realised function unchanged. For deeper networks, analogous permutations can be applied independently within each hidden layer. This permutation symmetry reflects that hidden units have no intrinsic ordering. Analogous symmetries also arise in more modern architectures, such as channel permutations in convolutional neural networks (CNNs), head permutations in multi-head attention, and, more broadly, automorphisms of the underlying computation graph [104, 70, 37].

A.2 Graph signal processing

Given a feature matrix $\boldsymbol{F} \in \mathbb{R}^{N \times d}$, we construct an undirected graph $G = ([N], E, \boldsymbol{A}, \boldsymbol{F})$, where each node i corresponds to a row of $\boldsymbol{F}$ and carries feature vector $\boldsymbol{F}_{i,:} \in \mathbb{R}^d$. The edge set E is built from local proximity $\|\boldsymbol{F}_{j,:} - \boldsymbol{F}_{i,:}\|_2$, e.g., a k-nearest-neighbour graph, and the adjacency matrix $\boldsymbol{A}$ is the Gaussian kernel $a_{ij} = \exp(-\|\boldsymbol{F}_{j,:} - \boldsymbol{F}_{i,:}\|_2^2/\sigma^2)$, where $\sigma > 0$ is a scale parameter, often chosen as the median pairwise distance scaled by a constant factor [80–82]. Let $\boldsymbol{D}$ denote the diagonal degree matrix and let $\boldsymbol{\Delta}$ be a self-adjoint graph shift operator (GSO), such as the graph Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ or the normalised Laplacian $\boldsymbol{N} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$ [84–87]. Writing the eigendecomposition as $\boldsymbol{\Delta} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^\top$, with eigenvalues ordered by increasing magnitude, the columns of $\boldsymbol{\Phi}$ define the graph Fourier basis and the eigenvalues are the graph frequencies [87, 89, 90]. For a graph signal $\boldsymbol{x} \in \mathbb{R}^N$, the graph Fourier transform is $\widehat{\boldsymbol{x}} = \boldsymbol{\Phi}^\top \boldsymbol{x}$, and a spectral filter h acts as $h(\boldsymbol{\Delta})\boldsymbol{x} = \boldsymbol{\Phi} h(\boldsymbol{\Lambda}) \boldsymbol{\Phi}^\top \boldsymbol{x}$, so that graph convolution is realized as pointwise multiplication in the spectral domain, in direct analogy with the classical convolution theorem [105–110]. Moreover, $\boldsymbol{\Delta}$ can be viewed as a discrete analogue of the Laplace-Beltrami operator $\Delta_{\text{LBO}} f = |g|^{-1/2} \partial_i (|g|^{1/2} g^{ij} \partial_j f)$ on a Riemannian manifold [111–115]. Note that the corresponding graph operator converges to the underlying manifold Laplace-Beltrami operator [112–114, 116–118].

Below, we give a dense graph functional shift that is not a permutation.

Example 7. Consider the complete graph on three vertices. Its symmetric normalised Laplacian is

$$\boldsymbol{\Delta} = \frac{3}{2}\boldsymbol{I} - \frac{1}{2}\mathbf{1}\mathbf{1}^\top. \tag{17}$$

The eigenvalue 0 has eigenspace $\operatorname{span}\{\mathbf{1}\}$, while the eigenvalue $3/2$ has the two-dimensional eigenspace $\mathbf{1}^\perp$. Hence any orthogonal rotation inside $\mathbf{1}^\perp$ commutes with Δ. For example,

$$\boldsymbol{U} = \frac{1}{3}\begin{bmatrix} 2 & -1 & 2 \\ 2 & 2 & -1 \\ -1 & 2 & 2 \end{bmatrix} \tag{18}$$

satisfies $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{U}\boldsymbol{\Delta} = \boldsymbol{\Delta}\boldsymbol{U}$. It is dense and is not a permutation matrix.

A.3 Graph-functional versus neural-functional

At layer l, the neuron graph G_l has one vertex per neuron or channel and graph-signal space

$$\mathcal{F}(G_l) = \{u : V_l \to \mathbb{R}\} \cong \mathbb{R}^{d_l}. \tag{19}$$

A neuron-indexed vector is a function on the graph vertices, where each descriptor coordinate and each activation vector across neurons is a graph signal. The graph shift operator $\boldsymbol{\Delta}_l$ acts on this

function space, while a correspondence $T_l : \mathcal{F}(G_l^A) \to \mathcal{F}(G_l^B)$ maps between two such spaces. This is the functional map sense in which we use the term *graph-functional*.

There is also an operator-functional interpretation. For any scalar spectral function h, the operator

$$h(\boldsymbol{\Delta}_l) = \boldsymbol{\Phi}_l h(\boldsymbol{\Delta}_l) \boldsymbol{\Phi}_l^\top \tag{20}$$

is a graph spectral functional, such as a diffusion, smoothing, or filtering operator. If an orthogonal map $\boldsymbol{U}_l$ commutes with $\boldsymbol{\Delta}_l$, then it also commutes with every such functional

$$h(\boldsymbol{\Delta}_l)\boldsymbol{U}_l = \boldsymbol{U}_l h(\boldsymbol{\Delta}_l). \tag{21}$$

Thus, the graph operator and all of its scalar spectral filters cannot distinguish graph signals related by U_l, motivating the term *graph-functional symmetry*.

This terminology does *not* imply preservation of the complete neural network input-output function $f_{\boldsymbol{\theta}}$. A graph-functional symmetry or alignment preserves, exactly or approximately, the operator structure of the constructed neuron graph. Exact neural-function preservation additionally requires architectural compatibility across adjacent layers and compatibility with the nonlinearity, for example $\sigma(\boldsymbol{Q}\boldsymbol{z}) = \boldsymbol{Q}\sigma(\boldsymbol{z})$ on the set of reachable preactivations (see §F). Thus, we use *symmetry* for square orthogonal commutants and use *alignment*, *correspondence*, or *intertwiner* for general nonorthogonal or rectangular maps.

B Extended related work

B.1 Layers as graphs built from weights or activations

A number of works represent neural networks or their hidden layers as graphs, but usually for purposes different from alignment and mode connectivity. On the weight side, [119] constructs weighted stratified graphs from network parameters and studies their persistent-topological summaries, showing that weight-derived graph structure can serve as a meaningful descriptor of network complexity. Subsequent work has refined and critiqued this perspective, for example, by analysing limitations of neural persistence and proposing deeper graph-based topological summaries of networks [120]. On the activation side, several papers build graphs whose structure is induced by neuron responses on data. [121] study activation graphs through persistent homology, while [122, 123] introduce co-activation graphs in which neurons are connected according to statistical dependence or correlation in their activations. Related graph constructions have also been used for monitoring and uncertainty estimation, as in activation-graph persistence methods [124], and for topological analysis and visualisation of learned activation spaces [125, 126]. A complementary line of work imposes graph structure directly on hidden units as an inductive bias or regularizer, for example, through graph-spectral regularisation of latent representations [127]. These works collectively support the idea that hidden layers admit meaningful graph structure, whether extracted from weights, activations, or their induced topology. Our approach builds on this broad viewpoint, but differs in objective: rather than using layer graphs only as summaries for interpretability, topology, or monitoring, we use them as the primary objects for defining layerwise correspondence operators between independently trained networks. Specifically, our graph is built either from the incoming weight vectors or from activations of hidden units, and the resulting graph operators are used to define the spectral transport that underlies re-basin and linear path loss barriers.

B.2 Graph operators, graph Laplacians, and graph signal processing

Our method is closely related to the broader literature on graph operators and graph signal processing, which provides the mathematical foundation for treating hidden-layer features as signals supported on a graph. The graph Laplacian and related graph shift operators are the standard objects used to encode the geometry of a graph through their spectra and eigenspaces [84]. In graph signal processing, signals defined on graph vertices can be analysed, filtered, and compared through the spectral decomposition of the underlying graph operator [86, 87, 128]. This point of view is central to our framework. Once a layer is represented as a graph of hidden units, built either from incoming weight vectors or from activations, the corresponding graph shift operator provides a canonical notion of layer geometry, and its eigenbasis provides the spectral coordinates in which we estimate alignments. From this perspective, our correspondence maps are not arbitrary linear transforms in feature space, but

operators acting on graph signals that are regularised to remain compatible with the graph structure itself.

The spectral viewpoint we adopt also connects to the functional calculus of graph operators developed in spectral graph theory and graph wavelet analysis. Spectral decompositions of graph Laplacians allow one to define graph Fourier modes, graph filters, and multiscale operators that depend on the intrinsic geometry of the graph rather than on any ambient Euclidean coordinates [129, 86, 87]. Our use of the graph operator is conceptually similar: the Laplacian spectrum is not merely a descriptive summary of the layer graph, but the coordinate system in which graph-compatible correspondences become simple to express. Specifically, the commutator term in spectral re-basin penalises transports that mix incompatible spectral components, so that alignment is encouraged to preserve the action of the graph operator rather than only the raw descriptors. The main difference of our method from standard matching objectives in weight space is that the graph operator is the structural object that defines what it means for two layers to be aligned.

More broadly, our work fits into the geometric deep learning perspective, where graphs, manifolds, and operators are used to express non-Euclidean structure, symmetry, and invariance in learning systems [130, 131]. However, while most of that literature studies how to design neural architectures that process graph-structured data, our goal is different. We use graph operators to study the internal organisation of neural networks themselves. In this sense, the graph is not the input domain but the hidden layer, and the relevant question is not how to perform learning on a graph, but how to compare, align, and transport two independently trained layers in a way that respects their intrinsic operator geometry. This operator-theoretic viewpoint is what makes graph signal processing natural for our setting: it supplies both the language for defining graph functional symmetries and the spectral tools needed to construct the layerwise transports used in spectral re-basin.

B.3 Graph functional symmetries and operator-level transformations

Our framework is also related to a broader line of work that studies symmetry and correspondence at the level of graph operators rather than only at the level of node permutations. A particularly close connection is the notion of *graph functional shifts* [132, 133], where the relevant symmetries are orthogonal operators commuting with a graph shift operator and therefore need not be induced by explicit node permutations. This perspective is important for our work because it provides the right language for moving beyond the classical re-basin story: once a layer is represented as a graph, the natural symmetry group is no longer the full permutation group, but the set of operator transformations that preserve the graph-functional structure of the layer. In a related direction, [134] emphasises that meaningful permutation symmetries on a graph are restricted by its automorphism group, reinforcing the point that arbitrary coordinate permutations are too broad once graph structure is taken seriously. Our notion of graph functional symmetry builds on this shift in viewpoint, but is tailored to neural alignment: rather than using graph symmetries to design equivariant architectures, we use them to define which layerwise transformations should count as structure-preserving correspondences between trained networks.

At the level of correspondence, our work is also closely connected to the functional map literature, where maps between objects are represented as linear operators between spaces of functions. The foundational work of [88] showed that correspondences between shapes can be encoded as small linear maps in spectral coordinates, and that structural constraints such as operator commutativity provide an effective way to regularise these maps. This idea was developed further by works such as [135], which enforce descriptor preservation through commutativity constraints, and [136], which introduces structured regularisation terms for functional map estimation. Our spectral re-basin framework inherits this operator-level philosophy, but applies it to a different setting. The carrier set is not a geometric surface or a sample manifold, but hidden units in a neural layer, represented as a graph built from weights or activations. The unknown correspondence is therefore not a map used for shape matching or retrieval, but a layerwise transport that reparameterizes one network in the coordinates of another. In this sense, our method imports the functional map tool into the re-basin problem: the spectral correspondence matrix plays the role of a functional map, the Laplacian commutator term enforces graph compatibility, and the resulting operator is then pushed back into parameter space to construct transported models and study linear path loss barriers.

What distinguishes our setting from these prior works is therefore not the use of spectral operators itself, but the downstream role of the operator. In graph symmetry and equivariance work, the operator

defines admissible transformations of graph signals or graph neural networks. In functional map methods, the operator is the main output and is judged by the quality of the recovered correspondence. In our case, the operator is an intermediate object inside a broader alignment pipeline: it is estimated so that two independently trained networks become comparable in a common layerwise coordinate system, and it is evaluated by the quality of the transported endpoint and the interpolation barrier that remains after rebasing. This explains why graph functional symmetries and operator-level commutativity are so useful for our method: they provide a principled extension of permutation symmetry to non-bijective, graph-structured neural correspondences, while remaining directly tied to the re-basin and linear path loss barriers objectives of this paper.

B.4 Graph alignment, spectral matching, and operator-level correspondences

Our notion of layerwise alignment is related to a long line of work on graph matching and spectral correspondence, but differs in both its object of alignment and its downstream goal. Classical graph matching methods sought node correspondences directly, often through combinatorial formulations or spectral relaxations of matching objectives. Early spectral approaches such as [137] showed that graph spectra can be useful for hard graph matching, while later work surveyed graph matching as a broad pattern recognition problem and developed spectral matching procedures based on adjacency or Laplacian eigenvectors and partitions [138, 139]. Similar to these approaches, we use graph operators and their spectra to move beyond raw coordinate comparisons. However, our goal is not to recover a node permutation between two external graphs. Instead, the graphs are internal neural layers, and the purpose of the spectral correspondence is to define a transport that makes two trained networks comparable for re-basin and linear path loss barriers.

A closer conceptual related work is the operator-level correspondence view developed in the functional map literature. Functional maps represent correspondences as linear maps between spaces of functions rather than as pointwise assignments, and regularise these maps through compatibility with intrinsic operators such as Laplace-Beltrami operators [88]. In this language, our condition $\boldsymbol{\Delta}_l^B \boldsymbol{T}_l \approx \boldsymbol{T}_l \boldsymbol{\Delta}_l^A$ is naturally interpreted as an intertwining relation between two graph signal spaces. Subsequent work in functional maps expanded this perspective to descriptor-preserving commutativity, structured regularisation, soft correspondences, partial correspondences, and coupled bidirectional maps [135, 136, 140–142]. These works are directly relevant to our setting because they show that useful correspondences need not be exact bijections and need not be represented as hard pointwise matches. They can instead be structured linear operators constrained by the intrinsic operators of the domains. Our method adopts this operator-theoretic viewpoint, but applies it to a different setting: the carrier sets are graphs of hidden units built from weights or activations, and the learned operator is ultimately used to transport model parameters rather than only to recover a geometric or combinatorial correspondence.

Another line of related work is graph matching and operator correspondence provided by GRASP [143, 144], which transfers the functional-map philosophy from continuous shapes to discrete graphs. In GRASP, graph alignment is formulated as the problem of finding a mapping between functions on graph nodes, using Laplacian eigenvalues and eigenvectors to capture multiscale structural information and then recovering node alignments from these corresponding functions [143, 144]. This is close to our framework, because it shows that graph alignment can be posed directly at the level of graph operators rather than only through node-wise assignments. The key difference is again the downstream role. In GRASP and related graph-alignment methods, the recovered map is evaluated as a graph correspondence in its own right. In our case, the correspondence operator is only an intermediate object inside a broader re-basin pipeline: it is estimated so that one trained network can be transported into the coordinates of another, or into a shared comparison architecture, and is then evaluated by the quality of the transported endpoint and the interpolation barrier that remains after rebasing.

More recent extensions of the functional-map viewpoint into learning and representation alignment further support this perspective. Deep functional maps cast dense correspondence as structured prediction in the space of functional operators, while latent functional maps extend the same formalism to the alignment of learned latent spaces and representations [145–147]. These works reinforce the idea that operator-level correspondences can be estimated in settings far beyond classical shape matching. Our method is closest to them in spirit, but differs in three respects that are central to the present paper: *(i)* the domains being aligned are neural layers represented as graphs of hidden units rather than shapes, generic graphs, or latent sample clouds, *(ii)* the learned map is designed to

support parameter transport rather than only representation transfer, and *(iii)* the quality of the map is judged by re-basin and linear path loss barriers rather than by retrieval or matching accuracy. This provides the mathematical backdrop for our notion of intertwining correspondence: spectral re-basin brings spectral graph matching and functional-map regularisation into the *weight-space alignment* problem, and uses the resulting operator not as an end in itself but as the mechanism for constructing meaningful low-barrier interpolations between trained networks.

B.5 Recent findings on permutations, multiple models, and width for linear path loss barriers

Recent theoretical work by Ito et al. [67, 148, 149] further clarifies the role of permutations in linear mode connectivity and is closely related to the perspective adopted in our paper. Their analysis of permutation-based weight matching shows that its effectiveness is not well explained by a simple reduction of Euclidean distance between two models, rather, the important effect is that the permutation reorganises structurally relevant directions in weight space, so that the interpolated model better preserves the functional directions that matter for prediction [67]. This interpretation is compatible with our viewpoint: what matters for connectivity is not arbitrary closeness in parameter space, but whether the transformation aligns the directions and layerwise structures that control model behaviour. Ito et al. also extend the permutation-based picture from two models to the multi-model setting, showing that pairwise permutation search does not automatically yield a coherent common basin for several independently trained models and that multi-model connectivity requires a coordinated alignment procedure [148]. This resonates with our extension of spectral re-basin to multiple networks (see §I), where the objective is likewise to move beyond isolated pairwise matches and construct a shared comparison space. At the same time, their recent width-based analysis shows that sufficiently wide networks can exhibit strong linear mode connectivity even without explicitly searching for permutations [149]. This result is complementary to our framework: it suggests that permutations are not the only mechanism behind low-barrier interpolation, while our method asks what more general notion of layerwise correspondence should replace permutations once one leaves the strict same-width setting. These works sharpen the current picture of linear path loss barriers: permutation matching can be structurally meaningful, multi-model connectivity demands coordinated alignment, and width can itself promote connectivity, but these observations still leave open the broader non-bijective setting that motivates our operator-level, graph-based notion of correspondence.

C Additional background

C.1 Parameter symmetries

We briefly review the classical notion of parameter symmetry in neural networks [24] and explain how it relates to re-basin. The main argument is that many apparently different parameter vectors can represent exactly the same function. As a consequence, raw Euclidean distance in weight space is not a faithful measure of functional proximity [67], and interpolation barriers can arise from coordinate choices rather than from true separation in function space. This is the motivation of permutation-based re-basin, and also the reason why our method seeks a more general notion of layerwise correspondence once exact coordinate symmetries become too restrictive.

For simplicity, we consider feed-forward networks without biases. Let

$$\boldsymbol{W}_l \in \mathbb{R}^{d_l \times d_{l-1}} \text{ for } l = 1, \dots, L, \tag{22}$$

and define the network by

$$\boldsymbol{z}_0(\boldsymbol{x}) = \boldsymbol{x}, \quad \boldsymbol{z}_l(\boldsymbol{x}) = \sigma_l(\boldsymbol{W}_l \boldsymbol{z}_{l-1}(\boldsymbol{x})) \text{ for } l = 1, \dots, L-1, \tag{23}$$

with output

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}_L \boldsymbol{z}_{L-1}(\boldsymbol{x}), \quad \boldsymbol{\theta} = (\boldsymbol{W}_1, \dots, \boldsymbol{W}_L). \tag{24}$$

Throughout, each hidden activation σ_l acts coordinatewise.

Proposition 8. *Let $\boldsymbol{T}_0 = \boldsymbol{I}_{d_0}$ and $\boldsymbol{T}_L = \boldsymbol{I}_{d_L}$. In addition, let $\boldsymbol{T}_l \in \mathbb{R}^{d_l \times d_l}$ be invertible for $l = 1, \dots, L-1$. Suppose that the activations commute with these hidden coordinate changes, i.e.,*

$$\sigma_l(\boldsymbol{T}_l \boldsymbol{z}) = \boldsymbol{T}_l \sigma_l(\boldsymbol{z}) \text{ for all } \boldsymbol{z} \in \mathbb{R}^{d_l}, \quad , l = 1, \dots, L-1. \tag{25}$$

Let the transformed weights be denoted by $\widetilde{\boldsymbol{W}}_l = \boldsymbol{T}_l \boldsymbol{W}_l \boldsymbol{T}_{l-1}^{-1}$ for $l = 1, \dots, L$. Then, $f_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for all $\boldsymbol{x}$.

Proof. We prove by induction that

$$\widetilde{\boldsymbol{z}}_l(\boldsymbol{x}) = \boldsymbol{T}_l \boldsymbol{z}_l(\boldsymbol{x}) \text{ for } l = 0, \ldots, L-1,$$

where $\widetilde{\boldsymbol{z}}_l$ is the hidden activations of the transformed network. The claim is trivial for $l = 0$, since $\boldsymbol{T}_0 = \boldsymbol{I}_{d_0}$. Assume it holds at layer $l-1$, then

$$\begin{aligned}
\widetilde{\boldsymbol{z}}_l(\boldsymbol{x}) &= \sigma_l(\widetilde{\boldsymbol{W}}_l \widetilde{\boldsymbol{z}}_{l-1}(\boldsymbol{x})) \\
&= \sigma_l(\boldsymbol{T}_l \boldsymbol{W}_l \boldsymbol{T}_{l-1}^{-1} \boldsymbol{T}_{l-1} \boldsymbol{z}_{l-1}(\boldsymbol{x})) \\
&= \sigma_l(\boldsymbol{T}_l \boldsymbol{W}_l \boldsymbol{z}_{l-1}(\boldsymbol{x})) \\
&= \boldsymbol{T}_l \sigma_l(\boldsymbol{W}_l \boldsymbol{z}_{l-1}(\boldsymbol{x})) \\
&= \boldsymbol{T}_l \boldsymbol{z}_l(\boldsymbol{x}).
\end{aligned}$$

At the output layer,

$$\begin{aligned}
f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}) &= \widetilde{\boldsymbol{W}}_L \widetilde{\boldsymbol{z}}_{L-1}(\boldsymbol{x}) \\
&= \boldsymbol{T}_L \boldsymbol{W}_L \boldsymbol{T}_{L-1}^{-1} \boldsymbol{T}_{L-1} \boldsymbol{z}_{L-1}(\boldsymbol{x}) \\
&= \boldsymbol{W}_L \boldsymbol{z}_{L-1}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x}),
\end{aligned}$$

□

Prop. 8 shows that exact symmetry arises from hidden coordinate changes that commute with the nonlinear activation.

Corollary 9 (Hidden-unit permutation symmetry)**.** *Let $\boldsymbol{P}_l \in \mathbb{R}^{d_l \times d_l}$ be permutation matrices for $l = 1, \ldots, L-1$, and let $\boldsymbol{P}_0 = \boldsymbol{I}_{d_0}$ and $\boldsymbol{P}_L = \boldsymbol{I}_{d_L}$. Since coordinatewise activations commute with permutations, i.e.,*

$$\sigma_l(\boldsymbol{P}_l \boldsymbol{z}) = \boldsymbol{P}_l \sigma_l(\boldsymbol{z}), \tag{26}$$

the transformed weights $\widetilde{\boldsymbol{W}}_l = \boldsymbol{P}_l \boldsymbol{W}_l \boldsymbol{P}_{l-1}^\top$ for $l = 1, \ldots, L$ leads to an exact parameter symmetry, i.e., $f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for all $\boldsymbol{x}$.

This is the symmetry class used in weight-based permutation [4, 5]. Two independently trained models may therefore represent similar functions but use different hidden-unit orderings, which makes them look artificially far apart in parameter space until one network is permuted into the coordinates of the other.

Corollary 10 (Positive diagonal scaling symmetry)**.** *Assume that every hidden activation σ_l is positively homogeneous, that is,*

$$\sigma_l(\alpha \boldsymbol{z}) = \alpha \sigma_l(\boldsymbol{z}) \text{ for all } \alpha > 0, \tag{27}$$

e.g., ReLU. Let $\boldsymbol{D}_l \in \mathbb{R}^{d_l \times d_l}$ be diagonal matrices with strictly positive diagonal entries for $l = 1, \ldots, L-1$ and set $\boldsymbol{D}_0 = \boldsymbol{I}_{d_0}$ and $\boldsymbol{D}_L = \boldsymbol{I}_{d_L}$. Then,

$$\sigma_l(\boldsymbol{D}_l \boldsymbol{z}) = \boldsymbol{D}_l \sigma_l(\boldsymbol{z}), \tag{28}$$

and the transformed weights $\widetilde{\boldsymbol{W}}_l = \boldsymbol{D}_l \boldsymbol{W}_l \boldsymbol{D}_{l-1}^\top$ for $l = 1, \ldots, L$ leads to $f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for all $\boldsymbol{x}$.

This corollary highlights that even in standard multilayer networks, there may be continuous exact symmetries, not only discrete ones. From the standpoint of alignment, this means that the space of function-preserving reparameterizations can be richer than hidden-unit relabeling alone. Our method is motivated by an even broader point: once one moves to different widths, split-clone relations, or deliberately asymmetric architectures, exact coordinate-level symmetries may disappear or become too small to explain the observed connectivity, and one needs a more flexible layerwise notion of correspondence.

Although the classical parameter symmetry viewpoint explains why raw weight-space comparisons can be misleading, it is too restrictive for the settings studied in this paper. First, exact layerwise changes of coordinates as in Eq. (25) require square invertible maps $\boldsymbol{T}_l$. This already rules out width mismatch, where $d_l^A \neq d_l^B$ and no such exact coordinate symmetry can exist between the two layers. Second, exact parameter symmetries are coordinate-level statements. They say that two parameterisations implement the same function, but they do not tell us how to align two different functions that have similar hidden structure without being equivalent. Third, recent work suggests that strong linear mode connectivity may persist even when explicit permutation symmetry is weakened or removed, for example in sufficiently wide or asymmetric models [149, 37, 98]. This indicates that

connectivity should not be identified with permutation symmetry alone. These limitations motivate the viewpoint adopted in our paper. We keep the layerwise transport form inspired by exact symmetries, but replace exact coordinate commutation by graph-functional compatibility. In this way, spectral re-basin can be viewed as a generalisation of classical parameter symmetry: when exact hidden-unit relabelings are available, they are included as special cases, while in non-bijective or symmetry-broken settings the method seeks a broader operator-level correspondence that still preserves the layerwise geometry relevant for interpolation.

C.2 Functional calculus

We briefly review the functional calculus and the spectral viewpoint adopted throughout the paper.

Spectral decomposition of self-adjoint operators. Let $\mathbf{\Delta}$ be a self-adjoint operator on a finite-dimensional Hilbert space, for example a real symmetric matrix $\mathbf{\Delta} \in \mathbb{R}^{N\times N}$ or a Hermitian matrix $\mathbf{\Delta} \in \mathbb{C}^{N\times N}$. By the spectral theorem, there exists an orthonormal basis of eigenvectors and a diagonal matrix of real eigenvalues such that

$$\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^*, \qquad \mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_N),$$

where the columns $\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_N$ of $\mathbf{\Phi}$ satisfy

$$\mathbf{\Delta}\boldsymbol{\phi}_i = \lambda_i \boldsymbol{\phi}_i.$$

Equivalently, $\mathbf{\Delta}$ admits the spectral expansion

$$\mathbf{\Delta} = \sum_{i=1}^{N} \lambda_i \boldsymbol{P}_i, \qquad \boldsymbol{P}_i := \boldsymbol{\phi}_i \boldsymbol{\phi}_i^*,$$

where $\boldsymbol{P}_i$ is the orthogonal projector onto the i-th eigendirection.

In the graph setting, $\mathbf{\Delta}$ is typically chosen to be a graph shift operator, such as a graph Laplacian, a normalised Laplacian, or a symmetrised adjacency-type operator. Its eigenvectors provide the graph Fourier basis, and its eigenvalues play the role of graph frequencies [86, 87, 84].

Functional calculus. Once $\mathbf{\Delta}$ has been diagonalised, any scalar function defined on its spectrum can be applied to the operator by acting on the eigenvalues. Namely, if $f : \mathbb{R} \to \mathbb{C}$ is defined on the spectrum $\sigma(\mathbf{\Delta}) = \{\lambda_1, \dots, \lambda_N\}$, then

$$f(\mathbf{\Delta}) := \mathbf{\Phi} f(\mathbf{\Lambda}) \mathbf{\Phi}^*, \qquad f(\mathbf{\Lambda}) := \mathrm{diag}(f(\lambda_1), \dots, f(\lambda_N)). \tag{29}$$

Equivalently,

$$f(\mathbf{\Delta}) = \sum_{i=1}^{N} f(\lambda_i) \boldsymbol{P}_i. \tag{30}$$

Thus one may think of $f(\mathbf{\Delta})$ as being obtained in three steps: diagonalize $\mathbf{\Delta}$, apply f to the eigenvalues, and transform back to the original basis. This construction is basis-independent and depends only on the operator and the function f. Note that functions of $\mathbf{\Delta}$ always commute with $\mathbf{\Delta}$:

$$f(\mathbf{\Delta})\mathbf{\Delta} = \mathbf{\Delta} f(\mathbf{\Delta}),$$

since both are diagonal in the same eigenbasis.

Graph Fourier transform and scalar spectral filters. Let $\mathbf{\Delta}$ be a graph shift operator with eigendecomposition

$$\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$$

in the real symmetric case. For a graph signal $x \in \mathbb{R}^N$, its graph Fourier coefficients are

$$\widehat{\boldsymbol{x}} = \mathbf{\Phi}^\top \boldsymbol{x}.$$

A scalar spectral filter is obtained by choosing a function $f : \mathbb{R} \to \mathbb{R}$ and defining

$$f(\mathbf{\Delta})x = \mathbf{\Phi} f(\Lambda) \mathbf{\Phi}^\top \boldsymbol{x} = \sum_{i=1}^{N} f(\lambda_i)\, \boldsymbol{\phi}_i \boldsymbol{\phi}_i^\top x.$$

Thus each graph Fourier mode $\boldsymbol{\phi}_i$ is preserved as a spatial direction, while its amplitude is reweighted by the scalar response $f(\lambda_i)$. In this sense, spectral filtering on a graph is the direct analogue of frequency-domain filtering in classical Fourier analysis [105, 129].

Note that if

$$f(\lambda) = \sum_{k=0}^{K} a_k \lambda^k,$$

then

$$f(\boldsymbol{\Delta}) = \sum_{k=0}^{K} a_k \boldsymbol{\Delta}^k.$$

This provides an operator expression in the original vertex basis and shows that many graph filters can be implemented without computing the full eigendecomposition.

D Technical comparison to OT fusion, CCA merge, and latent functional maps

In this section, we distinguish our approach in detail from OT fusion [39], CCA merge [40], and latent functional maps [146]: although all aim to generalise beyond permutations, they rely on different structural priors and solve different alignment problems.

D.1 Comparison to OT fusion

In this subsection, we compare optimal-transport (OT) fusion [39, 65, 44] and spectral re-basin in the setting studied in this paper. The two approaches address the same high-level question: how can one align two layers when the correspondence between hidden units is not necessarily bijective? However, they formalise this question in different mathematical languages, and this difference propagates through the entire pipeline.

At a coarse level, OT fusion views a layer as a finite collection of atoms, typically neurons or channels, with a local pairwise transport cost. The unknown object is a coupling that redistributes mass from the source atoms to the target atoms while satisfying marginal constraints. Spectral re-basin, by contrast, views a layer as a graph of hidden units endowed with an intrinsic operator geometry. The unknown object is a linear correspondence operator that should both align the chosen descriptors and approximately preserve the action of the graph shift operator. The distinction can be summarised informally as follows: OT fusion matches mass under a local cost, whereas spectral re-basin matches linear operators under a graph-functional compatibility condition. This does not mean that OT is ignorant of geometry. OT can encode geometry through the choice of the cost matrix, yet once the pairwise cost and marginals are fixed, the OT optimisation problem has no additional global structural condition analogous to the intertwining penalty used in spectral re-basin.

Conceptual distinction. The first conceptual difference lies in what is considered invariant under the alignment. In OT, what matters is whether mass can be transported cheaply from source atoms to target atoms. The notion of similarity is therefore local and pairwise: two neurons are considered compatible if their transport cost is small, and a whole alignment is considered good if the total transport cost is small while respecting the row and column marginal constraints. In spectral re-basin, the notion of similarity is more global. A layer is treated as a graph whose hidden units carry signals and whose structure is represented by a graph shift operator $\boldsymbol{\Delta}_l$. The goal is to align both features, such as activations or weight vectors, and the way the layer behaves as a graph signal space. This is why the correspondence map $\boldsymbol{T}_l$ is encouraged to satisfy an intertwining relation of the form $\boldsymbol{\Delta}_l^B \boldsymbol{T}_l \approx \boldsymbol{T}_l \boldsymbol{\Delta}_l^A$. Equivalently, in the spectral basis, $\boldsymbol{\Lambda}_l^B \boldsymbol{C}_l \approx \boldsymbol{C}_l \boldsymbol{\Lambda}_l^A$, where $\boldsymbol{T}_l = \boldsymbol{\Phi}_l^B \boldsymbol{C}_l {\boldsymbol{\Phi}_l^A}^\top$. Thus OT and spectral re-basin preserve different kinds of structure. OT preserves marginals and low local cost. Spectral re-basin preserves descriptor agreement with approximate graph functional symmetry.

A second conceptual difference is the admissible form of the correspondence. OT couplings are nonnegative by construction and are constrained to preserve mass. Spectral re-basin places no such constraints on the correspondence map. The learned operator may change sign, may be rectangular, and need not preserve total mass. Note that this flexibility is important in the width-mismatch settings.

A third difference is what each method is trying to achieve downstream. In OT fusion, the coupling is usually used directly to average or fuse parameters. In spectral re-basin, the learned correspondence is used to transport one model into the coordinate system of another model, or into a shared comparison architecture, before studying endpoint quality and linear interpolation barriers.

D.2 Comparison to CCA merge

We next compare CCA merge [40] with spectral re-basin. Note that both CCA merge and our spectral re-basin share the same motivation at a high level: both start from the observation that one-to-one permutation matching is often too restrictive [4, 5, 94]. In many layers, especially when features are distributed or partially redundant, a useful alignment may require dense linear combinations of neurons rather than a strict relabeling of coordinates. However, once one moves beyond permutations, the two methods generalise the classical re-basin story in different directions [9, 11].

CCA merge generalises permutations by searching for correlated subspaces [150–152], following from the observation that two models need not match neuron by neuron, and instead, they may realise similar representations after projecting into suitable linear combinations of their features (based on covariance matrices). Our spectral re-basin generalises permutations in a different way. Rather than centring the problem on cross-covariance, it asks for a correspondence operator that aligns the chosen descriptors while also approximately preserving the graph-functional structure of the layer. Thus the contrast is not simply "dense" versus "sparse" matching. It is a contrast between two different structural notions of alignment: covariance geometry in CCA, and operator geometry in spectral re-basin.

Conceptual distinction. The first conceptual difference is how each method models a hidden layer. In CCA merge, a layer is treated as a feature space whose meaningful directions may be dense linear combinations of neurons. The object of interest is the covariance structure of these features across examples, and alignment is defined by finding directions in the two models that are maximally correlated. Thus, CCA merge focuses on shared subspaces: if two models have the same representation up to different dense linear mixtures, CCA is designed to recover those mixtures.

In our spectral re-basin, a layer is treated as both a feature space and a graph signal space. The hidden units form a graph structure built from activations or weights, and this graph is represented by a graph shift operator $\boldsymbol{\Delta}_l$. The correspondence is then asked to do two things at the same time: fit the descriptors of the two layers and respect the action of the graph shift operators. In other words, spectral re-basin is looking for directions that remain meaningful with respect to the layerwise relational geometry.

This leads to a second conceptual distinction. CCA is most naturally suited to settings where the relevant common structure is distributed across many neurons and can be captured by a correlated latent subspace. Spectral re-basin is most naturally suited to settings where the global organisation of the layer matters, for example, when one wants the alignment to remain coherent under width mismatch, split-clone structure, or other non-bijective relations that are better described at the level of graph operators of the functionals than at the level of pairwise correlations alone.

D.3 Comparison to latent functional maps

Finally, we compare Latent Functional Maps (LFM) [146] and our spectral re-basin framework in detail. The comparison is important because the two methods share part of the same spectral toolbox [88, 112]: both use graph Laplacians, spectral bases, and the functional-map formalism. For that reason, a superficial reading could suggest that spectral re-basin is simply LFM applied to a different dataset or with a different loss. That is not the right interpretation. The two approaches share a mathematical language, but they act on different carrier sets, optimise different objectives, and are designed for different downstream questions.

At a high level, LFM studies the relation between representational spaces. Given two latent spaces X_1 and X_2, it treats samples from those spaces as points lying on two manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$, approximates these manifolds by kNN graphs built on the *samples*, and then estimates a functional map between the corresponding spaces of functions. The resulting map is used to compare spaces, recover correspondences, and transfer representations in tasks such as retrieval and stitching [153, 42, 154]. Spectral re-basin asks a different question. Our nodes are not samples in latent space, instead, they are *hidden units within a layer*, $\boldsymbol{W}_l^X \in \mathbb{R}^{d_l^X \times d_{l-1}^X}$, where the d_l^X rows of $\boldsymbol{W}_l^X$ are the incoming weight vectors of the neurons in that layer. The graph is not intended to approximate a sample manifold, but to encode the relational geometry of neurons. The functional map is therefore not an end in itself. It is an intermediate object used to construct a layerwise transport, rebase one

network into the coordinates of another (or a shared comparison architecture), and then analyse the resulting endpoint and linear interpolation path.

In short, LFM asks how two representational spaces are related. On the other hand, spectral re-basin is a *weight-space learning* method [26–30, 68, 32, 33, 75]: it learns layerwise correspondence operators only as an intermediate step toward transporting model parameters into a common comparison architecture, where linear interpolation barriers can be analysed. Hence, the scientific object being aligned is different.

Conceptual distinction. The distinction from LFM to our spectral re-basin explains why the theoretical statements in this paper are specific to low linear path loss barriers.

First, our results are about transporting model parameters. The map estimated by spectral re-basin is for the transported endpoint $\boldsymbol{W}_l^{A\to B} = \boldsymbol{T}_l \boldsymbol{W}_l^A \boldsymbol{T}_{l-1}^\dagger$ and by the loss barrier along the induced linear path. LFM does not by itself provide such a parameter transport or such a barrier analysis, because its main object is a relation between representational spaces rather than a relation between parameterisations of full networks.

Second, our non-bijective theory is layerwise and support-aware. In width-mismatch or split-clone settings, the relevant object is often a rectangular transport supported on a non-bijective relation between hidden units. LFM is designed primarily for representation alignment, correspondence recovery, and transfer between latent spaces, and partial or non-bijective mappings are explicitly noted there as a limitation of the current framework rather than the main organising use case.

Third, we demonstrate a utility for operator compatibility without exact parameter symmetry. The learned map is evaluated by whether it produces a good transported model and a low-barrier interpolation path, not by retrieval, stitching, or latent transfer quality. LFM is naturally coupled to the latter tasks, whereas spectral re-basin is naturally coupled to the former.

E Proofs

We include the proof of our theoretical result for the proposed spectral re-basin presented in §4. The numbering of the statements follows the numbering used in the paper. For convenience, we restate the claim of each statement before giving its proof.

E.1 Proof of Proposition 4

Proposition 4. *Let $\boldsymbol{\Delta}_l^X = \boldsymbol{\Phi}_l^X \boldsymbol{\Lambda}_l^X \boldsymbol{\Phi}_l^{X\top}$ with $X \in \{A, B\}$. Graph functional alignments are the transformations $\boldsymbol{T}_l$ that satisfy in the spectral domain*

$$\boldsymbol{\Lambda}_l^B \boldsymbol{C}_l = \boldsymbol{C}_l \boldsymbol{\Lambda}_l^A,$$

where $\boldsymbol{C}_l = \boldsymbol{\Phi}_l^{B\top} \boldsymbol{T}_l \boldsymbol{\Phi}_l^A$ is the spectral representation of $\boldsymbol{T}_l$ in the GSO eigenbasis. In addition, $\|\boldsymbol{\Delta}_l^B \boldsymbol{T}_l - \boldsymbol{T}_l \boldsymbol{\Delta}_l^A\|_F^2 = \sum_{i,j} (\lambda_{l,i}^B - \lambda_{l,j}^A)^2 (\boldsymbol{C}_l)_{ij}^2$.

Proof. Fix a layer index l. For readability, we suppress the subscript l throughout the proof and write $\boldsymbol{\Delta}^A, \boldsymbol{\Delta}^B, \boldsymbol{\Phi}^A, \boldsymbol{\Phi}^B, \boldsymbol{\Lambda}^A, \boldsymbol{\Lambda}^B, \boldsymbol{T}, \boldsymbol{C}$. Thus

$$\boldsymbol{\Delta}^X = \boldsymbol{\Phi}^X \boldsymbol{\Lambda}^X \boldsymbol{\Phi}^{X\top}, \qquad X \in \{A, B\}.$$

Since the eigenbases are assumed to be orthonormal, we have

$$\boldsymbol{\Phi}^{X\top} \boldsymbol{\Phi}^X = \boldsymbol{I}, \qquad \boldsymbol{\Phi}^X \boldsymbol{\Phi}^{X\top} = \boldsymbol{I}.$$

The spectral representation of $\boldsymbol{T}$ is

$$\boldsymbol{C} = \boldsymbol{\Phi}^{B\top} \boldsymbol{T} \boldsymbol{\Phi}^A.$$

Multiplying this identity on the left by $\boldsymbol{\Phi}^B$ and on the right by $\boldsymbol{\Phi}^{A\top}$ gives

$$\boldsymbol{T} = \boldsymbol{\Phi}^B \boldsymbol{C} \boldsymbol{\Phi}^{A\top}.$$

We first show that the spatial-domain intertwining condition

$$\boldsymbol{\Delta}^B \boldsymbol{T} = \boldsymbol{T} \boldsymbol{\Delta}^A$$

is equivalent to the spectral-domain condition

$$\boldsymbol{\Lambda}^B \boldsymbol{C} = \boldsymbol{C} \boldsymbol{\Lambda}^A.$$

Using the eigendecompositions and the expression for $\boldsymbol{T}$, we get

$$\begin{aligned}
\boldsymbol{\Delta}^B \boldsymbol{T} &= \left(\boldsymbol{\Phi}^B \boldsymbol{\Lambda}^B \boldsymbol{\Phi}^{B\top} \right) \left(\boldsymbol{\Phi}^B \boldsymbol{C} \boldsymbol{\Phi}^{A\top} \right) \\
&= \boldsymbol{\Phi}^B \boldsymbol{\Lambda}^B \left(\boldsymbol{\Phi}^{B\top} \boldsymbol{\Phi}^B \right) \boldsymbol{C} \boldsymbol{\Phi}^{A\top} \\
&= \boldsymbol{\Phi}^B \boldsymbol{\Lambda}^B \boldsymbol{C} \boldsymbol{\Phi}^{A\top}.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\boldsymbol{T} \boldsymbol{\Delta}^A &= \left(\boldsymbol{\Phi}^B \boldsymbol{C} \boldsymbol{\Phi}^{A\top} \right) \left(\boldsymbol{\Phi}^A \boldsymbol{\Lambda}^A \boldsymbol{\Phi}^{A\top} \right) \\
&= \boldsymbol{\Phi}^B \boldsymbol{C} \left(\boldsymbol{\Phi}^{A\top} \boldsymbol{\Phi}^A \right) \boldsymbol{\Lambda}^A \boldsymbol{\Phi}^{A\top} \\
&= \boldsymbol{\Phi}^B \boldsymbol{C} \boldsymbol{\Lambda}^A \boldsymbol{\Phi}^{A\top}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\boldsymbol{\Delta}^B \boldsymbol{T} - \boldsymbol{T} \boldsymbol{\Delta}^A &= \boldsymbol{\Phi}^B \boldsymbol{\Lambda}^B \boldsymbol{C} \boldsymbol{\Phi}^{A\top} - \boldsymbol{\Phi}^B \boldsymbol{C} \boldsymbol{\Lambda}^A \boldsymbol{\Phi}^{A\top} \\
&= \boldsymbol{\Phi}^B \left(\boldsymbol{\Lambda}^B \boldsymbol{C} - \boldsymbol{C} \boldsymbol{\Lambda}^A \right) \boldsymbol{\Phi}^{A\top}.
\end{aligned}$$

It follows immediately that if $\boldsymbol{\Lambda}^B \boldsymbol{C} = \boldsymbol{C} \boldsymbol{\Lambda}^A$, then

$$\boldsymbol{\Delta}^B \boldsymbol{T} - \boldsymbol{T} \boldsymbol{\Delta}^A = \boldsymbol{0}.$$

Conversely, if

$$\boldsymbol{\Delta}^B \boldsymbol{T} - \boldsymbol{T} \boldsymbol{\Delta}^A = \boldsymbol{0},$$

then multiplying on the left by $\boldsymbol{\Phi}^{B\top}$ and on the right by $\boldsymbol{\Phi}^A$ gives

$$\begin{aligned}
\boldsymbol{0} &= \boldsymbol{\Phi}^{B\top} \left(\boldsymbol{\Delta}^B \boldsymbol{T} - \boldsymbol{T} \boldsymbol{\Delta}^A \right) \boldsymbol{\Phi}^A \\
&= \boldsymbol{\Phi}^{B\top} \boldsymbol{\Phi}^B \left(\boldsymbol{\Lambda}^B \boldsymbol{C} - \boldsymbol{C} \boldsymbol{\Lambda}^A \right) \boldsymbol{\Phi}^{A\top} \boldsymbol{\Phi}^A \\
&= \boldsymbol{\Lambda}^B \boldsymbol{C} - \boldsymbol{C} \boldsymbol{\Lambda}^A.
\end{aligned}$$

Hence

$$\boldsymbol{\Lambda}^B \boldsymbol{C} = \boldsymbol{C} \boldsymbol{\Lambda}^A.$$

Thus the spatial-domain intertwining condition is equivalent to the stated spectral-domain condition.

It remains to prove the Frobenius-norm identity. From the computation above,

$$\boldsymbol{\Delta}^B \boldsymbol{T} - \boldsymbol{T} \boldsymbol{\Delta}^A = \boldsymbol{\Phi}^B \left(\boldsymbol{\Lambda}^B \boldsymbol{C} - \boldsymbol{C} \boldsymbol{\Lambda}^A \right) \boldsymbol{\Phi}^{A\top}.$$

Since multiplication by orthonormal matrices preserves the Frobenius norm, we have

$$\| \boldsymbol{\Delta}^B \boldsymbol{T} - \boldsymbol{T} \boldsymbol{\Delta}^A \|_F = \| \boldsymbol{\Lambda}^B \boldsymbol{C} - \boldsymbol{C} \boldsymbol{\Lambda}^A \|_F.$$

For completeness, this invariance follows from the trace representation of the Frobenius norm. If $\boldsymbol{M}$ is any compatible matrix, then

$$\begin{aligned}
\| \boldsymbol{\Phi}^B \boldsymbol{M} \boldsymbol{\Phi}^{A\top} \|_F^2 &= \operatorname{tr} \left(\left(\boldsymbol{\Phi}^B \boldsymbol{M} \boldsymbol{\Phi}^{A\top} \right)^\top \left(\boldsymbol{\Phi}^B \boldsymbol{M} \boldsymbol{\Phi}^{A\top} \right) \right) \\
&= \operatorname{tr} \left(\boldsymbol{\Phi}^A \boldsymbol{M}^\top \boldsymbol{\Phi}^{B\top} \boldsymbol{\Phi}^B \boldsymbol{M} \boldsymbol{\Phi}^{A\top} \right) \\
&= \operatorname{tr} \left(\boldsymbol{\Phi}^A \boldsymbol{M}^\top \boldsymbol{M} \boldsymbol{\Phi}^{A\top} \right) \\
&= \operatorname{tr} \left(\boldsymbol{M}^\top \boldsymbol{M} \boldsymbol{\Phi}^{A\top} \boldsymbol{\Phi}^A \right) \\
&= \operatorname{tr} \left(\boldsymbol{M}^\top \boldsymbol{M} \right) \\
&= \| \boldsymbol{M} \|_F^2.
\end{aligned}$$

Now let

$$\boldsymbol{M} := \boldsymbol{\Lambda}^B \boldsymbol{C} - \boldsymbol{C} \boldsymbol{\Lambda}^A.$$

Because $\mathbf{\Lambda}^A$ and $\mathbf{\Lambda}^B$ are diagonal, their diagonal entries are the eigenvalues λ_j^A and λ_i^B, respectively. Hence the (i,j)-entry of $\boldsymbol{M}$ is

$$\begin{aligned} M_{ij} &= \left(\mathbf{\Lambda}^B \boldsymbol{C}\right)_{ij} - \left(\boldsymbol{C}\mathbf{\Lambda}^A\right)_{ij} \\ &= \lambda_i^B C_{ij} - C_{ij}\lambda_j^A \\ &= \left(\lambda_i^B - \lambda_j^A\right) C_{ij}. \end{aligned}$$

Therefore,

$$\begin{aligned} \|\mathbf{\Delta}^B \boldsymbol{T} - \boldsymbol{T}\mathbf{\Delta}^A\|_F^2 &= \|\mathbf{\Lambda}^B \boldsymbol{C} - \boldsymbol{C}\mathbf{\Lambda}^A\|_F^2 \\ &= \sum_{i,j} M_{ij}^2 \\ &= \sum_{i,j} \left(\lambda_i^B - \lambda_j^A\right)^2 C_{ij}^2. \end{aligned}$$

Restoring the layer index l gives

$$\|\mathbf{\Delta}_l^B \boldsymbol{T}_l - \boldsymbol{T}_l \mathbf{\Delta}_l^A\|_F^2 = \sum_{i,j} \left(\lambda_{l,i}^B - \lambda_{l,j}^A\right)^2 (\boldsymbol{C}_l)_{ij}^2.$$

□

E.2 Proof of Proposition 5

Proposition 5. *Let $\boldsymbol{E}_l(\boldsymbol{T}) = \boldsymbol{W}_l^B \boldsymbol{T}_{l-1} - \boldsymbol{T}_l \boldsymbol{W}_l^A$ and $\boldsymbol{W}_l^{A\to B} = \boldsymbol{T}_l \boldsymbol{W}_l^A \boldsymbol{T}_{l-1}^\dagger$. Then,*

$$\|\boldsymbol{W}_l^B - \boldsymbol{W}_l^{A\to B}\|_F \le \|\boldsymbol{W}_l^B(\boldsymbol{I} - \boldsymbol{T}_{l-1}\boldsymbol{T}_{l-1}^\dagger)\|_F + \|\boldsymbol{E}_l(\boldsymbol{T})\|_F \|\boldsymbol{T}_{l-1}^\dagger\|_2.$$

Proof. From the definition of $\boldsymbol{E}_l(\boldsymbol{T})$, we have

$$\boldsymbol{T}_l \boldsymbol{W}_l^A = \boldsymbol{W}_l^B \boldsymbol{T}_{l-1} - \boldsymbol{E}_l(\boldsymbol{T}).$$

Therefore

$$\begin{aligned} \boldsymbol{W}_l^B - \boldsymbol{W}_l^{A\to B} &= \boldsymbol{W}_l^B - \boldsymbol{T}_l \boldsymbol{W}_l^A \boldsymbol{T}_{l-1}^\dagger \\ &= \boldsymbol{W}_l^B - (\boldsymbol{W}_l^B \boldsymbol{T}_{l-1} - \boldsymbol{E}_l(\boldsymbol{T}))\boldsymbol{T}_{l-1}^\dagger \\ &= \boldsymbol{W}_l^B(\boldsymbol{I} - \boldsymbol{T}_{l-1}\boldsymbol{T}_{l-1}^\dagger) + \boldsymbol{E}_l(\boldsymbol{T})\boldsymbol{T}_{l-1}^\dagger. \end{aligned}$$

Taking norms and using submultiplicativity completes the proofs. □

E.3 Proof of Proposition 6

Proposition 6. *If $\mathcal{L}$ is twice differentiable with $\|\nabla^2 \mathcal{L}(\boldsymbol{\theta}(t))\|_2 \le \beta$ for $t \in [0,1]$, then the loss barrier $\mathcal{B}(\boldsymbol{\theta}(t))$ is bounded by*

$$\mathcal{B}(\boldsymbol{\theta}(t)) \le \frac{\beta}{8} \sum_{l=1}^{L} \alpha_l^2(\boldsymbol{T}).$$

Proof. Let

$$\boldsymbol{\theta}_0 := \boldsymbol{\theta}^{A\to B}(\boldsymbol{T}), \qquad \boldsymbol{\theta}_1 := \boldsymbol{\theta}^B,$$

and define

$$\boldsymbol{v} := \boldsymbol{\theta}_1 - \boldsymbol{\theta}_0 = \boldsymbol{\theta}^B - \boldsymbol{\theta}^{A\to B}(\boldsymbol{T}).$$

The linear interpolation between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ can then be written as

$$\boldsymbol{\theta}(t) = (1-t)\boldsymbol{\theta}_0 + t\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 + t\boldsymbol{v}, \qquad t \in [0,1].$$

Define the one-dimensional function

$$g : [0,1] \to \mathbb{R}, \qquad g(t) := \mathcal{L}(\boldsymbol{\theta}(t)).$$

Since $\mathcal{L}$ is twice differentiable and $\boldsymbol{\theta}(t)$ is affine in t, the chain rule gives

$$g'(t) = \nabla \mathcal{L}(\boldsymbol{\theta}(t))^\top \boldsymbol{v},$$

and

$$g''(t) = \boldsymbol{v}^\top \nabla^2 \mathcal{L}(\boldsymbol{\theta}(t))\boldsymbol{v}.$$

Using the spectral norm bound on the Hessian, we obtain, for every $t \in [0,1]$,

$$\begin{aligned} |g''(t)| &= \left|\boldsymbol{v}^\top \nabla^2 \mathcal{L}(\boldsymbol{\theta}(t))\boldsymbol{v}\right| \\ &\le \|\nabla^2 \mathcal{L}(\boldsymbol{\theta}(t))\|_2 \|\boldsymbol{v}\|_2^2 \\ &\le \beta \|\boldsymbol{v}\|_2^2. \end{aligned}$$

Set

$$M := \beta \|\boldsymbol{v}\|_2^2.$$

Thus $|g''(t)| \le M$ for all $t \in [0,1]$.

We now derive the one-dimensional interpolation bound. For any fixed $t \in [0,1]$, the following identity holds:

$$g(t) = (1-t)g(0) + tg(1) - \int_0^t s(1-t)g''(s)\,ds - \int_t^1 t(1-s)g''(s)\,ds.$$

This identity follows by applying integration by parts twice to g on the intervals $[0,t]$ and $[t,1]$. Since the kernels $s(1-t)$ and $t(1-s)$ are nonnegative on their respective intervals, and since $|g''(s)| \le M$, we get

$$\begin{aligned} g(t) &\le (1-t)g(0) + tg(1) + M\left(\int_0^t s(1-t)\,ds + \int_t^1 t(1-s)\,ds\right) \\ &= (1-t)g(0) + tg(1) + M\left(\frac{(1-t)t^2}{2} + \frac{t(1-t)^2}{2}\right) \\ &= (1-t)g(0) + tg(1) + \frac{M}{2}t(1-t). \end{aligned}$$

Substituting $M = \beta\|\boldsymbol{v}\|_2^2$ gives

$$g(t) \le (1-t)g(0) + tg(1) + \frac{\beta}{2}t(1-t)\|\boldsymbol{v}\|_2^2.$$

Moreover,

$$(1-t)g(0) + tg(1) \le \max\{g(0), g(1)\}.$$

Therefore, for every $t \in [0,1]$,

$$g(t) - \max\{g(0), g(1)\} \le \frac{\beta}{2}t(1-t)\|\boldsymbol{v}\|_2^2.$$

Taking the maximum over $t \in [0,1]$ yields

$$\begin{aligned} \max_{t\in[0,1]} g(t) - \max\{g(0), g(1)\} &\le \frac{\beta}{2} \max_{t\in[0,1]} t(1-t)\|\boldsymbol{v}\|_2^2 \\ &= \frac{\beta}{2} \cdot \frac{1}{4}\|\boldsymbol{v}\|_2^2 \\ &= \frac{\beta}{8}\|\boldsymbol{v}\|_2^2. \end{aligned}$$

By the definition of the loss barrier,

$$\mathcal{B}(\boldsymbol{\theta}(t)) = \max_{t\in[0,1]} \mathcal{L}(\boldsymbol{\theta}(t)) - \max\{\mathcal{L}(\boldsymbol{\theta}(0)), \mathcal{L}(\boldsymbol{\theta}(1))\},$$

so the preceding inequality gives

$$\mathcal{B}(\boldsymbol{\theta}(t)) \le \frac{\beta}{8}\|\boldsymbol{v}\|_2^2.$$

It remains to bound $\|\boldsymbol{v}\|_2^2$. Since the parameter vector is obtained by vectorising and concatenating the layer-wise weight matrices, we have

$$\begin{aligned} \|\boldsymbol{v}\|_2^2 &= \left\|\boldsymbol{\theta}^B - \boldsymbol{\theta}^{A\to B}(\boldsymbol{T})\right\|_2^2 \\ &= \sum_{l=1}^{L} \left\|\boldsymbol{W}_l^B - \boldsymbol{W}_l^{A\to B}\right\|_F^2. \end{aligned}$$

By the assumed layer-wise bound

$$\left\|\boldsymbol{W}_l^B - \boldsymbol{W}_l^{A\to B}\right\|_F \le \alpha_l(\boldsymbol{T}),$$

we obtain

$$\|\boldsymbol{v}\|_2^2 \le \sum_{l=1}^{L} \alpha_l^2(\boldsymbol{T}).$$

Combining this with the previous loss-barrier bound gives

$$\mathcal{B}(\boldsymbol{\theta}(t)) \le \frac{\beta}{8} \sum_{l=1}^{L} \alpha_l^2(\boldsymbol{T}).$$

□

E.4 The effects of spectral truncation on Prop. 5 and Prop. 6

We derive a truncation-aware decomposition of the full coordinate-space residual. Let $\boldsymbol{P}_l^X = \boldsymbol{\Phi}_l^X(\boldsymbol{\Phi}_l^X)^\top$ be the projector onto the q_l^X retained graph modes, and let $\boldsymbol{T}_l = \boldsymbol{\Phi}_l^B \boldsymbol{C}_l(\boldsymbol{\Phi}_l^A)^\top$. For the weight-alignment residual $\boldsymbol{E}_l = \boldsymbol{W}_l^B\boldsymbol{T}_{l-1} - \boldsymbol{T}_l\boldsymbol{W}_l^A$, we obtain $\boldsymbol{E}_l = \boldsymbol{\Phi}_l^B\widehat{\boldsymbol{E}}_l(\boldsymbol{\Phi}_{l-1}^A)^\top + (\boldsymbol{I} - \boldsymbol{P}_l^B)\boldsymbol{W}_l^B\boldsymbol{T}_{l-1} - \boldsymbol{T}_l\boldsymbol{W}_l^A(\boldsymbol{I} - \boldsymbol{P}_{l-1}^A)$, where $\widehat{\boldsymbol{E}}_l$ is the projected residual computed in the retained spectral bases. Frobenius orthogonality gives $||\boldsymbol{E}_l||_F^2 = ||\widehat{\boldsymbol{E}}_l||_F^2 + ||(\boldsymbol{I} - \boldsymbol{P}_l^B)\boldsymbol{W}_l^B\boldsymbol{T}_{l-1}||_F^2 + ||\boldsymbol{T}_l\boldsymbol{W}_l^A(\boldsymbol{I} - \boldsymbol{P}_{l-1}^A)||_F^2$. Prop. 5 then gives $||\boldsymbol{W}_l^B - \boldsymbol{W}_l^{A\to B}||_F \le ||\boldsymbol{W}_l^B(\boldsymbol{I} - \boldsymbol{T}_{l-1}\boldsymbol{T}_{l-1}^\dagger)||_F + ||\boldsymbol{T}_{l-1}^\dagger||_2\sqrt{||\widehat{\boldsymbol{E}}_l||_F^2 + ||(\boldsymbol{I} - \boldsymbol{P}_l^B)\boldsymbol{W}_l^B\boldsymbol{T}_{l-1}||_F^2 + ||\boldsymbol{T}_l\boldsymbol{W}_l^A(\boldsymbol{I} - \boldsymbol{P}_{l-1}^A)||_F^2}$.

Prop. 6 is a generic smoothness result of endpoint distance. The intended condition is the bounded curvature of the scalar loss along the specific interpolation segment. Define $\beta_{\text{path}} = \max_{t\in[0,1]} ||\nabla^2\mathcal{L}(\boldsymbol{\theta}(t))||_2$. If $\mathcal{L}$ is C^2 in a neighbourhood of the interpolation path, then β_{path} is finite. This requires no global Hessian bound. For ReLU and other nonsmooth activations, the objective is generally piecewise smooth and may not be twice differentiable at activation-boundary crossings. A fully nonsmooth extension would require an additional pathwise weak-curvature assumption and is not claimed by the current proposition.

In addition, note that the proof only requires curvature along $\boldsymbol{v} = \boldsymbol{\theta}^B - \boldsymbol{\theta}^{A\to B}$. For $\boldsymbol{v} \neq \boldsymbol{0}$, we define

$$\beta_v = \sup_{t\in[0,1]} \frac{|\boldsymbol{v}^\top\nabla^2\mathcal{L}(\boldsymbol{\theta}(t))\boldsymbol{v}|}{||\boldsymbol{v}||_2^2}, \tag{31}$$

which gives $\mathcal{B} \le \frac{\beta_v}{8}||\boldsymbol{v}||_2^2$. This is a weaker pathwise assumption than a uniform spectral-norm bound on the full Hessian and can be substantially tighter.

We can sharpen the assumption, which is stronger than what the proof requires, and make the meaning of β clearer. Let $\boldsymbol{v} = \boldsymbol{\theta}^B - \boldsymbol{\theta}^{A\to B}$ and $g(t) = \mathcal{L}(\boldsymbol{\theta}^{A\to B} + t\boldsymbol{v})$. The proof needs a lower bound on the directional curvature, $g''(t) = \boldsymbol{v}^\top\nabla^2\mathcal{L}(\boldsymbol{\theta}(t))\boldsymbol{v} \ge -\beta_-||\boldsymbol{v}||_2^2$. Equivalently, a sufficient pathwise condition is $\nabla^2\mathcal{L}(\boldsymbol{\theta}(t)) \succeq -\beta_-\boldsymbol{I}$. β_- measures the largest negative curvature along the segment. Positive curvature bends the loss below the chord and does not produce an interpolation barrier, and only negative curvature can cause the interior of the segment to exceed both endpoint losses. The present condition $||\nabla^2\mathcal{L}(\boldsymbol{\theta}(t))||_2 \le \beta$ implies the one-sided condition with $\beta_- \le \beta$, but the one-sided version is much less restrictive. Specifically, if the objective is convex along the interpolation segment, one can take $\beta_- = 0$, and the bound correctly gives a zero loss barrier. The statement $\mathcal{B}(\boldsymbol{\theta}(t)) \le \frac{\beta_-}{8}\sum_{l=1}^{L}\alpha_l(\boldsymbol{T})^2$, separates two effects: *(i)* the term $\sum_l \alpha_l(\boldsymbol{T})^2$ controls the parameter distance between the transported endpoint and the target endpoint, and *(ii)* the curvature term β_- converts this endpoint discrepancy into a loss-barrier bound by measuring curvature along the interpolation path.

F Layerwise graph spectral symmetry and neural function-realisability

Let $\boldsymbol{\Delta}_l \in \mathbb{R}^{d_l\times d_l}$ be a symmetric graph shift operator with eigendecomposition $\boldsymbol{\Delta}_l = \boldsymbol{\Phi}_l\boldsymbol{\Lambda}_l\boldsymbol{\Phi}_l^\top$, where $\boldsymbol{\Phi}_l$ is orthonormal and $\boldsymbol{\Lambda}_l = \operatorname{diag}(\lambda_{l,1},\ldots,\lambda_{l,d_l})$. For a scalar function f defined on the

spectrum $\sigma(\boldsymbol{\Delta}_l)$, we define the spectral graph functional $f(\boldsymbol{\Delta}_l) := \boldsymbol{\Phi}_l f(\boldsymbol{\Lambda}_l)\boldsymbol{\Phi}_l^\top$, where $f(\boldsymbol{\Lambda}_l) := \operatorname{diag}\left(f(\lambda_{l,1}), \ldots, f(\lambda_{l,d_l})\right)$.

Proposition 11. *Let $\boldsymbol{U}_l \in \mathcal{U}_{\boldsymbol{\Delta}_l}$. Then, for every function f defined on $\sigma(\boldsymbol{\Delta}_l)$,*

$$f(\boldsymbol{\Delta}_l)\boldsymbol{U}_l = \boldsymbol{U}_l f(\boldsymbol{\Delta}_l). \tag{32}$$

Proof. Let the distinct eigenvalues of $\boldsymbol{\Delta}_l$ be $\mu_1, \ldots, \mu_r$. Since $\boldsymbol{\Delta}_l$ is symmetric, it admits the spectral decomposition

$$\boldsymbol{\Delta}_l = \sum_{k=1}^{r} \mu_k \boldsymbol{P}_k,$$

where $\boldsymbol{P}_k$ is the orthogonal projector onto the eigenspace associated with μ_k. Consequently,

$$f(\boldsymbol{\Delta}_l) = \sum_{k=1}^{r} f(\mu_k)\boldsymbol{P}_k.$$

Since $\boldsymbol{U}_l \in \mathcal{U}_{\boldsymbol{\Delta}_l}$, we have

$$\boldsymbol{U}_l\boldsymbol{\Delta}_l = \boldsymbol{\Delta}_l\boldsymbol{U}_l.$$

We now show that $\boldsymbol{U}_l$ commutes with each spectral projector $\boldsymbol{P}_k$. For each k, the projector $\boldsymbol{P}_k$ can be written as a polynomial in $\boldsymbol{\Delta}_l$:

$$\boldsymbol{P}_k = \prod_{\substack{s=1 \\ s\neq k}}^{r} \frac{\boldsymbol{\Delta}_l - \mu_s \boldsymbol{I}}{\mu_k - \mu_s}.$$

Because $\boldsymbol{U}_l$ commutes with $\boldsymbol{\Delta}_l$, it also commutes with every polynomial in $\boldsymbol{\Delta}_l$. Hence,

$$\boldsymbol{U}_l\boldsymbol{P}_k = \boldsymbol{P}_k\boldsymbol{U}_l \qquad \text{for every } k = 1, \ldots, r.$$

Therefore,

$$\begin{aligned}
\boldsymbol{U}_l f(\boldsymbol{\Delta}_l) &= \boldsymbol{U}_l \sum_{k=1}^{r} f(\mu_k)\boldsymbol{P}_k \\
&= \sum_{k=1}^{r} f(\mu_k)\boldsymbol{U}_l\boldsymbol{P}_k \\
&= \sum_{k=1}^{r} f(\mu_k)\boldsymbol{P}_k\boldsymbol{U}_l \\
&= f(\boldsymbol{\Delta}_l)\boldsymbol{U}_l.
\end{aligned}$$

□

Corollary 12. *Assume that $\boldsymbol{\Delta}_l \succeq 0$. Let $\boldsymbol{U}_l$ be an orthogonal matrix satisfying*

$$\|\boldsymbol{U}_l\boldsymbol{\Delta}_l - \boldsymbol{\Delta}_l\boldsymbol{U}_l\|_2 \leq \varepsilon. \tag{33}$$

Then, for every $t \geq 0$,

$$\|\boldsymbol{U}_l \exp(-t\boldsymbol{\Delta}_l) - \exp(-t\boldsymbol{\Delta}_l)\boldsymbol{U}_l\|_2 \leq t\varepsilon. \tag{34}$$

Proof. Define

$$\boldsymbol{H}_t := \exp(-t\boldsymbol{\Delta}_l).$$

We use the Duhamel identity for the commutator between $\boldsymbol{U}_l$ and $\boldsymbol{H}_t$. For $s \in [0, t]$, define

$$\boldsymbol{F}(s) := \exp(-(t-s)\boldsymbol{\Delta}_l)\boldsymbol{U}_l \exp(-s\boldsymbol{\Delta}_l).$$

Then

$$\boldsymbol{F}(0) = \exp(-t\boldsymbol{\Delta}_l)\boldsymbol{U}_l$$

and

$$\boldsymbol{F}(t) = \boldsymbol{U}_l \exp(-t\boldsymbol{\Delta}_l).$$

Differentiating $\boldsymbol{F}(s)$ gives

$$\begin{aligned}
\boldsymbol{F}'(s) &= \exp(-(t-s)\boldsymbol{\Delta}_l)\boldsymbol{\Delta}_l\boldsymbol{U}_l \exp(-s\boldsymbol{\Delta}_l) - \exp(-(t-s)\boldsymbol{\Delta}_l)\boldsymbol{U}_l\boldsymbol{\Delta}_l \exp(-s\boldsymbol{\Delta}_l) \\
&= \exp(-(t-s)\boldsymbol{\Delta}_l)\left(\boldsymbol{\Delta}_l\boldsymbol{U}_l - \boldsymbol{U}_l\boldsymbol{\Delta}_l\right)\exp(-s\boldsymbol{\Delta}_l).
\end{aligned}$$

By the fundamental theorem of calculus,

$$\boldsymbol{U}_l \exp(-t\boldsymbol{\Delta}_l) - \exp(-t\boldsymbol{\Delta}_l)\boldsymbol{U}_l = \boldsymbol{F}(t) - \boldsymbol{F}(0) = \int_0^t \boldsymbol{F}'(s)\, ds.$$

Taking spectral norms and using submultiplicativity,

$$\begin{aligned}&\|\boldsymbol{U}_l \exp(-t\boldsymbol{\Delta}_l) - \exp(-t\boldsymbol{\Delta}_l)\boldsymbol{U}_l\|_2 \\ &\leq \int_0^t \|\exp(-(t-s)\boldsymbol{\Delta}_l)\|_2 \|\boldsymbol{\Delta}_l\boldsymbol{U}_l - \boldsymbol{U}_l\boldsymbol{\Delta}_l\|_2 \|\exp(-s\boldsymbol{\Delta}_l)\|_2\, ds.\end{aligned}$$

Since $\boldsymbol{\Delta}_l \succeq 0$, all eigenvalues of $\exp(-r\boldsymbol{\Delta}_l)$ lie in $[0,1]$ for every $r \geq 0$. Therefore,

$$\|\exp(-r\boldsymbol{\Delta}_l)\|_2 \leq 1 \qquad \text{for all } r \geq 0.$$

Using the assumed commutator bound,

$$\|\boldsymbol{U}_l \exp(-t\boldsymbol{\Delta}_l) - \exp(-t\boldsymbol{\Delta}_l)\boldsymbol{U}_l\|_2 \leq \int_0^t \varepsilon\, ds = t\varepsilon.$$

□

Proposition 11 shows that an exact graph functional symmetry is invisible to all spectral graph functionals. If $\boldsymbol{U}_l$ commutes with $\boldsymbol{\Delta}_l$, then applying $\boldsymbol{U}_l$ before or after any spectral filter $f(\boldsymbol{\Delta}_l)$ gives the same result. Thus, the graph shift operator cannot distinguish between coordinates related by $\boldsymbol{U}_l$.

Equivalently, graph functional symmetries preserve the eigenspace structure of $\boldsymbol{\Delta}_l$. When an eigenvalue has multiplicity greater than one, $\boldsymbol{U}_l$ may rotate coordinates inside the corresponding eigenspace without changing any spectral functional of the graph. Therefore, the symmetry reveals which hidden directions are indistinguishable from the perspective of graph spectral measurements.

Corollary 12 gives the corresponding approximate statement for heat diffusion. If $\boldsymbol{U}_l\boldsymbol{\Delta}_l$ is close to $\boldsymbol{\Delta}_l\boldsymbol{U}_l$, then $\boldsymbol{U}_l$ also nearly commutes with the diffusion operator $\exp(-t\boldsymbol{\Delta}_l)$. The error grows at most linearly in the diffusion time t. Hence a low-defect graph symmetry identifies neurons or hidden directions that behave similarly under graph diffusion, especially at short and moderate diffusion scales.

However, graph functional symmetry is a property of the layer graph. It does not automatically imply that the same transformation is a true symmetry of the neural network function. To obtain a neural-parameter symmetry, the coordinate transformation must also be realisable through the network architecture and nonlinearities.

Definition 13. Let $f_{\boldsymbol{\theta}}$ be a neural network, and let

$$\boldsymbol{h}^l_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^{d_l} \tag{35}$$

denote the hidden state at layer l. An invertible matrix $\boldsymbol{U}_l \in \mathbb{R}^{d_l \times d_l}$ is *function-realizable* at layer l if there exists a parameter vector $\boldsymbol{\theta}^{(\boldsymbol{U}_l)}$ of the same architecture such that, for every input $\boldsymbol{x}$,

$$f_{\boldsymbol{\theta}^{(\boldsymbol{U}_l)}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{36}$$

and

$$\boldsymbol{h}^l_{\boldsymbol{\theta}^{(\boldsymbol{U}_l)}}(\boldsymbol{x}) = \boldsymbol{U}_l \boldsymbol{h}^l_{\boldsymbol{\theta}}(\boldsymbol{x}). \tag{37}$$

Thus, $\boldsymbol{U}_l$ is function-realisable if it can be implemented as a change of hidden coordinates while leaving the input-output function of the network unchanged.

Proposition 14. *Consider one hidden layer with preactivation and a hidden state*

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{u} + \boldsymbol{b}, \qquad \boldsymbol{h} = \sigma(\boldsymbol{z}), \tag{38}$$

followed by the affine map

$$\boldsymbol{o} = \boldsymbol{V}\boldsymbol{h} + \boldsymbol{c}. \tag{39}$$

Let $\boldsymbol{Q} \in \mathrm{GL}(d)$ and define transformed parameters

$$\boldsymbol{W}^{\boldsymbol{Q}} = \boldsymbol{Q}\boldsymbol{W}, \qquad \boldsymbol{b}^{\boldsymbol{Q}} = \boldsymbol{Q}\boldsymbol{b}, \qquad \boldsymbol{V}^{\boldsymbol{Q}} = \boldsymbol{V}\boldsymbol{Q}^{-1}. \tag{40}$$

Then the transformed network satisfies

$$\boldsymbol{h}^{\boldsymbol{Q}}(\boldsymbol{x}) = \boldsymbol{Q}\boldsymbol{h}(\boldsymbol{x}), \qquad \boldsymbol{o}^{\boldsymbol{Q}}(\boldsymbol{x}) = \boldsymbol{o}(\boldsymbol{x}) \qquad \textit{for all } \boldsymbol{x}, \tag{41}$$

if and only if

$$\sigma(\boldsymbol{Q}\boldsymbol{z}) = \boldsymbol{Q}\sigma(\boldsymbol{z}) \tag{42}$$

for every preactivation $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{u} + \boldsymbol{b}$ reached by the original network.

Proof. For an input $\boldsymbol{x}$, let

$$\boldsymbol{z}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{b}$$

be the original preactivation, and let

$$\boldsymbol{h}(\boldsymbol{x}) = \sigma(\boldsymbol{z}(\boldsymbol{x}))$$

be the corresponding hidden state. Under the transformed parameters, the preactivation becomes

$$\begin{aligned}\boldsymbol{z}^{\boldsymbol{Q}}(\boldsymbol{x}) &= \boldsymbol{W}^{\boldsymbol{Q}}\boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{b}^{\boldsymbol{Q}} \\ &= \boldsymbol{Q}\boldsymbol{W}\boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{Q}\boldsymbol{b} \\ &= \boldsymbol{Q}\left(\boldsymbol{W}\boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{b}\right) \\ &= \boldsymbol{Q}\boldsymbol{z}(\boldsymbol{x}).\end{aligned}$$

Therefore, the transformed hidden state is

$$\boldsymbol{h}^{\boldsymbol{Q}}(\boldsymbol{x}) = \sigma\left(\boldsymbol{z}^{\boldsymbol{Q}}(\boldsymbol{x})\right) = \sigma\left(\boldsymbol{Q}\boldsymbol{z}(\boldsymbol{x})\right).$$

Suppose first that

$$\sigma(\boldsymbol{Q}\boldsymbol{z}) = \boldsymbol{Q}\sigma(\boldsymbol{z})$$

for every reachable preactivation $\boldsymbol{z}$. Then, for every input $\boldsymbol{x}$,

$$\begin{aligned}\boldsymbol{h}^{\boldsymbol{Q}}(\boldsymbol{x}) &= \sigma\left(\boldsymbol{Q}\boldsymbol{z}(\boldsymbol{x})\right) \\ &= \boldsymbol{Q}\sigma\left(\boldsymbol{z}(\boldsymbol{x})\right) \\ &= \boldsymbol{Q}\boldsymbol{h}(\boldsymbol{x}).\end{aligned}$$

The transformed output is then

$$\begin{aligned}\boldsymbol{o}^{\boldsymbol{Q}}(\boldsymbol{x}) &= \boldsymbol{V}^{\boldsymbol{Q}}\boldsymbol{h}^{\boldsymbol{Q}}(\boldsymbol{x}) + \boldsymbol{c} \\ &= \boldsymbol{V}\boldsymbol{Q}^{-1}\boldsymbol{Q}\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{c} \\ &= \boldsymbol{V}\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{c} \\ &= \boldsymbol{o}(\boldsymbol{x}).\end{aligned}$$

Thus the transformed network realises the hidden-coordinate change $\boldsymbol{h}^{\boldsymbol{Q}} = \boldsymbol{Q}\boldsymbol{h}$ while preserving the output.

Conversely, suppose the transformed network satisfies

$$\boldsymbol{h}^{\boldsymbol{Q}}(\boldsymbol{x}) = \boldsymbol{Q}\boldsymbol{h}(\boldsymbol{x})$$

for every input $\boldsymbol{x}$. Since

$$\boldsymbol{h}^{\boldsymbol{Q}}(\boldsymbol{x}) = \sigma\left(\boldsymbol{Q}\boldsymbol{z}(\boldsymbol{x})\right)$$

and

$$\boldsymbol{h}(\boldsymbol{x}) = \sigma\left(\boldsymbol{z}(\boldsymbol{x})\right),$$

we obtain

$$\sigma\left(\boldsymbol{Q}\boldsymbol{z}(\boldsymbol{x})\right) = \boldsymbol{Q}\sigma\left(\boldsymbol{z}(\boldsymbol{x})\right)$$

for every input $\boldsymbol{x}$. Equivalently,

$$\sigma(\boldsymbol{Q}\boldsymbol{z}) = \boldsymbol{Q}\sigma(\boldsymbol{z})$$

for every preactivation $\boldsymbol{z}$ reached by the original network.

□

Proposition 14 separates graph-level symmetry from network-level symmetry. A graph functional symmetry $\boldsymbol{U}_l \in \mathcal{U}_{\boldsymbol{\Delta}_l}$ may preserve all spectral functionals of the layer graph, but it becomes an exact neural-parameter symmetry only if the network can realise the corresponding hidden-coordinate transformation.

The obstruction is the nonlinearity. The affine maps can absorb an invertible coordinate change by replacing $\boldsymbol{W}$ with $\boldsymbol{Q}\boldsymbol{W}$ and $\boldsymbol{V}$ with $\boldsymbol{V}\boldsymbol{Q}^{-1}$. However, the activation must also satisfy

$$\sigma(\boldsymbol{Q}\boldsymbol{z}) = \boldsymbol{Q}\sigma(\boldsymbol{z}) \tag{43}$$

on the set of reachable preactivations. If this equivariance condition fails, then the transformed hidden representation is not $\boldsymbol{Q}\boldsymbol{h}$, and the coordinate change is not an exact network symmetry.

G Additional remarks on spectral re-basin

G.1 Commutation with the graph shift operator

We here discuss what the commutativity condition present in graph functional shifts means. We have the following result.

Proposition 15. *If a transformation $\boldsymbol{T}_l$ commutes with the graph shift operator $\boldsymbol{\Delta}_l$, i.e., $\boldsymbol{\Delta}_l\boldsymbol{T}_l = \boldsymbol{T}_l\boldsymbol{\Delta}_l$, then $\boldsymbol{T}_l$ consists in independent linear transformations within the eigenspaces $\boldsymbol{\Delta}_l$. Additionally, if $\boldsymbol{T}_l$ is orthogonal, then $\boldsymbol{T}_l$ performs independent rotations within the eigenspaces.*

Proof. Let $\boldsymbol{T}_l$ be any square matrix commuting with the graph shift operator $\boldsymbol{\Delta}_l$, i.e.,

$$\boldsymbol{\Delta}_l\boldsymbol{T}_l = \boldsymbol{T}_l\boldsymbol{\Delta}_l.$$

Denote $\boldsymbol{\Phi}_l, \boldsymbol{\Lambda}_l$ the orthogonal eigenvectors and eigenvalues of $\boldsymbol{\Delta}_l$, thus $\boldsymbol{\Delta}_l = \boldsymbol{\Phi}_l\boldsymbol{\Lambda}_l\boldsymbol{\Phi}_l^\top$, and $\boldsymbol{M}_l = \boldsymbol{\Phi}_l^\top\boldsymbol{T}_l\boldsymbol{\Phi}_l$ the spectral representation of $\boldsymbol{T}_l$ in $\boldsymbol{\Phi}_l$. Then commutativity becomes equivalent in the spectral domain to

$$\boldsymbol{\Lambda}_l\boldsymbol{M}_l = \boldsymbol{M}_l\boldsymbol{\Lambda}_l.$$

Looking at the equation entrywise, this means that

$$((\lambda_l)_i - (\lambda_l)_j)\,(\boldsymbol{M}_l)_{ij} = 0 \qquad \forall i, j,$$

where $(\lambda_l)_i$ is the i-th diagonal entry of $\boldsymbol{\Lambda}_l$ (eigenvalue of $\boldsymbol{T}_l$). As this is equivalent to $(\boldsymbol{M}_l)_{ij} = 0$ for different eigenvalues $(\lambda_l)_i \neq (\lambda_l)_j$, the commutativity condition is equivalent to the transformation only linearly recombining eigenvectors within eigenspaces, banning cross-eigenspace combinations.

If $\boldsymbol{T}_l$ is additionally constrained to be an orthogonal transformation $\boldsymbol{U}_l$, then its spectral representation $\boldsymbol{M}_l$ is also orthogonal. Since it is block diagonal (if eigenvalues are ordered monotonically), with blocks according to eigenspaces, due to the commutativity constraint, this means that each block is an orthogonal transformation (rotation) on an eigenspace. □

In other words, Proposition 15 means that commutants $\boldsymbol{T}_l$ with the Laplacian are expressed in the spectral basis, ordered monotonically by eigenvalues, as a block-diagonal matrix with square blocks for each eigenspace (eigenvectors corresponding to the same eigenvalue). Note that this doesn not mean that $\boldsymbol{T}_l$ is block diagonal, but its spectral representation $\boldsymbol{M}_l = \boldsymbol{\Phi}_l^\top\boldsymbol{T}_l\boldsymbol{\Phi}_l$ is.

G.2 Explicit solution to spectral re-basin

We here derive the optimal solution to the spectral re-basin objective. Denote $\boldsymbol{F}_l$ the features, in practice either weights $\boldsymbol{W}_l$ or activations $\boldsymbol{Z}_l$ when aligning weights or activations respectively, with spectral representations denoted as $\widehat{\boldsymbol{F}}_l^X$.

Let $\widetilde{\boldsymbol{F}}_l^B$ be the following shorthand

$$\widetilde{\boldsymbol{F}}_l^B = \begin{cases} \widehat{\boldsymbol{F}}_l^B\boldsymbol{C}_{l-1} & \text{when aligning weights } \boldsymbol{W}_l^X \\ \widehat{\boldsymbol{F}}_l^B & \text{when aligning activations } \boldsymbol{Z}_l^X. \end{cases} \tag{44}$$

Then the spectral re-basin objective can be rewritten whether aligning weights or activations as

$$\mathcal{L}(\boldsymbol{C}_l) = \|\widetilde{\boldsymbol{F}}_l^B - \boldsymbol{C}_l\widehat{\boldsymbol{F}}_l^A\|_F^2 + \mu\|\boldsymbol{\Lambda}_l^B\boldsymbol{C}_l - \boldsymbol{C}_l\boldsymbol{\Lambda}_l^A\|_F^2. \tag{45}$$

Let $\boldsymbol{G}_l$, $\boldsymbol{K}_l$, and $\boldsymbol{H}_l$ be the following matrices

$$\boldsymbol{G}_l = \widehat{\boldsymbol{F}}_l^A\widehat{\boldsymbol{F}}_l^{A\top}, \tag{46}$$

$$\boldsymbol{H}_l = \widetilde{\boldsymbol{F}}_l^B\widehat{\boldsymbol{F}}_l^{A\top}, \tag{47}$$

$$\boldsymbol{K}_l = \boldsymbol{G}_l \otimes \boldsymbol{I}_{k_l^B} + \mu\left(\boldsymbol{I}_{k_l^A} \otimes (\boldsymbol{\Lambda}_l^B)^2 - 2\,\boldsymbol{\Lambda}_l^A \otimes \boldsymbol{\Lambda}_l^B + (\boldsymbol{\Lambda}_l^A)^2 \otimes \boldsymbol{I}_{k_l^B}\right), \tag{48}$$

where $\otimes$ is the Kronecker product. We then have the following result.

Proposition 16. *The optimal spectral transformation minimising the spectral alignment objective $\boldsymbol{C}_l^* = \arg\min_{\boldsymbol{C}_l}\mathcal{L}(\boldsymbol{C}_l)$ is given by*

$$\boldsymbol{C}_l^* = \mathrm{unvec}_{k_l^B, k_l^A}\left(\boldsymbol{K}_l^\dagger\,\mathrm{vec}(\boldsymbol{H}_l)\right).$$

Proof. The gradient of the loss with respect to $\boldsymbol{C}_l$ is

$$\nabla_{\boldsymbol{C}_l}\mathcal{L}(\boldsymbol{C}_l) = 2(\boldsymbol{C}_l\boldsymbol{G}_l - \boldsymbol{H}_l) + 2\mu((\boldsymbol{\Lambda}_l^B)^2\boldsymbol{C}_l - 2\boldsymbol{\Lambda}_l^B\boldsymbol{C}_l\boldsymbol{\Lambda}_l^A + \boldsymbol{C}_l(\boldsymbol{\Lambda}_l^A)^2).$$

Setting the gradient to zero yields the optimality condition

$$\boldsymbol{C}_l^*\boldsymbol{G}_l + \mu((\boldsymbol{\Lambda}_l^B)^2\boldsymbol{C}_l^* - 2\boldsymbol{\Lambda}_l^B\boldsymbol{C}_l^*\boldsymbol{\Lambda}_l^A + \boldsymbol{C}_l^*(\boldsymbol{\Lambda}_l^A)^2) = \boldsymbol{H}_l,$$

or equivalently

$$\mu(\boldsymbol{\Lambda}_l^B)^2\boldsymbol{C}_l^* - 2\mu\,\boldsymbol{\Lambda}_l^B\boldsymbol{C}_l^*\boldsymbol{\Lambda}_l^A + \boldsymbol{C}_l^*(\boldsymbol{G}_l + \mu\,(\boldsymbol{\Lambda}_l^A)^2) = \boldsymbol{H}_l.$$

This is a generalised Sylvester matrix equation. We can solve the equation explicitly row-wise or resort to a compact vectorised expression. Applying the vectorisation identity

$$\mathrm{vec}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}) = (\boldsymbol{B}^\top \otimes \boldsymbol{A})\,\mathrm{vec}(\boldsymbol{X}),$$

and using that $\boldsymbol{G}_l$ is symmetric while $\boldsymbol{\Lambda}_l^A$ and $\boldsymbol{\Lambda}_l^B$ are diagonal, we obtain the explicit compact solution

$$\mathrm{vec}(\boldsymbol{C}_l^*) = \boldsymbol{K}_l^\dagger\,\mathrm{vec}(\boldsymbol{H}_l).$$

When $\boldsymbol{K}_l$ is invertible, for instance if $\boldsymbol{G}_l$ is positive definite, then the Moore-Penrose pseudoinverse reduces to the inverse $\boldsymbol{K}_l^\dagger = \boldsymbol{K}_l^{-1}$. Finally, unvectorising we have

$$\boldsymbol{C}_l^* = \mathrm{unvec}_{k_l^B,k_l^A}\left(\boldsymbol{K}_l^\dagger\,\mathrm{vec}(\boldsymbol{H}_l)\right).$$

□

G.3 Mixed weight space spectral re-basin

When performing weight-based re-basin, we transform the weights $\boldsymbol{W}_l^A$ of network A to align with those of network B and thus of B's architecture. For weights, unlike activations, we need to account for realignments at the previous layers. Thus the transfered weights $\boldsymbol{W}_l^{A\to B}$ must satisfy

$$\boldsymbol{W}_l^{A\to B}\boldsymbol{T}_{l-1}^* = \boldsymbol{T}_l^*\boldsymbol{W}_l^A. \tag{49}$$

The issue in our spectral formulation is that we do not constrain spectral solutions $\boldsymbol{C}_l$ to correspond to invertible (for same width architectures) or full rank (different widths) transforms $\boldsymbol{T}_l$. Thus, we must resort to the Moore-Penrose pseudoinverse instead to compute $\boldsymbol{W}_l^{A\to B}$ (Eq. (8)), which might bring errors compared to an invertible transformation by dropping information from A but not from B.

While $\boldsymbol{W}_l^{A\to B}$ aims to bring the weights of network A to the same loss basin as that of network B, albeit with the use of pseudo-inverts, we could instead avoid them by mapping both weights $\boldsymbol{W}_l^A$ and $\boldsymbol{W}_l^B$ to a shared space of weights of a new network that mixes between the architectures of A and B. This would treat A and B equally, unlike the asymmetric treatment in our main setting.

To achieve this, note that the optimal alignment $\boldsymbol{T}_l^*$ aims to satisfy the symmetry $\boldsymbol{W}_l^B\boldsymbol{T}_{l-1}^* \approx \boldsymbol{T}_l^*\boldsymbol{W}_l^A$, rather than $\boldsymbol{W}_l^B \approx \boldsymbol{T}_l^*\boldsymbol{W}_l^A{\boldsymbol{T}_{l-1}^*}^\dagger$. Thus, we suggest to transform both $\boldsymbol{W}_l^A$ and $\boldsymbol{W}_l^B$ to $\boldsymbol{W}_l^{(A\to B)/2} \in \mathbb{R}^{d_l^B\times D_{l-1}^A}$ and $\boldsymbol{W}_l^{(B\to A)/2} \in \mathbb{R}^{d_l^B\times D_{l-1}^A}$ belonging to a shared weight space and basin without approximation induced from using pseudo inverts

$$\boldsymbol{W}_l^{(A\to B)/2} = \boldsymbol{T}_l^*\boldsymbol{W}_l^A \quad\text{and}\quad \boldsymbol{W}_l^{(B\to A)/2} = \boldsymbol{W}_l^B\boldsymbol{T}_{l-1}^*. \tag{50}$$

Note that if the transformations are not invertible, this formulation treats equally both networks A and B, as transforms are applied to the weights of both networks. Interpolation can then be done between the warped weights $\boldsymbol{W}_l^{(A\to B)/2}$ and $\boldsymbol{W}_l^{(A\to B)/2}$. However, these weights cannot be directly fed to A or B's neural architectures. Instead, they need to be fed to an architecture oscillating between the widths of A and B. Rather than running in a new architecture, should we want to return in say B's (resp. A's) architecture, we would need to use the pseudoinverse ${\boldsymbol{T}_{l-1}^*}^\dagger$ (resp. ${\boldsymbol{T}_l^*}^\dagger$) to the right (resp. left) of $\boldsymbol{W}_l^{(A\to B)/2}$ (resp. $\boldsymbol{W}_l^{(B\to A)/2}$).

G.4 Effect of matching direction

Spectral re-basin is directional in general. Given two models A and B, our main formulation learns maps $\boldsymbol{T}_l^{A\to B} : \mathbb{R}^{d_l^A} \to \mathbb{R}^{d_l^B}$ and transports model A into the parameter space of model B. The resulting endpoint $\boldsymbol{\theta}^{A\to B}$ can then be interpolated with $\boldsymbol{\theta}^B$. One could instead learn maps

$\boldsymbol{T}_l^{B\to A} : \mathbb{R}^{d_l^B} \to \mathbb{R}^{d_l^A}$ and interpolate $\boldsymbol{\theta}^{B\to A}$ with $\boldsymbol{\theta}^A$. These two procedures are not equivalent in general.

For the weight-based variant, the forward direction $A \to B$ solves

$$\min_{\boldsymbol{C}_l^{A\to B}} \left\| \widehat{\boldsymbol{W}}_l^B \boldsymbol{C}_{l-1}^{A\to B} - \boldsymbol{C}_l^{A\to B} \widehat{\boldsymbol{W}}_l^A \right\|_F^2 + \mu \left\| \boldsymbol{\Lambda}_l^B \boldsymbol{C}_l^{A\to B} - \boldsymbol{C}_l^{A\to B} \boldsymbol{\Lambda}_l^A \right\|_F^2 . \tag{51}$$

The reverse direction $B \to A$ instead solves

$$\min_{\boldsymbol{C}_l^{B\to A}} \left\| \widehat{\boldsymbol{W}}_l^A \boldsymbol{C}_{l-1}^{B\to A} - \boldsymbol{C}_l^{B\to A} \widehat{\boldsymbol{W}}_l^B \right\|_F^2 + \mu \left\| \boldsymbol{\Lambda}_l^A \boldsymbol{C}_l^{B\to A} - \boldsymbol{C}_l^{B\to A} \boldsymbol{\Lambda}_l^B \right\|_F^2 . \tag{52}$$

The minimisers of Eq. (51) and Eq. (52) are not necessarily inverses of each other. They would coincide up to inversion only in an idealised case where the maps are square and invertible, the spectral bases are not truncated, and the two models admit an exact zero-residual correspondence. In the regimes studied in this paper, these assumptions generally do not hold: the maps can be rectangular, the spectral representation is truncated to the low-frequency subspace, and the correspondence is approximate.

The directionality is important when the widths differ. If A is narrower than B, then $\boldsymbol{T}_l^{A\to B}$ is a tall map that embeds the lower-dimensional neuron space of A into the wider neuron space of B. The transported weights

$$\boldsymbol{W}_l^{A\to B} = \boldsymbol{T}_l^{A\to B} \boldsymbol{W}_l^A \left(\boldsymbol{T}_{l-1}^{A\to B} \right)^\dagger \tag{53}$$

therefore have the same shape as $\boldsymbol{W}_l^B$, so the interpolation is performed in the wider architecture. In contrast, the reverse direction $B \to A$ compresses the wider model into the narrower architecture:

$$\boldsymbol{W}_l^{B\to A} = \boldsymbol{T}_l^{B\to A} \boldsymbol{W}_l^B \left(\boldsymbol{T}_{l-1}^{B\to A} \right)^\dagger . \tag{54}$$

This reverse interpolation is well-defined, but it evaluates a different path in a different parameter space. The two midpoint models are not the same object.

This directionality is also reflected in the layerwise error bound. For $A \to B$, the bound contains the projection term

$$\Pi_l^{A\to B} = \left\| \boldsymbol{W}_l^B \left(\boldsymbol{I} - \boldsymbol{T}_{l-1}^{A\to B} \left(\boldsymbol{T}_{l-1}^{A\to B} \right)^\dagger \right) \right\|_F . \tag{55}$$

When $\boldsymbol{T}_{l-1}^{A\to B}$ is rectangular, this term measures how much of the target layer $\boldsymbol{W}_l^B$ lies outside the subspace reachable from the transported source model. The corresponding reverse-direction term is different as it is computed with $\boldsymbol{T}_{l-1}^{B\to A}$ and the target layer $\boldsymbol{W}_l^A$. Thus, the theoretical barrier bound is also directional.

In the main experiments, we fix the matching direction rather than treating direction as an additional hyperparameter. For identical-width experiments, we choose one endpoint as the reference model B and transport the other endpoint A into its coordinate system. For mismatched-width experiments, we always transport the narrower model into the wider model, i.e., $\times 1 \to \times 4$ and $\times 4 \to \times 8$. This convention matches the zero-padding protocol used by the baselines: all midpoint models are evaluated in the wider architecture. It also avoids conflating alignment quality with compression error from forcing a wider endpoint into a narrower architecture.

For activation-based spectral re-basin, the same directional issue applies. The forward activation objective learns $\boldsymbol{T}_l^{A\to B}$ by solving

$$\min_{\boldsymbol{C}_l^{A\to B}} \left\| \widehat{\boldsymbol{Z}}_l^B - \boldsymbol{C}_l^{A\to B} \widehat{\boldsymbol{Z}}_l^A \right\|_F^2 + \mu \left\| \boldsymbol{\Lambda}_l^B \boldsymbol{C}_l^{A\to B} - \boldsymbol{C}_l^{A\to B} \boldsymbol{\Lambda}_l^A \right\|_F^2 , \tag{56}$$

whereas the reverse direction exchanges the roles of A and B. Since the learned maps are then converted into transported weights, the final interpolation path remains direction-dependent for the same reasons as in the weight-based variant.

We do not impose a cycle-consistency constraint such as

$$\boldsymbol{T}_l^{B\to A} \boldsymbol{T}_l^{A\to B} \approx \boldsymbol{I} \quad \text{or} \quad \boldsymbol{T}_l^{A\to B} \boldsymbol{T}_l^{B\to A} \approx \boldsymbol{I}. \tag{57}$$

Such a constraint would bias the method toward bijective correspondences and would work against the main purpose of our framework: allowing non-bijective relations such as splitting, merging, and cross-width neuron correspondences. Thus, directionality is an expected feature of spectral re-basin in the mismatched-width setting, not an artefact of the implementation.

H Complexity comparison to weight matching and activation matching

A practical distinction from permutation-based re-basin is that the core alignment step in spectral re-basin is continuous rather than combinatorial. In Git Re-Basin, activation matching solves a layerwise linear assignment problem after collecting activations, while weight matching approximates a harder joint permutation problem by repeated coordinate-descent updates of layerwise assignments; the exact joint formulation is NP-hard for deep networks, and the straight-through estimator is more expensive still because it repeatedly projects through the permutation solver during optimisation [5]. By contrast, our spectral method solves for a coefficient matrix C in reduced spectral coordinates. This turns the alignment subproblem into a ridge-regularised least-squares system on $k \times k$ variables, where k is the number of retained graph modes. Thus, spectral re-basin replaces repeated combinatorial search over hidden-unit permutations with a small continuous solve, while still encoding layerwise functional structure through the graph operators.

For a layer of width d, activation matching requires a forward pass to collect activations and then solves a linear assignment problem (LAP) on a $d \times d$ cost matrix, so after the activation pass its per-layer matching cost scales roughly cubically with width. Weight matching is structurally more expensive: the exact joint objective is NP-hard, and practical solvers instead perform repeated LAP updates across layers until convergence. STE is costlier still, because it repeatedly invokes that projection inside a training loop. By contrast, in spectral re-basin, the unknown is a continuous $k \times k$ spectral coefficient matrix C, where k denotes the number of retained modes and is typically much smaller than the layer width. The descriptor projections cost approximately $O((d_A + d_B)km)$ for m descriptor channels, while the main optimisation reduces to a ridge-regularised linear system in k^2 variables. In the diagonalised or whitened setting, this system even decouples entrywise. Thus, the computational simplification is genuine: the difficult step is no longer a search over permutations in S_d, but the solution of a reduced continuous problem in spectral coordinates.

I Matching multiple networks

So far, we have considered spectral re-basin as a pairwise procedure: given two networks A and B, we estimate layerwise maps that transport one model into the coordinate system of the other. This is sufficient for studying linear path loss barriers between two independently trained models, but it becomes unsatisfactory once more than two networks are involved.

The difficulty is that pairwise matches are generally not globally consistent. If we align network A to B, and then B to C, there is no reason for the induced alignment from A to C to agree with the one obtained by matching A and C directly. In other words, pairwise matching is typically reference-dependent and may violate cycle consistency. This is nontrivial when the goal is to analyse families of solutions, construct shared low-loss regions, or merge many models at once [40, 148].

To overcome this, we extend spectral re-basin from pairwise alignment to multiway alignment. Instead of matching each network to a chosen reference, we introduce a shared latent coordinate system at every layer (similar to in spirit to §G.3) and match all networks to this common space simultaneously. This yields three advantages. First, it removes the arbitrary choice of a reference model. Second, it makes the correspondences synchronised across the whole collection. Third, it produces a common barycentric representation that can be used to control pairwise distances and interpolation barriers among all matched networks.

I.1 Set up

Let $\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(M)}$ be M feed-forward networks. For network $m \in \{1, \dots, M\}$, and let $\boldsymbol{W}_l^{(m)} \in \mathbb{R}^{d_l^{(m)} \times d_{l-1}^{(m)}}$ be the corresponding weights for $l = 1, \dots, L$. As before, we assume a common input dimension d_0 and common output dimension d_L, while hidden widths may vary with m. At each hidden layer l, network m induces a graph shift operator $\boldsymbol{\Delta}_l^{(m)} \approx \boldsymbol{\Phi}_l^{(m)} \boldsymbol{\Lambda}_l^{(m)} {\boldsymbol{\Phi}_l^{(m)}}^\top$, with truncated spectral rank $k_l^{(m)}$. We also define the spectral weights $\widehat{\boldsymbol{W}}_l^{(m)} = {\boldsymbol{\Phi}_l^{(m)}}^\top \boldsymbol{W}_l^{(m)} \boldsymbol{\Phi}_{l-1}^{(m)}$. For the shared latent space, we choose dimensions as $q_0 = d_0$, $q_L = d_L$, and $q_l \le \min_m k_l^{(m)}$ for $1 \le l \le L-1$. The integer q_l is the dimension of the common layer-l latent space to which all networks are aligned.

We denote by $\overline{\boldsymbol{W}}_l \in \mathbb{R}^{q_l \times q_{l-1}}$ the shared latent weight matrix at layer l, and by $\overline{\boldsymbol{\Lambda}}_l \in \mathbb{R}^{q_l \times q_l}$ a diagonal latent graph-spectrum. For each network m, the layerwise map into the latent space is represented spectrally by $\boldsymbol{C}_l^{(m)} \in \mathbb{R}^{q_l \times k_l^{(m)}}$, $\boldsymbol{C}_0^{(m)} = \boldsymbol{I}_{d_0}$, and $\boldsymbol{C}_L^{(m)} = \boldsymbol{I}_{d_L}$. Thus $\boldsymbol{C}_l^{(m)}$ plays the same role as the pairwise spectral map, but now it transports network m into the shared latent coordinates rather than into another single network.

I.2 A shared latent objective

The guiding principle is the same as in the pairwise case: after transport, each network should look like the same latent model, both in terms of its layerwise weights and in terms of the graph geometry of its hidden representations. This leads to the multiway objective

$$\sum_{m=1}^{M} \sum_{l=1}^{L} \left\| \overline{\boldsymbol{W}}_l \boldsymbol{C}_{l-1}^{(m)} - \boldsymbol{C}_l^{(m)} \widehat{\boldsymbol{W}}_l^{(m)} \right\|_F^2 + \sum_{m=1}^{M} \sum_{l=1}^{L-1} \mu_l \left\| \overline{\boldsymbol{\Lambda}}_l \boldsymbol{C}_l^{(m)} - \boldsymbol{C}_l^{(m)} \boldsymbol{\Lambda}_l^{(m)} \right\|_F^2. \tag{58}$$

The first term is the multiway feature-alignment. It requires the transported layer of each network to agree with the same latent weight matrix $\overline{\boldsymbol{W}}_l$. The second term is the multiway graph-functional term. It requires every network's layer graph to be compatible with the same latent spectral template $\overline{\boldsymbol{\Lambda}}_l$.

This formulation contains pairwise matching as a special case. If one fixes $\overline{\boldsymbol{W}}_l$ and $\overline{\boldsymbol{\Lambda}}_l$ to those of a chosen reference network, Eq. (58) reduces to matching every model independently to that reference. Specifically, Eq. (58) is the barycentric extension of pairwise spectral re-basin. In the pairwise setting, the target network provides the coordinate system and the feature template. In the multiway setting, both are replaced by learned latent objects $\overline{\boldsymbol{W}}_l$ and $\overline{\boldsymbol{\Lambda}}_l$. The resulting maps are therefore synchronised by construction: every network is matched to the same hidden geometry and the same latent layerwise model. Note that for fixed correspondence maps, the latent templates have simple least-squares updates, which can be obtained by the least-squares Fréchet mean of the transported spectral weights.

I.3 Consistency of induced pairwise maps

Once all networks are aligned to the same latent space, we can recover induced pairwise correspondences between any two networks i and j by composing through the shared coordinates:

$$\boldsymbol{P}_l^{(j \leftarrow i)} := \boldsymbol{C}_l^{(j)\dagger} \boldsymbol{C}_l^{(i)}. \tag{59}$$

This is the multiway analogue of a pairwise spectral map. The advantage is that these induced maps inherit consistency from the shared latent representation.

Proposition 17. *Assume that, at layer l, every map $\boldsymbol{C}_l^{(m)}$ has full row rank, so that*

$$\boldsymbol{C}_l^{(m)} \boldsymbol{C}_l^{(m)\dagger} = \boldsymbol{I}_{q_l}. \tag{60}$$

Then the induced pairwise maps in Eq. (59) satisfy the cycle-consistency relation

$$\boldsymbol{P}_l^{(k \leftarrow j)} \boldsymbol{P}_l^{(j \leftarrow i)} = \boldsymbol{P}_l^{(k \leftarrow i)} \qquad \text{for all } i, j, k. \tag{61}$$

Proof. By definition, $\boldsymbol{P}_l^{(k \leftarrow j)} \boldsymbol{P}_l^{(j \leftarrow i)} = \boldsymbol{C}_l^{(k)\dagger} \boldsymbol{C}_l^{(j)} \boldsymbol{C}_l^{(j)\dagger} \boldsymbol{C}_l^{(i)}$. Since $\boldsymbol{C}_l^{(j)} \boldsymbol{C}_l^{(j)\dagger} = \boldsymbol{I}_{q_l}$, this simplifies to $\boldsymbol{C}_l^{(k)\dagger} \boldsymbol{C}_l^{(i)} = \boldsymbol{P}_l^{(k \leftarrow i)}$. □

Prop. 17 shows that pairwise alignments estimated independently do not usually compose correctly, and the relation between three networks may depend on which intermediate anchor is used. In contrast, once all networks are matched to a single latent space, the induced pairwise maps are automatically synchronised.

I.4 Multi-network re-basin and pairwise barrier

To connect the multiway construction back to linear path loss barriers, we define the rebased spectral weights of network m in the shared latent coordinates:

$$\overline{\boldsymbol{W}}_l^{(m \to \star)} := \boldsymbol{C}_l^{(m)} \widehat{\boldsymbol{W}}_l^{(m)} \boldsymbol{C}_{l-1}^{(m)\dagger}. \tag{62}$$

This is the multi-network analogue of the pairwise rebased weight $\boldsymbol{W}_l^{A\to B}$: it expresses network m in the shared latent coordinates. We next define the layerwise residual of network m relative to the latent barycenter:

$$R_l^{(m)} := \left\| \overline{\boldsymbol{W}}_l \boldsymbol{C}_{l-1}^{(m)} - \boldsymbol{C}_l^{(m)} \widehat{\boldsymbol{W}}_l^{(m)} \right\|_F. \tag{63}$$

This is the multi-network version of the pairwise alignment residual. It measures how well the transported layer of network m agrees with the shared latent weight matrix.

Because the maps may be rectangular, we also define the projection error

$$\Pi_l^{(m)} := \left\| \overline{\boldsymbol{W}}_l \left(\boldsymbol{I} - \boldsymbol{C}_{l-1}^{(m)} \boldsymbol{C}_{l-1}^{(m)\dagger} \right) \right\|_F. \tag{64}$$

This measures how much of the latent layer-$(l-1)$ space is not reachable by network m under the transport map. It vanishes when $\boldsymbol{C}_{l-1}^{(m)}$ is full row rank.

Combining these two effects, we define

$$\alpha_l^{(m)} := \Pi_l^{(m)} + R_l^{(m)} \left\| \boldsymbol{C}_{l-1}^{(m)\dagger} \right\|_2. \tag{65}$$

The quantity $\alpha_l^{(m)}$ is the total layerwise deviation of the rebased network m from the latent barycenter at layer l. By the same argument as in the pairwise case, we obtain the bound

$$\left\| \overline{\boldsymbol{W}}_l^{(m\to\star)} - \overline{\boldsymbol{W}}_l \right\|_F \le \alpha_l^{(m)}. \tag{66}$$

Thus each network is controlled layer by layer by its distance to the common latent model. Note that controlling distance to the same barycenter also controls distances between every pair of rebased networks. Indeed, by the triangle inequality,

$$\left\| \overline{\boldsymbol{W}}_l^{(i\to\star)} - \overline{\boldsymbol{W}}_l^{(j\to\star)} \right\|_F \le \alpha_l^{(i)} + \alpha_l^{(j)}. \tag{67}$$

This leads to the following multi-network barrier bound.

Proposition 18. *Fix two indices $i, j \in \{1, \dots, M\}$, and let $\overline{\boldsymbol{\theta}}^{(i\to\star)} = \left\{ \overline{\boldsymbol{W}}_l^{(i\to\star)} \right\}_{l=1}^L$ and $\overline{\boldsymbol{\theta}}^{(j\to\star)} = \left\{ \overline{\boldsymbol{W}}_l^{(j\to\star)} \right\}_{l=1}^L$. Consider the linear interpolation in the shared latent space, $\overline{\boldsymbol{\theta}}^{(ij)}(t) = (1-t)\,\overline{\boldsymbol{\theta}}^{(i\to\star)} + t\,\overline{\boldsymbol{\theta}}^{(j\to\star)}$ for $t \in [0,1]$. Assume that the loss function $\mathcal{L}$ is twice differentiable along this segment and that $\left\| \nabla^2 \mathcal{L}\left(\overline{\boldsymbol{\theta}}^{(ij)}(t) \right) \right\|_2 \le \beta$ for all $t \in [0,1]$. Then*

$$\max_{t\in[0,1]} \mathcal{L}\left(\overline{\boldsymbol{\theta}}^{(ij)}(t) \right) - \max\left\{ \mathcal{L}\left(\overline{\boldsymbol{\theta}}^{(i\to\star)} \right), \mathcal{L}\left(\overline{\boldsymbol{\theta}}^{(j\to\star)} \right) \right\} \le \frac{\beta}{8} \sum_{l=1}^{L} \left(\alpha_l^{(i)} + \alpha_l^{(j)} \right)^2. \tag{68}$$

Proof. By Eq. (67), we have $\left\| \overline{\boldsymbol{W}}_l^{(i\to\star)} - \overline{\boldsymbol{W}}_l^{(j\to\star)} \right\|_F \le \alpha_l^{(i)} + \alpha_l^{(j)}$. Hence the squared parameter distance between the two rebased models satisfies $\left\| \overline{\boldsymbol{\theta}}^{(i\to\star)} - \overline{\boldsymbol{\theta}}^{(j\to\star)} \right\|_2^2 \le \sum_{l=1}^L \left(\alpha_l^{(i)} + \alpha_l^{(j)} \right)^2$. The claim then follows from the same one-dimensional smoothness argument. □

Prop. 18 is the multi-network analogue of the pairwise bound. It shows that once every model is close to a shared latent barycenter, then every pair of rebased models is close to one another, and consequently every pairwise linear interpolation barrier is controlled.

J Experimental details

Here we provide the experimental setups and details used in §5, including matching on identical architectures (§5.1), matching across mismatched widths (§5.2), and matching asymmetric networks of different widths (§5.3). Throughout the experiments, we compare two independently trained endpoint networks, denoted by A and B. Unless otherwise stated, model B is treated as the reference coordinate system, and all alignment methods first map model A into the coordinate system or the

width of model B before evaluating the midpoint of the linear interpolation. The experiments are conducted on NVIDIA HGX H100.

J.1 Evaluation protocol

Let $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^B$ denote the trained parameters of two endpoint networks. For an alignment method that produces transformed parameters $\boldsymbol{\theta}^{A\to B}$, we evaluate the line segment

$$\boldsymbol{\theta}(t) = (1-t)\boldsymbol{\theta}^{A\to B} + t\boldsymbol{\theta}^B, \qquad t \in [0,1]. \tag{69}$$

For the main merging on identical architectures and mismatched widths, we report the test accuracy at the midpoint $t = 1/2$:

$$\boldsymbol{\theta}_{\text{mid}} = \frac{1}{2}\boldsymbol{\theta}^{A\to B} + \frac{1}{2}\boldsymbol{\theta}^B. \tag{70}$$

For the asymmetric-network experiments, we report the midpoint test loss barrier

$$\mathcal{B}_{1/2} = \mathcal{L}\left(\frac{1}{2}\boldsymbol{\theta}^{A\to B} + \frac{1}{2}\boldsymbol{\theta}^B\right) - \frac{1}{2}\left(\mathcal{L}(\boldsymbol{\theta}^{A\to B}) + \mathcal{L}(\boldsymbol{\theta}^B)\right). \tag{71}$$

When no alignment is applied, $\boldsymbol{\theta}^{A\to B}$ is simply $\boldsymbol{\theta}^A$ in the identical-width case and the zero-padded embedding of $\boldsymbol{\theta}^A$ in the cross-width case. We report the mean and standard deviation over eight independently trained model pairs.

For methods using activations, we use only training-set data to compute alignments. Test data are used only for final evaluation. All methods are evaluated under the same trained endpoints, the same alignment batches, and the same test set. For architectures with batch-normalisation layers, we reset the running batch-normalisation statistics of the merged/interpolated model using the training data before test evaluation, following prior work's practice. The endpoint accuracies and ensemble accuracies reported are computed using the same evaluation pipeline.

J.2 Datasets and base architectures

J.2.1 CIFAR experiments

For identical-width and mismatched-width experiments, we use VGG11 on CIFAR-10 and ResNet20 on CIFAR-100. The VGG11 experiments use width multipliers $\times 1$, $\times 4$, and $\times 8$. The ResNet20 experiments also use width multipliers $\times 1$, $\times 4$, and $\times 8$. The identical-architecture experiments compare pairs with the same architecture and the same width. The mismatched-width experiments compare models from the same architecture family but with different widths: *(i)* VGG11 $\times 1 \to \times 4$, *(ii)* VGG11 $\times 4 \to \times 8$, *(iii)* ResNet20 $\times 1 \to \times 4$, and *(iv)* ResNet20 $\times 4 \to \times 8$. In all cross-width experiments, the smaller model is aligned to the wider model.

Following the setup used by Stoica et al. (2024) [94], CIFAR models are trained with a CLIP-style classification objective. Specifically, the class names are embedded using a fixed CLIP text encoder, producing class vectors $\{\boldsymbol{c}_y\}_{y=1}^C \subset \mathbb{R}^{512}$. The neural network outputs an embedding $g_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^{512}$. During training and evaluation, the output embedding is normalised and classified by cosine similarity against the CLIP class vectors $s_y(\boldsymbol{x}) = 100 \cdot \frac{g_{\boldsymbol{\theta}}(\boldsymbol{x})^\top}{\|g_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2}\boldsymbol{c}_y$, and the model is trained with cross-entropy over the scores $\{s_y(\boldsymbol{x})\}_{y=1}^C$. We train the CIFAR models for 200 epochs using stochastic gradient descent with momentum 0.9 and weight decay 5×10^{-4}. When training with the CLIP-style objective, we use a base learning rate of 0.4 and a linear learning-rate schedule: the learning-rate multiplier is warmed up from 0 to 1 during the first 5 epochs and then linearly decayed to 0 by the end of training. The effective learning rate at epoch e is therefore 0.4 times the corresponding schedule multiplier. We use mixed-precision training. For CIFAR-10, images are normalised with mean $(0.4914, 0.4822, 0.4465)$ and standard deviation $(0.2470, 0.2435, 0.2616)$. For CIFAR-100, images are normalised with mean $(0.5071, 0.4867, 0.4408)$ and standard deviation $(0.2675, 0.2565, 0.2761)$. The training transform consists of random horizontal flipping, random cropping with padding of 4, conversion to a tensor, and normalisation. The test transform consists of conversion to a tensor followed by normalisation.

J.2.2 Asymmetric network experiments

For the asymmetric network experiments, we use the $\boldsymbol{W}$-Asym construction of [37]. A $\boldsymbol{W}$-Asym layer fixes a subset of entries in each linear map to random constants and keeps these entries untrained throughout optimisation. This breaks hidden-unit or channel permutation symmetries by assigning each neuron or channel a distinct fixed weight. The remaining entries are trainable. We refer to [37] for more details.

We evaluate three width-mismatched asymmetric settings: *(i)* MLP $128 \to 256$ on MNIST dataset, *(ii)* ResNet20 $\times 4 \to \times 8$ on CIFAR-10 dataset, and *(iii)* GIN $128 \to 256$ on ogbn-arXiv. The MLP is a 4-layer network with LayerNorm. The ResNet20 uses the LayerNorm version from the asymmetric network setup. The GIN uses three message-passing layers with mean aggregation. For MNIST, we use no data augmentation. For CIFAR-10, we use random cropping and horizontal flipping during training. For ogbn-arXiv, training is transductive node classification on the full graph, and we follow the full-batch training setup from [37].

For each random seed, the $\boldsymbol{W}$-Asym construction is sampled independently. That is, both the binary mask determining which entries are fixed and the corresponding fixed random constants are resampled for every trained endpoint. Consequently, two models in an alignment pair differ not only in random initialisation and width, but also in the specific symmetry-breaking mechanism used during training. During training of a given endpoint, the entries selected by its own mask are kept fixed, while the remaining entries are optimised. After training, we form the effective weight tensors of each endpoint, containing both trained entries and fixed entries, and use these effective tensors for alignment, interpolation, and evaluation. Thus, the intermediate models along the interpolation path are not constrained to preserve either endpoint's original $\boldsymbol{W}$-Asym mask. This makes the setting more challenging than matching asymmetric networks that share the same symmetry breaking mechanism as the two endpoints may have incompatible fixed masks, incompatible fixed constants, and different widths.

J.3 Construction of width-mismatched padded baselines

Some baselines, such as direct averaging, weight matching, activation matching, and the standard same-width version of CCA Merge, require the two models to have matching parameter shapes. For cross-width experiments, we therefore construct a function-preserving zero-padded embedding of the smaller model into the larger architecture. These methods are marked with *.

For a fully-connected layer with weight matrix $\boldsymbol{W}_l^A \in \mathbb{R}^{d_l^A \times d_{l-1}^A}$ and target width $d_l^B \geq d_l^A$, we embed the smaller model by adding zero rows for new output units and zero columns for new input units. The next layer is padded consistently so that the added units do not affect the network output. For a convolutional layer with weights $\boldsymbol{W}_l^A \in \mathbb{R}^{C_{\text{out}}^A \times C_{\text{in}}^A \times k_h \times k_w}$, we add zero output filters and zero input-channel slices to match $\mathbb{R}^{C_{\text{out}}^B \times C_{\text{in}}^B \times k_h \times k_w}$. For affine normalisation parameters, added channels are initialised to the identity/zero defaults: scale equal to 1, bias equal to 0, running mean equal to 0, and running variance equal to 1. The padded channels are therefore inactive, and the padded network computes exactly the same function as the original smaller network before interpolation.

For residual architectures, padding is applied consistently across all branches that meet at a residual addition. Projection shortcuts are padded in the same way as convolutional layers. For output layers, the output dimension is not padded because the class or CLIP-embedding output space is already shared and only the input dimension to the output layer is padded.

J.4 Spectral re-basin implementation

We evaluate two variants of our method: weight-based spectral re-basin and activation-based spectral re-basin. Both variants construct a layerwise neuron graph and solve a spectral least-squares problem to obtain a transformation from the neuron space of model A to the neuron space of model B.

J.4.1 Layer descriptors

For the weight-based variant, each neuron or channel is represented by its incoming weight vector. For a linear layer, the descriptor matrix is simply the row-flattened weight matrix $\boldsymbol{F}_l^X = \boldsymbol{W}_l^X \in \mathbb{R}^{d_l^X \times d_{l-1}^X}$. For a convolutional layer, each output channel is treated as one node, and its filter tensor

is flattened across input channels and spatial dimensions $\boldsymbol{F}_l^X = \mathrm{reshape}\left(\boldsymbol{W}_l^X\right) \in \mathbb{R}^{C_{\mathrm{out}}^X \times (C_{\mathrm{in}}^X k_h k_w)}$. For the activation-based variant, each neuron or channel is represented by its activation pattern on the alignment data. For a linear layer with activations $\boldsymbol{Z}_l^X \in \mathbb{R}^{d_l^X \times n}$, rows are neurons and columns are examples. For a convolutional layer with an activation tensor $\boldsymbol{Z}_l^X \in \mathbb{R}^{n \times C_l^X \times H_l \times W_l}$, we flatten the batch and spatial axes and use $\boldsymbol{F}_l^X \in \mathbb{R}^{C_l^X \times (nH_lW_l)}$ as the descriptor matrix.

J.4.2 Neuron graph

Given a descriptor matrix $\boldsymbol{F}_l^X \in \mathbb{R}^{d_l^X \times p_l}$, we construct a weighted undirected k-nearest-neighbour graph over the d_l^X rows. Distances are Euclidean distances between descriptor rows $d_{ij}^2 = \left\|(\boldsymbol{F}_l^X)_{i,:} - (\boldsymbol{F}_l^X)_{j,:}\right\|_2^2$. We connect each node to its k nearest neighbours, symmetrise the graph by taking the union of directed nearest-neighbour edges, and set the adjacency weight to

$$(\boldsymbol{A}_l^X)_{ij} = \exp\left(-\frac{d_{ij}^2}{(\sigma_l^X)^2}\right) \quad \text{if } i \sim j, \qquad (\boldsymbol{A}_l^X)_{ij} = 0 \quad \text{otherwise.} \tag{72}$$

The kernel bandwidth is $\sigma_l^X = \tau_\sigma \cdot \mathrm{median}\{d_{ij} : i < j\}$, where τ_σ is swept over the grid specified in the §J.4.5. The number of nearest neighbours k is swept over the same grid-search protocol. We use the symmetric normalised graph Laplacian

$$\boldsymbol{\Delta}_l^X = \boldsymbol{I} - (\boldsymbol{D}_l^X)^{-1/2} \boldsymbol{A}_l^X (\boldsymbol{D}_l^X)^{-1/2}, \tag{73}$$

where $\boldsymbol{D}_l^X$ is the diagonal degree matrix. The eigendecomposition $\boldsymbol{\Delta}_l^X = \boldsymbol{\Phi}_l^X \boldsymbol{\Lambda}_l^X \boldsymbol{\Phi}_l^{X\top}$ is computed with eigenvalues ordered increasingly. We retain the lowest-frequency $q_l^X = \min\{d_l^X, 50\}$ eigenvectors. If the full basis is used, this is equivalent to setting $q_l^X = d_l^X$.

J.4.3 Weight-based spectral re-basin

For weight-based alignment, we solve layer by layer. The previous-layer transformation is needed because a weight matrix maps from layer $l-1$ to layer l. In the spectral domain, the optimisation problem is

$$\min_{\boldsymbol{C}_l} \left\| \widehat{\boldsymbol{W}}_l^B \boldsymbol{C}_{l-1} - \boldsymbol{C}_l \widehat{\boldsymbol{W}}_l^A \right\|_F^2 + \mu \left\| \boldsymbol{\Lambda}_l^B \boldsymbol{C}_l - \boldsymbol{C}_l \boldsymbol{\Lambda}_l^A \right\|_F^2, \tag{74}$$

where $\widehat{\boldsymbol{W}}_l^X = \boldsymbol{\Phi}_l^{X\top} \boldsymbol{W}_l^X \boldsymbol{\Phi}_{l-1}^X$. The boundary maps are identities at the input and output $\boldsymbol{T}_0 = \boldsymbol{I}, \boldsymbol{T}_L = \boldsymbol{I}$. For convolutional layers, the input transformation is applied over the input-channel dimension only. Equivalently, after flattening the convolutional kernel into a matrix, the aligned convolution is computed as

$$\overline{\boldsymbol{W}}_l^{A\to B} = \boldsymbol{T}_l \overline{\boldsymbol{W}}_l^A \left(\boldsymbol{T}_{l-1}^\dagger \otimes \boldsymbol{I}_{k_h k_w}\right), \tag{75}$$

where $\overline{\boldsymbol{W}}_l$ denotes the channel-flattened convolutional weight and $\otimes$ is the Kronecker product. The result is reshaped back to a convolutional-kernel format.

After solving Eq. (74), the coordinate-space transformation is $\boldsymbol{T}_l = \boldsymbol{\Phi}_l^B \boldsymbol{C}_l \boldsymbol{\Phi}_l^{A\top}$. The transformed weight is

$$\boldsymbol{W}_l^{A\to B} = \boldsymbol{T}_l \boldsymbol{W}_l^A \boldsymbol{T}_{l-1}^\dagger. \tag{76}$$

The Moore-Penrose pseudoinverse is computed by SVD. For numerical stability, the least-squares solve is performed in double precision, and the final transformed parameters are cast back to the model dtype.

J.4.4 Activation-based spectral re-basin

For activation-based alignment, each layer is solved independently because activation descriptors already live in the layer representation space. The spectral objective is

$$\min_{\boldsymbol{C}_l} \left\| \widehat{\boldsymbol{Z}}_l^B - \boldsymbol{C}_l \widehat{\boldsymbol{Z}}_l^A \right\|_F^2 + \mu \left\| \boldsymbol{\Lambda}_l^B \boldsymbol{C}_l - \boldsymbol{C}_l \boldsymbol{\Lambda}_l^A \right\|_F^2, \tag{77}$$

where $\widehat{\boldsymbol{Z}}_l^X = \boldsymbol{\Phi}_l^{X\top} \boldsymbol{Z}_l^X$. The resulting map is again $\boldsymbol{T}_l = \boldsymbol{\Phi}_l^B \boldsymbol{C}_l \boldsymbol{\Phi}_l^{A\top}$. We convert the learned activation maps into transported weights using the same formula as in the weight-based case $\boldsymbol{W}_l^{A\to B} = \boldsymbol{T}_l \boldsymbol{W}_l^A \boldsymbol{T}_{l-1}^\dagger$. This gives a single parameter-space endpoint $\boldsymbol{\theta}^{A\to B}$.

J.4.5 Optimisation and hyperparameters

For each dataset-architecture-width setting and each spectral re-basin variant, we train one additional endpoint pair using independent random seeds and reserve it solely for hyperparameter selection. This validation pair is excluded from the eight independently trained endpoint pairs used to compute the reported means and standard deviations. Before endpoint training, we also reserve 10% of the original training data as a validation split: 5,000 examples for CIFAR-10 and CIFAR-100, 6,000 examples for MNIST, and the official validation split for ogbn-ArXiv. The remaining training data are used to train endpoint models, construct activation descriptors, fit the spectral transformations, and recalibrate batch-normalisation statistics. Validation examples are used only to rank candidate hyperparameter configurations; they are not used to construct neuron graphs or fit the layerwise maps. Test data are used only for final evaluation.

For each benchmark setting and spectral variant, we run one 30-trial Optuna [155] study with the TPE sampler, consisting of 10 startup trials followed by 20 guided trials. Each trial selects one global tuple (k, τ_σ, μ) shared across all hidden layers. The maps C_l are solved separately at each layer, but their hyperparameters are not tuned independently by layer. The search space is $k \in \{3, 5, \ldots, 41\}$, $\tau_\sigma \in \{0.1, 1, 2, 5, 10\}$, $\mu \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. The spectral rank is fixed to $q_l^X = \min\{d_l^X, 50\}$.

Candidate configurations are ranked by midpoint accuracy on the held-out validation pair for the classification experiments and by midpoint validation loss barrier for the asymmetric-network experiments. After selection, the best tuple is frozen and applied unchanged to all eight reported model pairs. We use the same held-out pair, validation split, and freezing principle for every tuned baseline, including CCA. Note that spectral re-basin evaluates more candidate configurations than the four CCA ridge values.

Activation descriptors, pairwise descriptor distances, and nearest-neighbour rankings are computed once and cached across trials. For a fixed candidate, the layerwise spectral maps are obtained by direct convex least-squares solves. A 30-trial study requires approximately 0.5–1.5 hours for most settings and at most approximately 2 hours for the widest VGG setting. Applying a frozen configuration to a new CNN pair takes approximately 1–3 minutes. A separate 30-trial sweep is not performed for each of the eight reported pairs. The sensitivity analysis shows that ten trials recover most of the 30-trial performance and that gains beyond twenty trials are marginal.

J.5 Baselines

Naïve averaging. For identical architectures, naïve averaging directly averages the two endpoint parameter vectors $\boldsymbol{\theta}_{\text{mid}} = \frac{1}{2}\boldsymbol{\theta}^A + \frac{1}{2}\boldsymbol{\theta}^B$. For mismatched-width experiments, the smaller model is first embedded into the larger architecture using the function-preserving zero-padding. We then average the padded smaller model with the larger model. This baseline is marked with *.

Weight matching. Weight matching refers to the weight-based permutation matching in Ainsworth et al. (2023) [5]. For each hidden layer or channel group, it solves a linear assignment problem to find a permutation that maximises weight similarity between the two endpoint networks. The permutation is then propagated through adjacent layers so that the transformed model remains functionally equivalent in the same-width case. For cross-width experiments, weight matching is applied after zero-padding the smaller model to the larger width, and is marked with *. We do not include weight matching for asymmetric networks because the $\boldsymbol{W}$-Asym construction explicitly removes the hidden-unit or channel permutation symmetries exploited by this baseline.

Activation matching. Activation matching also solves a linear assignment problem, but the matching score is computed from activation correlations instead of raw weights. Activations are collected on the same alignment data used by all activation-based methods. In cross-width experiments, activation matching is applied after function-preserving zero-padding and is marked with *.

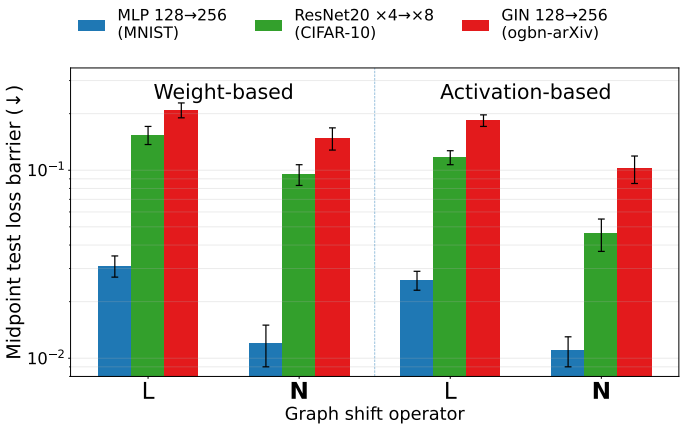


Figure 7: Ablation on the choice of the graph shift operator for matching asymmetric networks of different widths. We compare the combinatorial graph Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ with the normalised graph Laplacian $\boldsymbol{N} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$. The midpoint test loss barrier (↓) is reported. The normalised Laplacian consistently yields lower barriers, indicating that degree normalisation improves spectral alignment across neuron graphs with different widths and heterogeneous local densities.

OT fusion. OT fusion computes a soft transport plan between neurons or channels using activation-based costs and uses the transport plan to fuse the networks. Unlike the zero-padded baselines, OT fusion can be applied directly to different widths. However, the transported network produced by OT fusion is generally not a function-preserving reparameterisation of the source model, because a soft transport can mix units rather than merely relabeling or padding them. We still include OT fusion because it is a natural non-bijective alignment baseline.

CCA merge. CCA merge aligns networks by finding linear combinations of activations that maximise canonical correlations. In the identical-width setting, we use regularised CCA as in Horoi et al. (2024) [40]. The regularisation parameter is selected on a validation pair and then fixed for all reported pairs. In the mismatched-width setting, we evaluate two variants. The first, CCA merge*, applies zero-padding to match widths before applying the same CCA merge pipeline. The second, CCA merge†, applies CCA directly across different widths by adding a ridge term to the covariance matrices. This regularisation stabilises the CCA solve when the covariance matrices have different ranks or are ill-conditioned. We test $\varepsilon \in \{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ and report the best performance.

K Additional experiments

K.1 Ablation studies

K.1.1 Graph shift operators

In the main experiments, we use the normalised graph Laplacian $\boldsymbol{N} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$ as the graph shift operator, where $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ is the combinatorial graph Laplacian. We ablate this choice by replacing $\boldsymbol{N}$ with $\boldsymbol{L}$ in the asymmetric cross-width setting. Fig. 7 reports the midpoint test loss barrier for both weight-based and activation-based spectral re-basin.

The normalised Laplacian consistently outperforms the combinatorial Laplacian across all datasets, architectures, and descriptor choices. The same trend is observed for the weight-based variant. This behaviour is expected in our setting as the neuron graphs can have highly non-uniform degree distributions. Neurons may form dense clusters when several units implement similar or redundant features, while specialised neurons may be more isolated. The combinatorial Laplacian $\boldsymbol{L}$ reflects both the geometry of the graph and the absolute node degrees. As a result, its spectrum can be strongly affected by differences in local density, width, activation scale, or the sampled symmetry-breaking masks. Since our functional alignment penalty compares graph frequencies through $\|\boldsymbol{\Lambda}^B\boldsymbol{C} - \boldsymbol{C}\boldsymbol{\Lambda}^A\|_F^2$, such degree-induced spectral shifts can make otherwise compatible modes appear mismatched.

The normalised Laplacian mitigates this issue by measuring variation relative to the local connectivity of each node. Thus, it reduces the influence of degree heterogeneity and makes the low-frequency eigenspaces more comparable across graphs of different widths. Moreover, for undirected graphs with nonnegative weights, the spectrum of $\boldsymbol{N}$ is bounded in $[0, 2]$ [84, 85], which improves the conditioning of the spectral alignment objective.

K.1.2 Alternative descriptors

Spectral re-basin depends on the choice of layerwise descriptors used to construct the neuron graph and the feature-alignment term. In the main experiments, we use either weight descriptors or activation descriptors. We additionally investigated two alternative choices: Fisher-based descriptors and heat-kernel signatures (HKS). Fisher descriptors compare neurons by their local sensitivity to the training objective, estimated through the empirical Fisher information. HKS descriptors compare nodes by their multi-scale diffusion geometry on the neuron graph. Given a graph shift operator $\boldsymbol{\Delta}_l = \boldsymbol{\Phi}_l \boldsymbol{\Lambda}_l \boldsymbol{\Phi}_l^\top$, the HKS of node i at diffusion time t is given by

$$\mathrm{HKS}_l(i; t) = \sum_k \exp(-t\lambda_{l,k}) \boldsymbol{\Phi}_{l,ik}^2, \tag{78}$$

and we use the values over multiple diffusion times as a node descriptor.

Table 1: Ablation on alternative descriptors for spectral re-basin. We evaluate VGG11 $\times 8$ on CIFAR-10 in the identical-architecture, different-seed setting and report midpoint test accuracy (%) as mean $\pm$ standard deviation. Each row specifies the descriptor used to construct the neuron graph and the descriptor used in the feature-alignment term. Fisher- and HKS-based variants perform substantially worse than the default weight- and activation-based descriptors, suggesting that descriptors tied directly to the network computation provide more reliable correspondences for re-basing.

	VGG11$\times$8
Fisher graph + Fisher feature	69.31$\pm$2.49%
Fisher graph + HKS feature	59.24$\pm$1.64%
Weight graph + Fisher feature	54.28$\pm$2.44%
Weight graph + HKS feature	61.45$\pm$2.04%
Activation graph + Fisher feature	62.33$\pm$1.40%
Activation graph + HKS feature	61.54$\pm$2.21%
Weight graph + Weight feature	86.20$\pm$1.86%
Activation graph + Activation feature	87.25$\pm$1.39%

We tested variants in which the neuron graph is built either from weights or from Fisher descriptors, and the functional map feature term is driven by weights, Fisher descriptors, or HKS descriptors. Table 1 reports the midpoint test accuracy for VGG11 $\times 8$ on CIFAR-10 under identical-architecture matching with different random seeds, comparing Fisher- and HKS-based graph/feature descriptors against the default weight- and activation-based spectral re-basin variants. These alternatives did not improve over the weight- and activation-based choices used in the main experiments. Fisher-based variants retained a noticeable interpolation barrier, both when Fisher descriptors were used to construct the graph and when they were used only as the node descriptors for feature alignment. HKS-based variants were also less reliable and were sensitive when computed on graphs induced by Fisher descriptors. By contrast, weight and activation descriptors produced substantially flatter connectivity profiles in the same setting.

This negative result is informative. In spectral re-basin, the functional-alignment term constrains the map $\boldsymbol{C}_l$ to couple spectrally compatible graph modes, but it does not by itself determine which modes should correspond semantically across networks. That information is supplied by the feature-alignment descriptor. A useful descriptor should therefore satisfy two properties: it should be discriminative for the computational role of a neuron, and it should be stable across independently trained models. Weight descriptors satisfy this approximately because they directly describe the layerwise linear maps, while activation descriptors are even more directly tied to the realised function because they compare neuron responses on data.

Fisher descriptors are less suited to this role. The empirical Fisher measures local loss sensitivity around a particular endpoint, rather than the representation implemented by the neuron. It can vary

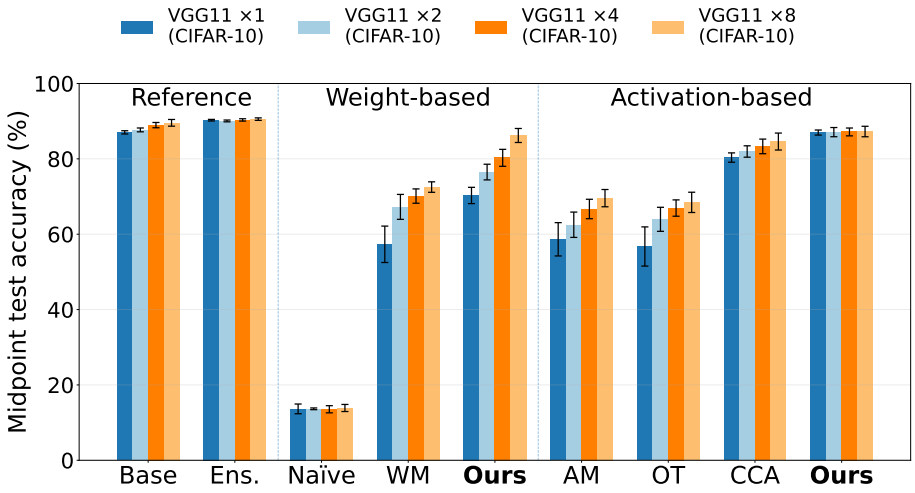


Figure 8: Identical-architecture matching across width expansions for VGG11 on CIFAR-10. For each width multiplier, we merge pairs of VGG11 models with the same architecture and same width, but trained from different random seeds.

with the data subset, the scale of the loss, the confidence of the trained model, and the coordinate parametrisation of the network. For example, in ReLU networks, function-preserving rescalings of incoming and outgoing weights can change parameter-wise Fisher magnitudes without changing the realised function. Thus, two neurons implementing similar features may have different Fisher profiles, while neurons implementing different features may have similar loss sensitivities. This makes Fisher information a weak descriptor for cross-network neuron correspondence.

HKS has a different limitation. It is an intrinsic descriptor of the graph geometry: it describes how heat diffuses around a node, rather than what feature or activation that node represents. This is useful in shape analysis, where intrinsic geometry is often the object being matched, but neuron graphs contain repeated local structures, duplicated or redundant units, and heterogeneous local densities. As a result, many neurons can have similar HKS values even when their computational roles differ. HKS can therefore encourage matching nodes with similar graph positions rather than matching neurons with compatible functions. Moreover, because HKS is computed from the graph shift operator, it inherits any noise or instability in the graph construction; this issue is amplified when the underlying graph is built from noisy Fisher descriptors.

For these reasons, we use weights and activations as the default descriptors in the main experiments. The Fisher and HKS ablations suggest that spectral re-basin benefits most from descriptors that remain close to the actual network computation.

K.2 Matching on identical architectures with different width expansions

We further study how matching performance changes as the width of the architecture increases. In this experiment, the two endpoint models in each pair have the same architecture and the same width multiplier, but are trained from different random seeds. Thus, unlike the cross-width setting in §5.2, no width mismatch is present within a pair. This setting isolates the effect of overparameterisation on re-basing performance.

We evaluate VGG11 on CIFAR-10 and ResNet20 on CIFAR-100 with width multipliers $\times 1$, $\times 2$, $\times 4$, and $\times 8$. For each width, we report the test accuracy at the midpoint of the interpolated parameters, $\frac{1}{2}\boldsymbol{\theta}^{A\to B} + \frac{1}{2}\boldsymbol{\theta}^{B}$, averaged over independently trained model pairs. The average accuracy of the endpoint models and their ensemble are shown as references.

Fig. 8 reports the results for VGG11 on CIFAR-10. Naïve weight interpolation fails across all widths, showing that independently trained models remain in different parameter basins even when they share the same architecture. Permutation-based weight matching improves the midpoint performance, and its accuracy increases with width. Our weight-based spectral re-basin consistently outperforms

weight matching at every width and benefits strongly from width expansion. Among activation-based methods, CCA merge is a strong baseline, but the activation-based spectral re-basin achieves the best performance across all widths. It remains close to the average base-model accuracy even at standard width and is stable across width multipliers.

Fig. 9 shows the corresponding results for ResNet20 on CIFAR-100. This setting is more challenging: at width $\times 1$, all matching methods exhibit a substantial gap from the base models. However, performance improves markedly as the width increases. The activation-based spectral re-basin is again the strongest method at every width. These results indicate that width expansion makes independently trained models easier to connect, but also that spectral re-basin provides consistent gains beyond standard permutation, OT, and CCA-based alignment methods.

These results show that spectral re-basin is effective across a range of width expansions. The activation-based variant is the most robust, while the weight-based variant consistently improves over permutation-based weight matching. This supports the view that graph-spectral correspondences capture a broader class of neuron-level equivalences than one-to-one relabelings, especially in overparameterised networks.

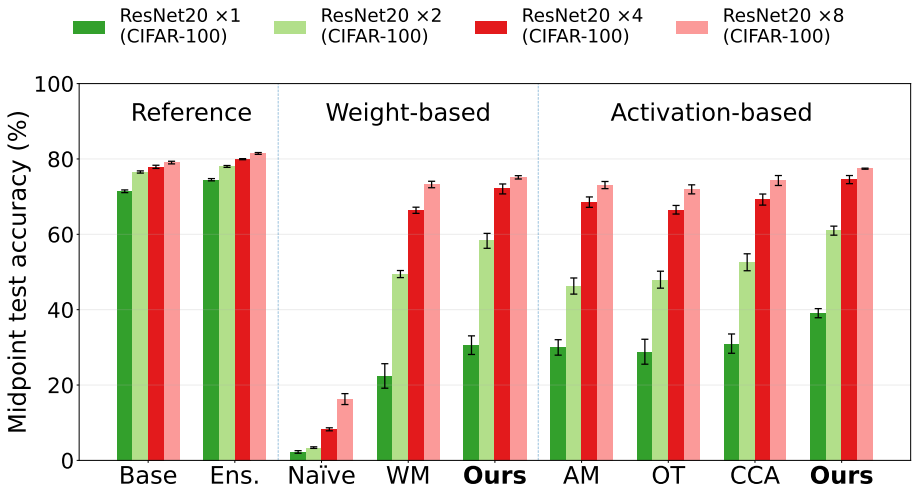


Figure 9: Identical-architecture matching across width expansions for ResNet20 on CIFAR-100. For each width multiplier, we merge pairs of ResNet20 models with the same architecture and same width, but trained from different random seeds.

K.3 End-point preserving after spectral re-basin

A desirable matching method should not only improve the midpoint model, but also preserve the performance of the transported endpoint. This is important for non-permutation-based methods, since a low interpolation barrier or strong midpoint performance can be misleading if the transformation substantially damages the original model. Hence, we evaluate the accuracy of the transported endpoint $\boldsymbol{\theta}^{A\to B}$ before interpolation.

Fig. 10 reports this diagnostic for cross-width matching of ResNet20 $\times 4 \to \times 8$ on CIFAR-100. As expected, the zero-padding and permutation-based methods preserve the endpoint accuracy of $\boldsymbol{\theta}^{A}$, since they correspond to function-preserving transformations. However, their midpoint performance remains substantially below the base-model accuracies, especially for naïve merging and activation matching. By contrast, spectral re-basin approximately preserves the endpoint while also improving the midpoint. This suggests that the learned graph-spectral maps do not merely improve interpolation by sacrificing endpoint quality, rather, they produce a high-performing model expressed in the coordinate system of the target network. In contrast, OT fusion and CCA-based methods are far from endpoint-preserving in this setting. Although OT fusion obtains a reasonable midpoint accuracy, the transformed endpoint drops. CCA merge degrades the endpoint even more substantially. These results indicate that matching dimensionality or aligning subspaces is not sufficient to preserve the realised

function under width mismatch. Therefore, spectral re-basin provides a more reliable trade-off: it keeps the transported endpoint accurate while substantially reducing the interpolation barrier.

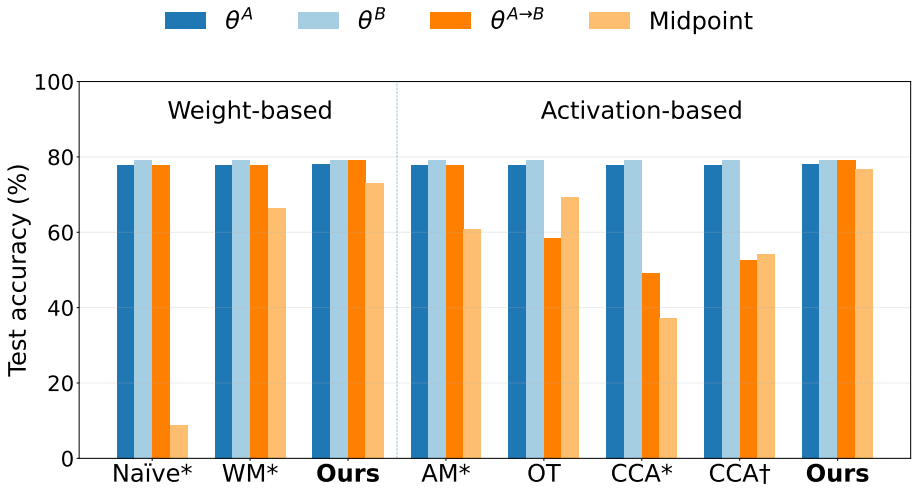


Figure 10: Endpoint preservation after re-basing for cross-width matching of ResNet20 $\times 4 \to \times 8$ on CIFAR-100. We report test accuracy (%) of the original endpoints $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^B$, the transported endpoint $\boldsymbol{\theta}^{A\to B}$, and the midpoint model $\frac{1}{2}\boldsymbol{\theta}^{A\to B} + \frac{1}{2}\boldsymbol{\theta}^B$. Methods marked with * use zero-padding to match parameter dimensions, while methods marked with † use regularised covariance matrices for CCA-based alignment.

K.4 Effect of alignment direction

Our cross-width experiments mainly consider the direction from a smaller model to a larger model. This direction is natural because the smaller network can be embedded into the larger parameter space, for example by zero-padding hidden channels. However, spectral re-basin learns rectangular correspondences and is therefore directional: mapping from a narrow model to a wide model is an expansion, whereas mapping from a wide model to a narrow model is a compression. Hence, we evaluate the reverse direction, where the larger model is rebased into the width of the smaller model.

This setting is more challenging as there is no trivial function-preserving analogue of padding when compressing a larger model into a smaller one. For this reason, we do not include naïve padding, weight matching, or activation matching in the wide-to-narrow direction, since these methods rely on matched dimensionality or permutation-based correspondences. We compare against OT fusion and CCA merge, which can potentially produce non-square alignments. For CCA merge in the wide-to-narrow direction, we truncate the canonical directions to match the target width.

Fig. 11 reports midpoint test accuracy for ResNet20 on CIFAR-100 under both directions, $\times 4 \to \times 8$ and $\times 8 \to \times 4$. The reverse direction is harder for most methods. OT fusion drops substantially, indicating that its transport map is less effective when the source representation must be compressed. CCA merge improves in the reverse direction compared with the narrow-to-wide case, but still remains well below the spectral re-basin variants.

Spectral re-basin is considerably more stable across directions. Thus, although compression into a narrower representation is more restrictive, the learned graph-spectral maps still recover strong midpoint performance. This supports the directional interpretation of our method: rectangular spectral maps can handle both expansion and compression, but the latter may incur additional information loss through the projection and pseudoinverse terms discussed in Prop. 5.

K.5 Matched-rank ablation of the graph-spectral contribution

We isolate the graph-spectral contribution of the activation-based method in three regimes: same-width VGG11 $\times 8$ on CIFAR-10, cross-width ResNet20 $\times 4 \to \times 8$ on CIFAR-100, and asymmetric cross-width ResNet20 $\times 4 \to \times 8$ on CIFAR-10. We compare the full method with *(i)* the same graph

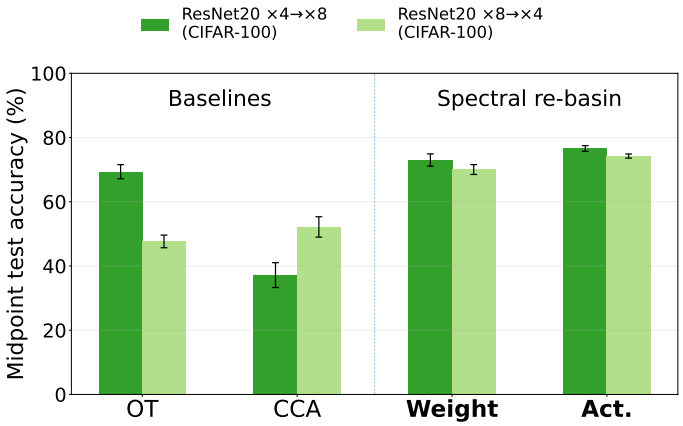


Figure 11: Effect of alignment direction for cross-width matching of ResNet20 on CIFAR-100. We report midpoint test accuracy (%) as mean ± standard deviation for matching from a narrower model to a wider model, $\times 4 \to \times 8$, and from a wider model to a narrower model, $\times 8 \to \times 4$. Permutation-based and padding-based baselines are omitted in the wide-to-narrow direction because there is no direct function-preserving compression analogue. For CCA merge in the wide-to-narrow direction, canonical directions are truncated to match the target width.

bases and $\mu = 0$, *(ii)* PCA/SVD bases, *(iii)* random orthonormal bases, *(iv)* coordinate-space reduced-rank regression, and *(v)* unrestricted coordinate-space ridge regression. All low-rank alternatives use rank 50 whereas the unrestricted ridge baseline has greater capacity.

Three comparisons separate the effects. First, comparing the full method with the same graph basis at $\mu = 0$ isolates the graph-intertwining regulariser. We see that the larger gains in cross-width and symmetry-broken settings indicate that operator compatibility is useful for non-bijective correspondences. Second, the graph-basis $\mu = 0$ variant consistently outperforms the graph-free matched-rank controls, supporting the graph-spectral basis as a useful structural low-rank parameterisation. Third,the unrestricted coordinate ridge performs worse despite having more degrees of freedom than the rank-50 methods, so the gains can not be explained by fitting a dense linear map.

Across all configurations, $\mu \in \{0.1, 1\}$ is selected in 93% of cases, $\mu \in \{10^{-2}, 10^{-3}\}$ in 6%, and $\mu = 0$ in 1%. Because the feature and functional losses have layer-dependent scales, the raw magnitude of μ should not be interpreted as an effect size.

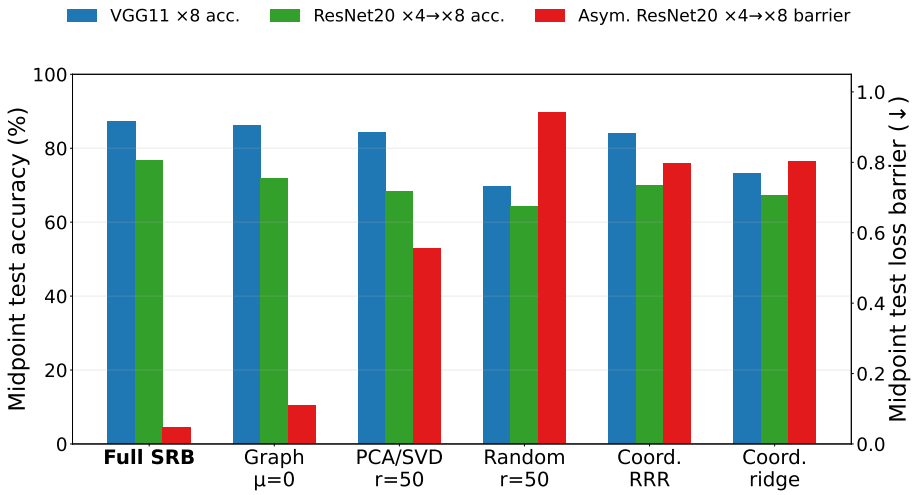


Figure 12: Matched-rank ablation of the graph-spectral parameterisation and intertwining regulariser.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We introduced *spectral re-basin*, an operator-theoretic framework for studying linear mode connectivity. We represent each layer as a graph of hidden units, and study the geometry induced by the corresponding graph shift operator. This construction leads to graph functional symmetries within a single model and intertwining correspondences between different models. Our analysis shows that the spectral alignment residual controls the discrepancy between the rebased and target weights, and further yields an upper bound on the linear interpolation barrier. The main contributions are summarized in §1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the work in §6, where we outline the constraints of the proposed method.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Right after each stated theoretical result, we indicate that the proof and the theoretical analysis can be found in §E.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: §J provides detailed descriptions of the proposed method. The method can be re-implemented, and the results are reproducible.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide detailed descriptions of our method's implementation and hyperparameters in §J.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide detailed information on experimental setup, architecture construction, and baseline comparison in §J.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In §5, we report performance (%) as mean $\pm$ standard deviation for identical architecture matching and matching on mismatched-width architectures. For matching mismatched-width asymmetric networks, we report the loss barrier as mean $\pm$ standard deviation.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Detailed descriptions of the type of computer resources and time of execution are provided in §J-K.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We respect the NeurIPS code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: This paper presents work whose goal is to advance the field of linear mode connectivity and model matching. The paper has no foreseeable societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide credit to the original papers of all external data, code, and models used in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets introduced in the paper are well documented. Detailed descriptions of our method's implementation are provided in §J.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing nor research with human subjects was involved in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No crowdsourcing nor research with human subjects was involved in this work.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only for editing purposes, such as for grammar, spelling, and word choice.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.