Deep Networks as Paths on the Manifold of Neural Representations

Richard D. Lange [1] **Devin Kwok** [2] **Jordan Matelsky** [*1] **Xinyue Wang** [*1] **David Rolnick** [2] **Konrad P. Kording** [1]

Abstract

Deep neural networks implement a sequence of layer-by-layer operations that are each relatively easy to understand, but the resulting overall computation is generally difficult to understand. An intuitive hypothesis is that the role of each layer is to reformat information to reduce the "distance" to the desired outputs. With this spatial analogy, the layer-wise computation implemented by a deep neural network can be viewed as a path along a high-dimensional manifold of neural representations. With this framework, each hidden layer transforms its inputs by taking a step of a particular size and direction along the manifold, ideally moving towards the desired network outputs. We formalize this intuitive idea by leveraging recent advances in *metric* representational similarity. We extend existing representational distance methods by defining and characterizing the *manifold* that neural representations live on, allowing us to calculate quantities like the shortest path or tangent direction separating representations between hidden layers of a network or across different networks. We then demonstrate these tools by visualizing and comparing the paths taken by a collection of trained neural networks with a variety of architectures, finding systematic relationships between model depth and width, and properties of their paths.

1. Introduction

A core design principle of modern neural networks is that they progressively transform inputs through a series of layers until the information is in a format that is immediately usable for some task (Rumelhart et al., 1988; LeCun et al., 2015). This idea of composing simple operations to gradually construct more complicated functions is both central to artificial neural networks and how neuroscientists conceptualize various functions in the brain (Kriegeskorte, 2015; Richards et al., 2019; Barrett et al., 2019).

Our work is motivated by a spatial analogy for such information-processing: we imagine that outputs are "far" from inputs if the mapping between them is complex, or "close" if it is simple. In this spatial analogy, one layer of a neural network contributes a single step, and the composition of many steps transports representations along a path towards the desired target representation. This spatial analogy enables us to translate abstract deep learning concepts into intuitions. Formalizing this intuition requires a method to quantify if any two representations are "close" (similar) or "far" (dissimilar). This is the purview of representational similarity tools, which were developed to compare neural representations across disparate systems such as fMRI scans of human brains and hidden activity in a deep neural network (Kriegeskorte, 2009; Kornblith et al., 2019). Recent work introduced *metrics* for representational dissimilarity (Williams et al., 2021; Shahbazi et al., 2021), which is an important step towards the kind of spatial reasoning about neural representations that we are interested in.

Embedding hidden-layers on a smooth manifold endows every operation in the network with a sense of both distance and direction. If a layer simply scales or rotates its inputs, the length of the step it takes is zero. But, if a layer meaningfully transforms its inputs, we can quantify both *how much* and *in what direction* it was transformed. Further, the manifold defines a theoretical shortest path or **geodesic** from any layer of the network to any other layer, including the target labels, and we can compare this to the path actually taken by the model. Results of these sorts of analyses will in general depend strongly on one's choice of representational dissimilarity metric and properties of manifold that it implies, and in principle there are infinitely many metrics one could choose from. Here, we build on previously proposed dissimilarity metrics to showcase the methodology. Developing new representational dissimilarity metrics with their geometric properties in mind will be an interesting avenue for future work.

The main contributions of this paper are (1) We propose

[*]Equal contribution [1]Department of Neurobiology, University of Pennsylvania, Philadelphia, USA [2]Mila Québec AI Institute, McGill Unviersity, Montréal, Canada. Correspondence to: Richard D. Lange <rdlange@seas.upenn.edu>.

Proceedings of the 2^{nd} Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023. *Copyright 2023 by the author(s).*

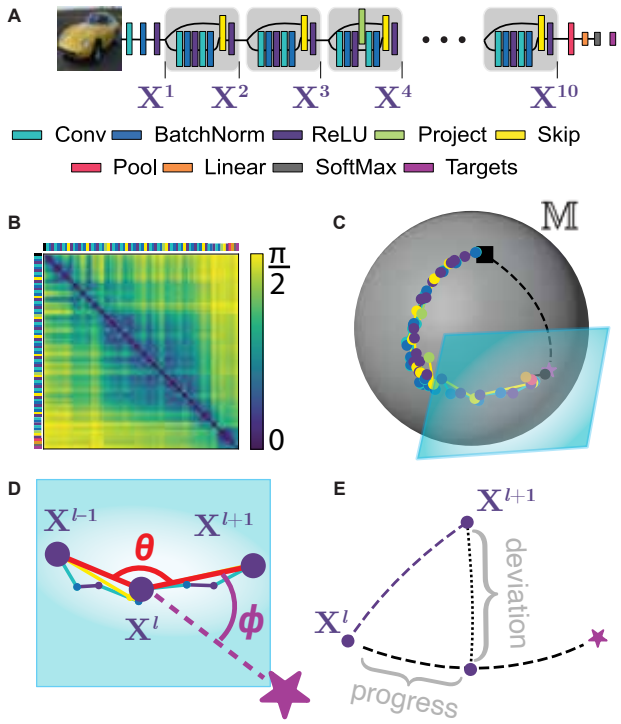

Figure 1. Introduction using ResNet-20 model trained on CIFAR-10, evaluated using Angular CKA (section 2.4). **A)** Schematic of model architecture with color-coded layers. Gray boxes correspond to residual blocks. **B)** Pairwise distance between layers shows that adjacent layers are "nearby" while inputs and outputs are "far apart" on the manifold. **C)** Low-dimensional visualization of the network's path on the 2-sphere. We used spherical multi-dimensional scaling (MDS) to embed distances in B on a 20D hypersphere, followed by spherical PCA to reduce to 2D. We additionally calculated embedding positions for input images (black square), target class labels (purple star), and fifteen points calculated from the geodesic between input and labels (black dashed line). **D)** Visualization of the tangent space of the manifold, with which we can compare *directions* from one layer's representation to another. For residual networks, we treat each residual block as a single "step" (red lines). We refer to θ as **interior angle** of the path at layer l, and ϕ angle as the **target angle** comparing the direction from layer l to $l+1$ to the direction pointing towards the targets. **E)** Visualization of projecting a point onto the geodesic spanning two other points, decomposing neural network operations into **progress** in the direction of the targets and **deviation** in orthogonal directions.

and quantitatively evaluate a spatial "path" analogy for deep neural networks; (2) We extend recently proposed representational distance metrics by analyzing geometric properties of the manifold implied by these metrics; (3) We provide these tools in an open-source toolbox;[1] (4) We empirically compare paths taken by different models, different datasets, and changes over training.

[1] https://github.com/wrongu/repsim

2. Preliminaries

2.1. Related Work

One motivation for thinking of neural networks as paths is that it provides a compelling analogy for the way that complex functions (deep networks) can be composed out of simple parts (layers). Indeed, it is well known that both deeper (Poole et al., 2016; Raghu et al., 2017; Rolnick & Tegmark, 2017) and wider (Hornik et al., 1989) neural network architectures can express a larger class of functions than their shallower or narrower counterparts. However, much less is known about how implementing a *particular* complex function constrains the role of individual layers and intermediate representations in the intervening layers between input and output. Our work is in line with other recent efforts to characterize the features learned in hidden layers as smoothly varying between inputs and outputs (Chan et al., 2020; Yang et al., 2022; He & Su, 2022; Yu et al., 2020; 2023). Our work is a first step, so to speak, towards formalizing this notion of composing simple functions in *geometric* terms.

There is a rich literature applying geometric concepts like distance between representations to formalize notions of "similarity" in neuroscience and psychology (Edelman, 1998; Jäkel et al., 2008; Rodriguez & Granger, 2017; Hénaff et al., 2019; Kriegeskorte & Wei, 2021). However, there is a crucial difference between measuring similarity or distance between points in a given space, and *measuring distances between representational spaces themselves*. The former includes questions like, "how far apart are the activation vectors for two inputs in a given layer?" The latter, which is the subject of this paper, asks instead, "how far apart are two layers' representations, considering all inputs?"

Our work is most closely related to, and draws much inspiration from recent advances in representational similarity analysis (RSA). In particular, Kornblith et al. (2019) showed that a kernel method for testing statistical dependence, known as CKA (Gretton et al., 2005; Cortes et al., 2012), is closely related to classic RSA (Kriegeskorte, 2009). In follow-up work, Nguyen et al. (2021) used CKA to make layer-by-layer comparisons between wide and deep networks. Independently, both Williams et al. (2021) and Shahbazi et al. (2021) developed methods to compute metrics between neural representations. Shahbazi et al. (2021) proposed using the so-called **Affine-Invariant Riemannian** metric on the space of symmetric positive-definite matrices (abbreviated AIR-SPD below) (Pennec, 2006; 2019). Williams et al. (2021) derived a metric variation of CKA which we call **Angular CKA**, as well as a family of **Generalized Shape**

Metrics. We extend this prior work by computing not just pairwise distances, but by also introducing a suite of tools for analyzing the geometry of the manifold implied by each of these distance metrics. Finally, whereas Williams et al. (2021) and Shahbazi et al. (2021) compare representations *across models*, we compare representations *within a single model* to study the transformation of information from inputs to outputs through hidden layers.

2.2. Distance metrics between neural representations

Representational dissimilarity is quantified by some function $\mathrm{d}(\mathbf{X}, \mathbf{Y}) : \mathbb{X} \times \mathbb{X} \to \mathbb{R}^+$ that takes in two matrices of neural data and outputs a nonnegative value for their dissimilarity. Here, $\mathbb{X} = \bigcup_{n=1,2,3,\ldots} \mathbb{R}^{m \times n}$ is the space of all $m \times n$ matrices for all n. The matrices $\mathbf{X}$ and $\mathbf{Y}$ could be, for instance, the values of two hidden layers in a network with n_x and n_y units, respectively, in response to m inputs. We flatten convolutional layers, in which case n is the product of height, width, and feature dimensions. We encode target labels as one-hot vectors, i.e. targets are encoded in a $m \times 10$ matrix for the CIFAR-10 dataset (Krizhevsky, 2009). Note that $\mathbf{X}$ and $\mathbf{Y}$ may be different layers of the same model or layers from different models, as long as they are evaluated on the same inputs.

There is considerable leeway in choosing the representational dissimilarity function $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ in terms of what features of $\mathbf{X}$ and $\mathbf{Y}$ it is sensitive or invariant to. Previous work has argued that any sensible dissimilarity function should be nonnegative, so $\mathrm{d}(\mathbf{X}, \mathbf{Y}) \geq 0$, and should return zero between any **equivalent** representations, so $\mathrm{d}(\mathbf{X}, \mathbf{Y}) = 0 \Leftrightarrow \mathbf{X} \sim \mathbf{Y}$, where $\mathbf{X} \sim \mathbf{Y}$ means that $\mathbf{X}$ and $\mathbf{Y}$ are in the same equivalence class. For example, we may wish to design the function d so that $\mathrm{d}(\mathbf{X}, \mathbf{Y}) = 0$ if $\mathbf{Y}$ is a shifted copy of $\mathbf{X}$, or if it is a non-degenerate scaling, rotation, or affine transformation of $\mathbf{X}$ (Kornblith et al., 2019; Williams et al., 2021; Shahbazi et al., 2021). A second desirable property is that d is **symmetric**, so $\mathrm{d}(\mathbf{X}, \mathbf{Y}) = \mathrm{d}(\mathbf{Y}, \mathbf{X})$. A third is that d satisfies the **triangle inequality**, or $\mathrm{d}(\mathbf{X}, \mathbf{Y}) \leq \mathrm{d}(\mathbf{X}, \mathbf{Z}) + \mathrm{d}(\mathbf{Z}, \mathbf{Y})$. As argued by Williams et al. (2021), a representational dissimilarity function that fails to satisfy the triangle inequality can lead to errant results when, for instance, clustering or embedding representations based on their pairwise dissimilarity. A function that satisfies all of the above properties – equivalence, symmetry, and the triangle inequality – qualifies as a **metric**[2] on the space of neural representations $\mathbb{X}$ (Burago et al., 2001).

We are interested in using metrics between neural representations to explore how representations evolve as they are transformed through the hidden layers of deep networks. Not all metrics are sufficient for the kind of spatial reasoning – that is, not all metrics can be interpreted as *distances*. For example, consider the trivial metric

$$\mathrm{d}(\mathbf{X}, \mathbf{Y}) = \begin{cases} 0 & \text{if } \mathbf{X} \sim \mathbf{Y} \\ 1 & \text{otherwise} \end{cases}.$$

This is a valid metric according to the equivalence, symmetry, and triangle inequality criteria, but it is useless for characterizing distances. To be interpretable as a measure of distance, $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ must satisfy an intuitive fourth condition called **rectifiability**: the distance between any two points must be realizable as the (infimum of the) sum of distances of segments along a path between them (Burago et al., 2001). While not all metrics are rectifiable (such as the trivial metric above), this condition is unsurprisingly met by many sensible metrics, including those already developed by Williams et al. (2021) and Shahbazi et al. (2021). In fact, all metrics considered in this paper are **Riemannian metrics**, which not only implies that they are rectifiable, but further has the property that points (i.e. neural representations in each hidden layer) live on a smooth manifold (Burago et al., 2001). The rectifiability property allows us to smoothly *interpolate* neural representations along a geodesic as well as compute projections and angles, and Riemannian structure allows us to meaningfully compare the *direction* of steps taken by different layers using the tangent space of the manifold.

All prior distance metrics use a two-stage approach to defining $\mathrm{d}(\mathbf{X}, \mathbf{Y})$: first, $\mathbf{X}$ and $\mathbf{Y}$ are mapped to a common manifold $\mathbb{M}$ through an **embedding function** $\mathrm{f} : \mathbb{X} \to \mathbb{M}$, then distance is computed using a distance metric defined on $\mathbb{M}$. More precisely,

$$\mathrm{d}(\mathbf{X}, \mathbf{Y}) \equiv \mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}), \tag{1}$$

where $\tilde{\mathbf{X}} = \mathrm{f}(\mathbf{X})$ is the result of mapping $\mathbf{X}$ from $\mathbb{X}$ to $\mathbb{M}$. In all cases we consider here, $\mathbb{M}$ is a Riemannian manifold. In practice, equivalence relations can be built into this two-stage approach in either stage: $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ can be made invariant to changes in the scale of $\mathbf{X}$, for instance, either by imposing a canonical scale in the embedding stage f, or by preserving scale in f but using a scale-invariant metric $\mathrm{d}_{\mathbb{M}}$. Appendix A provides details for all metrics we consider here, the parameters that govern their behavior, what transformations on $\mathbb{X}$ they are invariant to, etc.

In addition to these desiderata on how the metric space is defined, a practical concern is computing $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ accurately and efficiency while selecting finitely many inputs on which to evaluate the representations $\mathbf{X}$ and $\mathbf{Y}$. For instance, in our experiments below, we evaluate hidden representations $\mathbf{X}$ using $m = 1000$ randomly selected images from the test

[2] More precisely, it is a "metric" on the quotient space of the equivalence class $\mathbf{X} \sim \mathbf{Y}$, or a "pseudometric" on $\mathbb{X}$, but we suppress this distinction throughout to avoid excess verbiage; see (Williams et al., 2021).

set. When m is small (and 1000 may be small relative to the size of the hidden layer), $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ – and other geometric quantities of interest – may be biased or high variance. In order to work with these representational manifolds in practice, we need the geometry to be asymptotically consistent, so the limit

$$\lim_{m\to\infty} \mathrm{d}(\mathbf{X}_m, \mathbf{Y}_m)$$

exists, where $\mathbf{X}_m$ and $\mathbf{Y}_m$ each have m rows. In the $m \to \infty$ limit, distances between neural representations become a kind of distance between probability distributions, but in different spaces since in general $n_x \neq n_y$. We would like to remove as much bias and variance as possible, so that d_m is a good estimator of d_∞ using a feasible value for m.

2.3. Geodesics, projections, and angles between neural representations

Having embedded a matrix of neural data $\mathbf{X}$ as a point $\tilde{\mathbf{X}} = f(\mathbf{X})$ on a Riemannian manifold $\mathbb{M}$, new kinds of analyses become available going beyond mere pairwise representational distances, such as **geodesics**, **logarithmic** and **exponential maps**, **projections**, and **angles** between neural representations. In this section, we will give a brief and intuitive introduction to each of these concepts as they apply to neural representations. For a formal introduction to manifolds, metrics, and Riemanninan geometry, we refer the reader to (Burago et al., 2001; do Carmo, 1992).

Imagine the manifold $\mathbb{M}$ as a sphere, where the embedded neural representations $\tilde{\mathbf{X}}$ – as well as the embedded inputs (images) and targets (labels) – are points on the sphere (Figure 1b). A **geodesic** $\gamma(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, t)$ is a smooth curve between $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ in $\mathbb{M}$, parameterized by t, that traces out the shortest path between $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ (Burago et al., 2001). On the sphere, geodesics are arc segments of great circles. In practice, we can assume there is a unique shortest path between any two embeddings of neural data. We use geodesics to quantify the "efficiency" of the transformations implemented by the layers of a deep network by comparing to the hypothetical shortest path towards the targets (but note that the geodesic depends on the metric).

The **tangent space** of a manifold can be thought of as a flat hyperplane that is tangent to the manifold at a point. The tangent space provides a flat coordinate system where we can reason about vectors and directions along the surface of the manifold. The **logarithmic map** $\mathbf{V} = \log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}})$ computes a vector $\mathbf{V}$ in the tangent space of the base point $\tilde{\mathbf{X}}$ that points in the direction of $\tilde{\mathbf{Y}}$, and the **exponential map** $\tilde{\mathbf{Y}} = \exp_{\tilde{\mathbf{X}}}(\mathbf{V})$ is its inverse. We use the tangent space around embedded neural representations to reason about *directions* towards or away from other representations. For instance, we use it to compute the **angle** $\theta(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, \tilde{\mathbf{Z}})$ between triplets of embedded representations, defined as

$$\cos\Big(\theta(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, \tilde{\mathbf{Z}})\Big) = \frac{\langle \log_{\tilde{\mathbf{Y}}}(\tilde{\mathbf{X}}), \log_{\tilde{\mathbf{Y}}}(\tilde{\mathbf{Z}})\rangle_{\tilde{\mathbf{Y}}}}{\sqrt{\langle \log_{\tilde{\mathbf{Y}}}(\tilde{\mathbf{X}}), \log_{\tilde{\mathbf{Y}}}(\tilde{\mathbf{X}})\rangle_{\tilde{\mathbf{Y}}} \langle \log_{\tilde{\mathbf{Y}}}(\tilde{\mathbf{Z}}), \log_{\tilde{\mathbf{Y}}}(\tilde{\mathbf{Z}})\rangle_{\tilde{\mathbf{Y}}}}}. \quad (2)$$

Here, the inner product $\langle \mathbf{V}, \mathbf{U}\rangle_{\tilde{\mathbf{Y}}}$ is in the tangent space at $\tilde{\mathbf{Y}}$ and depends on the local metric tensor (do Carmo, 1992).

Finally, we are interested in **projecting** points onto the geodesic spanned by another two points. Projecting allows us to decompose tangent vectors (e.g. steps taken by neural network layers) into a component that is pointing in a particular direction (e.g. towards the target outputs) and an orthogonal component (Figure 1E). The projection of $\tilde{\mathbf{X}}$ onto the geodesic spanning $\tilde{\mathbf{Y}}$ and $\tilde{\mathbf{Z}}$ can be found by minimizing the distance from $\tilde{\mathbf{X}}$ to $\exp_{\tilde{\mathbf{Y}}}(t \log_{\tilde{\mathbf{Y}}}(\tilde{\mathbf{Z}}))$ with respect to t.[3] Details on how we solve for t numerically can be found in Appendix C.

2.4. Angular CKA

Angular CKA was introduced by Williams et al. (2021) (eq (60)) as a metric variation on CKA, which is a well-established method for (non-metric) representational similarity analysis (Kornblith et al., 2019). Angular CKA is defined as the arccosine of CKA, which is itself derived from the Hilbert-Schmidt independence criterion (HSIC) (Gretton et al., 2005; Cortes et al., 2012; Kornblith et al., 2019). Because HSIC and CKA measure statistical dependence, distance measured by Angular CKA is small when the rows of $\mathbf{X}$ and $\mathbf{Y}$ are strongly statistically dependent, and large when they are independent. As originally argued by Kornblith et al., CKA has desirable properties as a measure of neural similarity because it is invariant to shifts, scaling, and rotations of the original data $\mathbf{X}$, but is not invariant to arbitrary affine transforms. While Angular CKA was originally introduced simply as a method for computing a metric between neural representations, here we exploit the fact that Angular CKA is the arc-length on a hypersphere to compute additional geometric properties of the space.

Formally, Angular CKA is defined as

$$\mathrm{d}(\mathbf{X}, \mathbf{Y}) = \arccos\left(\frac{\mathrm{HSIC}(\mathbf{X}, \mathbf{Y})}{\sqrt{\mathrm{HSIC}(\mathbf{X}, \mathbf{X})\mathrm{HSIC}(\mathbf{Y}, \mathbf{Y})}}\right) \quad (3)$$

The bias and variance of a finite-m estimator of Angular CKA depends primarily on the bias and variance of the estimator of HSIC. Gretton et al. originally proposed the estimator

$$\mathrm{HSIC}(\mathbf{X}, \mathbf{Y}) \propto \langle \boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H}, \boldsymbol{H}\boldsymbol{G}_{\mathbf{Y}}\boldsymbol{H}\rangle_F \,, \quad (4)$$

[3] When $0 \leq t \leq 1$, this is a projection onto the geodesic spanning from $\tilde{\mathbf{Y}}$ to $\tilde{\mathbf{Z}}$, but this expression in terms of logarithmic and exponential maps extends to the $t < 0$ or $t > 1$ cases.

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner-product, $\boldsymbol{G}_{\mathbf{X}}$ is the $m \times m$ Gram matrix between rows of $\mathbf{X}$, and $\boldsymbol{H} \equiv \mathbb{I} - m^{-1}\mathbf{1}\mathbf{1}^\top$ is the centering matrix. This Gram matrix may optionally be computed using a kernel, but following (Kornblith et al., 2019), we set $\boldsymbol{G}_{\mathbf{X}} = \mathbf{X}\mathbf{X}^\top$. This expression of HSIC as an inner-product leads to a straightforward characterization of Angular CKA as the arc-length a hypersphere.

Unfortunately, (4) has substantial bias of $\mathcal{O}(m^{-1})$ that requires infeasibly large values of m to overcome (Figure E.1). This bias has been addressed in different ways by different authors. Song et al. introduced an unbiased estimator of HSIC, but unlike (4) their estimator cannot be written as an inner-product in a vector space, and thus we cannot use it our geometric calculations. Nguyen et al. further reduce variance by estimating the numerator and denominator of (3) separately using the unbiased estimator of Song et al. in small batches, but this approach similarly will not work for our geometric calculations.

To address these issues, we derived a new estimator of HSIC that simultaneously (i) has low bias at $\mathcal{O}(m^{-2})$ and (ii) can be written as an inner product in a vector space, thus retaining the desired geometric properties. Our estimator is

$$\text{HSIC}(\mathbf{X}, \mathbf{Y}) = \frac{2}{m(m-3)} \langle \text{tril}(\boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H}), \text{tril}(\boldsymbol{H}\boldsymbol{G}_{\mathbf{Y}}\boldsymbol{H}) \rangle_F \,, \tag{5}$$

where tril is a function that extracts just the lower-triangular part of its matrix argument, excluding the diagonal. Proof of the $\mathcal{O}(m^{-2})$ bias of (5) can be found in Appendix B. In sum, we compute finite-sample estimate of Angular CKA as

$$\text{d}(\mathbf{X}, \mathbf{Y}) = \arccos\left(\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}} \right\rangle_F\right) , \tag{6}$$

where the embedding function $\tilde{\mathbf{X}} = f(\mathbf{X})$ is given by

$$\tilde{\mathbf{X}} \equiv \frac{\text{tril}(\boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H})}{||\text{tril}(\boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H})||_F} \,. \tag{7}$$

Further details on the geometry of Angular CKA can be found in Appendix A, including expressions for the logarithmic map, exponential map, and geodesics, all of which follow from the well-known geometry of hyperspheres.

2.5. Shape Metrics

Williams et al. (2021) additionally proposed using a generalization of Procrustes distance and Kendall's shape space as a dissimilarity metric for neural representations. Shape-space and Procrustes distance are a well-studied case of a Riemannian manifold between point clouds (Nava-Yazdani et al., 2020). The basic idea of shape metrics is as follows: $m \times n$ matrices of neural data are first transformed into a common $m \times p$ space and interpreted as a "shape" consisting of m distinct $p-$dimensional keypoints. Then, shift-, scale- and rotation-invariance are added explicitly by centering, scaling, and rotationally aligning pairs of shapes so that they maximally align with each other before computing distances between keypoints.

We embed each $m \times n$ matrix of neural data $\mathbf{X}$ (where n may vary) into the shared ambient space of $m \times p$ matrices (where p is fixed) by first subtracting the mean of each column, then projecting neural data $\mathbf{X}$ onto its top $p = 100$ principal components, or padding with zeros, depending on whether $n > p$ or $n < p$. Matrices are then scaled to unit Frobenius norm. These operations embed neural data $\mathbf{X}$ into points $\tilde{\mathbf{X}}$ in the pre-shape space (Nava-Yazdani et al., 2020). Distances in shape space are then given by

$$\text{d}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = \min_{\boldsymbol{Q} \in SO(p)} \arccos\left(\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}\boldsymbol{Q} \right\rangle_F\right) \tag{8}$$

The minimization over $\boldsymbol{Q} \in SO(p)$ rotates $\tilde{\mathbf{Y}}$ to maximally align with $\tilde{\mathbf{X}}$. Formally, this means that shape distance is distance in the quotient space of pre-shape space after applying the *equivalence relation* $\tilde{\mathbf{X}} \sim \tilde{\mathbf{Y}} \Leftrightarrow \exists \boldsymbol{Q} \in SO(p) : \tilde{\mathbf{X}} = \tilde{\mathbf{Y}}\boldsymbol{Q}$.

Our presentation differs slightly from that of Williams et al. (2021). There, the authors include a "partial whitening" stage as part of the embedding, and consider more restricted classes of rotations than $SO(p)$ that eliminate permutations across spatial dimensions when comparing convolutional layers. Here, we opted not to perform whitening because full whitening makes the metric invariant to affine transformations in the original n-dimensional neural space, and others have argued that neural dissimilarity should be sensitive to second moments (Kornblith et al., 2019). Further, partial whitening distorts the pre-shape space, thorough treatment of which we leave for future work (see Appendix A and Table A.1). We additionally opted not to use restricted rotations because we are interested in distances between both convolutional and non-convolutional layers. Note that restricting large convolutional layers to $p = 100$ dimensions has a similar effect of restricting the analysis only to relevant features of the data, and reduces bias for finite m, though we note that this technique incurs moderate bias when computing distances to one-hot labels (Figure E.2).

Solving for $\boldsymbol{Q}$ that minimizes (8) – or equivalently maximally aligning the individual keypoints of $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}\boldsymbol{Q}$ – is known as the orthogonal Procrustes problem, and its solution is given by

$$\boldsymbol{Q}^* = \boldsymbol{V}^\top \boldsymbol{U}^\top$$

where $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is the singular value decomposition of $\tilde{\mathbf{X}}^\top \tilde{\mathbf{Y}}$.

2.6. Affine Invariant metric on SPD matrices

Shahbazi et al. proposed using a well-studied metric between symmetric positive definite (SPD) matrices as a dissimilarity metric for neural data. This metric is known as the Affine-Invariant Riemannian metric on the manifold of SPD matrices (Pennec, 2006; 2019), or AIR-SPD.

The AIR-SPD metric requires positive definite matrices; all rank-deficient symmetric matrices are effectively at a distance of infinity. This is addressed by the original authors by using the $\mathbf{X}\mathbf{X}^\top$ Gram matrix when $m < n$ or $\mathbf{X}^\top\mathbf{X}$ otherwise, but the latter solution requires $n_\mathbf{X} = n_\mathbf{Y}$ when comparing different layers. While rank deficiency in the case of $n < m$ could in principle be addressed using a kernel to compute the Gram matrix, this runs up against numerical stability issues in practice. Worse, it does not fix rank deficiency due to repeated rows, and by design, the target outputs of a classification task contain repeated rows when inputs are from the same class. This makes the AIR-SPD metric ill-suited for the problem of quantifying distance or directions towards target outputs. Finally, we note that others have argued that affine invariance is a bug rather than a feature when comparing neural data (Kornblith et al., 2019).

For these reasons, we leave further investigation of the AIR-SPD metric to future work.

3. Experiments

We are interested in characterizing geometric properties of the "paths" that standard neural networks take on their way from inputs (e.g. images) to outputs (e.g. labels). These paths are high-dimensional objects on curved manifolds, but we can get interpretable glimpses of their overall structure by plotting summary statistics of the path's structure and how they depend on the model architecture, training, the dataset the model was trained on.

For our main analyses, we trained a variety of standard "wide" and "deep" ResNet models (He et al., 2016; Nguyen et al., 2021), as well as models from the VGG family (Simonyan & Zisserman, 2015), on the CIFAR-10 dataset (Krizhevsky, 2009). For all training, we used standard procedures developed for repeatable experimentation on neural networks,[4] and each model was trained using 5 different seeds.

First, we confirmed that both Angular CKA and the Angular Shape Metric are suitable for our original goal of quantifying the gradual progress that neural networks make towards targets through their layers. We began by calculating pairwise distances on the manifold between all layers in a trained model as well as distances from each layer to the embedding of the targets. We then used multidimensional scaling (MDS) to embed each layer as a point on a 20-dimensional[5] hypersphere, then used principal component analysis (PCA) to summarize the main axes of variation of the network's path (Williams et al., 2021). The top two PCs for an example model are shown in Figure 1B-C. We indeed find that for both metrics, the path taken by the network does indeed gradually approach the targets, and that the first two principal components tend to cover (i) the direction spanning the input-output geodesic, and (ii) a primary "out and back" axis of deviation of the model's path away from the geodesic.

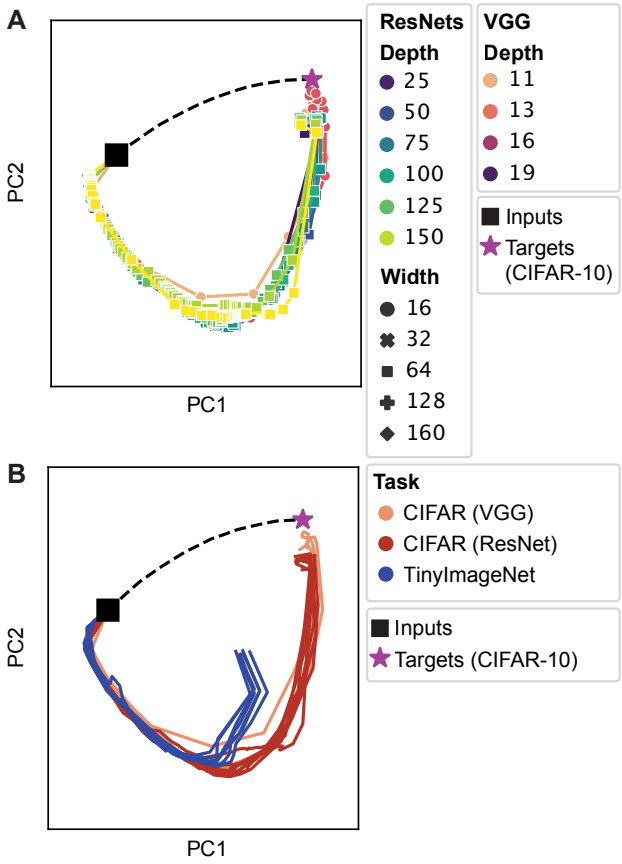

Figure 2. Visualizations of representation paths in low dimension. **A)** Several models of varying architecture, depth, and width were trained on the same task of CIFAR-10 image classification. All models take nearly the same path, departing from the inputs-targets geodesic, regardless of architecture. **B)** Several models from above are compared against identical models applied on the TinyImageNet task instead. These paths are visually distinct, and terminate at a visibly different destination than the CIFAR-10 task target.

[4] https://github.com/facebookresearch/open_lth

[5] chosen as the smallest dimensionality where the stress of the embedding improves on the stress of a 2D embedding by at least 99%.

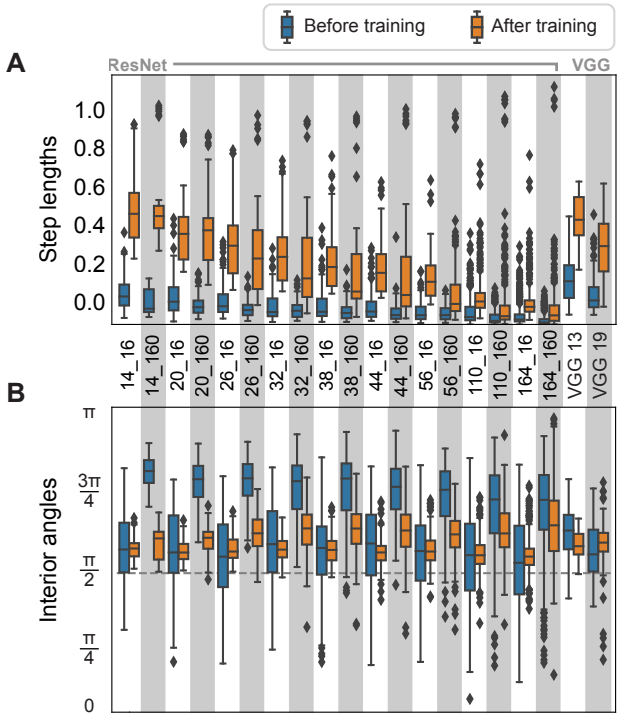

Figure 3. Step lengths and interior angles θ for a variety of architectures, compared before and after training. Resnet models are labeled "depth_width". **A)** We compute the length of the step taken by each residual bock of ResNets (or each convolutional block of VGG), $\mathrm{d}(\tilde{\mathbf{X}}^l, \tilde{\mathbf{X}}^{l+1})$. Three notable trends emerge: first, step sizes are generally longer after training than before. Second, this effect is greater on average for shallower models. Third, although the average step size is small for very deep models, there are large outlying step lengths that are of roughly the same size for all models. **B)** We computed the interior angle (see Figure 1D) for all triplets of layers $(\tilde{\mathbf{X}}^{l-1}, \tilde{\mathbf{X}}^l, \tilde{\mathbf{X}}^{l+1})$. It is expected that angles are more variable before training when step sizes are on average smaller. The primary effect of training is to concentrate these angles at values just above $\pi/2$, and in fact we see a significant *sharpening* of interior angles for wide and shallow models (Figure **??**).

We next asked to what extent paths taken by different model architectures are similar or different. For this, we again computed pairwise distances between layers within each model as well as across different models, and followed the same MDS and PCA procedure above to jointly embed them into the same space. Figure 2A shows the first two principal components after jointly embedding ResNet models of various widths and depths, as well as models from the VGG family, all trained on CIFAR-10 (see Figure E.4 for additional PCs and E.6 for the Angular Shape Metric). These analyses are initially suggestive that, when trained on the same dataset, the path taken by models of different architectures are highly similar (Li et al., 2015; Nguyen et al., 2021). However, could it be that these low-dimensional visualizations show intrinsic properties of neural networks unrelated to the particular task? To answer this question, we trained another set of models with similar architecture on the related but different dataset known as TinyImagenet.[6] We then co-embedded paths taken by models trained on different datasets in the same space by passing images from the CIFAR-10 test set through all models including those trained on the other task. The first two PCs using Angular CKA are shown in Figure 2B (see Figure E.5 for additional PCs and E.7 for the Angular Shape Metric). Even in the first 2 PCs, we see a "fork" in the paths between models trained on different image classification datasets. This suggests that there is *nontrivial* similarity between the different architectures shown in Figure 2A. Further, it is consistent with the idea that two image-classification datasets like CIFAR-10 and TinyImagenet are similar enough that they share common processing in the first few layers (Yosinski et al., 2014). Ultimately, these low-dimensional path visualizations confirm our original spatial intuitions, namely that representational distance metrics can be used to characterize the gradual layer-wise transformation of information through the layers of a deep network and to distinguish internal computations of different networks.

Next, we turned to the question of how some simple summary statistics of model's paths – including the size of the steps taken by each layer as well as **internal angles** (Figure 1D) – are affected by the model architecture and how they change with training. Before running these analyses, we hypothesized that (i) step sizes would be smaller for deeper models, as they are able to break a problem into a larger number of steps, and (ii) that interior angles would begin close to orthogonal and become more straight through training.

Our first hypothesis about step lengths was borne out by the data (Figures 3A, **??**A). Importantly, this increase in step length is not merely a function of an increase in the magnitude of the neural network weights, as we are using metrics that are invariant to overall scale. An increase in step length here means that representations at the output of each residual block become *more dissimilar* from the input to the block in terms of second- or higher-moments.

Surprisingly, our second hypothesis about interior angles straightening with training was not borne out by the data (Figure 3B, **??**B). Surprisingly we see the *opposite* trend in shallow and wide networks, with interior angles becoming on average *less* straight with training (Figure **??**). Analysis of **target angles** can be found in Figure E.10).

Finally, we leveraged **projections** of points onto geodesics (Figure 1E) to decompose the step taken by each residual block (or convolutional block for VGG models) into a

[6] available at http://cs231n.stanford.edu/tiny-imagenet-200.zip

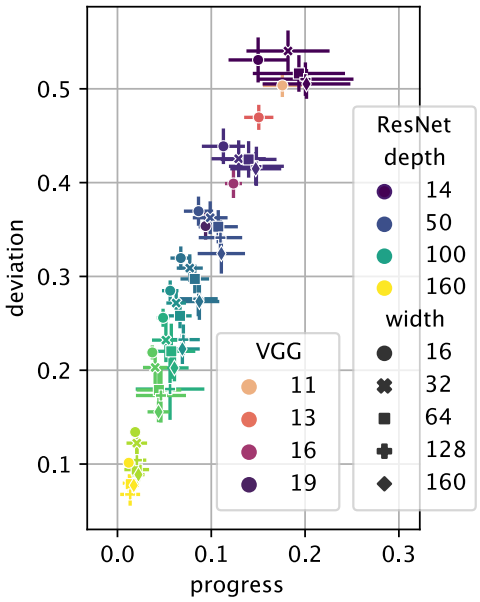

Figure 4. Summarizing average "progress" towards targets and "deviation" in orthogonal directions by every step of the models. Error bars show standard error of the mean. See Figure E.3 for individual datapoints and the Angular Shape Metric.

"progress" component that points along the geodesic from $\hat{\mathbf{X}}^l$ to the targets, and a "deviation" component in orthogonal directions (Figure 4; see also Figure E.3)). Two main trends are apparent: first, increasing depth reduces overall step sizes close to linearly, reducing both progress and deviation both for ResNet and for VGG models. Second, increasing ResNet width trends down and right, increasing width and reducing deviation. This suggests that wider models have the capacity to take more direct paths towards targets, perhaps because additional width reduces constraints on the set of possible steps that each layer can take.

4. Discussion

We investigated two families of representational distance metrics — Angular CKA, and the Angular Shape Metric (Williams et al., 2021). Surprisingly, we found that trained networks take rather circuitous paths according to both metric families, deviating far from the shortest paths from inputs to targets. There are three potential explanations for this. First, networks may be taking short paths according to some metric other than those we investigated here, implying that the metrics we used may not reflect the most natural notion of distance between representations. Second, neural networks may fail to take efficient paths. The distance metrics we consider are all differentiable, and so an interesting question for future work is whether networks can be regularized to take shorter paths, and whether such regularization will improve or reduce their performance or generalization ability. Third, it could be that networks take the shortest and most direct path which is possible under some architectural constraints, which may prevent the hidden layers of the network from moving directly along the geodesic. This explanation must be at least partially true, since the dimensionality of the representation space $\mathbb{M}$ generally exceeds the number of parameters in each layer/block of the network, and our analysis of deviation versus progress in Figure 4 is consistent with the idea that wider networks – with more degrees of freedom per layer – deviate less.

We were also surprised to discover that according to all metrics we investigated, network paths tended to be *jagged* in the sense that interior angles are predominantly of 90° angle turns, and learning had little effect. Although it is well known that random directions in high-dimensional spaces such as the representation space $\mathbb{M}$ tend to be nearly always orthogonal, this does not explain why we found that path angles are straighter at initialization and become more acute during training. This is puzzling because we expected gradient descent to encourage all layers to point in the same direction towards the targets. This result also contrasts with recent work by Chan et al. (2020) that suggests that a sequence of residual blocks can be interpreted as a sequence of small gradient steps optimizing a *rate reduction* objective; in their interpretation, all layers ought to be moving in the same general direction. However, this is not what we find in our models trained by backpropagation and where interior angles are quantified using Angular CKA or the Angular Shape Metric. Ultimately, ours is an empirical finding which suggests that future theoretical work is needed to interpret the *direction* of steps taken in representation space in the context of a given representational distance metric, and to understand which directions are realizable by a given network architecture.

As described in the introduction, we are motivated to develop a general spatial analogy for neural information-processing, where complex transformations of representations require functions that cover more "distance" than simple ones. To this end, a measure of representational distance ought to reflect the *function complexity* of transforming $\mathbf{X}$ into $\mathbf{Y}$. In this work, we chose to extend and compare existing representational distance metrics in order to build directly on previous work, but the metrics we evaluated here may not be interpretable as measures of function complexity. In fact, such a $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ characterizing function complexity of transforming $\mathbf{X}$ into $\mathbf{Y}$ will likely not be a standard distance metric at all. For intance, the complexity of a function and its inverse are in general not equal, and so it may be desirable to have $\mathrm{d}(\mathbf{X}, \mathbf{Y}) < \mathrm{d}(\mathbf{Y}, \mathbf{X})$ if the transformation from $\mathbf{X}$ to $\mathbf{Y}$ can be implemented by a simpler function than the inverse. An exciting avenue for future work is thus to derive new measures of representational dissimilarity specifically for characterizing information-processing in spatial

terms, as was our original goal.

The analogy of neural networks as paths in a representation space brings together ideas about representational similarity and the expressivity of deep networks, marrying these techniques with intuitive and mathematically rigorous geometric concepts. Our work takes a first step in exploring the possibilities of this new geometric framework, and we anticipate that it will spark new insights about model design, model training, and model comparison.

References

Barrett, D. G., Morcos, A. S., and Macke, J. H. Analyzing biological and artificial neural networks: challenges with opportunities for synergy? *Current Opinion in Neurobiology*, 55:55–64, 2019. ISSN 18736882. doi: 10.1016/j.conb.2019.01.007. URL https://doi.org/10.1016/j.conb.2019.01.007.

Burago, D., Burago, Y., and Ivanov, S. *A Course in Metric Geometry*. American Mathematical Society, 2001.

Chan, K. H. R., Yu, Y., You, C., Qi, H., Wright, J., and Ma, Y. Deep Networks from the Principle of Rate Reduction, October 2020. URL http://arxiv.org/abs/2010.14765. arXiv:2010.14765 [cs, math, stat].

Cortes, C., Mohri, M., and Rostamizadeh, A. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13:795–828, 2012. ISSN 15324435.

Diedrichsen, J. and Kriegeskorte, N. Representational models: A common framework for understanding encoding, pattern-component, and representational-similarity analysis. *PLoS Computational Biology*, 13(4):1–33, 2017. URL https://journals.plos.org/ploscompbiol/article/file?id=10.1371/journal.pcbi.1005508&type=printable.

do Carmo, M. *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 1992. ISBN 9783764334901. URL https://books.google.com/books?id=uXJQQgAACAAJ.

Edelman, S. Representation is representation of similarities. *Behavioral and Brain Sciences*, 21(4):449–498, 1998. ISSN 0140525X. doi: 10.1017/S0140525X98001253.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. Measuring statistical dependence with Hilbert-Schmidt norms. In Jain, S., Simon, H. U., and Tomita, E. (eds.), *Lecture Notes in Artificial Intelligence*, volume 3734, pp. 63–77. Springer-Verlag, Berlin, 2005. ISBN 354029242X. doi: 10.1007/11564089_7.

He, H. and Su, W. J. A Law of Data Separation in Deep Learning, October 2022. URL http://arxiv.org/abs/2210.17020. arXiv:2210.17020 [cs, math, stat].

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. *CVPR*, 2016.

Hénaff, O. J., Goris, R. L., and Simoncelli, E. P. Perceptual straightening of natural videos. *Nature Neuroscience*, 22(6):984–991, 2019. ISSN 15461726. doi: 10.1038/s41593-019-0377-4. URL http://dx.doi.org/10.1038/s41593-019-0377-4.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 08936080. doi: 10.1016/0893-6080(89)90020-8.

Jäkel, F., Schölkopf, B., and Wichmann, F. A. Similarity, kernels, and the triangle inequality. *Journal of Mathematical Psychology*, 52(5):297–303, 2008. ISSN 00222496. doi: 10.1016/j.jmp.2008.03.001. URL http://dx.doi.org/10.1016/j.jmp.2008.03.001.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of Neural Network Representations Revisited. *ICML*, 36, 2019. URL http://arxiv.org/abs/1905.00414.

Kriegeskorte, N. Relating population-code representations between man, monkey, and computational models. *Frontiers in Neuroscience*, 3(3):363–373, 2009. ISSN 16624548. doi: 10.3389/neuro.01.035.2009. URL http://journal.frontiersin.org/article/10.3389/neuro.01.035.2009/abstract.

Kriegeskorte, N. Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annual Review of Vision Science*, 1(1):417–446, 2015. ISSN 2374-4642. doi: 10.1101/029876.

Kriegeskorte, N. and Wei, X.-X. Neural tuning and representational geometry. *Nature Reviews Neuroscience*, 22 (11):703–718, 2021.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bengio, Y., and Hinton, G. E. Deep learning. *Nature*, 521(7553):436–444, 2015. ISSN 0028-0836. doi: 10.1038/nature14539. URL http://www.nature.com/nature/journal/v521/n7553/full/nature14539.html.

Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. Convergent learning: Do different neural networks learn the same representations? *JMLR: Workshop and Conference Proceedings*, 44:196–212, 2015. arXiv:1511.07543.

Miolane, N., Guigui, N., Brigant, A. L., Mathe, J., Hou, B., Thanwerdas, Y., Heyder, S., Peltre, O., Koep, N., Zaatiti, H., Hajri, H., Cabanes, Y., Gerald, T., Chauchat, P., Shewmake, C., Brooks, D., Kainz, B., Donnat, C., Holmes, S., and Pennec, X. Geomstats: A python package for riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9, 2020. URL http://jmlr.org/papers/v21/19-027.html.

Nava-Yazdani, E., Hege, H.-C., Sullivan, T. J., and von Tycowicz, C. Geodesic analysis in kendall's shape space with epidemiological applications. *Journal of Mathematical Imaging and Vision*, 62(4):549–559, 2020.

Nguyen, T., Raghu, M., and Kornblith, S. Do Wide and Deep Networks Learn the Same Things? Uncovering How Neural Network Representations Vary with Width and Depth. *ICLR*, pp. 1–25, 2021. URL http://arxiv.org/abs/2010.15327.

Nguyen, T., Raghu, M., and Kornblith, S. On the Origins of the Block Structure Phenomenon in Neural Network Representations, February 2022. URL http://arxiv.org/abs/2202.07184. arXiv:2202.07184 [cs].

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pennec, X. Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, 2006. ISSN 09249907. doi: 10.1007/s10851-006-6228-4.

Pennec, X. *Manifold-valued image processing with SPD matrices*. Elsevier Ltd, 2019. ISBN 9780128147269. doi: 10.1016/B978-0-12-814725-2.00010-8. URL https://doi.org/10.1016/B978-0-12-814725-2.00010-8.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. *arXiv*, 2016. URL http://arxiv.org/pdf/1606.05340v2.pdf.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Jascha Soh. On the Expressive Power of Deep Fully Circulant Neural Networks. *ICML*, 70:2847–2854, 2017. URL http://arxiv.org/abs/1901.10255.

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., de Berker, A., Ganguli, S., Gillon, C. J., Hafner, D., Kepecs, A., Kriegeskorte, N., Latham, P., Lindsay, G. W., Miller, K. D., Naud, R., Pack, C. C., Poirazi, P., Roelfsema, P., Sacramento, J., Saxe, A., Scellier, B., Schapiro, A. C., Senn, W., Wayne, G., Yamins, D., Zenke, F., Zylberberg, J., Therien, D., and Kording, K. P. A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770, 2019. ISSN 15461726. doi: 10.1038/s41593-019-0520-2. URL http://dx.doi.org/10.1038/s41593-019-0520-2.

Rodriguez, A. M. and Granger, R. The differential geometry of perceptual similarity. *arXiv preprint arXiv:1708.00138*, 2017. URL http://arxiv.org/abs/1708.00138.

Rolnick, D. and Tegmark, M. The power of deeper networks for expressing natural functions. *ICLR*, pp. 1–14, 2017. URL http://arxiv.org/abs/1705.05502.

Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. *Parallel distributed processing*, volume 1. IEEE New York, 1988.

Shahbazi, M., Shirali, A., Aghajan, H., and Nili, H. Using distance on the Riemannian manifold to compare representations in brain and in models. *NeuroImage*, 239 (June):118271, 2021. ISSN 10959572. doi: 10.1016/j.neuroimage.2021.118271. URL https://doi.org/10.1016/j.neuroimage.2021.118271.

Shoemake, K. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pp. 245–254, 1985.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Skopek, O., Ganea, O.-E., and Bécigneul, G. Mixed-curvature variational autoencoders. *arXiv preprint arXiv:1911.08411*, 2019.

Song, L., Smola, A., Gretton, A., Borgwardt, K. M., and Bedo, J. Supervised feature selection via dependence estimation. *ACM International Conference Proceeding Series*, 227:823–830, 2007. doi: 10.1145/1273496.1273600. arXiv: 0704.2668.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M.,

Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Williams, A. H., Kunz, E., Kornblith, S., and Linderman, S. Generalized shape metrics on neural representations. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 4738–4750. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/252a3dbaeb32e7690242ad3b556e626b-Paper.pdf.

Yang, A. X., Robeyns, M., Milsom, E., Schoots, N., and Aitchison, L. A theory of representation learning in deep neural networks gives a deep generalisation of kernel methods, July 2022. URL http://arxiv.org/abs/2108.13097. arXiv:2108.13097 [cs, stat].

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, pp. 3320–3328, 2014. URL http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks. Publisher: Curran Associates, Inc.

Yu, Y., Chan, K. H. R., You, C., Song, C., and Ma, Y. Learning Diverse and Discriminative Representations via the Principle of Maximal Coding Rate Reduction, June 2020. URL http://arxiv.org/abs/2006.08558. arXiv:2006.08558 [cs, math, stat].

Yu, Y., Buchanan, S., Pai, D., Chu, T., Wu, Z., Tong, S., Haeffele, B. D., and Ma, Y. White-Box Transformers via Sparse Rate Reduction, June 2023.

Metric Family	Options	Invariances			Dimensionality d	Isometric to
		scale	rotation	affine		
Angular CKA	linear kernel	yes	yes	no	$\frac{m(m-1)}{2} - 1$	$\mathbb{S}^d$
Angular CKA	nonlinear kernel	no†	no†	no	$\frac{m(m-1)}{2} - 1$	$\mathbb{S}^d$
Angular Shape	$p < \infty$, $\alpha = 1$	yes	yes	no	$(m-1)p - \frac{p(p-1)}{2} - 1$	$\mathbb{S}^d$
Angular Shape	$p < \infty$, $0 < \alpha < 1$	yes	yes	no	$(m-1)p - \frac{p(p-1)}{2} + \alpha(p-1)$	(see note$^\circ$)
Angular Shape	$p < \infty$, $\alpha = 0$	yes	yes	yes‡	$(m-1)p - \frac{p(p+1)}{2}$	$\mathbb{S}^d$
Euclidean Shape	$p < \infty$, $\alpha = 1$	no	yes	no	$(m-1)p - \frac{p(p-1)}{2}$	$\mathbb{R}^d$
Euclidean Shape	$p < \infty$, $0 < \alpha < 1$	no	yes	no	$(m-1)p - \frac{p(p-1)}{2} + \alpha p$	(see note$^\circ$)
Euclidean Shape	$p < \infty$, $\alpha = 0$	(converges to Angular Shape when $\alpha = 0$)				
AIR-SPD	(unused in this paper; see section 2.6 for rationale)					

Table A.1. Properties of distance metrics between neural representations. All metrics are translation-invariant by construction. What each metric is invariant to and properties of their manifolds depend on various configuration options. Angular CKA, Angular Shape, and Euclidean Shape were introduced as metrics for neural data by Williams et al.. Option for Angular CKA is choice of kernel. Options for Shapes include (i) Angular or Euclidean base metric, (ii) dimensionality of neural space p, and (iii) whether to whiten the data ($\alpha = 0$) or not ($\alpha = 1$). In this work we do not consider the case of unbounded p.
† depends on the choice of kernel. Many kernels are rotation-invariant, i.e. $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x}\boldsymbol{Q}, \mathbf{y}\boldsymbol{Q})$ for $\boldsymbol{Q} \in SO(n)$, in which case the metric becomes rotation-invariant. Nonlinear kernels may also be constructed to be scale-invariant, e.g. using a squared exponential kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/\tau^2)$ with the length scale τ automatically adapted to the scale of the data $\mathbf{X}$.
‡ only if affine transform does not affect the subspace spanned by the top p principal components.
$^\circ$ The manifold $\mathbb{M}$ is a continuous deformation of $\mathbb{R}^{d_{\alpha=1}}$ (in the euclidean case) or $\mathbb{S}^{d_{\alpha=1}}$ (in the angular case) to $\mathbb{S}^{d_{\alpha=0}}$, parameterized by α. With partial whitening ($0 < \alpha < 1$), natural way to measure dimensionality is using the sum of singular values of the local metric tensor on the manifold expressed in the coordinate frame of the ambient ($\alpha = 1$ metric). This leads to a real-valued measure of dimensionality that linearly interpolates between the integer-valued dimensionalities for $\alpha = 0$ and $\alpha = 1$.

A. Detailed information on metrics

As summarized in equation (1), all distance metrics between neural repreoentations operate in two stages: first, a layer's activity $\mathbf{X} \in \mathbb{X}$ is transformed into a point in some canonical space $\mathbb{M}$ through an **embedding function** f, and second distances are measured in that shared space. In the following subsections we give details for each metric in Table A.1.

We say a metric is **scale-invariant** if $\mathrm{d}(\mathbf{X}, \alpha\mathbf{X}) = 0$ for all scalars $\alpha \neq 0$ and $\mathbf{X} \in \mathbb{X}$ (that is, $\mathbf{X}$ is a matrix of neural data in the original space, before embedding on a manifold). A metric is **shift-invariant** if $\mathrm{d}(\mathbf{X}, \mathbf{X} + \mathbf{1}\mathbf{b}^\top) = 0$ for any $n-$dimensional vector $\mathbf{b}$. A metric is **rotation-invariant** if $\mathrm{d}(\mathbf{X}, \mathbf{X}\boldsymbol{R}) = 0$ for any $n \times n$ orthonormal matrix $\boldsymbol{R}$. A metric is **affine-invariant** if $\mathrm{d}(\mathbf{X}, \mathbf{X}\boldsymbol{A}) = 0$ for any full-rank $n \times n$ matrix $\boldsymbol{A}$.

Our Python implementation of various quantities from Riemannian geometry draws much inspiration from the `geomstats` Python package (Miolane et al., 2020). Our analyses in the main paper were done using

- Angular CKA with $m = 1000$ and the linear kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$, using our estimator of HSIC in (5).
- The Angular Shape metric with $m = 1000$, $p = 100$, $\alpha = 1$.

We begin with an introduction to Angular CKA, where we also review some key terms from Riemannian geometry.

A.1. Angular CKA

Angular CKA was introduced by Williams et al. (2021) (eq (60) in their supplement). It is defined as the arccosine of centered kernel alignment (CKA), which is itself derived from the Hilbert-Schmidt independence criterion (HSIC) (Gretton et al., 2005; Cortes et al., 2012; Kornblith et al., 2019). Because HSIC and CKA measure statistical dependence, distance

measured by Angular CKA is high when the rows of $\mathbf{X}$ and $\mathbf{Y}$ are statistically independent, and low when they are highly correlated.

Angular CKA is equivalent to arc-length distance on the spherical manifold consisting of *centered* and *normalized* $m \times m$ Gram matrices. Let $\boldsymbol{G}_{\mathbf{X}}$ denote the Gram matrix of $\mathbf{X}$, i.e. $\boldsymbol{G}_{\mathbf{X}ij}$ is given by the inner-product between the ith and jth rows of $\mathbf{X}$. Optionally, this inner-product may be computed using a kernel; following previous work (Kornblith et al., 2019; Nguyen et al., 2021), results in this paper use Linear CKA, i.e. we use $\boldsymbol{G}_{\mathbf{X}} = \mathbf{X}\mathbf{X}^\top$, but our python package supports the use of other kernels. The *normalized* and *centered* Gram matrix is given by

$$\tilde{\boldsymbol{G}}_{\mathbf{X}} = \frac{\boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H}}{||\boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H}||_{\mathrm{F}}}$$

where $\boldsymbol{H} = \mathbb{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ is the $m \times m$ **centering matrix**, and $||\cdot||_{\mathrm{F}}$ is the Frobenius norm of a matrix. As stated in the main text, we address the bias of HSIC by dropping the diagonal elements of $\boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H}$ in both the numerator and denominator (or equivalently, taking the upper- or lower-triangular elements only). Thus, the embedding function we use for Angular CKA is given by

$$\tilde{\mathbf{X}} \equiv \frac{\mathrm{tril}(\boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H})}{||\mathrm{tril}(\boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H})||}\,. \qquad \text{((7) restated)}$$

The Riemannian manifold $\mathbb{M}$ for Angular CKA consists of all such centered, normalized, symmetric positive definite matrices; it is a **sphere** in sense that $\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{X}} \right\rangle_{\mathrm{F}} = 1$, where $\langle \boldsymbol{A}, \boldsymbol{B} \rangle_{\mathrm{F}} = \mathrm{Tr}(\boldsymbol{A}^\top \boldsymbol{B})$ is the Frobenius inner-product.

Distance according to Angular CKA is equal to arc length on the sphere consisting of centered and normalized Gram matrices:

$$\begin{aligned} \mathrm{d}(\mathbf{X}, \mathbf{Y}) &= \mathrm{d}_{\mathbb{M}}(\mathrm{f}(\mathbf{X}), \mathrm{f}(\mathbf{Y})) \\ &= \mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \\ &= \arccos\left(\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}} \right\rangle_{\mathrm{F}}\right)\,. \end{aligned} \qquad \text{((3) restated)}$$

Because Angular CKA is an arc length, its geodesics lie along great circles on the hypersphere. We therefore can compute points along the geodesic in closed-form using the SLERP formula (Shoemake, 1985):

$$\mathrm{geodesic}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, t) = \frac{\sin((1-t)\Omega)}{\sin(\Omega)}\tilde{\mathbf{X}} + \frac{\sin(t\Omega)}{\sin(\Omega)}\tilde{\mathbf{Y}}\,, \qquad \text{(A.1)}$$

where $t \in [0, 1]$ is the fraction of distance along the geodesic from $\mathbf{X}$ to $\mathbf{Y}$, and $\Omega = \mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$.

The **tangent space** for Angular CKA is the space of all symmetric $m \times m$ matrices, and the inner-product in the tangent space is simply the Frobenius inner-product. The **logarithmic map** computes tangent vectors from a base point that point towards another point. In the case of Angular CKA, the logarithmic map from $\tilde{\mathbf{X}}$ to $\tilde{\mathbf{Y}}$ is a tangent vector (symmetric matrix) at $\tilde{\mathbf{X}}$ given by

$$\log_{\tilde{\mathbf{X}}}\left(\tilde{\mathbf{Y}}\right) = \boldsymbol{W}\arccos\left(\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}} \right\rangle_{\mathrm{F}}\right) \qquad \text{(A.2)}$$

where $\boldsymbol{W}$ is the unit tangent vector at $\tilde{\mathbf{X}}$ pointing towards $\tilde{\mathbf{Y}}$, given by

$$\boldsymbol{W} = \frac{\tilde{\mathbf{Y}} - \tilde{\mathbf{X}}\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}} \right\rangle_{\mathrm{F}}}{||\tilde{\mathbf{Y}} - \tilde{\mathbf{X}}\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}} \right\rangle_{\mathrm{F}}||_{\mathrm{F}}}\,.$$

The **exponential map** is the inverse of the logarithmic map – it is a function that "extrapolates" a tangent vector $\boldsymbol{W}$ from a point to give another point on the manifold. In the case of Angular CKA, the exponential map is given by

$$\exp_{\tilde{\mathbf{X}}}(\boldsymbol{W}) = \cos\left(||\boldsymbol{W}||_{\mathrm{F}}\right)\tilde{\mathbf{X}} + \mathrm{sinc}\left(||\boldsymbol{W}||_{\mathrm{F}}\right)\boldsymbol{W} \qquad \text{(A.3)}$$

where $\mathrm{sinc}(x) = \frac{\sin(x)}{x}$. For proofs of the exponential and logarithmic map on hyperspheres, see Skopek et al. (2019) Theorem A.8.

For all metrics, we compute **angles** between triplets of points by computing the inner-product of their tangent vectors. In the case of Angular CKA in particular, let $\boldsymbol{W}_{\mathbf{XY}}$ denote the tangent vector pointing from $\tilde{\mathbf{X}}$ to $\tilde{\mathbf{Y}}$, i.e. the result of $\log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}})$. Then,

$$\theta(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, \tilde{\mathbf{Z}}) = \arccos\left(\frac{\langle \boldsymbol{W}_{\mathbf{YX}} \boldsymbol{W}_{\mathbf{YZ}} \rangle_{\mathrm{F}}}{||\boldsymbol{W}_{\mathbf{YX}}||_{\mathrm{F}} ||\boldsymbol{W}_{\mathbf{Y\tilde{Z}}}||_{\mathrm{F}}}\right) \tag{A.4}$$

is the angle of the $\tilde{\mathbf{X}}\tilde{\mathbf{Y}}\tilde{\mathbf{Z}}$ triangle.

A.1.1. Invariances of Angular CKA

The invariances of Angular CKA depend on the kernel used to compute the Gram matrix. In the simplest case of **Linear CKA**, the Gram matrix is simply $\boldsymbol{G}_{\mathbf{X}} = \mathbf{X}\mathbf{X}^\top$. The resulting metric is

- **shift-invariant** due to centering the Gram matrix.
- **scale-invariant** due to normalizing the Gram matrix.
- **rotation-invariant** since $(\mathbf{X}\boldsymbol{R})(\mathbf{X}\boldsymbol{R})^\top = \mathbf{X}\boldsymbol{R}\boldsymbol{R}^\top\mathbf{X}^\top = \mathbf{X}\mathbf{X}^\top$ for any orthonormal $\boldsymbol{R}$.

However, Angular CKA with is not invariant to arbitrary affine transformations – a feature it inherits from CKA and has been argued to be an important feature of CKA (Kornblith et al., 2019). Note that when using a nonlinear kernel to compute the Gram matrix, the resulting metric may lose these invariances. However, Angular CKA with a nonlinear kernel may still be shift-, scale-, and rotation-invariant if the kernel itself has those invariances. For example the squared exponential kernel

$$\boldsymbol{G}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(||\mathbf{x}_i - \mathbf{x}_j||_2^2 / \tau^2\right) \tag{A.5}$$

is naturally shift- and rotation-invariant, and it can be further made scale-invariant by setting the length scale τ automatically based on the scale of the data.

A.2. Shape Metrics

Williams et al. (2021) proposed using a generalization of Procrustes distance and Kendall's shape space to measure metric distance between neural representations. Shape-space and Procrustes distance are a well-studied case of a Riemannian manifold between point clouds (Nava-Yazdani et al., 2020). Williams et al. (2021) consider two different shape metrics – one *angular* shape metric and one *Euclidean* shape metric. The key idea behind both of these metrics is as follows: $m \times n$ matrices of neural data are first transformed into a common $m \times p$ space and interpreted as a point cloud consisting of $p-$dimensional points. Then, any two point clouds are scaled and rotated so that they maximally align with each other. The final distance is then computed as some measure of discrepancy between these maximally-aligned point clouds. The behavior of these shape metrics is tuned using two hyperparameters: the dimensionality p, and a partial whitening parameter α.

The role of the **embedding function** for shape metrics is to convert $n-$dimensional neural data into a canonical zero-mean $p-$dimensional space (i.e. $\mathbb{M} = \mathbb{R}^{m \times p}$ is the space of all $m \times p$ matrices whose column means are all zero). Williams et al. (2021) also include a partial whitening stage as part of the embedding. This space of $m \times p$ zero-mean (and sometimes scaled) matrices is called the **pre shape space** (Nava-Yazdani et al., 2020). In the case where $n < p$, this conversion from n to p dimensions is done by simply padding $\mathbf{X}$ with $p - n$ columns of all zeros. In the case where $p < n$, we reduce the dimensionality of $\mathbf{X}$ by keeping only the top p principal components. Formally, let $\bar{\mathbf{X}} = \mathbf{X} - \frac{1}{m}\sum_{i=1}^{m} \mathbf{X}_i$ be the matrix of neural data with its mean subtracted, then

$$\tilde{\mathbf{X}} = \mathrm{f}(\mathbf{X}) = \begin{cases} \text{whiten}([\bar{\mathbf{X}},\ \mathbf{0}], \alpha) & \text{if } n \leq p \\ \text{whiten}(\bar{\mathbf{X}}\boldsymbol{U}_{:p}, \alpha) & \text{if } n > p \end{cases} \tag{A.6}$$

where $\boldsymbol{U}_{:p}$ stands for the first p principal components of $\bar{\mathbf{X}}$, as unit column vectors, and $\mathbf{0}$ is a $m \times (p - n)$ matrix of all zeros. The partial whitening function begins by computing the eigen-decomposition of its input, $m^{-1}\bar{\mathbf{X}}^\top\bar{\mathbf{X}} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ (here, $\boldsymbol{V}$ is a $p \times p$ orthonormal matrix containing the top principal components of $\bar{\mathbf{X}}$, and $\boldsymbol{\Sigma}$ is a diagonal matrix of variances). Then, the partial whitening stage is

$$\text{whiten}(\bar{\mathbf{X}}, \alpha) = \bar{\mathbf{X}}\boldsymbol{V}\left(\alpha\mathbb{I}_p + (1-\alpha)\boldsymbol{\Sigma}^{-\frac{1}{2}}\right)\boldsymbol{V}^\top.$$

Note that when $\alpha = 0$, this is equivalent to ZCA whitening, and when $\alpha = 1$ it leaves $\bar{\mathbf{X}}$ unchanged. All shape metric results we report are with $p = 100$ and $\alpha = 1$. We use $\alpha = 1$ because this is most comparable to Angular CKA in terms of its invariances.

Both the angular and Euclidean shape metrics require *aligning* by rotating the embedded points by minimizing $||\tilde{\mathbf{X}} - \tilde{\mathbf{Y}}\boldsymbol{R}||_{\mathrm{F}}$ where $\boldsymbol{R}$ is a $p \times p$ orthonormal matrix. This is known as the orthogonal Procrustes problem, and its solution is given by

$$\boldsymbol{R} = \boldsymbol{V}^\top \boldsymbol{U}^\top$$

where $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \tilde{\mathbf{X}}^\top\tilde{\mathbf{Y}}$ is a singular value decomposition of $\tilde{\mathbf{X}}^\top\tilde{\mathbf{Y}}$. The **generalized** shape metrics introduced by Williams et al. (2021) include further restrictions on $\mathbf{R}$, such as considering rotations across channel but not spatial dimensions of convolutional layers, but we omit these restrictions in our work.

In the case of **angular** shape metrics, distance is defined as

$$\mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = \arccos\left(\frac{\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}\boldsymbol{R} \right\rangle_{\mathrm{F}}}{||\tilde{\mathbf{X}}||_{\mathrm{F}}||\tilde{\mathbf{Y}}||_{\mathrm{F}}}\right). \tag{A.7}$$

In the case of **Euclidean** shape metrics, distance is defined as

$$\mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = \frac{1}{m}\sum_{i=1}^{m} ||\tilde{\mathbf{X}}_i - \tilde{\mathbf{Y}}_i\boldsymbol{R}||\,. \tag{A.8}$$

We compute **geodesics** in shape space after finding $\boldsymbol{R}$ to align $\tilde{\mathbf{Y}}$ to $\tilde{\mathbf{X}}$. Then, the geodesic from $\tilde{\mathbf{X}}$ to $\tilde{\mathbf{Y}}\boldsymbol{R}$ in the **angular** case is given by the SLERP formula as in (A.1):

$$\mathrm{geodesic}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, t) = \frac{\sin((1-t)\Omega)}{\sin(\Omega)}\tilde{\mathbf{X}} + \frac{\sin(t\Omega)}{\sin(\Omega)}\tilde{\mathbf{Y}}\boldsymbol{R}, \tag{A.9}$$

where $\Omega = \mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ is the angular shape distance. Note that this means that $\mathrm{geodesic}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, 1)$ results in a point that is *equivalent* but not *identical* to $\tilde{\mathbf{Y}}$.

Tangent vectors for Euclidean shape metrics can be any $m \times p$ matrix whose column-wise mean is zero. In the case of angular shape metrics, the tangent space is further restricted to the tangent space of the hypersphere of unit-Frobenius-norm matrices (i.e. a tangent vector $\boldsymbol{W}$ at $\tilde{\mathbf{X}}$ must satisfy $\left\langle \tilde{\mathbf{X}}, \boldsymbol{W} \right\rangle_{\mathrm{F}} = 0$ in the angular case). The tangent space is further divided into so-called **horizontal** and **vertical** subspaces, where the vertical subspace captures changes to $\tilde{\mathbf{X}}$ that leave distance invariant, i.e. rotations that are removed by alignment by $\boldsymbol{R}$, and the horizontal subspace captures changes that affect the metric (Nava-Yazdani et al., 2020). The vertical component of a tangent vector $\boldsymbol{W}$ at point $\tilde{\mathbf{X}}$ is given by $\mathrm{vert}_{\tilde{\mathbf{X}}}(\boldsymbol{W}) = \tilde{\mathbf{X}}\boldsymbol{A}$, where $\boldsymbol{A} \in \mathbb{R}^{p\times p}$ is the solution to the following Sylvester equation:

$$\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}\boldsymbol{A} + \boldsymbol{A}\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}} = \boldsymbol{W}^\top\tilde{\mathbf{X}} - \tilde{\mathbf{X}}^\top\boldsymbol{W}\,.$$

Following the example of Miolane et al. (2020), we use the `solve_sylvester` function from Scipy to compute this (Virtanen et al., 2020). The horizontal component of a tangent vector is given by simply subtracting the vertical part of $\boldsymbol{W}$:

$$\mathrm{horz}_{\tilde{\mathbf{X}}}(\boldsymbol{W}) = \boldsymbol{W} - \mathrm{vert}_{\tilde{\mathbf{X}}}(\boldsymbol{W})\frac{\langle \mathrm{vert}_{\tilde{\mathbf{X}}}(\boldsymbol{W}), \boldsymbol{W}\rangle_{\mathrm{F}}}{||\mathrm{vert}_{\tilde{\mathbf{X}}}(\boldsymbol{W})||_{\mathrm{F}}}\,.$$

To compute the **angle** between any triplet of representations, we use the inner-product of tangent vectors, as in (A.4), but using only the *horizontal* part of each tangent vector. As in Angular CKA, we compute horizontal tangent vectors from $\tilde{\mathbf{X}}$ to $\tilde{\mathbf{Y}}$ using the **logarithmic map**, which in the case of shape metrics is given by

$$\text{horizontal } \log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}}) = \tilde{\mathbf{Y}}\boldsymbol{R} - \tilde{\mathbf{X}} \tag{A.10}$$

in the Euclidean case, or

$$\text{horizontal } \log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}}) = \boldsymbol{W}\arccos\left(\frac{\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}\boldsymbol{R} \right\rangle_{\mathrm{F}}}{||\tilde{\mathbf{X}}||_{\mathrm{F}}||\tilde{\mathbf{Y}}||_{\mathrm{F}}}\right) \tag{A.11}$$

where $\boldsymbol{W}$ is the unit tangent vector at $\tilde{\mathbf{X}}$ pointing towards $\tilde{\mathbf{Y}}$, given by

$$\boldsymbol{W} = \frac{\tilde{\mathbf{Y}}\boldsymbol{R} - \tilde{\mathbf{X}}\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}\boldsymbol{R} \right\rangle_{\mathrm{F}}}{||\tilde{\mathbf{Y}}\boldsymbol{R} - \tilde{\mathbf{X}}\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}\boldsymbol{R} \right\rangle_{\mathrm{F}}||_{\mathrm{F}}}.$$

(Nava-Yazdani et al., 2020). As in (A.7) and (A.8), $\boldsymbol{R}$ is the rotation matrix that optimally aligns $\tilde{\mathbf{Y}}$ to $\tilde{\mathbf{X}}$.

A.2.1. Invariances of Shape Metrics

The invariances of the shape metrics depend on a variety of hyperparameter settings.

- All shape metrics are **shift-invariant** because the embedding function $\tilde{\mathbf{X}} = \mathrm{f}(\mathbf{X})$ subtracts the mean.
- All shape metrics are **rotation-invariant** because of the Procrustes alignment procedure, and because rotation does not affect the principal component projection nor the zero-padding step of (A.6).
- The angular shape metric is **scale-invariant** because (A.7) divides by the norms of $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$.
- The Euclidean shape metric is not scale-invariant in general, but it is for the special case of $\alpha = 0$, since scale is removed by whitening. In fact, the Euclidean and Angular shape metrics coincide with each other entirely when $\alpha = 0$.
- Neither angular nor Euclidean shape metrics is **affine-invariant** in general, but both can become affine-invariant for the special case of $\alpha = 0$, since full-rank affine transforms are removed by whitening as long as $n \leq p$. However, in the $n > p$ case, an affine transformation may amplify or suppress the principal components of the data, and as a result it can affect the embedding stage (A.6). Thus, these metrics are only truly affine-invariant even with $\alpha = 0$ *within the top-p principal components' subspace*.

A.2.2. Additional challenges of partial whitening

Note that (partial) whitening with $\alpha < 1$ adds additional constraints on the space. When using (A.7) the Frobenius norm of $\tilde{\mathbf{X}}$ is constrained to be 1, and the structure of the space is spherical. This constraint on the Frobenius norm of $\tilde{\mathbf{X}}$ is equivalent to saying that the sum of the singular values of $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ is constrained. After full whitening with $\alpha = 0$, $\tilde{\mathbf{X}}$ is further constrained so that all singular values of $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ are *equal*. This adds an additional $p - 1$ constraints and thus reduces the dimensionality of the space by $p - 1$ dimensions; see Table A.1. The standard equations for geodesics and the tangent space above do not take this constraint into account; for instance, using $\tilde{\mathbf{Z}} = \mathrm{geodesic}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, t)$ with both $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ whitened, $\tilde{\mathbf{Z}}$ will not in general be whitened. This makes our definition of Shape Metrics with $\alpha < 1$ valid metrics, but not valid *length metrics* (namely, distances are not rectifiable because they do not respect the whitening constraint). By focusing on $\alpha = 1$ in the main paper, we circumvent this issue and retain all length and Riemannian structure of the metric. We leave proper treatment of the geometric structure of the $\alpha < 1$ case to future work.

A.3. Affine Invariant Riemannian Metric

The Affine Invariant Riemannian (AIR) metric is a metric between symmetric positive definite (SPD) matrices, originally derived for use in image processing (Pennec, 2006; 2019), and recently it was proposed to use it as a metric between neural representations by first converting neural data into a SPD matrix (Shahbazi et al., 2021). The **embedding function** can therefore be any function that maps $m \times n$ matrices in $\mathbb{X}$ into $\mathbb{M} = Sym_k^+$ for some k. Shahbazi et al. (2021) considered two possibilities for the embedding stage: either using the $m \times m$ Gram matrix $\mathrm{f}(\mathbf{X}) = \boldsymbol{G}_{\mathbf{X}}$, or using the $n \times n$ data covariance matrix $\mathrm{f}(\mathbf{X}) = \mathrm{cov}(\mathbf{X}) = \frac{1}{m-1}\mathbf{X}^\top\boldsymbol{H}\mathbf{X}$. These correspond to complementary perspectives on the nature of neural representation, analogous to the difference between representational similarity analysis and pattern component analysis (Diedrichsen & Kriegeskorte, 2017).

The challenge when using the $m \times m$ Gram matrix approach is that, without further regularization, a $m \times m$ Gram matrix has rank n when $n < m$, which implies that it cannot be SPD (and the metric considers all rank-deficient matrices to be infinitely far away). To address this, our toolbox implements the AIR metric between neural representations with additional regularization options. In the Gram matrix case, we regularize in two ways: first, we compute the Gram matrix using a kernel that implicitly has an infinite feature space (so that m is much less than the number of features). This alleviates the

rank-deficiency problem in cases where $n < m$ but rows are unique. However, when rows of $\mathbf{X}$ contain duplicates (notably, this is true for the target labels), $\boldsymbol{G}_{\mathbf{K}}$ is still rank-deficient. To address this, we include a second regularization stage where we add a small diagonal ridge with magnitude ϵ. The full embedding function in the Gram matrix case is given by

$$\mathrm{f}(\mathbf{X}) = \boldsymbol{G}_{\mathbf{X}} + \epsilon \mathbb{I}_m \tag{A.12}$$

where the ijth element of $\boldsymbol{G}_{\mathbf{X}}$ is given by $k(\mathbf{X}_i, \mathbf{X}_j)$. For our results in the paper, we use $\epsilon = 0.05$ and a squared exponential kernel for k as in (A.5), setting the length scale τ automatically to the median pairwise Euclidean distance between rows of $\mathbf{X}$.

The main challenge when using the covariance matrix approach is that it cannot be directly applied to compare layers with different numbers of neurons n. To address this, we first convert from $m \times n$ matrices of neural data into a common $m \times p$ size, using the same method as we use for the shape metrics as in (A.6), but without the whitening stage. We can then embed all layers into a common space of $p \times p$ covariance matrices. As in the Gram matrix case, we again run into rank-deficiency issues when $n < m$ (e.g. for the one-hot embedding of targets for which $n = 10$), and so we again regularize by adding a diagonal ridge to the resulting covariance matrices. The full embedding function in the covariance matrix case is given by

$$\mathrm{f}(\mathbf{X}) = \begin{cases} \mathrm{cov}\left([\mathbf{X},\, \mathbf{0}]\right) + \epsilon \mathbb{I}_p & \text{if } n \leq p \\ \mathrm{cov}\left(\mathbf{X}\boldsymbol{U}_{:p}\right) + \epsilon \mathbb{I}_p & \text{if } n > p \end{cases} \tag{A.13}$$

(compare with (A.6)).

Let $\mathbf{P} = \mathrm{f}(\mathbf{X})$ and $\mathbf{Q} = \mathrm{f}(\mathbf{Y})$ be SPD matrices (we are using $\mathbf{P}$ and $\mathbf{Q}$ instead of $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ to use a consistent notation with Pennec (2019)). The AIR metric distance is defined as

$$\mathrm{d}(\mathbf{X}, \mathbf{Y}) = \mathrm{d}_{\mathbb{M}}(\mathbf{P}, \mathbf{Q}) = \sum_i \log(d_i)^2 \tag{A.14}$$

where d_i is the ith eigenvalue of $\mathbf{P}^{-\frac{1}{2}}\mathbf{Q}\mathbf{P}^{-\frac{1}{2}}$ (Pennec, 2006; 2019). Since $\mathbf{P}$ is SPD, its singular value decomposition can be written $\mathbf{P} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where $\boldsymbol{\Sigma}$ is a diagonal matrix and $\boldsymbol{V}$ is orthonormal. Following Pennec (2019), we use element-wise square root, exp, and log operations on the singular values to define the matrix square root, matrix exponential, and matrix logarithm:

$$\begin{aligned} \mathrm{pow}\left(\mathbf{P}, k\right) &= \boldsymbol{V}\,\mathtt{pow}\left(\boldsymbol{\Sigma}, k\right)\boldsymbol{V}^\top \\ \exp\left(\mathbf{P}\right) &= \boldsymbol{V}\,\mathtt{exp}\left(\boldsymbol{\Sigma}\right)\boldsymbol{V}^\top \\ \log\left(\mathbf{P}\right) &= \boldsymbol{V}\,\mathtt{log}\left(\boldsymbol{\Sigma}\right)\boldsymbol{V}^\top \end{aligned}$$

where the operations on the left hand side are *matrix* power, exponential, and log, whereas `pow`, `exp`, and `log` operations are performed element-wise on the diagonal of $\boldsymbol{\Sigma}$. $\mathbf{P}^k$ is equivalent to $\mathrm{pow}\left(\mathbf{P}, k\right)$.

The **geodesics** from $\mathbf{P}$ to $\mathbf{Q}$ is given by

$$\mathrm{geodesic}(\mathbf{P}, \mathbf{Q}, t) = \mathbf{P}^{\frac{1}{2}}\left(\mathbf{P}^{-\frac{1}{2}}\mathbf{Q}\mathbf{P}^{-\frac{1}{2}}\right)^t\mathbf{P}^{\frac{1}{2}} \tag{A.15}$$

(combining equations (3.12) and (3.13) in (Pennec, 2019)).

Tangent vectors in this space are symmetric matrices, and the **logarithmic map** is given by

$$\log_{\mathbf{P}}(\mathbf{Q}) = \mathbf{P}^{\frac{1}{2}}\log\left(\mathbf{P}^{-\frac{1}{2}}\mathbf{Q}\mathbf{P}^{-\frac{1}{2}}\right)\mathbf{P}^{\frac{1}{2}} \tag{A.16}$$

(see equation (3.12) in (Pennec, 2019)). The **exponential map** of the tangent vector $\boldsymbol{W}$ at $\mathbf{P}$ is given by

$$\exp_{\mathbf{P}}(\boldsymbol{W}) = \mathbf{P}^{\frac{1}{2}}\exp\left(\mathbf{P}^{-\frac{1}{2}}\boldsymbol{W}\mathbf{P}^{-\frac{1}{2}}\right)\mathbf{P}^{\frac{1}{2}} \tag{A.17}$$

(see equation (3.13) in (Pennec, 2019)). One can easily verify that $\exp_{\mathbf{P}}(\log_{\mathbf{P}}(\mathbf{Q})) = \mathbf{Q}$.

As before, we compute **angles** between triplets of representations $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ by computing the inner product of the $\log_{\mathbf{Y}}(\mathbf{X})$ and $\log_{\mathbf{Y}}(\mathbf{Z})$ tangent vectors, but unlike the previous metrics the definition of inner products for the AIR metric is not simply the Frobenius inner product. For the AIR metric, the **inner product** of tangent vectors $\boldsymbol{W}$ and $\boldsymbol{V}$ at $\mathbf{P}$ is defined as

$$\langle \boldsymbol{W}, \boldsymbol{V} \rangle_{\mathbf{P}} \equiv \left\langle \mathbf{P}^{-1}\boldsymbol{W}, \mathbf{P}^{-1}\boldsymbol{V} \right\rangle_{\mathrm{F}} = \mathrm{Tr}\left(\boldsymbol{W}^\top\mathbf{P}^{-\top}\mathbf{P}^{-1}\boldsymbol{V}\right). \tag{A.18}$$

A.3.1. Invariances of Affine Invariant Riemannian Metric

We will treat the Gram matrix (A.12) and the covariance matrix (A.13) cases separately. In the Gram matrix case,

- The AIR metric is **shift-invariant**, **scale-invariant**, and/or **rotation-invariant** if and only if the kernel used to compute $\boldsymbol{G}_{\mathbf{X}}$ has the corresponding invariance. Because we use a squared-exponential kernel with a length scale that adapts to the data scale, we have all three invariances.
- The AIR metric is, despite its name, not **affine-invariant** in the sense we are interested in, since affine transformations of $\mathbf{X}$ will in general affect $\boldsymbol{G}_{\mathbf{X}}$ through the nonlinear kernel (e.g. the squared exponential kernel with an *isotropic* length scale is sensitive to non-isotropic scaling of $\mathbf{X}$).

In the covariance matrix case,

- The AIR metric is **shift-invariant** because covariance subtracts the mean.
- The AIR metric is **scale-** and **rotation-invariant** due to the eponymous "affine-invariances" of the metric itself (Pennec, 2006; 2019).
- As in the case of shape metrics discussed above, the AIR metric may or may not be invariant to arbitrary affine transformations of $\mathbf{X}$ due to the restriction to the top p principal components in the embedding stage. Only in the case where $n > p$ and $\boldsymbol{A}$ is a matrix such that $\mathbf{X}\boldsymbol{A}$ changes the subspace of the top p principal components, then the resulting metric is not invariant to $\boldsymbol{A}$.

B. Proof of HSIC estimator bias

Recall from equation (5) that we defined

$$\text{HSIC}_{\text{ours}}(\boldsymbol{X}, \boldsymbol{Y}) \equiv \frac{2}{m(m-3)} \left\langle \text{tril}(\boldsymbol{H}\boldsymbol{G}_{\mathbf{X}}\boldsymbol{H}), \text{tril}(\boldsymbol{H}\boldsymbol{G}_{\mathbf{Y}}\boldsymbol{H}) \right\rangle_F ,$$

where $\text{tril}(\boldsymbol{A})$ is a function that zeros out all but the lower triangle of $\boldsymbol{A}$, zeroing the diagonal as well.

To simplify notation, let $\boldsymbol{K} = \boldsymbol{G}_{\boldsymbol{X}}$ and $\boldsymbol{L} = \boldsymbol{G}_{\boldsymbol{Y}}$. Defining $\text{diag}(\boldsymbol{A})$ to be the diagonal of $\boldsymbol{A}$ as a column vector, and using the fact that $\boldsymbol{K}$ and $\boldsymbol{L}$ are symmetric Gram matrices, our estimator is equivalent to

$$\text{HSIC}_{\text{ours}}(\boldsymbol{X}, \boldsymbol{Y}) \equiv \frac{1}{m(m-3)} \left(\langle \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}, \boldsymbol{H}\boldsymbol{L}\boldsymbol{H} \rangle_F - \text{diag}(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H})^\top \text{diag}(\boldsymbol{H}\boldsymbol{L}\boldsymbol{H}) \right) .$$

Using symmetry and idempotency of $\boldsymbol{H}$, and the cyclic property of the trace, $\langle \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}, \boldsymbol{H}\boldsymbol{L}\boldsymbol{H} \rangle_F = \text{Tr}(\boldsymbol{K}\boldsymbol{H}\boldsymbol{L}\boldsymbol{H})$ which is equivalent to a multiple of the biased estimator found by Gretton et al. (2005):

$$\begin{aligned}
\text{HSIC}_{\text{Gretton}}(\boldsymbol{X}, \boldsymbol{Y}) &\equiv \frac{\text{Tr}(\boldsymbol{K}\boldsymbol{H}\boldsymbol{L}\boldsymbol{H})}{(m-1)^2} = \frac{1}{(m-1)^2} \left(a_{\text{Gretton}} - \frac{2}{m} b_{\text{Gretton}} + \frac{1}{m^2} c_{\text{Gretton}} \right) \\
a_{\text{Gretton}} &\equiv \text{Tr}(\boldsymbol{K}\boldsymbol{L}) \\
b_{\text{Gretton}} &\equiv \mathbf{1}^\top \boldsymbol{K}\boldsymbol{L}\mathbf{1} \\
c_{\text{Gretton}} &\equiv \mathbf{1}^\top \boldsymbol{K}\mathbf{1}\mathbf{1}^\top \boldsymbol{L}\mathbf{1}
\end{aligned}$$

Song et al. (2007) introduced an *unbiased* estimator of HSIC, given by

$$\text{HSIC}_{\text{Song}}(\boldsymbol{X}, \boldsymbol{Y}) \equiv \frac{1}{m(m-3)} \left(a_{\text{Song}} - \frac{2}{m-2} b_{\text{Song}} + \frac{1}{(m-1)(m-2)} c_{\text{Song}} \right)$$

$$\begin{aligned}
a_{\text{Song}} &\equiv \text{Tr}(\hat{\boldsymbol{K}}\hat{\boldsymbol{L}}) && = \text{Tr}(\boldsymbol{K}\boldsymbol{L}) - \text{diag}(\boldsymbol{K})^\top \text{diag}(\boldsymbol{L}) \\
b_{\text{Song}} &\equiv \mathbf{1}^\top \hat{\boldsymbol{K}}\hat{\boldsymbol{L}}\mathbf{1} && = \mathbf{1}^\top \boldsymbol{K}\boldsymbol{L}\mathbf{1} - \mathbf{1}^\top \boldsymbol{K}\text{diag}(\boldsymbol{L}) - \mathbf{1}^\top \boldsymbol{L}\text{diag}(\boldsymbol{K}) + \text{diag}(\boldsymbol{K})^\top \text{diag}(\boldsymbol{L}) \\
c_{\text{Song}} &\equiv \mathbf{1}^\top \hat{\boldsymbol{K}}\mathbf{1}\mathbf{1}^\top \hat{\boldsymbol{L}}\mathbf{1} && = \mathbf{1}^\top \boldsymbol{K}\mathbf{1}\mathbf{1}^\top \boldsymbol{L}\mathbf{1} - \mathbf{1}^\top \boldsymbol{K}\mathbf{1}\,\text{Tr}(\boldsymbol{L}) - \mathbf{1}^\top \boldsymbol{L}\mathbf{1}\,\text{Tr}(\boldsymbol{K}) + \text{Tr}(\boldsymbol{K})\,\text{Tr}(\boldsymbol{L}) .
\end{aligned}$$

where the hat symbol $\hat{\boldsymbol{A}}$ denotes that the diagonal of $\boldsymbol{A}$ has been set to zero. For ease of manipulation, we include expressions for a_{Song}, b_{Song}, and c_{Song} in terms of $\boldsymbol{K}$ and $\boldsymbol{L}$.

Both our estimator and Song et al's estimator share the same general strategy of mitigating bias by removing the diagonal, since the diagonal of the Gram matrices contain *non-independent* samples. The primary difference is that it is impossible to express $\text{HSIC}_{\text{Song}}$ as an inner-product of real-valued vectors, due to the subtraction of $\frac{2}{m-2}b_{\text{Song}}$.

Our strategy for proving that our estimator has $\mathcal{O}(m^{-2})$ bias by taking the difference $\text{HSIC}_{\text{ours}}(\boldsymbol{X}, \boldsymbol{Y}) - \text{HSIC}_{\text{Song}}(\boldsymbol{X}, \boldsymbol{Y})$ and inspecting the asymptotics of the remaining terms, since

$$\begin{aligned}\text{bias} &= \mathbb{E}[\text{HSIC}_{\text{ours}}(\boldsymbol{X}, \boldsymbol{Y})] - \text{HSIC}_{True} \\ &= \mathbb{E}[\text{HSIC}_{\text{ours}}(\boldsymbol{X}, \boldsymbol{Y})] - \mathbb{E}[\text{HSIC}_{\text{Song}}(\boldsymbol{X}, \boldsymbol{Y})] \\ &= \mathbb{E}[\text{HSIC}_{\text{ours}}(\boldsymbol{X}, \boldsymbol{Y}) - \text{HSIC}_{\text{Song}}(\boldsymbol{X}, \boldsymbol{Y})]\,.\end{aligned}$$

First, we rewrite our estimator as the difference between the original biased estimator and a product of matrix diagonals:

$$\text{HSIC}_{\text{ours}}(\boldsymbol{X}, \boldsymbol{Y}) \equiv \frac{1}{m(m-3)}\left((m-1)^2\text{HSIC}_{\text{Gretton}}(\boldsymbol{X}, \boldsymbol{Y}) - \text{diag}(\boldsymbol{HKH})^\top \text{diag}(\boldsymbol{HLH})\right)\,.$$

Using the definition $\boldsymbol{H} = \mathbb{I} - m^{-1}\mathbf{1}\mathbf{1}^\top$, the diagonal term expands as:

$$\begin{aligned}\text{diag}(\boldsymbol{HKH})^\top \text{diag}(\boldsymbol{HLH}) &= a_{\text{diag}} - \frac{2}{m}b_{\text{diag}} + \frac{1}{m^2}c_{\text{diag}} \\ a_{\text{diag}} &\equiv \text{diag}(\boldsymbol{K})^\top \text{diag}(\boldsymbol{L}) \\ b_{\text{diag}} &\equiv \mathbf{1}^\top \boldsymbol{K}\text{diag}(\boldsymbol{L}) + \mathbf{1}^\top \boldsymbol{L}\text{diag}(\boldsymbol{K}) \\ c_{\text{diag}} &\equiv \mathbf{1}^\top \boldsymbol{K}\mathbf{1}\,\text{Tr}(\boldsymbol{L}) + \mathbf{1}^\top \boldsymbol{L}\mathbf{1}\,\text{Tr}(\boldsymbol{K}) + 4\mathbf{1}^\top \boldsymbol{KL}\mathbf{1} - \frac{3}{m}\mathbf{1}^\top \boldsymbol{K}\mathbf{1}\mathbf{1}^\top \boldsymbol{L}\mathbf{1}\,.\end{aligned}$$

Taking the difference between the estimators gives:

$$\begin{aligned}&\text{HSIC}_{\text{ours}}(\boldsymbol{X}, \boldsymbol{Y}) - \text{HSIC}_{\text{Song}}(\boldsymbol{X}, \boldsymbol{Y}) \\ &= \frac{1}{m(m-3)}\left(a_{\text{Gretton}} - a_{\text{diag}} - a_{\text{Song}}\right) \\ &\quad - \frac{2}{m^2(m-2)(m-3)}\left((m-2)\left(b_{\text{Gretton}} - b_{\text{diag}}\right) - mb_{\text{Song}}\right) \\ &\quad + \frac{1}{m^3(m-1)(m-2)(m-3)}\left((m-1)(m-2)\left(c_{\text{Gretton}} - c_{\text{diag}}\right) - m^2 c_{\text{Song}}\right)\,.\end{aligned}$$

Next, we examine each of a, b, and c separately. The a terms cancel exactly:

$$a_{\text{Gretton}} - a_{\text{diag}} - a_{\text{Song}} = \text{Tr}(\boldsymbol{KL}) - \text{diag}(\boldsymbol{K})^\top \text{diag}(\boldsymbol{L}) - \text{Tr}(\boldsymbol{KL}) + \text{diag}(\boldsymbol{K})^\top \text{diag}(\boldsymbol{L}) = 0\,.$$

The b terms give the difference:

$$\begin{aligned}(m-2)\left(b_{\text{Gretton}} - b_{\text{diag}}\right) - mb_{\text{Song}} &= (m-2)\left(\mathbf{1}^\top \boldsymbol{KL}\mathbf{1} - \mathbf{1}^\top \boldsymbol{K}\text{diag}(\boldsymbol{L}) - \mathbf{1}^\top \boldsymbol{L}\text{diag}(\boldsymbol{K})\right) \\ &\quad - m\left(\mathbf{1}^\top \boldsymbol{KL}\mathbf{1} - \mathbf{1}^\top \boldsymbol{K}\text{diag}(\boldsymbol{L}) - \mathbf{1}^\top \boldsymbol{L}\text{diag}(\boldsymbol{K}) + \text{diag}(\boldsymbol{K})^\top \text{diag}(\boldsymbol{L})\right) \\ &= -2\mathbf{1}^\top \boldsymbol{KL}\mathbf{1} + 2\mathbf{1}^\top \boldsymbol{K}\text{diag}(\boldsymbol{L}) + 2\mathbf{1}^\top \boldsymbol{L}\text{diag}(\boldsymbol{K}) - m\text{diag}(\boldsymbol{K})^\top \text{diag}(\boldsymbol{L}) \\ &= -2\mathbf{1}^\top \boldsymbol{KL}\mathbf{1} + O(m^2)\,.\end{aligned}$$

Note the order of each component depends on the degrees of freedom in the equivalent summation. For example, $\mathbf{1}^\top \boldsymbol{K}\text{diag}(\boldsymbol{L}) = \sum_{i=1}^m \sum_{j=1}^m \boldsymbol{K}_{ij}\boldsymbol{L}_{ii}$ is a sum over m^2 products.

The c terms give the difference:

$$\begin{aligned}
&(m-1)(m-2)\left(c_{\text{Gretton}} - c_{\text{diag}}\right) - m^2 c_{\text{Song}} \\
&\quad = (m-1)(m-2)\left(\mathbf{1}^\top \boldsymbol{K}\mathbf{1}\mathbf{1}^\top \boldsymbol{L}\mathbf{1} - \mathbf{1}^\top \boldsymbol{K}\mathbf{1}\operatorname{Tr}(\boldsymbol{L}) - \mathbf{1}^\top \boldsymbol{L}\mathbf{1}\operatorname{Tr}(\boldsymbol{K}) - 4\mathbf{1}^\top \boldsymbol{K}\boldsymbol{L}\mathbf{1} + \frac{3}{m}\mathbf{1}^\top \boldsymbol{K}\mathbf{1}\mathbf{1}^\top \boldsymbol{L}\mathbf{1}\right) \\
&\qquad - m^2\left(\mathbf{1}^\top \boldsymbol{K}\mathbf{1}\mathbf{1}^\top \boldsymbol{L}\mathbf{1} - \mathbf{1}^\top \boldsymbol{K}\mathbf{1}\operatorname{Tr}(\boldsymbol{L}) - \mathbf{1}^\top \boldsymbol{L}\mathbf{1}\operatorname{Tr}(\boldsymbol{K}) + \operatorname{Tr}(\boldsymbol{K})\operatorname{Tr}(\boldsymbol{L})\right) \\
&\quad = -\frac{7m-6}{m}\mathbf{1}^\top \boldsymbol{K}\mathbf{1}\mathbf{1}^\top \boldsymbol{L}\mathbf{1} + (3m-2)\left(\mathbf{1}^\top \boldsymbol{K}\mathbf{1}\operatorname{Tr}(\boldsymbol{L}) + \mathbf{1}^\top \boldsymbol{L}\mathbf{1}\operatorname{Tr}(\boldsymbol{K})\right) \\
&\qquad - 4(m-1)(m-2)\mathbf{1}^\top \boldsymbol{K}\boldsymbol{L}\mathbf{1} - m^2\operatorname{Tr}(\boldsymbol{K})\operatorname{Tr}(\boldsymbol{L}) \\
&\quad = -4(m-1)(m-2)\mathbf{1}^\top \boldsymbol{K}\boldsymbol{L}\mathbf{1} + \mathcal{O}(m^4)\,.
\end{aligned}$$

Finally, substituting a, b, and c into the bias equation and cancelling shows that our estimator's bias is of order $\mathcal{O}(m^{-2})$:

$$\begin{aligned}
&\text{HSIC}_{\text{ours}}(\boldsymbol{X}, \boldsymbol{Y}) - \text{HSIC}_{\text{Song}}(\boldsymbol{X}, \boldsymbol{Y}) \\
&\quad = -2\frac{-2\mathbf{1}^\top \boldsymbol{K}\boldsymbol{L}\mathbf{1} + \mathcal{O}(m^2)}{m^2(m-2)(m-3)} + \frac{-4(m-1)(m-2)\mathbf{1}^\top \boldsymbol{K}\boldsymbol{L}\mathbf{1} + \mathcal{O}(m^4)}{m^3(m-1)(m-2)(m-3)} \\
&\quad = \frac{4\left(m(m-1) - (m-1)(m-2)\right)}{m^3(m-1)(m-2)(m-3)}\mathbf{1}^\top \boldsymbol{K}\boldsymbol{L}\mathbf{1} + \mathcal{O}(m^{-2}) \\
&\quad = \frac{8(m+1)}{m^3(m-1)(m-2)(m-3)}\mathbf{1}^\top \boldsymbol{K}\boldsymbol{L}\mathbf{1} + \mathcal{O}(m^{-2}) \\
&\quad = \mathcal{O}(m^{-2})
\end{aligned}$$

□

C. Numerical details and algorithms

For all metrics we study here, we have closed-form expressions for geodesics, logarithmic maps, exponential maps, inner-products in the tangent space, and parallel transport (although we do not use parallel transport in this paper). Let $\langle \mathbf{U}, \mathbf{V}\rangle_{\tilde{\mathbf{X}}}$ denote the inner product of tangent vectors $\mathbf{U}$ and $\mathbf{V}$ at the point $\tilde{\mathbf{X}} \in \mathbb{M}$. The angle between $\mathbf{U}$ and $\mathbf{V}$ is

$$\arccos\left(\frac{\langle \mathbf{U}, \mathbf{V}\rangle_{\tilde{\mathbf{X}}}}{\sqrt{\langle \mathbf{U}, \mathbf{U}\rangle_{\tilde{\mathbf{X}}}\langle \mathbf{V}, \mathbf{V}\rangle_{\tilde{\mathbf{X}}}}}\right).$$

In the main text, we used this to compute the **interior angle** of a network's path at layer l, using this formula with $\mathbf{U} = \log_{\tilde{\mathbf{X}}^l}(\tilde{\mathbf{X}}^{l-1})$ and $\mathbf{V} = \log_{\tilde{\mathbf{X}}^l}(\tilde{\mathbf{X}}^{l+1})$. We also used it to compute the **target angle** at layer l, using $\mathbf{U} = \log_{\tilde{\mathbf{X}}^l}(\tilde{\mathbf{X}}^{l+1})$ and $\mathbf{V} = \log_{\tilde{\mathbf{X}}^l}(\tilde{\mathbf{T}})$ where $\tilde{\mathbf{T}}$ denotes the embedding of the target outputs (i.e. embedding of the one-hot class vectors).

We used an iterative algorithm to compute the **projection** of a point $\tilde{\mathbf{Z}}$ onto the geodesic spanning $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$. Specifically, we used an iterative procedure that reaches the correct projection in a single iteration on flat (isometric to Euclidean) manifolds. In Euclidean space, the projection of $\mathbf{z}$ onto the vector spanning $\mathbf{x} \to \mathbf{y}$ is given by

$$\begin{aligned}
\text{projection_length}(\mathbf{z}, \mathbf{x}, \mathbf{y}) &= (\mathbf{z} - \mathbf{x})^\top \left(\frac{\mathbf{y} - \mathbf{x}}{||\mathbf{y} - \mathbf{x}||}\right) \\
\text{project}(\mathbf{z}, \mathbf{x}, \mathbf{y}) &= \mathbf{x} + \text{projection_length}(\mathbf{z}, \mathbf{x}, \mathbf{y}) \times \left(\frac{\mathbf{y} - \mathbf{x}}{||\mathbf{y} - \mathbf{x}||}\right)
\end{aligned}$$

The analogue in curved spaces is given by

$$\begin{aligned}
\text{projection_length}(\tilde{\mathbf{Z}}, \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) &= \left\langle \log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Z}}), \frac{\log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}})}{||\log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}})||} \right\rangle \\
\text{project}(\tilde{\mathbf{Z}}, \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) &= \exp_{\tilde{\mathbf{X}}}\left(\text{projection_length}(\tilde{\mathbf{Z}}, \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \times \frac{\log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}})}{||\log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}})||}\right)
\end{aligned}$$

Our algorithm for projection on curved manifolds iteratively solves for t, the projection length onto the tangent vector from $\tilde{\mathbf{X}}$ to $\tilde{\mathbf{Y}}$ as follows:

1. initialize $t = 0$
2. calculate the base point $\tilde{\mathbf{B}} \equiv \exp_{\tilde{\mathbf{X}}}\left(t \times \frac{\log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}})}{||\log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}})||}\right)$
3. calculate how far to update t using the formula for flat spaces: $\Delta t \equiv \text{projection\_length}(\tilde{\mathbf{Z}}, \tilde{\mathbf{B}}, \tilde{\mathbf{Y}})$
4. update $t \leftarrow t + \Delta t$ and go to step 2 unless converged, in which case return $\tilde{\mathbf{B}}$.

D. Models and training details

We trained a collection of convolutional networks including both residual networks (He et al., 2016) and VGG (Simonyan & Zisserman, 2014) on CIFAR-10 (Krizhevsky, 2009) using PyTorch (Paszke et al., 2019). We used the open-source OpenLTH framework for training and checkpointing models, using the default hyperparameters for each model.

Following Nguyen et al. (2021), we trained Residual networks of varying widths and depths. The "width" refers to the number of feature channels per convolutional layer, and took on values of $\{16, 32, 64, 128, 160\}$ (corresponding to the base size of 16 multiplied by $\{1\times, 2\times, 4\times, 8\times, 10\times\}$). The "depth" controls the number of residual blocks, according to the formula $\#\text{blocks} = (\text{depth} - 2)/2$, since each block contains two convolutional layers, and there are two additional preprocessing/projection layers before/after the blocks. We trained models of depths $\{14, 20, 26, 32, 38, 44, 56, 110\}$. The VGG architecture supports "depths" of $\{11, 13, 16, 18\}$, all at the same width. The test accuracy of all models is shown in Table D.2. We analyzed the representational distances and geometry of a subset of these, focusing on depths 14 and 38 (for all widths), and widths 16 and 64 (for all depths).

All models were trained using the default training hyperparameters of OpenLTH. Specifically, all models were trained by stochastic gradient descent for 160 epochs with a batch size of 128 (390.6 batches per epoch of 50k training items), an initial learning rate of 0.1 reducing to 0.01 and 0.001 after 80 and 120 epochs respectively, momentum of 0.9, and weight decay of 0.0001. During training, images were augmented by random horizontal flips and random ± 4 and ± 8 pixel left/right or up/down shifts (padding with zeros) for CIFAR-10 and Tiny ImageNet.

When evaluating TinyImagenet models on CIFAR-10 data to co-embed them in the same space, we upscaled the CIFAR-10 images from 32×32 to 64×64 using bilinear interpolation built in to PyTorch.

E. Additional figures

Architecture (depth/width)	CIFAR-10 test accuracy (mean ± std)
VGG 11	0.919 ± 0.002
VGG 13	0.935 ± 0.001
VGG 16	0.934 ± 0.001
VGG 19	0.933 ± 0.001
ResNet 14/16	0.907 ± 0.003
ResNet 14/32	0.931 ± 0.001
ResNet 14/64	0.943 ± 0.001
ResNet 14/128	0.949 ± 0.001
ResNet 14/160	0.949 ± 0.001
ResNet 20/16	0.917 ± 0.002
ResNet 20/32	0.939 ± 0.002
ResNet 20/64	0.948 ± 0.001
ResNet 20/128	0.953 ± 0.001
ResNet 20/160	0.954 ± 0.001
ResNet 26/16	0.922 ± 0.002
ResNet 26/32	0.940 ± 0.002
ResNet 26/64	0.950 ± 0.001
ResNet 26/128	0.954 ± 0.001
ResNet 26/160	0.954 ± 0.002
ResNet 32/16	0.925 ± 0.002
ResNet 32/32	0.942 ± 0.002
ResNet 32/64	0.950 ± 0.001
ResNet 32/128	0.954 ± 0.001
ResNet 32/160	0.954 ± 0.003
ResNet 38/16	0.926 ± 0.001
ResNet 38/32	0.945 ± 0.001
ResNet 38/64	0.951 ± 0.002
ResNet 38/128	0.954 ± 0.004
ResNet 38/160	0.951 ± 0.002
ResNet 44/16	0.927 ± 0.001
ResNet 44/32	0.944 ± 0.001
ResNet 44/64	0.950 ± 0.001
ResNet 44/128	0.949 ± 0.003
ResNet 44/160	0.950 ± 0.002
ResNet 56/16	0.929 ± 0.002
ResNet 56/32	0.944 ± 0.002
ResNet 56/64	0.950 ± 0.001
ResNet 56/128	0.946 ± 0.007
ResNet 56/160	0.947 ± 0.003
ResNet 110/16	0.934 ± 0.002
ResNet 110/32	0.942 ± 0.002
ResNet 110/64	0.938 ± 0.003
ResNet 110/128	0.935 ± 0.009
ResNet 110/160	0.942 ± 0.005
ResNet 164/16	0.934 ± 0.003
ResNet 164/32	0.937 ± 0.004
ResNet 164/64	0.938 ± 0.006
ResNet 164/128	0.941 ± 0.012
ResNet 164/160	0.943 ± 0.009

Table D.2. Model architectures and performance on CIFAR-10.

Architecture (depth/width)	Tiny ImageNet test accuracy
ResNet 14/16	0.498
ResNet 14/32	0.570
ResNet 14/64	0.600
ResNet 14/128	0.637
ResNet 14/160	0.635
ResNet 20/16	0.535
ResNet 20/32	0.581
ResNet 20/64	0.620
ResNet 20/128	0.654
ResNet 20/160	0.659
ResNet 26/16	0.542
ResNet 26/32	0.588
ResNet 26/64	0.627
ResNet 26/128	0.666
ResNet 26/160	0.667
ResNet 32/16	0.550
ResNet 32/32	0.595
ResNet 32/64	0.638
ResNet 32/128	0.664
ResNet 32/160	0.670
ResNet 38/16	0.555
ResNet 38/32	0.595
ResNet 38/64	0.639
ResNet 38/128	0.674
ResNet 38/160	0.673
ResNet 44/16	0.564
ResNet 44/32	0.605
ResNet 44/64	0.648
ResNet 44/128	0.674
ResNet 44/160	0.675
ResNet 56/16	0.569
ResNet 56/32	0.608
ResNet 56/64	0.655

Table D.3. Model architectures and performance on Tiny ImageNet. (Note only one seed for each combination was run, so no error bounds are placed on the accuracy)

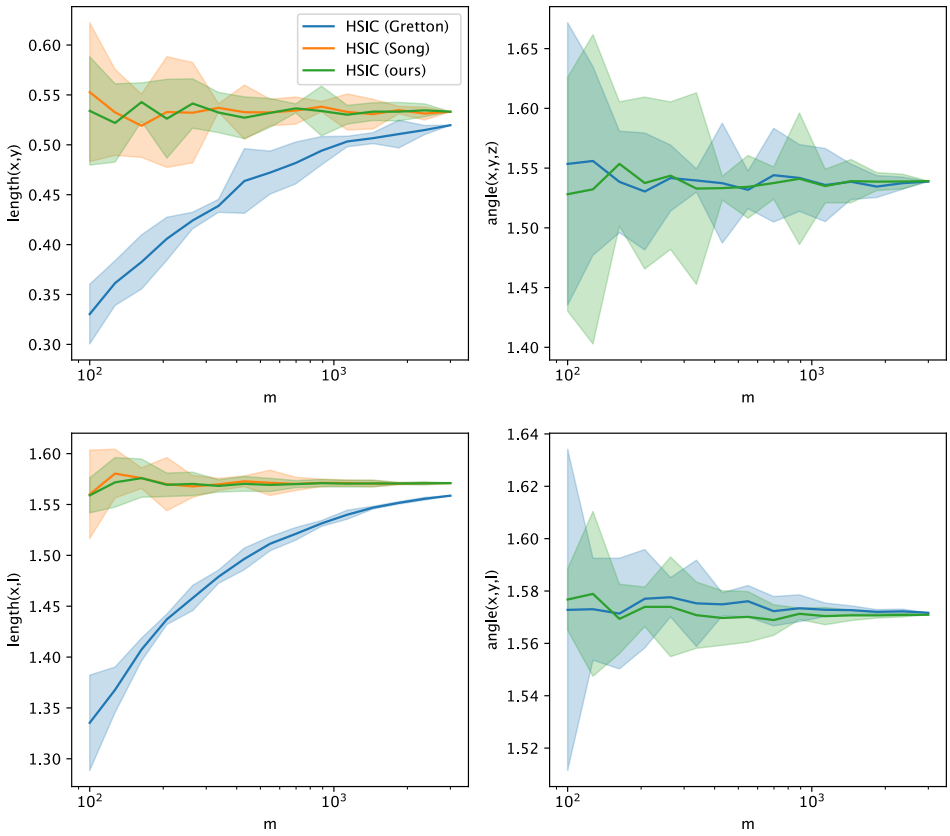

Figure E.1. Inspecting bias and variance of calculations of Angular CKA using different underlying estimators for HSIC. "Gretton" refers to Gretton et al. (2005), and the estimator given in equation (4). "Song" refers to Song et al. (2007) and the estimator given in equation (**??**). "Ours" refers to equation (5). The Song estimator is known to be unbiased, the Gretton estimator is known to have $\mathcal{O}(m^{-1})$ bias, and we prove in Appendix B that our estimator has $\mathcal{O}(m^{-2})$ bias. Lines and error bars show mean $\pm$ standard deviation of each quantity across four runs for each value of m. **Top left:** estimates of representational distance or length from one convolutional layer to another in an example ResNet. **Top right:** estimates of interior angles among three convolutional layers; note that the Song estimator is omitted because it is not expressible as an inner product in a vector space, and so we cannot compute geodesics or angles with it. **Bottom left:** estimates of distance from a convolutional layer to one-hot labels. **Bottom right:** estimates of target angle, i.e. the angle between $\tilde{\mathbf{X}}^l$, $\tilde{\mathbf{X}}^{l+1}$, and one-hot labels, for some layer l.

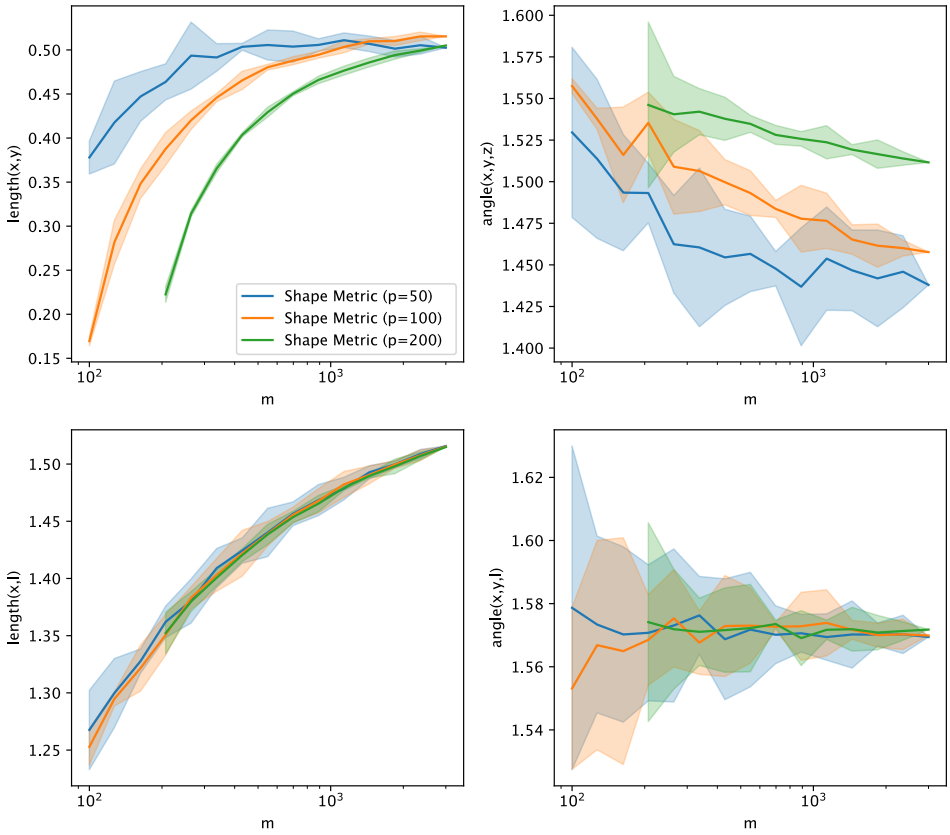

Figure E.2. Same as Figure E.1 but for the Angular Shape Metric. We reduce the dimensionality of convolutional layers to $p = 100$ and use $m = 1000$. Note that this means that estimates of length and interior angles may be slightly biased throughout.

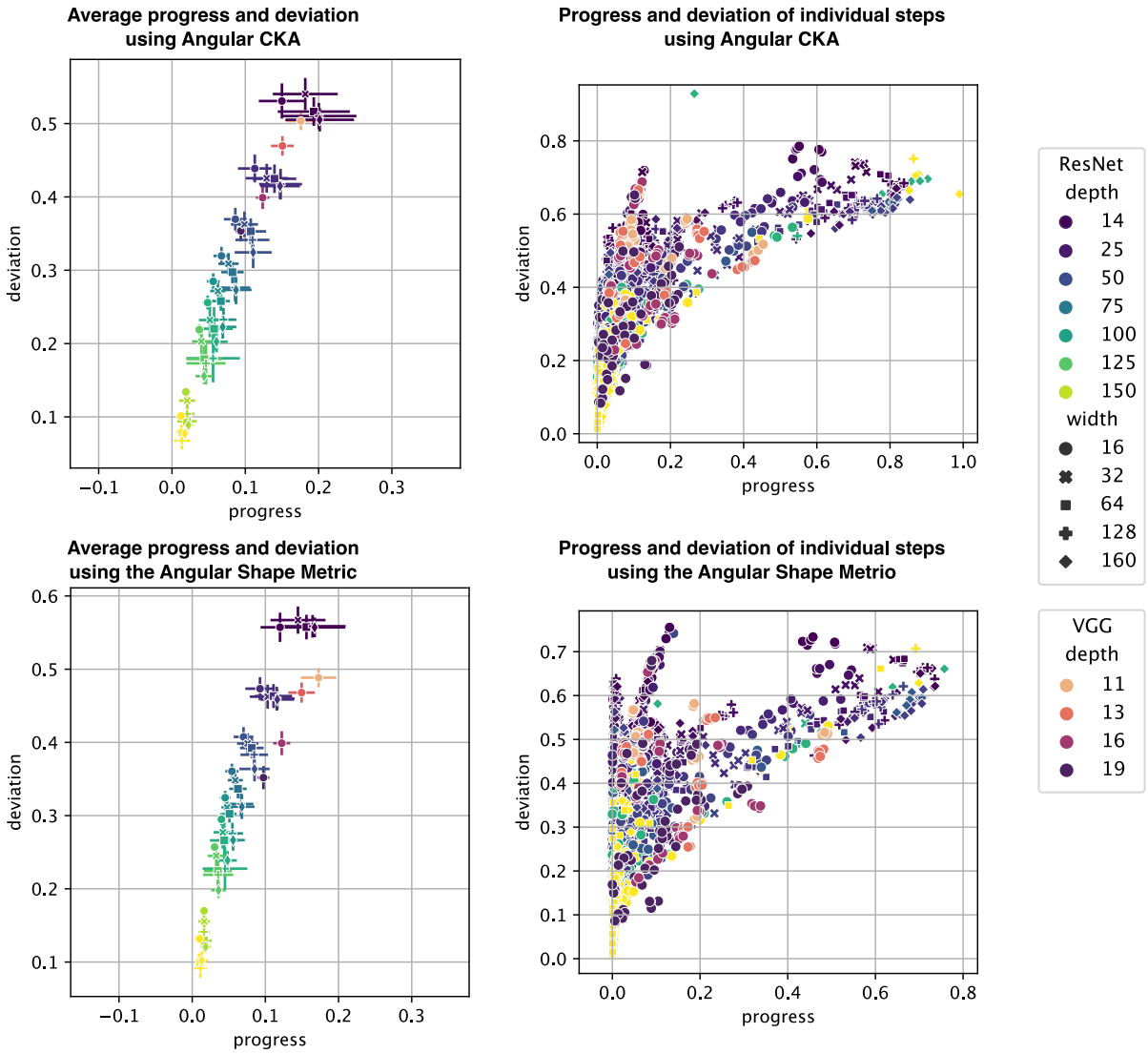

Figure E.3. Companion to Figure 4 in the main text. Here, we additionally show progress and deviation for every layer individually (right subplots) and identical analyses using the Angular Shape Metric (bottom row).

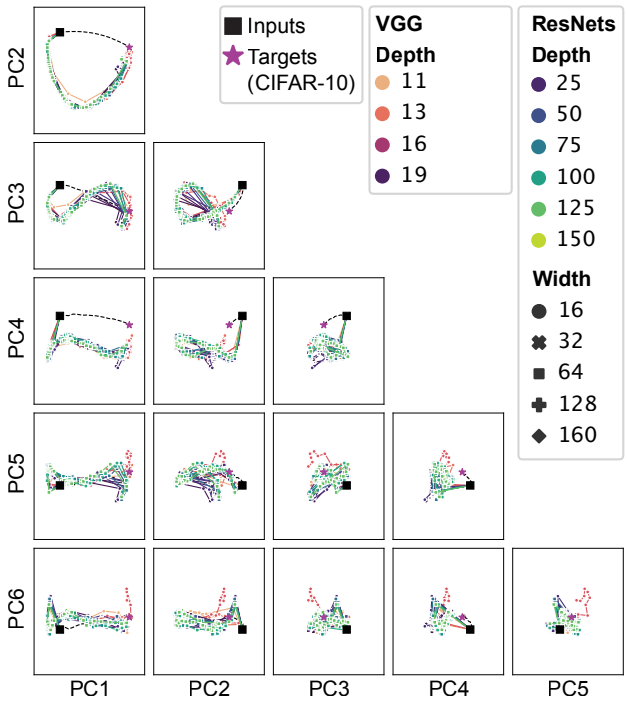

Figure E.4. Grid of first 5 principal components of path visualizations for various architectures on CIFAR-10. Figure 2A shows PC1 vs PC2 (top left of this figure).

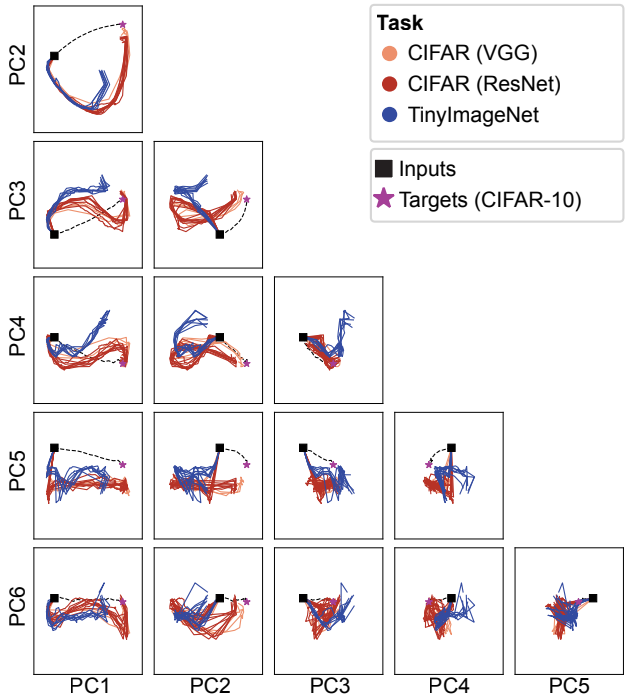

Figure E.5. Grid of first 5 principal components of path visualizations comparing models trained on CIFAR-10 to models trained on TinyImagenet. Figure 2B shows PC1 vs PC2 (top left of this figure).

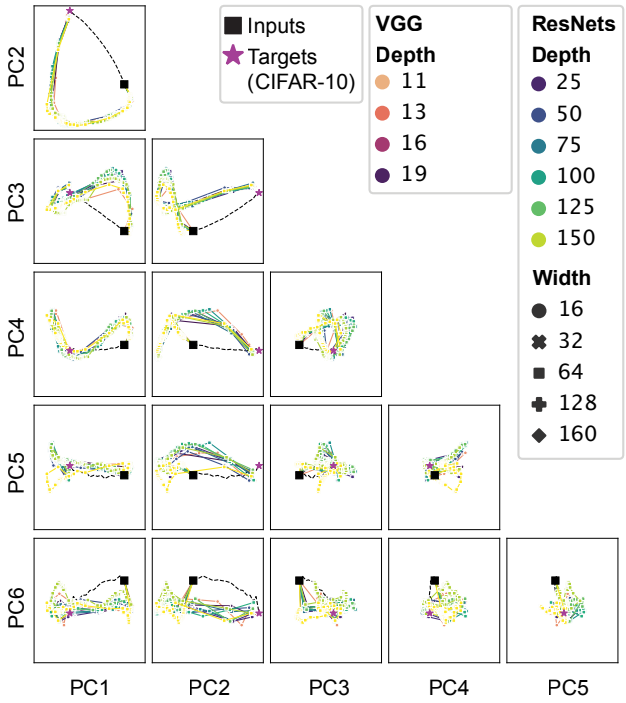

Figure E.6. Same as Figure E.4 – comparing paths taken by various model architectures trained on CIFAR-10 – but for Angular Shape Metric.

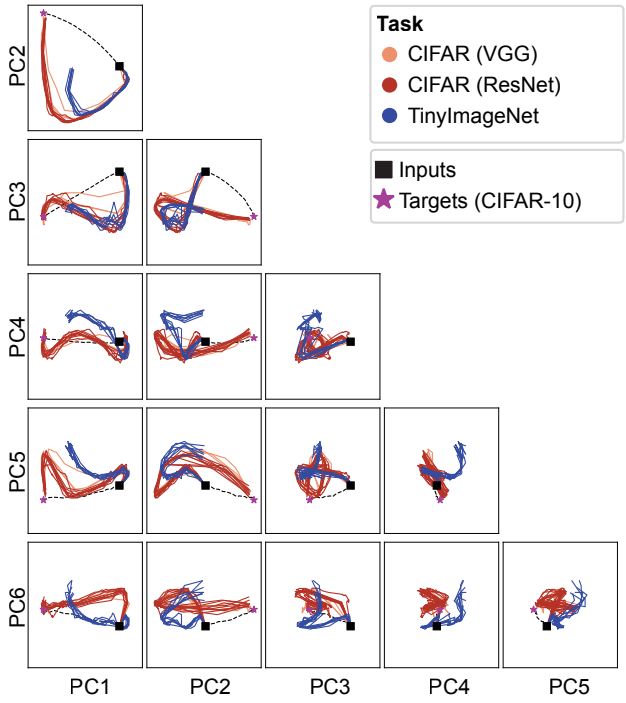

Figure E.7. Same as Figure E.5 – comparing paths taken by similar models trained on different datasets – but for the Angular Shape Metric.

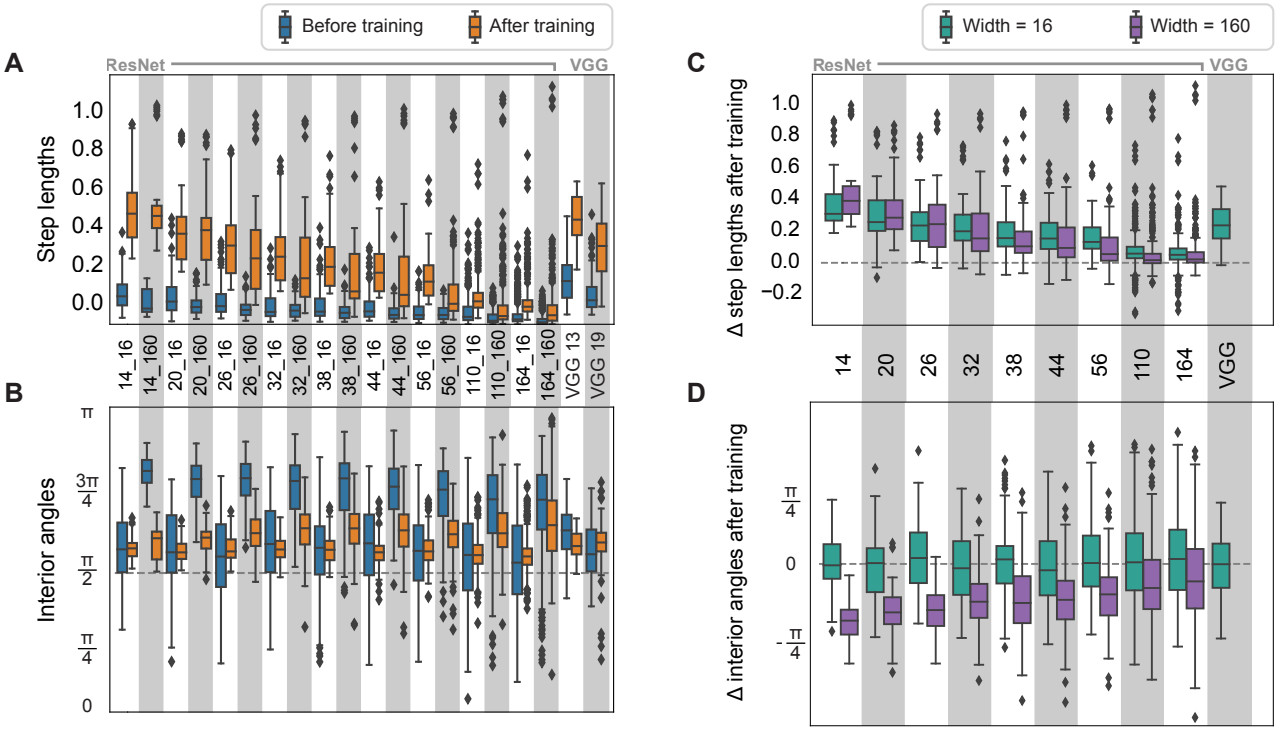

Figure E.8. **Angular CKA:** Elaborating on Figure 3 in the main text. Here, we reproduce **A)** the distances between layers per model before and after learning, and **B)** the interior angles between layers per model before and after training. We then show **C)** the *change* in step length and **D)** change in interior angles per layer per model before and after learning. The main effect we see is that interior angles trend towards becoming *more acute* with learning (the delta is negative), and this effect is more pronounced in shallower models than deep models.

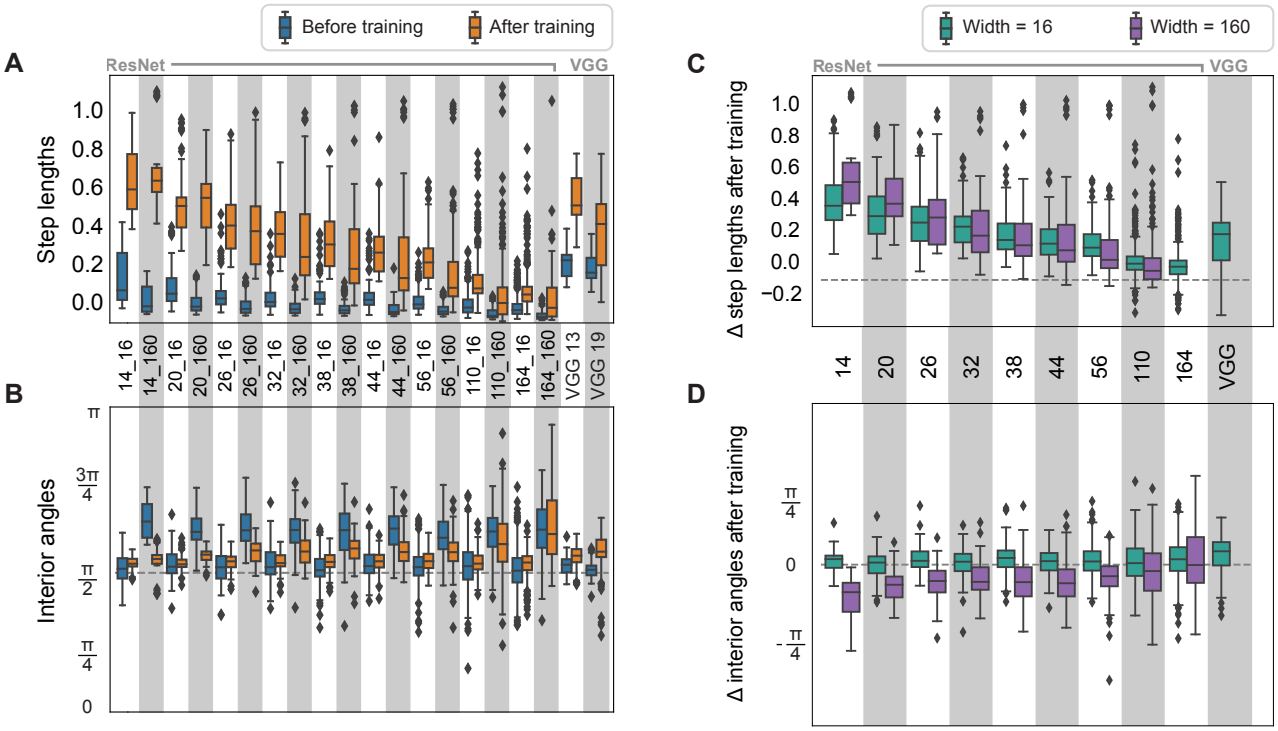

Figure E.9. **Angular Shape Metric:** The matching figure to Figure E.8, here showing results for the angular shape metric. The main trends are qualitatively conserved between metric spaces.

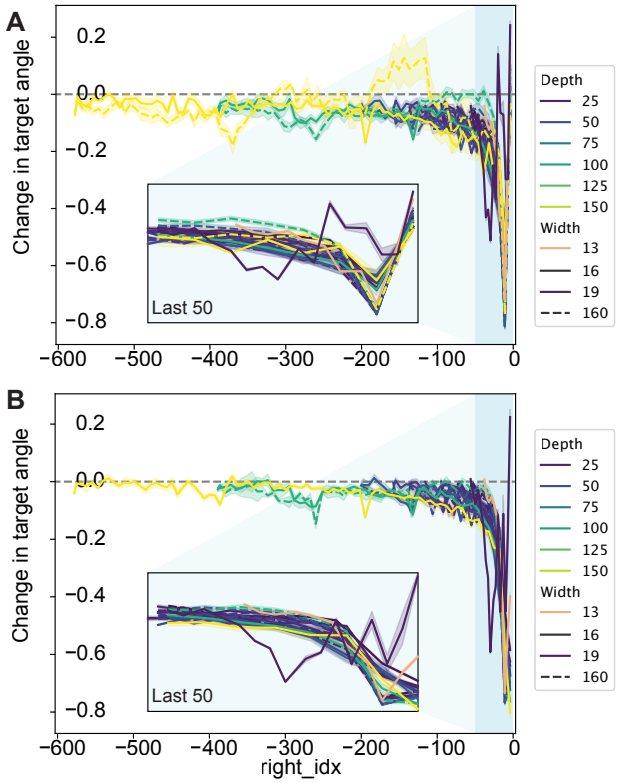

Figure E.10. Analysis of **target angles** (see Figure 1E. Angles close to zero mean that a layer is pointing in the direction of the targets. Here, we calculated the change in target angle for all layers before and after learning, and plotted as a function of the layer index *relative to the targets*. Consistent with the low dimensional MDS+PCA visualizations, we see that all models make *slight* progress in the direction of the targets in the early layers, and much more dramatic progress towards the targets in the last few layers. Insets zoom on the final few steps. (Note that the indices on the x-axis use a different convention than earlier analyses. Here, indices correspond to raw network components like Conv, ReLU, BatchNorm, etc., rather than residual blocks, and residual blocks consist of many individual components. The largest model is a ResNet-164 and has 81 residual blocks. The inset covers about 10 block-sized "steps" per model.)