NEURAL NETWORKS AS PATHS THROUGH THE SPACE OF REPRESENTATIONS

Anonymous authors
Paper under double-blind review

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. We consider a simple hypothesis for interpreting the layer-by-layer construction of useful representations: perhaps the role of each layer is to reformat information to reduce the "distance" to the desired outputs. With this framework, the layer-wise computation implemented by a deep neural network can be viewed as a path through a high-dimensional representation space. We formalize this intuitive idea of a "path" by leveraging recent advances in *metric* representational similarity. We extend existing representational distance methods by computing geodesics, angles, and projections of representations, going beyond mere layer distances. We then demonstrate these tools by visualizing and comparing the paths taken by ResNet and VGG architectures on CIFAR-10. We conclude by sketching additional ways that this kind of representational geometry can be used to understand and interpret network training, and to describe novel kinds of similarities between different models.

Keywords: representational similarity, metric spaces, geometry, neural network expressivity, visualization

1 INTRODUCTION

A core design principle of modern neural networks is that they process information serially, progressively transforming inputs 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 sets of simple units to construct more complicated functions is central to both 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 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, any 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. Formalizing this intuition requires a method to quantify if any two representations are "close" (similar) or "far" (dissimilar) (Kriegeskorte, 2009; Kornblith et al., 2019). In order to use this kind of spatial or geometric analogy for neural representations, we need some way to quantify the "distance" between representations. We build on recent work introducing *metrics* for quantifying representational dissimilarity (Williams et al., 2021; Shahbazi et al., 2021). Representational dissimilarity is quantified using a function $\mathrm{d}(\mathbf{X}, \mathbf{Y}) : \mathbb{X} \times \mathbb{X} \to \mathbb{R}^+$, where $\mathbb{X}$ is the space of all $m \times n$ matrices for fixed m and for all n, that takes as input two matrices of neural activity in response to m inputs, and returns a nonnegative measure of the dissimilarity of $\mathbf{X}$ and $\mathbf{Y}$. 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.

What are desirable properties of such a representational dissimilarity function? 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 say that $\mathbf{X}$ and $\mathbf{Y}$ are

equivalent representations 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 dissimilarity function that satisfies all of the above properties – equivalence, symmetry, and the triangle inequality – qualifies as a **metric**[1] on $\mathbb{X}$ (Burago et al., 2001). Examples of metrics between neural representations were recently developed independently by Williams et al. (2021) and Shahbazi et al. (2021).

Not all metrics are sufficient for the kind of spatial reasoning about representations we are interested in – 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 as a tool 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), it is perhaps unsurprising that this condition is 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 requires that points live on a smooth manifold (Burago et al., 2001). Each restriction on the type of metric comes with additional structure that we can use to inspect and visualize neural representations: rectifiability 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.

The main contributions of this paper are

- We put forward the spatial "path" analogy for deep neural networks, and quantitatively evaluate it using recently developed methods for representational distance.
- We develop a toolbox for analyzing geometric properties of sequences of neural network layers, and show how existing representational distance measures imply a rich set of geometric concepts beyond mere pairwise distance.
- We apply these techniques to compare paths taken by wide and deep residual networks, as well as four VGG architectures, all trained on CIFAR-10, extending the results of Nguyen et al. (2021).

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 and 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, little 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. In our path framework, relatively narrow layers take shorter steps, and chaining them together increases the total achievable length, while wide layers are capable of taking a single large step, and all such steps contribute by moving "closer" to the targets. Our work is a first step, so to speak, towards formalizing this notion of composing simple functions in *geometric* terms.

[1] It would be more precise to say 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. For a more thorough treatment of equivalence classes, see (Williams et al., 2021).

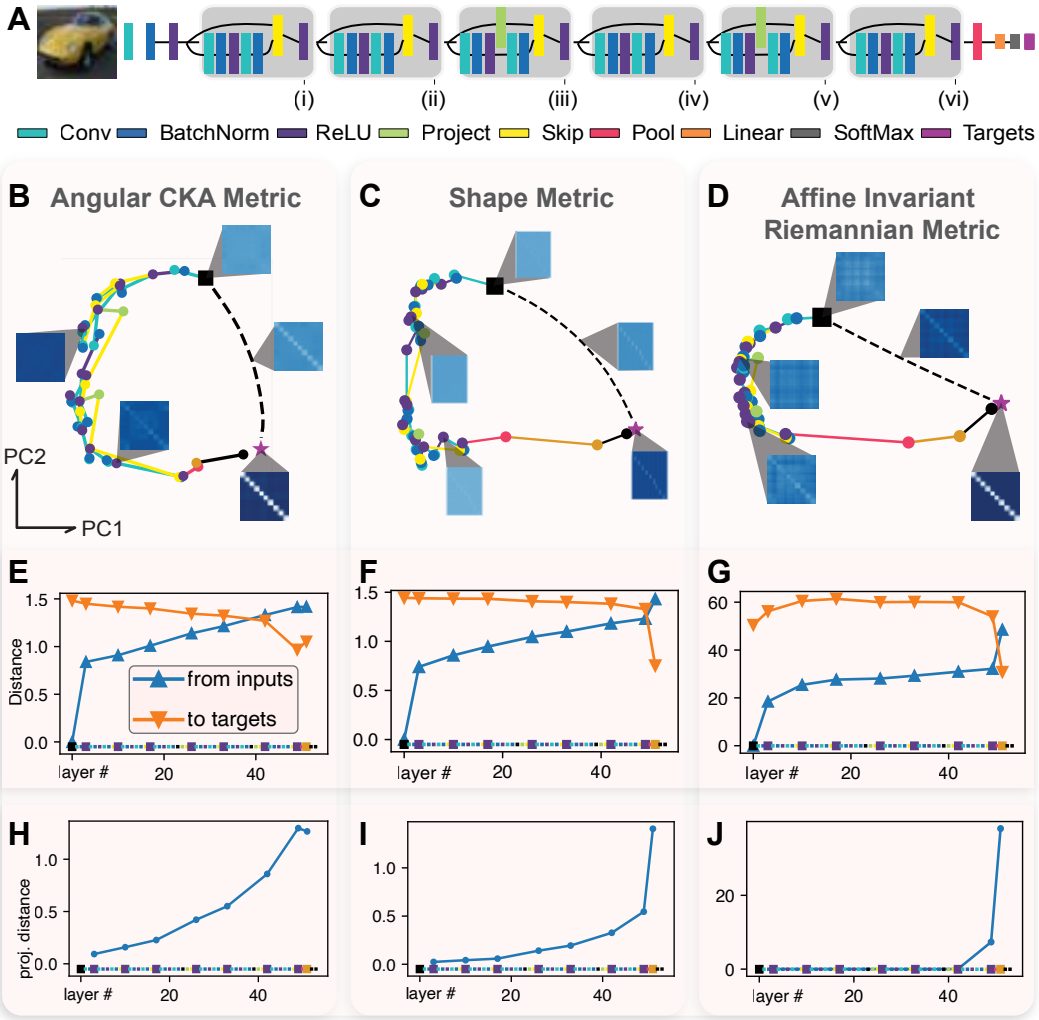

Figure 1: **A)** Schematic of Resnet-14 architecture with color-coded layers. **B, C, D)** 2D visualization of the high-dimensional "representation space," where each colored point along the path corresponds to a layer output in the model in **(A)**. The black square corresponds to input pixels, the purple star corresponds to output categories, and the black dashed line represents the geodesic from inputs to labels. Each column of the figure (**B, C, D**) visualizes the same neural network's path according to three different distance metrics: angular CKA (Williams et al., 2021) **(B)**, the angular shape metric (Williams et al., 2021) **(C)**, and the affine invariant Riemannian metric (Shahbazi et al., 2021) **(D)**. Inset matrices show matrices of neural data converted points in the corresponding metric space of each distance metric. Each panel shows five matrices: inputs, targets, two hidden layers, and the hypothetical representation at the midpoint of the geodesic between inputs and targets. **E, F, G)** Each panel shows the representational distance from inputs to hidden layers (▲) and from hidden layers to targets (▼). **H, I, J)** Distance from inputs to hidden layer representations after *projecting* each hidden representation onto the input-label geodesic.

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 and Granger, 2017; Hénaff et al., 2019; Kriegeskorte and 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 – which we elaborate on in section 3.2 below. Separately, both Williams et al. (2021) and Shahbazi et al. (2021) developed methods to compute Riemannian metrics between neural representations. We unify and extend their work by computing not just pairwise distance functions, but by also introducing a suite of tools for analyzing the geometry of representation-space induced by each of these distance functions. Finally, unlike Williams et al. (2021) and Shahbazi et al. (2021), 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

In our notation, a deep neural network processes an input $\mathbf{x}^0$ through a sequence of hidden layers, $\{\mathbf{x}^1, \ldots, \mathbf{x}^{L-1}\}$, and produces outputs $\mathbf{x}^L$, trained to match some target outputs (LeCun et al., 2015). In the common example of image classification, $\mathbf{x}^0$ is the input image, targets are the class label, and $\mathbf{x}^L$ a set of (log) class probabilities. We will use subscripts to index inputs, so $\mathbf{x}_i^l$ is the n_l-dimensional response of the lth hidden layer to the ith input, $\mathbf{x}_i^0$, with $i \in \{1, \ldots, m\}$. We use capital letters like $\mathbf{X}^l$, $\mathbf{X}^k$, or $\mathbf{Y}$ to refer to $m \times n$ matrices of neural representations in response to m inputs[2], and we treat target labels as one-hot vectors (i.e. if $\mathbf{Y}$ is the target representation, then it is a $m \times 10$ matrix for the CIFAR-10 dataset (Krizhevsky, 2009)). We will adopt the convention that $l < k$ if $\mathbf{x}^l$ is a direct ancestor of $\mathbf{x}^k$ through the layers of a network (layers will in general be partially but not strictly ordered). We are interested in functions $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ for that quantify the "dissimilarity" between two neural representations; when $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ satisfies the four conditions of a length metric – equivalence, symmetry, triangle inequality, and rectifiability – we will say that it measures their "representational distance." Note that $\mathbf{X}$ and $\mathbf{Y}$ may be different layers of the same model or layers from different models evaluated on the same inputs. We use $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ to refer to the distance function between arbitrary representations $\mathbf{X}$ and $\mathbf{Y}$, and $\mathrm{d}(\mathbf{X}^l, \mathbf{X}^k)$ when the layer indices l and k are contextually important.

Comparing representations is challenging because we often want to compare representations of different sizes, i.e. where $n_x \neq n_y$. Using $\mathbb{X}$ to denote the set of all $m \times n$ matrices (for all n), all representational distance metrics we consider here solve this problem using a two-stage approach: first, $\mathbf{X}$ and $\mathbf{Y}$ are mapped to a common space $\mathbb{M}$ 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 with metric $\mathrm{d}_{\mathbb{M}}$. Equivalence relations on $\mathbb{X}$ can be built into this two-stage approach in either stage. For instance, $\mathrm{d}(\mathbf{X}, \mathbf{Y})$ can be made invariant to changes in the scale of $\mathbf{X}$ either by imposing a canonical scale in f, or by preserving scale in f but using a scale-invariant metric $\mathrm{d}_{\mathbb{M}}$.

We implemented three existing families of representational distance, each of which can be understood as different choices for f, $\mathbb{M}$, and $\mathrm{d}_{\mathbb{M}}$, and extended them to further compute various geometric quantities including geodesics, projections, tangent vectors, and angles. The representational distance metrics we investigate here are **Angular CKA** (with a linear kernel), **Shape Metrics** (with angular distance), and the **Affine Invariant Riemannian Metric** (with a squared exponential kernel and ridge regularization) (Williams et al., 2021; Shahbazi et al., 2021). Table A.1 and Appendix A provide mathematical details on each.

To get an initial impression of each metric's behavior, Figure 1B-D shows a 2D visualization of the path taken by a Resnet-14 (architecture in Figure 1A). Following the suggestion of Williams et al. (2021), we generated these low-dimensional path visualizations by first computing pairwise

[2]We assume convolutional layers are flattened, in which case n is the product of height, width, and feature dimensions.

distances between all layers, as well as 15 equally-spaced points along the input-target geodesic. We then used multidimensional scaling to find approximate Euclidean coordinates for each layer and each point on the geodesic. Figure 1B-D shows the top two principal components of these Euclidean spaces, and provides a quick visual impression of the main features of the path taken by the model according to each metric. We emphasize that we use MDS only for these initial visualizations, and that all subsequent calculations are done using the native geometry of each metric. For Angular CKA in Figure 1B, the metric space $\mathbb{M}$ is the space of $m \times m$ centered and normalized Gram matrices). For the Shape Metric in Figure 1C, $\mathbb{M}$ is the space of $m \times p$ matrices after selecting only the top p principal components of $\mathbf{X}$. For the Affine Invariant Riemannian metric in Figure 1D, $\mathbb{M}$ is the space of $m \times m$ Gram matrices, which we further regularize to ensure positive semi-definiteness as required by the metric (Pennec, 2006; 2019).

2.3 From distance metrics to geometry

Our main methodological contribution is recognizing that any choice of representational distance metric implies rich geometric structure on the space of representations that goes beyond pairwise distances; representations are not merely separated by some distance, but also by some *direction*. We are releasing a standalone PyTorch (Paszke et al., 2019) library to compute all of these metrics and their associated geometric quantities, available at [redacted link during review process].

Interpolating along geodesics. The rectifiability property of distance metrics implies that we can instantiate points in $\mathbb{M}$ that *interpolate* $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$ along a shortest path between them. That is, for any $t \in [0, 1]$, we can construct a point $\tilde{\mathbf{Z}} = \texttt{geodesic}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, t)$ such that $\mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}}) + \mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{Z}}, \tilde{\mathbf{Y}}) = \mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ and $\mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Z}}) = t\, \mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$. (Shortest paths are unique in all cases we consider here, but in general there may be more than on $\tilde{\mathbf{Z}}$ that satisfies these expressions). This grants us the ability to interpret neural network layers not as discrete jumps from one representation to another, but as a piece-wise smooth curve from inputs to network outputs. In Figure 1C, we used this property to instantiate points along the shortest path connecting input images directly to output labels, and to compare this with the path taken by the network.

Projection. Points in $\mathbb{M}$ can be *projected* onto the geodesic spanning any other two points. This follows from the existence of geodesics: 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 $\texttt{geodesic}(\tilde{\mathbf{Y}}, \tilde{\mathbf{Z}}, t)$ with respect to t. In Figure 1H-J, we use this property to project each layer of a Resnet-14 onto the geodesic connecting inputs to targets. Compared with measuring raw distance from inputs to hidden layers or from hidden layers to outputs (Figure 1E-J), this measure of projected distance from inputs provides a useful 1-dimensional summary of the network's progress through the layers.

Angles. Any triplet of points in $\mathbb{M}$ defines an angle. We use $\theta(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, \tilde{\mathbf{Z}})$ to denote the angle formed by the $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ and $(\tilde{\mathbf{Z}}, \tilde{\mathbf{Y}})$ geodesics incident at $\tilde{\mathbf{Y}}$. We use this to quantify the **internal angle** along the network's path at layer l, $\theta(\tilde{\mathbf{X}}^j, \tilde{\mathbf{X}}^k, \tilde{\mathbf{X}}^l)$ – this measures the extent to which the steps from layer j to k and from k to l lie along a *straight* path according to a given metric.

A path through neural network layers may in principle be highly curved locally, but relatively straight when zooming out and considering a block consisting of multiple layers as a single step. For this reason, we treat each residual block as a single path step.

3 Experiments

3.1 How paths change over training

Whereas Figure 1 shows properties of the path taken by a well-trained Resnet-14 model, here we ask how properties of its path emerge over training. For this, we trained a new model of the same architecture, and logged model parameters every 10 batches. Since the vast majority of changes occur in the first epoch, Figure 2 shows how distances, projected distances, and internal angles of the model's path change for every 10 steps in the first epoch, plus an additional line after 20 epochs to show where values converge to.

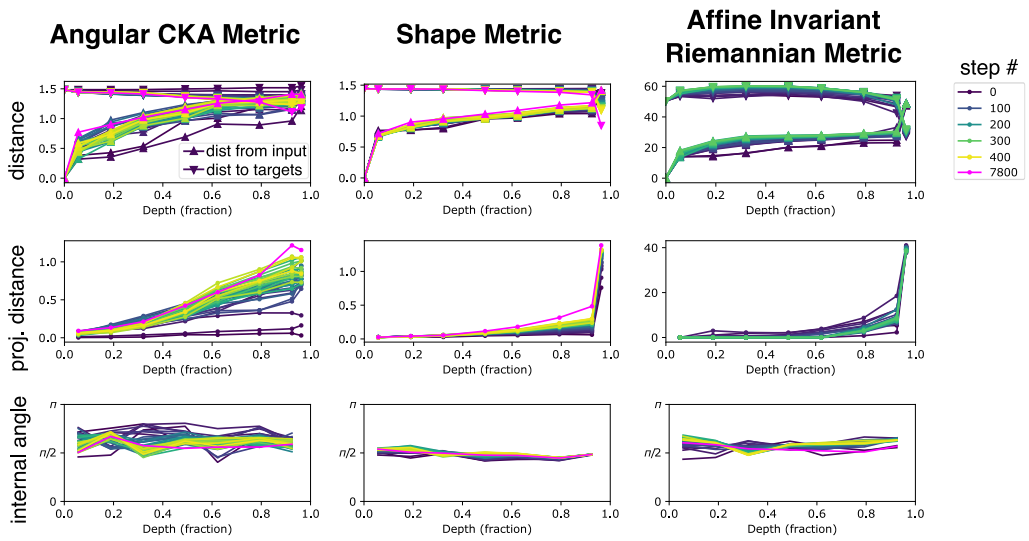

Figure 2: How paths change over training in a Resnet-14 according to each of three metrics (columns are different metrics). We show a line every 10 batches for the first epoch (blue - yellow color scale), and a magenta line for the result after 20 epochs. **Top row:** As in figure 1E-G, we plot the distance of each layer representation from the inputs (▲) and from each layer to the targets (▼). **Middle row:** As in Figure 1H-J, we project each layer of the path onto the geodesic from inputs to target. **Bottom row:** We compute the **internal angles** along triplets of adjacent residual blocks on the neural network's path. The closer this is to π, the more "straight" the network's path is.

As in Figure 1, the results for Angular CKA are the clearest and easiest to interpret. As expected, the network at initialization takes a path *somewhere*, but this path is originally not in the direction of the targets: the distance from inputs initially increases through the layers, while the distance to targets does not initially decrease (Figure 2, first row). In comparison, the *projected progress* paints a clearer picture of the gradual process of moving closer to the targets as training progresses (Figure 2, middle row): at initialization the path makes no progress, but by the time the first epoch completes (step 400), the net progress covers most of the distance to the targets. Results for the other metric are similar, but difficult to see because their distance is dominated by the last layers of the network.

We hypothesized that training would cause the network's path to become more *straight*, but surprisingly we found the opposite to be true. In the bottom row of Figure 2, we show the **internal angle** between steps along the path (one step is one residual block). According to the Angular CKA metric, internal angles begin slightly above $\pi/2$ at initialization, and concentrate towards $\pi/2$ as training progresses. This means that steps become *more orthogonal* through training. Results for the other metrics agree that steps are largely orthogonal, though they show no clear pattern over training.

3.2 Comparing paths of wide versus deep models

We replicate and extend the results of Nguyen et al. (2021) by comparing paths taken by "wide" versus "deep" models. In Figure 3, we show the same quantities we computed in Figure 2, but comparing different Resnet widths and depths. Our results confirm the main findings of Nguyen et al. (2021), namely that wide and deep models compute similar functions — in other words, wide-model paths and deep-model paths all follow roughly the same trajectory in representational space. This is not universally true of all models – we also compare with VGG, and see salient differences between the VGG and Resnet architectures.

To investigate how longer or shorter paths differ in their movement towards the target, we further decompose each layer's progress into the two orthogonal components that result when projecting each path segment onto the geodesic (Figure 4A). The components are "progress": the distance travelled by the segment along the geodesic from the current position to the targets; and "deviation":

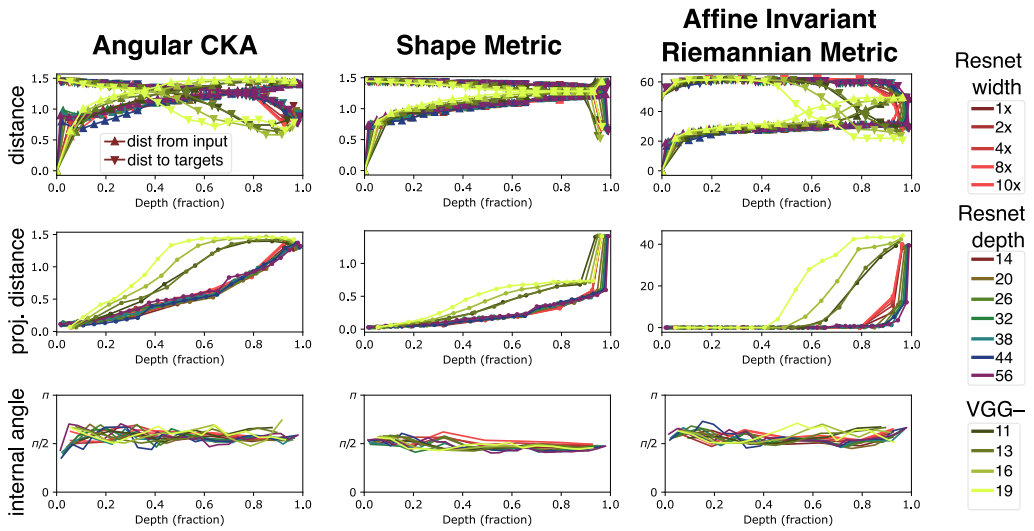

Figure 3: How paths change between different architectures according to each of the three metrics (columns). We compare residual networks by width (red shades) and depth (hues), as well as VGG networks by size (yellow/green shades). **Top row:** distance of each layer representation from the inputs (▲) and to targets (▼), as in Figure 1E-G. **Middle row:**, projection of each layer of onto the geodesic from inputs to target, as in Figure 1H-J. **Bottom row:** internal angles along triplets of adjacent residual blocks on the network's path, where π indicates a "straight" path.

the distance by which the segment moves away from this geodesic. In this case the geodesic is the shortest path from the network's *current* position to the targets, and differs for each segment.

Figure 4B plots the mean and standard deviation of the progress and deviation made by each path segment of a variety of residual and VGG network architectures. We observe that deep networks have significantly less deviation and slightly less progress per step compared with shallow networks. Among residual networks of equal depth, wide networks have less deviation per segment than narrow networks. This may be due to the regularizing effect of increased network width. Although VGG networks follow similar trends, they form their own distinct cluster on the top right of the figure (yellow points). VGG networks have overall higher ratio of progress to deviation than residual networks, which might be introduced by the shortcut connection and difference in the block design. In short, we find that in both convolutional based networks, even only in residual networks, different depth and width brings networks a different progress and deviation in their representation path.

4 Discussion

In this work, we interpret neural networks spatially as paths from inputs to outputs through a space of intermediate representations. These paths have rich geometric structure, inherited from computing distances between representations, and we can use that structure to build intuitions about network structure, training, and comparisons between models.

We investigated three families of representational distance metrics — Angular CKA, Shape Metrics, and the Affine Invariant metric on symmetric positive definite matrices (Williams et al., 2021; Shahbazi et al., 2021). Surprisingly, we found that trained networks take circuitous paths according to all of these metrics, and deviate far from the shortest paths from inputs to targets which are defined geometrically by each representational distance metric. 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 our metrics may not reflect the distance between representations in terms of ease of computation. 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.

We were also surprised to discover that according to all metrics we investigated, network paths tended to be *jagged*, and consist predominantly of 90° angle turns. 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 orthogonal during training. This is surprising because the training objective ostensibly should encourage all layers to point in the same direction towards the targets. This empirical finding 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.

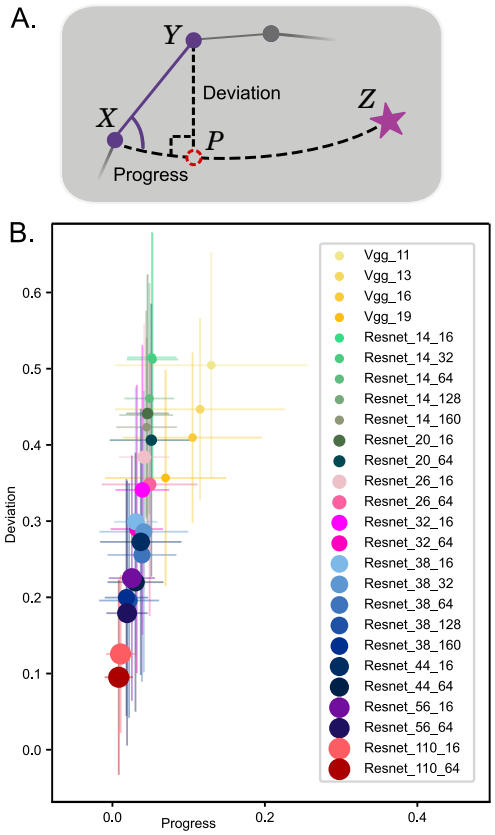

Figure 4: **A)** We decomposed each path step into a "progress" part – along the geodesic towards targets – and an orthogonal "deviation" part and measured the length of each. **B)** Average "progress" and "deviation" of all layers per model; error bars show standard deviation across layers. For Resnets, dot hue indicates depth and marker size width. VGG networks of varying depths are in yellow.

As described in the introduction, we are motivated to develop a general spatial analogy for information-processing, where complex transformations of representations cover more "distance" than simple ones. In this sense, a measure of representational distance ought to reflect the *function complexity* of transforming $\mathbf{X}$ into $\mathbf{Y}$. 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. An exciting avenue for future work is thus to derive a measure of representational distance directly from a measure of function complexity. Such a measure will likely violate the axioms of a distance metric. For example, 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. Notably, many of the geometric properties studied here (shortest paths, angles, projection, etc.) can be extended to the asymmetric case using the theory of Finsler rather than Riemannian manifolds (Burago et al., 2001).

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.

5 Reproducibility Statement

We have made an effort to make all of our code shareable and reproducible. Our model specifications, training recipe, and data augmentation are all based on standard research practice: we use default training options for all models taken from the OpenLTH framework. Our standalone PyTorch library for metric representational distance, along with example usage and unit tests to verify its performance, is available at [redacted github link], and our repository to train and analyze models specific to this paper is available at [redacted github link].

REFERENCES

David GT Barrett, Ari S. Morcos, and Jakob H. Macke. Analyzing biological and artificial neural networks: challenges with opportunities for synergy? *Current Opinion in Neurobiology*, 55: 55–64, 2019.

Dimitri Burago, Yuri Burago, and Sergei Ivanov. *A Course in Metric Geometry*. American Mathematical Society, 2001.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13:795–828, 2012.

Jörn Diedrichsen and Nikolaus Kriegeskorte. Representational models: A common framework for understanding encoding, pattern-component, and representational-similarity analysis. *PLoS Computational Biology*, 13(4):1–33, 2017.

Shimon Edelman. Representation is representation of similarities. *Behavioral and Brain Sciences*, 21(4):449–498, 1998.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In S. Jain, H. U. Simon, and E. Tomita, editors, *Lecture Notes in Artificial Intelligence*, volume 3734, pages 63–77. Springer-Verlag, Berlin, 2005.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CVPR*, 2016.

Olivier J. Hénaff, Robbe L.T. Goris, and Eero P. Simoncelli. Perceptual straightening of natural videos. *Nature Neuroscience*, 22(6):984–991, 2019.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Frank Jäkel, Bernhard Schölkopf, and Felix A. Wichmann. Similarity, kernels, and the triangle inequality. *Journal of Mathematical Psychology*, 52(5):297–303, 2008.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of Neural Network Representations Revisited. *ICML*, 36, 2019.

Nikolaus Kriegeskorte. Relating population-code representations between man, monkey, and computational models. *Frontiers in Neuroscience*, 3(3):363–373, 2009.

Nikolaus Kriegeskorte. Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annual Review of Vision Science*, 1(1):417–446, 2015.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Nina Miolane, Nicolas Guigui, Alice Le Brigant, Johan Mathe, Benjamin Hou, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, Hatem Hajri, Yann Cabanes, Thomas Gerald, Paul Chauchat, Christian Shewmake, Daniel Brooks, Bernhard Kainz, Claire Donnat, Susan Holmes, and Xavier Pennec. Geomstats: A python package for riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9, 2020.

Esfandiar Nava-Yazdani, Hans-Christian Hege, Timothy John Sullivan, and Christoph von Tycowicz. Geodesic analysis in kendall's shape space with epidemiological applications. *Journal of Mathematical Imaging and Vision*, 62(4):549–559, 2020.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do Wide and Deep Networks Learn the Same Things? Uncovering How Neural Network Representations Vary with Width and Depth. *ICLR*, pages 1–25, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Xavier Pennec. Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, 2006.

Xavier Pennec. *Manifold-valued image processing with SPD matrices*. Elsevier Ltd, 2019.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *arXiv*, 2016.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Soh. On the Expressive Power of Deep Fully Circulant Neural Networks. *ICML*, 70:2847–2854, 2017.

Blake A. Richards, Timothy P. Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, Colleen J. Gillon, Danijar Hafner, Adam Kepecs, Nikolaus Kriegeskorte, Peter Latham, Grace W. Lindsay, Kenneth D. Miller, Richard Naud, Christopher C. Pack, Panayiota Poirazi, Pieter Roelfsema, João Sacramento, Andrew Saxe, Benjamin Scellier, Anna C. Schapiro, Walter Senn, Greg Wayne, Daniel Yamins, Friedemann Zenke, Joel Zylberberg, Denis Therien, and Konrad P. Kording. A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770, 2019.

Antonio M Rodriguez and Richard Granger. The differential geometry of perceptual similarity. *arXiv preprint arXiv:1708.00138*, 2017.

David Rolnick and Max Tegmark. The power of deeper networks for expressing natural functions. *ICLR*, pages 1–14, 2017.

David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. IEEE New York, 1988.

Mahdiyar Shahbazi, Ali Shirali, Hamid Aghajan, and Hamed Nili. Using distance on the Riemannian manifold to compare representations in brain and in models. *NeuroImage*, 239(June):118271, 2021.

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

O. Tange. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1): 42–47, Feb 2011.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

Alex H Williams, Erin Kunz, Simon Kornblith, and Scott Linderman. Generalized shape metrics on neural representations. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4738–4750. Curran Associates, Inc., 2021.

A DETAILED INFORMATION ON METRICS

Distance Metric	**Manfiold** $\mathbb{M}$	**Distance** $\mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$
Angular CKA†	$Gram_m$	$\arccos\left(\frac{\langle\tilde{\mathbf{X}},\tilde{\mathbf{Y}}\rangle_{\mathrm{F}}}{\sqrt{\langle\tilde{\mathbf{X}},\tilde{\mathbf{X}}\rangle_{\mathrm{F}}\langle\tilde{\mathbf{Y}},\tilde{\mathbf{Y}}\rangle_{\mathrm{F}}}}\right)$
Angular Shape†	$\mathbb{R}^{m\times p}$	$\min_{\boldsymbol{R}} \arccos\left(\frac{\langle\tilde{\mathbf{X}},\tilde{\mathbf{Y}}\boldsymbol{R}\rangle_{\mathrm{F}}}{\sqrt{\langle\tilde{\mathbf{X}},\tilde{\mathbf{X}}\rangle_{\mathrm{F}}\langle\tilde{\mathbf{Y}},\tilde{\mathbf{Y}}\rangle_{\mathrm{F}}}}\right)$
Euclidean Shape†	$\mathbb{R}^{m\times p}$	$\min_{\boldsymbol{R}} \frac{1}{m}\sum_{i=1}^{m} \lvert\lvert\tilde{\mathbf{X}}_i - \tilde{\mathbf{Y}}_i\boldsymbol{R}\rvert\rvert_2$
Affine Invariant Riemannian	Sym_m^+ or Sym_p^+	$\sum_i \log(d_i)^2$ where d_i is the ith eigenvalue of $\tilde{\mathbf{X}}^{-\frac{1}{2}}\tilde{\mathbf{Y}}\tilde{\mathbf{X}}^{-\frac{1}{2}}$

Table A.1: Summary of representational distance metrics. $Gram_m$ is the set of $m \times m$ centered Gram matrices. Sym_m^+ is the set of $m \times m$ symmetric positive definite matrices. $Dist_m$ is the set of $m \times m$ pairwise-distance matrices. p is a hyperparameter used to set the dimensionality of points in $\mathbb{M}$. $||\mathbf{X}||_{\mathrm{F}}$ denotes the Frobenius norm, and $\langle\mathbf{X},\mathbf{Y}\rangle_{\mathrm{F}}$ denotes the Frobenius inner-product. $\boldsymbol{R}$ is a $p\times p$ orthonormal matrix, found by solving the orthogonal Procrustes problem to maximally align $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$. Metrics marked with "†" are due to Williams et al. (2021). The Affine Invariant Riemannian metric is due to Pennec (2006; 2019); Shahbazi et al. (2021). For further details on each metric and its implied geometry, see Appendix A.

As summarized in equation (1), all distance metrics between neural representations 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 $\alpha \neq 0$. 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 $\boldsymbol{A}$.

For all geometric calculations, we drew a subset of $m = 1000$ items randomly from the test set, and used the same subset throughout. We verified that $m = 1000$ was sufficient to reliably estimate the geometric quantities of interest by randomly resampling $m = 1000$ points from the test set multiple times and inspecting the variance of computed quantities (Figure A.1). Targets (class labels) are converted to 1-hot vectors before embedding.

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 $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$.
- The Angular Shape metric with $m = 1000$, $p = 100$, $\alpha = 0$.
- The Gram-matrix version of the Affine-Invariant Riemannian metric with $m = 1000$, $\epsilon = 0.1$, and a squared exponential kernel with length scale set to the median Euclidean distance of $\mathbf{X}$.

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. Note that both $\boldsymbol{G}$ and $\tilde{\boldsymbol{G}}$ are symmetric and positive definite matrices. 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 $\left\langle \tilde{\boldsymbol{G}}_{\mathbf{X}}, \tilde{\boldsymbol{G}}_{\mathbf{X}} \right\rangle_{\mathrm{F}} = 1$, where $\langle \boldsymbol{A}, \boldsymbol{B} \rangle_{\mathrm{F}} = \mathrm{Tr}(\boldsymbol{A}^\top \boldsymbol{B})$ is the Frobenius inner-product.

The **embedding function** for Angular CKA is $\mathrm{f}(\mathbf{X}) = \tilde{\boldsymbol{G}}_{\mathbf{X}}$. Distance according to Angular CKA is equal to arc length on the sphere consisting of centered and normalized Gram matrices:

$$\begin{aligned} \mathrm{d}(\mathbf{X}, \mathbf{Y}) &= \mathrm{d}_{\mathbb{M}}(\mathrm{f}(\mathbf{X}), \mathrm{f}(\mathbf{Y})) \\ &= \mathrm{d}_{\mathbb{M}}(\tilde{\boldsymbol{G}}_{\mathbf{X}}, \tilde{\boldsymbol{G}}_{\mathbf{Y}}) \\ &= \arccos\left(\left\langle \tilde{\boldsymbol{G}}_{\mathbf{X}}, \tilde{\boldsymbol{G}}_{\mathbf{Y}} \right\rangle_{\mathrm{F}}\right) . \end{aligned} \tag{A.1}$$

Because Angular CKA is an arc length, its geodesics lie along great circles $\tilde{\boldsymbol{G}}$-space. We therefore can compute points along the geodesic in closed-form using the SLERP formula:[3]

$$\mathrm{geodesic}(\tilde{\boldsymbol{G}}_{\mathbf{X}}, \tilde{\boldsymbol{G}}_{\mathbf{Y}}, t) = \frac{\sin((1-t)\Omega)}{\sin(\Omega)}\tilde{\boldsymbol{G}}_{\mathbf{X}} + \frac{\sin(t\Omega)}{\sin(\Omega)}\tilde{\boldsymbol{G}}_{\mathbf{Y}} , \tag{A.2}$$

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

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 $\tilde{\boldsymbol{G}}_{\mathbf{X}}$ to $\tilde{\boldsymbol{G}}_{\mathbf{Y}}$ is a tangent vector (symmetric matrix) at $\tilde{\boldsymbol{G}}_{\mathbf{X}}$ given by

$$\log_{\tilde{\boldsymbol{G}}_{\mathbf{X}}}\left(\tilde{\boldsymbol{G}}_{\mathbf{Y}}\right) = \boldsymbol{W}\arccos\left(\left\langle \tilde{\boldsymbol{G}}_{\mathbf{X}}, \tilde{\boldsymbol{G}}_{\mathbf{Y}} \right\rangle_{\mathrm{F}}\right) \tag{A.3}$$

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

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

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

$$\exp_{\tilde{\boldsymbol{G}}_{\mathbf{X}}}(\boldsymbol{W}) = \cos\left(||\boldsymbol{W}||_{\mathrm{F}}\right)\tilde{\boldsymbol{G}}_{\mathbf{X}} + \mathrm{sinc}\left(||\boldsymbol{W}||_{\mathrm{F}}\right)\boldsymbol{W} \tag{A.4}$$

where $\mathrm{sinc}(x) = \frac{\sin(x)}{x}$.

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 $\mathbf{X}$ to $\mathbf{Y}$, i.e. the result of $\log_{\tilde{\boldsymbol{G}}_{\mathbf{X}}}(\tilde{\boldsymbol{G}}_{\mathbf{Y}})$. Then,

$$\theta(\tilde{\boldsymbol{G}}_{\mathbf{X}}, \tilde{\boldsymbol{G}}_{\mathbf{Y}}, \tilde{\boldsymbol{G}}_{\mathbf{Z}}) = \arccos\left(\frac{\langle \boldsymbol{W}_{\mathbf{YX}}\boldsymbol{W}_{\mathbf{YZ}} \rangle_{\mathrm{F}}}{||\boldsymbol{W}_{\mathbf{YX}}||_{\mathrm{F}}||\boldsymbol{W}_{\mathbf{YZ}}||_{\mathrm{F}}}\right) \tag{A.5}$$

is the angle of the $\mathbf{XYZ}$ triangle.

[3] https://en.wikipedia.org/wiki/Slerp

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

$$\boldsymbol{G}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(||\mathbf{x}_i - \mathbf{x}_j||_2^2/\tau^2\right) \tag{A.6}$$

is naturally shift- and rotation-invariant, and it can me made further 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

$$\tilde{\mathbf{X}} = \mathrm{f}(\mathbf{X}) = \begin{cases} \text{whiten}([\bar{\mathbf{X}},\, \mathbf{0}], \alpha) & \text{if } n \leq p \\ \text{whiten}(\bar{\mathbf{X}}\boldsymbol{U}_{:p}, \alpha) & \text{if } n > p \end{cases} \tag{A.7}$$

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 eigendecomposition 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 $\alpha = 0$. We use $\alpha = 0$ because this entails further invariances, making metrics more interpretable across disparate layer shapes and types (see section A.2.1 below for details on shape metric 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

$$\mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = \arccos\left(\frac{\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}\boldsymbol{R}\right\rangle_{\mathrm{F}}}{||\tilde{\mathbf{X}}||_{\mathrm{F}}||\tilde{\mathbf{Y}}||_{\mathrm{F}}}\right) . \tag{A.8}$$

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

$$\mathrm{d}_{\mathbb{M}}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = \frac{1}{m}\sum_{i=1}^{m} ||\tilde{\mathbf{X}}_i - \tilde{\mathbf{Y}}_i\boldsymbol{R}|| . \tag{A.9}$$

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.2):

$$\mathrm{geodesic}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, t) = \frac{\sin((1-t)\Omega)}{\sin(\Omega)}\tilde{\mathbf{X}} + \frac{\sin(t\Omega)}{\sin(\Omega)}\tilde{\mathbf{Y}}\boldsymbol{R} , \tag{A.10}$$

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.5), 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

$$\text{horizontal } \log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}}) = \tilde{\mathbf{Y}}\boldsymbol{R} - \tilde{\mathbf{X}} \tag{A.11}$$

in the Euclidean case, or

$$\text{horizontal } \log_{\tilde{\mathbf{X}}}(\tilde{\mathbf{Y}}) = \boldsymbol{W}\arccos\left(\frac{\left\langle \tilde{\mathbf{X}}, \tilde{\mathbf{Y}}\boldsymbol{R}\right\rangle_{\mathrm{F}}}{||\tilde{\mathbf{X}}||_{\mathrm{F}}||\tilde{\mathbf{Y}}||_{\mathrm{F}}}\right) \tag{A.12}$$

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.8) and (A.9), $\boldsymbol{R}$ 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.7).
- The angular shape metric is **scale-invariant** because (A.8) 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.
- 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.7).

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 and 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

$$\mathrm{f}(\mathbf{X}) = \boldsymbol{G}_{\mathbf{X}} + \epsilon \mathbb{I}_m \tag{A.13}$$

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.6), 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.7), 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

$$\mathrm{f}(\mathbf{X}) = \begin{cases} \mathrm{cov}\left([\mathbf{X},\, \mathbf{0}]\right) + \epsilon \mathbb{I}_p & \text{if } n \leq p \\ \mathrm{cov}\left(\mathbf{X}\boldsymbol{U}_{:p}\right) + \epsilon \mathbb{I}_p & \text{if } n > p \end{cases} \tag{A.14}$$

(compare with (A.7)).

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

$$\mathrm{d}(\mathbf{X}, \mathbf{Y}) = \mathrm{d}_{\mathbb{M}}(\mathbf{P}, \mathbf{Q}) = \sum_i \log(d_i)^2 \tag{A.15}$$

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

$$\mathrm{geodesic}(\mathbf{P}, \mathbf{Q}, t) = \mathbf{P}^{\frac{1}{2}}\left(\mathbf{P}^{-\frac{1}{2}}\mathbf{Q}\mathbf{P}^{-\frac{1}{2}}\right)^t\mathbf{P}^{\frac{1}{2}} \tag{A.16}$$

(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

$$\log_{\mathbf{P}}(\mathbf{Q}) = \mathbf{P}^{\frac{1}{2}}\log\left(\mathbf{P}^{-\frac{1}{2}}\mathbf{Q}\mathbf{P}^{-\frac{1}{2}}\right)\mathbf{P}^{\frac{1}{2}} \tag{A.17}$$

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

$$\exp_{\mathbf{P}}(\boldsymbol{W}) = \mathbf{P}^{\frac{1}{2}}\exp\left(\mathbf{P}^{-\frac{1}{2}}\boldsymbol{W}\mathbf{P}^{-\frac{1}{2}}\right)\mathbf{P}^{\frac{1}{2}} \tag{A.18}$$

(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

$$\langle \boldsymbol{W}, \boldsymbol{V}\rangle_{\mathbf{P}} \equiv \left\langle \mathbf{P}^{-1}\boldsymbol{W}, \mathbf{P}^{-1}\boldsymbol{V}\right\rangle_{\mathrm{F}} = \mathrm{Tr}\left(\boldsymbol{W}^\top\mathbf{P}^{-\top}\mathbf{P}^{-1}\boldsymbol{V}\right) . \tag{A.19}$$

A.3.1 Invariances of Affine Invariant Riemannian Metric

We will treat the Gram matrix (A.13) and the covariance matrix (A.14) 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 MODELS AND TRAINING DETAILS

We trained a collection of convolutional networks including both residual networks (He et al., 2016) and VGG (Simonyan and Zisserman, 2014) on CIFAR-10 (Krizhevsky, 2009) using PyTorch (Paszke et al., 2019), managing compute jobs using GNU Parallel (Tange, 2011). We used the OpenLTH framework[4] 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 B.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 pixel left/right or up/down shifts (padding with zeros).

For the training analysis in Figure 2, we trained a second Resnet-14 with default width (16), and analyzed the representational distance and geometry once every 10 batches for the epoch, since the vast majority of the changes to the network performance and geometry occur in the first epoch.

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

Architecture (depth/width)	CIFAR-10 test accuracy (%)
Resnet 14/16	90.78
Resnet 14/32	92.67
Resnet 14/64	93.88
Resnet 14/128	94.57
Resnet 14/160	94.91
Resnet 20/16	91.28
Resnet 20/32	93.76
Resnet 20/64	94.84
Resnet 20/128	95.13
Resnet 20/160	95.22
Resnet 26/16	91.92
Resnet 26/32	93.99
Resnet 26/64	94.69
Resnet 26/128	95.24
Resnet 26/160	95.07
Resnet 32/16	93.04
Resnet 32/32	94.18
Resnet 32/64	94.83
Resnet 32/128	95.47
Resnet 32/160	94.79
Resnet 38/16	92.28
Resnet 38/32	94.07
Resnet 38/64	95.05
Resnet 38/128	94.71
Resnet 38/160	94.99
Resnet 44/16	93.06
Resnet 44/32	94.14
Resnet 44/64	95.34
Resnet 44/128	94.85
Resnet 44/160	94.91
Resnet 56/16	93.06
Resnet 56/32	94.46
Resnet 56/64	94.82
Resnet 56/128	95.22
Resnet 56/160	95.32
Resnet 110/16	93.37
Resnet 110/32	94.91
Resnet 110/64	94.87
Resnet 110/128	95.15
Resnet 110/160	95.02
VGG 11	91.93
VGG 13	93.37
VGG 16	93.45
VGG 19	93.33

Table B.2: Model architectures and performance.

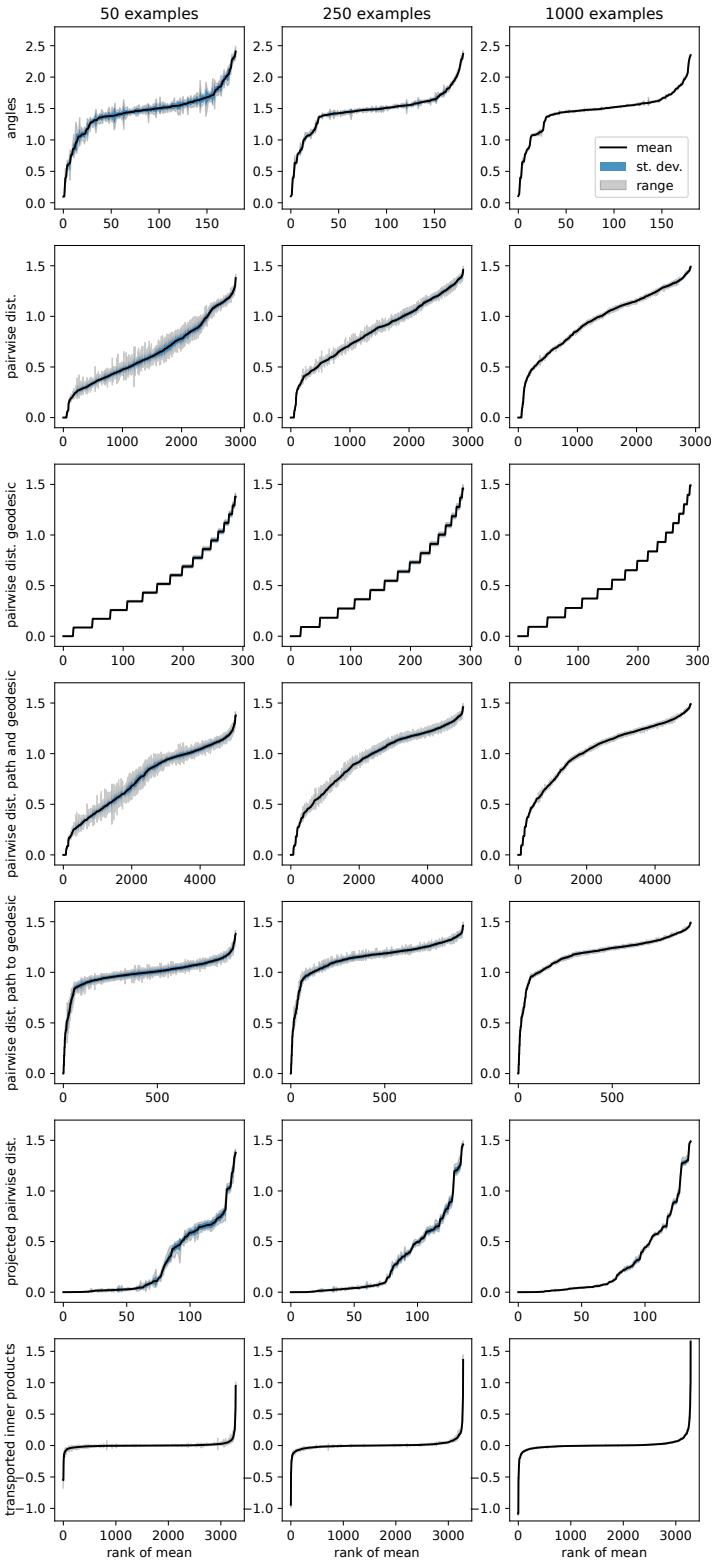

Figure A.1: 10-fold cross validation for quantities computed using the Angular CKA metric on random subsets of CIFAR-10 test examples. Rows correspond to different types of computed quantity, and columns correspond to the number of examples used. Quantities are ordered by the rank of their mean over CV folds (x-axis). Each plot shows the mean (line), standard deviation (blue region), and minimum to maximum range (grey region) per quantity over CV folds (y-axis).