Comparing Mapper Graphs of Artificial Neuron Activations

Youjia Zhou, Helen Jenne, Davis Brown, Madelyn Shapiro, Brett Jefferson,
Cliff Joslyn, Gregory Henselman-Petrusek, Brenda Praggastis, Emilie Purvine, Bei Wang

Abstract— The mapper graph is a popular tool from topological data analysis that provides a graphical summary of point cloud data. It has been used to study data from cancer research, sports analytics, neurosciences, and machine learning. In particular, mapper graphs have been used recently to visualize the topology of high-dimensional artificial neural activations from convolutional neural networks and large language models. However, a key question that arises from using mapper graphs across applications is how to compare mapper graphs to study their structural differences. In this paper, we introduce a distance between mapper graphs using tools from optimal transport. We demonstrate the utility of such a distance by studying the topological changes of neural activations across convolutional layers in deep learning, as well as by capturing the loss of structural information for a multiscale mapper.

Index Terms—Mapper graphs, hypergraphs, neuron activations, deep learning, optimal transport

1 Introduction

The mapper graph is a popular tool from topological data analysis that provides a graphical summary of point cloud data. First introduced by Singh et al. [67], it has been used to study data from cancer research [53], sports analytics [45], neuroscience [63], and machine learning [55, 58]. In particular, mapper graphs have been used recently to visualize the topology of high-dimensional artificial neural activations (i.e., outputs of neurons) from convolutional neural networks (CNNs) [55] and large language models (such as BERT and RoBERTa) [58].

However, a key question that arises from using mapper graphs across applications is how to compare these mapper graphs to study their structural differences. In this paper, we aim to address such a question. Our contributions include:

- We introduce distances between mapper graphs using existing tools from optimal transport. We consider these distances to be well suited to compare mapper graphs of neuron activations, as they are both easily *computable* and *interpretable* (Section 4).
- We demonstrate the utility of such a distance by studying the topological changes of neural activations across convolutional layers of CNNs (Section 5).
- We show that such distances are additionally applicable for capturing the loss of structural information for a sequence of mapper graphs that arise from a multiscale mapper [25] (Section 6).

We expect our framework to be applicable beyond deep learning whenever sequences of mapper graphs arise from data, such as shape analysis and scientific visualization in material sciences.

2 Related Work

Mapper in data analysis and visualization. The mapper graph is the 1-dimensional skeleton of the classic mapper construction designed to summarize high-dimensional data [67]. It is a data analysis and visualization tool by design, and has seen widespread applications.

A key strength of the mapper graph is that it is coordinate and deformation invariant [16], making it suitable for examining data across participants or subgroups [64]. In neuroscience and biomedicine, the mapper graph has been used to study genetic data from breast cancer [53], imaging and behavior data from patients diagnosed with fragile X syndrome (FXS) [62], and motor features in Parkinson's disease [46]. It recently has shown promise in generating insights from fMRI data [33], in particular, in "distilling complex brain dynamics into interactive and behaviorally relevant representations" [64]. In biophysics, the mapper graph has been used to study biomolecular folding pathways [74]. Outside of science and engineering, it has also been used to characterize NBA basketball players' performance statistics and to study voting data from the United States House of Representatives [45].

A number of open-source software packages are available for mapper computation and visualization, including general purpose tools *Mapper Interactive* [77], *KeplerMapper* [72], *giotta-tda* [69], *Gudhi* [70], and *Python Mapper* [49]. In application domains, *Pheno-Mapper* [78] and *Hyppo-X* [39] explore phenomics datasets. *DyNeuSR* is designed to explore topological properties and neurophysiological correlates of mapper graphs from neuroimaging datasets [33], and *NeuMapper* implements a neuroimaging-focused (e.g., fMRI) mapper pipeline.

Distances between mapper graphs. The mapper graph may be considered as a discrete approximation [13, 51] of a topological descriptor called the Reeb graph. A number of distances have been developed for Reeb graphs and their variants, including interleaving distance [14, 20, 21, 24, 48, 50], functional distortion distance [5, 7], functional contortion distance [3], edit distance [4, 6, 26, 68], Gromov-Hausdorff distance [17, 71], and more [8, 15, 65]. A major drawback of some of the above distances is that they are hard to compute. For instance, it is NP-hard to approximate the interleaving distance for merge trees (i.e., special cases of Reeb graphs) [48] within a factor of 3. The 0-interleaving distance between Reeb graphs is graph isomorphism complete [10,24]. Even the more general graph edit distance is NP-hard to compute exactly [76]. See [11, 73] for surveys.

In comparison with previous work, an important distinction is that we are not comparing mapper graphs as graphs, but as hypergraphs. Our framework models mapper graphs as hypergraphs and explicitly takes into consideration the matching between vertices and hyperedges during the comparative analysis. Whereas most existing distances are hard to compute (with a few exceptions), we introduce distances based on optimal transport that are easily computable via gradient descent (e.g., [54]) and the Python Optimal Transport library [30]. Furthermore, our distances are also interpretable in the sense that they offer probabilistic matchings between vertices and hyperedges (see Section 4).

Topology and deep learning. For simplicity, we use "(neuron) activations" to refer to the high-dimensional vector representations produced by outputs of neurons from a particular layer of a neural network and "activation space" to refer to the space of these vectors.

Topology-based methods, such as persistent homology [27] and

- *Y. Zhou and B. Wang are with the University of Utah. E-mails: zhouyj96180@gmail.com, beiwang@sci.utah.com*
- *H. Jenne, D. Brown, M. Shapiro, B. Jefferson, C. Joslyn, G. Henselman-Petrusek, B. Praggastis, and E. Purvine are with the Pacific Northwest National Laboratory. Emails: {helen.jenne, davis.brown, madelyn.shapiro, brett.jefferson, Cliff.Joslyn, gregory.roek,brenda.praggastis, Emilie.Purvine}@pnnl.gov*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

mapper graphs, have been utilized in machine learning, in particular, deep learning, in recent years. Following a recent survey [36], extrinsic topological features have been defined as the transformations of topological descriptors from data into feature vectors to be used as input to machine learning models, whereas intrinsic topological features are used to influence machine learning models themselves via architecture choices or regularization. From an intrinsic perspective, quantifiers based on persistent homology were developed to quantify the learnability or complexity of neural networks, such as *topological capacity* [34] and *neural persistence* [61]. Gebhart et al. [32] studied the topology of neural network activations using persistent homology. They introduced the *activation graph*, which is a sequence of bipartite graphs between neurons of adjacent layers whose edge weights capture how much an input "activates" the connections between the neurons. Lacombe et al. [42] monitored trained neural networks based on the topological properties of their activation graphs. Using mapper graphs, Gabrielsson et al. [31] showed that the weights of convolutional layers (referred to as *weight spaces* in contrast to our *activation spaces*) learn simple global structures, and such structures change across training.

Several previous works are most relevant to ours. Rathore et al. [57] introduced *TopoAct*, which uses mapper graphs to interactively visualize and explore neuron activations from image classifiers. Zhou et al. [79] visualized the topology of neuron activations in deep adversarial training. Rathore et al. [58] presented *TopoBERT*, a visual analytics system based on mapper graphs for exploring the fine-tuning process of transformer-based models, across multiple fine-tuning batch updates, subsequent layers of the model, and different natural language processing (NLP) tasks. Purvine et al. [55] modeled neuron activations as high-dimensional point clouds and applied persistent homology to quantify meaningful differences between layers. They also studied how mapper graphs of neuron activations evolve visually across layers and provide semantic insight into how deep models organize hierarchical class knowledge.

This work could be considered as a natural extension of previous work [55, 57] in studying the topology of neuron activations. However, the focus of this work is to develop distances useful for comparing mapper graphs that arise from CNNs, which may be applicable to comparing mapper graphs derived from other deep learning models and tasks such as weight spaces [31].

Visualization of neuron activations. An explosion of research in recent years has aimed to explain deep learning models by visualizing neuron activations; see [37] for a survey. Examples include *salience maps* [66], *activation maximization* [28], *DeepVis* [75], *multifaceted feature visualization* [52], *TCAV* (Testing with Concept Activation Vectors) [40], *activation atlas* [18], *SUMMIT* [38], and visual analytics of neuron vulnerability [44].

This work builds on previous research that constructs and visualizes mapper graphs of neuron activations [55, 57, 58]. However, instead of focusing on interactive exploration of neuron activations, this work addresses a key challenge in using mapper graphs in practice, that is, how to compare mapper graphs in an intuitive and computationally efficient way.

3 Background on Activations and Mapper Graphs

Neuron Activations. The input to a CNN is an image, and the output is a probability vector communicating the likelihood the input image belongs to each class. The intermediate outputs of neurons from each convolutional layer are referred to as the *activation tensors*, which may be sliced vertically into *spatial activations*. In a nutshell, a spatial activation is a high-dimensional vector formed by neuron outputs of an image patch at a fixed layer, as visualized in Figure 1. The dimension of a spatial activation (referred to as an activation or an activation vector) is the number of neurons/channels at a fixed layer.

For each training image from the CIFAR-10 dataset, we experiment with two types of spatial activations: the random activations and the foreground activations.

For random activations, we randomly sample a single spatial activation from a number of image patches at a fixed convolutional layer.

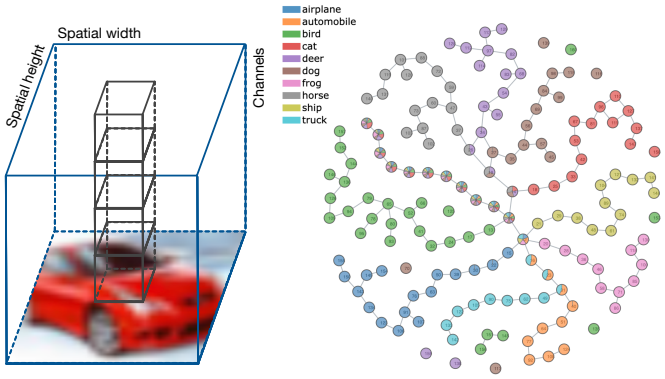

Fig. 1. Left: a spatial activation within an activation tensor for a CIFAR-10 image. Right: a mapper graph of random activations from CIFAR-10.

For foreground activations, we are interested in studying how foreground information of an input image manifests in the activation space following [55]. Each spatial activation corresponds to a subset of the foreground or background pixels in its *effective receptive field*, which is the region of the input image that the network has "seen" via contributions from previous layers [55]. To generate foreground activations, we first use *cv2.grabCut* from the OpenCV library [12] to perform image segmentation and identify the foreground pixels of an input image. We then choose a spatial activation with the largest percentage of foreground pixels in the effective receptive field, referred to as the foreground activation.

Using a ResNet-18 CNN model with 50K CIFAR-10 training images, for each type of activation, each layer gives rise to a high-dimensional point cloud with 50K points. Since ResNet-18 contains 16 convolutional layers, it gives rise to 16 point clouds with varying dimensions.

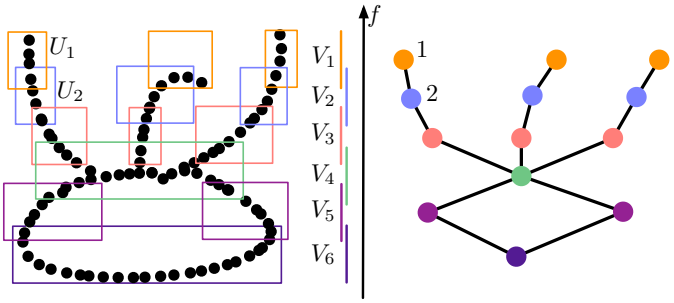

Fig. 2. A point cloud in $\mathbb{R}^2$ and its mapper graph.

Mapper graphs. Given a point cloud of neuron activations, we capture its topology via a graphical representation called the *mapper graph* [67] that represents clusters from the point cloud as mapper nodes and preserves cluster relations as mapper edges. We review its definition for a generic high-dimensional point cloud.

Let $\mathbb{X}$ be a d-dimensional point cloud (e.g., d would be the number of neurons at a fixed layer for a high-dimensional spatial activation). A *cover* of $\mathbb{X}$ is a set of open sets in $\mathbb{R}^d$, $\mathcal{U} = \{U_i\}_{i\in I}$ such that $\mathbb{X} \subset \cup_{i\in I} U_i$. We construct the 1-dimensional nerve of $\mathcal{U}$, denoted as $\mathcal{N}_1(\mathcal{U})$, as follows: each node i in $\mathcal{N}_1(\mathcal{U})$ represents a cover element U_i, and there is an edge between nodes i and j if $U_i \cap U_j$ is nonempty. As shown in Figure 2, given a 2-dimensional point cloud $\mathbb{X}$, the colored rectangles on the left are open sets in $\mathbb{R}^2$ that form a cover $\mathcal{U}$ of $\mathbb{X}$; the 1-dimensional nerve of $\mathcal{U}$ is shown on the right.

In the mapper construction introduced by Singh et al. [67], a cover of $\mathbb{X}$ is obtained by a scalar function defined on $\mathbb{X}$, referred to as a *filter* function. In our setting, we define a mapper graph with a single filter function $f : \mathbb{X} \to \mathbb{R}$. A *uniform cover* $\mathcal{V} = \{V_k\}_{k=1}^n$ of $f(\mathbb{X}) \subset \mathbb{R}$ is obtained using partially overlapping intervals of uniform lengths such that $f(\mathbb{X}) \subseteq \cup_k V_k$. The cover $\mathcal{U}$ of $\mathbb{X}$ is obtained by considering the clusters, obtained from a chosen clustering algorithm, induced by points in $f^{-1}(V_k)$ as cover elements (for each V_k). The 1-dimensional nerve of $\mathcal{U}$ is the *mapper graph* of $(\mathbb{X}, f)$.

We give Figure 2 as an example. A 2-dimensional point cloud $\mathbb{X}$ is sampled from the silhouette of a steaming pork bun (left). We use a height function $f : \mathbb{X} \to \mathbb{R}$ as its filter function. A cover $\mathcal{V} = \{V_1, \cdots, V_6\}$ of $f(\mathbb{X})$ is formed by six uniform intervals (middle). For each $k \in [1, 6]$, $f^{-1}(V_k)$ induces a number of clusters that are subsets of $\mathbb{X}$. These clusters are enclosed by rectangles and form the elements of a cover $\mathcal{U}$ of $\mathbb{X}$ (left). The mapper graph of $\mathbb{X}$ is shown on the right. For instance, $f^{-1}(V_1)$ induces three cover elements in orange (including U_1), whereas $f^{-1}(V_2)$ induces three cover elements in blue (including U_2). U_1 and U_2 become nodes 1 and 2 in the mapper graph, and since $U_1 \cap U_2 \neq \emptyset$, there is an edge connecting node 1 and node 2 in the mapper graph.

A number of parameters are needed to construct a mapper graph, including the filter function f, the number of cover elements n, and their percentage of overlap p, the metric $d_{\mathbb{X}}$ on $\mathbb{X}$ which is necessary for computing distances between points in the clustering step, and the clustering method. As shown in Fig. 2, $n = 6$, $p = 30\%$, $d_{\mathbb{X}}$ is the Euclidean distance, and the clustering method is the density-based DBSCAN [29] that contains two additional parameters ($minPts$: the minimum number of points required to form a dense region; ϵ: and the size of a neighborhood). With an appropriate choice of parameters, the mapper graph captures the shape of the input data; see [19, 77] for a discussion on parameter tuning.

Applying the above algorithm to a point cloud of neuron activations, we obtain a mapper graph that captures its topology, that is, the organizational principle behind neuron activations. As illustrated in Figure 1 (right), we give an example of a mapper graph obtained by random activations of CIFAR-10 training images at layer 16 of a ResNet-18 model. Since there are 512 neurons/channels at layer 16, each image gives rise to an activation vector in 512 dimensions. The filter function f is chosen to be the ℓ_2-norm of an activation vector, which reflects how strongly the neural network "reacts" to a given image. We set $n = 40$ and $p = 20\%$. For parameters associated with DBSCAN, $\epsilon = 8.71$, $minPts = 5$. As observed previously [77], the mapper graph not only clusters images from each class into a separate branch, but also highlights the hierarchical relationship among the different classes. Specifically, we see a clear, slightly delayed bifurcation between the automobile images (contained in the orange mapper nodes) and truck images (contained in the sky blue mapper nodes), since these images are more similar to each other than to images from other classes.

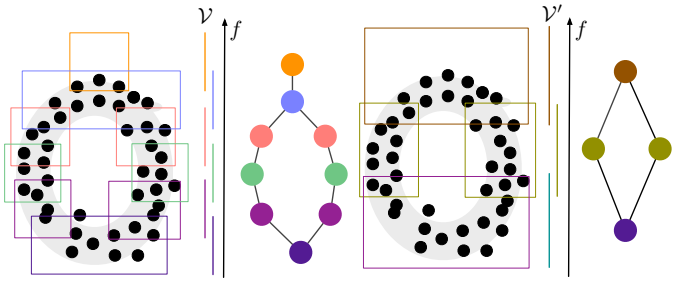

Fig. 3. Two mapper graphs at different scales. Points are sampled from a 2-dimensional annulus. Covers $\mathcal{V}$ and $\mathcal{V}'$ contain six and three intervals respectively.

Multiscale mapper. The basic idea behind the multiscale mapper [25] is that as the cover $\mathcal{V}$ varies, we obtain mapper graphs at different scales, as illustrated in Figure 3.

The multiscale mapper was developed to relate mapper graphs constructed at different scales. A key idea necessary to relate mapper graphs constructed at different scales is that of a map of covers. Let $\mathcal{V} = \{V_i\}_{i \in I}$ and $\mathcal{V}' = \{V'_j\}_{j \in J}$ be two covers of a topological space, then a set map $\phi : I \to J$ is a *map of covers* if $V_i \subseteq V'_{\phi(i)}$ for all $i \in I$. The multiscale mapper utilizes the observation that a map of covers induces a map between mapper graphs at different scales. For example, there is a map of covers between the two covers in Figure 3, which induces a map between the mapper graphs across two scales.

Given a collection of finite covers $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m$ of $f(\mathbb{X})$ such that there are well-behaved maps between them, we call it a *tower of covers* (across m scales) [25]. More generally, the well-behaved maps of covers are defined such that for any $1 \leq i, j, k \leq m$, the maps $\phi_{i,j} : \mathcal{V}_i \to \mathcal{V}_j$ satisfy $\phi_{i,i} = id$, and $\phi_{i,k} = \phi_{j,k} \circ \phi_{i,j}$; see [25] for technical details.

4 Method

To compare a pair of mapper graphs that arise from deep learning, we introduce a distance between them using existing tools from optimal transport. In our context, a mapper graph is a hypergraph in which each data point is a vertex in the hypergraph and a cluster of data points from the mapper algorithm forms a hyperedge. We treat each hypergraph as a measure hypernetwork using existing tools from optimal transport [22].

4.1 Modeling Mapper Graphs as Measure Networks

Formally, a *measure hypernetwork* is a tuple $\mathcal{H} = (X, \mu, Y, \nu, \omega)$, where X and Y are well-behaved topological spaces (i.e., Polish spaces), μ and ν are probability measures on X and Y, respectively, and $\omega : X \times Y \to \mathbb{R}$ is a well-behaved (e.g. measurable and bounded) weight function that captures relations between elements in X and elements in Y [22].

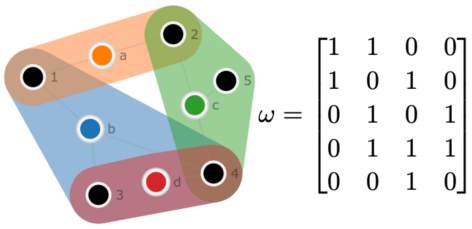

Fig. 4. An example of modeling a hypergraph (left) as a measure hypernetwork where ω (right) captures incidence relations between vertices and hyperedges. Black nodes represent vertices, whereas colored nodes and colored convex hulls both represent hyperedges.

In our context, we treat a mapper graph (specifically, the cover induced by the mapper graph) as a finite hypergraph $H = (X, Y)$, where X is a set of vertices, and Y is a set of hyperedges. Such a hypergraph may be modeled as a measure hypernetwork $\mathcal{H} = (X, \mu, Y, \nu, \omega)$ by introducing probability measures μ and ν on its vertices and hyperedges, respectively based on the hypergraph structure, and the weight function ω based on the node-hyperedge relations.

For example, we may choose μ and ν to be the uniform measures on vertices and hyperedges, respectively, i.e., $\mu(x) = 1/|X|$ for any $x \in X$ and $\nu(y) = 1/|Y|$ for any $y \in Y$. The weight function ω may encode the *incidence relation*, that is,

$$\omega(x, y) = \begin{cases} 1, & \text{if } x \in y; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The above formulation means that $\omega(x, y) = 1$ if a node $x \in X$ belongs to a hyperedge $y \in Y$. Figure 4 gives a simple example of modeling a hypergraph as a measure hypernetwork. Here, we have a vertex set $X = \{1, 2, 3, 4, 5\}$ and $\mu(x) = 1/5$ for each $x \in X$. We have a set of hyperedges $Y = \{a, b, c, d\}$, where $\mu(y) = 1/4$ for each $y \in Y$. ω is a $|X| \times |Y|$ matrix capturing the incidence relation, e.g., $\omega(2, 1) = 1$ means that vertex 2 belongs to hyperedge a.

Alternatively, to capture richer relations between vertices and hyperedges, ω may encode a *(weighted) shortest path relation*. To define ω, we need to first introduce the notion of a line graph. The line graph $L(H)$ of a hypergraph H is a graph whose node set is the set of the hyperedges of H, where two vertices in $L(H)$ are connected by an edge when their corresponding hyperedges in H have a nonempty intersection, see Figure 5 for an example.

Edges in a line graph $L(H)$ may be weighted, where a larger weight indicates a larger distance between a pair of vertices. For a pair of vertices $v, v' \in L(H)$, their assigned weight $w(v, v')$ may be inversely proportional to the intersection size of their corresponding hyperedges y and y' in H. That is, $w(v, v') = 1/|y \cap y'|$. Alternatively, $w(v, v')$ may be the Jaccard distance (one minus the Jaccard index) that measures the dissimilarity between y and y', $w(v, v') = 1 - \frac{|y \cap y'|}{|y \cup y'|}$. As illustrated

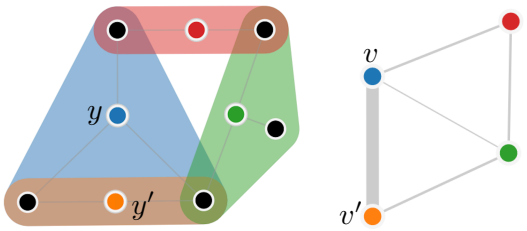

Fig. 5. An example of a hypergraph H (left) with its corresponding line graph $L(H)$ (right). Left: black nodes represent vertices in a hypergraph, whereas colored nodes and colored convex hulls both represent hyperedges. Right: colored nodes represent vertices of $L(H)$ that correspond to hyperedges in H. An edge in $L(H)$ is weighted by the Jaccard index between a pair of hyperedges in H that correspond to its end points.

in Figure 5 (right), edge thickness in the line graph scales with the Jaccard index between hyperedges. For example, hyperedges y and y' has a Jaccard index of $2/3$, then $w(v, v') = 1 - 2/3 = 1/3$.

Aksoy et al. [2] introduced the notion of *s-walk* between hyperedges y and y' in a hypergraph H, which is a sequence of hyperedges, $y = y_0, y_1, ..., y_k = y'$, where $y_{j-1} \neq y_j$ and $|y_{j-1} \cap y_j| \geq s$, for $1 \leq j \leq k$. In our context, we set $s = 1$ and work with 1-walks without repeated hyperedges (referred to as *paths*). By construction, a (shortest) path between hyperedges y and y' in H corresponds to a (shortest) path between vertices v and v' in $L(H)$.

We are now ready to introduce the shortest path relation between vertices and hyperedges in H. That is, we define $\omega(x, y)$ between a vertex $x \in X$ and a hyperedge $y \in Y$ to be the length of the shortest path from any hyperedge y' containing x to hyperedge y in H. Formally, we have

$$\omega(x, y) = \begin{cases} 0, & \text{if } x \in y; \\ \min_{y' \in Y, x \in y} d(y, y'), & \text{otherwise.} \end{cases} \tag{2}$$

Here, $d(y, y')$ is the shortest path distance between hyperedges y and y' in H.

4.2 Comparing Mapper Graphs Using Optimal Transport

Given two hypernetworks $\mathcal{H} = (X, \mu, Y, \nu, \omega)$ and $\mathcal{H}' = (X', \mu', Y', \nu', \omega')$ modeling mapper graphs, the Gromov-Wasserstein (GW) distance between them is defined as [22]:

$$d_{GW}(\mathcal{H}, \mathcal{H}')^2 = \min_{\pi \in C(\mu, \mu'), \xi \in C(\nu, \nu')} \left(\sum_{x \in X, y \in Y, x' \in X', y' \in Y'} (\omega(x, y) - \omega'(x', y'))^2 \pi(x, x') \xi(y, y') \right). \tag{3}$$

Here, π is a coupling of X and X', that is, a joint probability measure on $X \times X'$ such that its marginals agree with μ and μ', respectively. $C(\mu, \mu')$ is a set of couplings between μ and μ'. Similarly, ξ is a coupling of Y and Y', and $C(\nu, \nu')$ is a set of couplings between ν and ν'. Following [22], computing d_{GW} requires optimizing the couplings between vertices and hyperedges at the same time using co-optimal transport [59].

We give an example in Figure 6 with a pair of hypernetworks, $\mathcal{H} = (X, \mu, Y, \nu, \omega)$ (left) and $\mathcal{H}' = (X', \mu', Y', \nu', \omega')$ (right). For both hypernetworks, we set μ, ν (and μ', ν') to be proportional to the degrees of vertices and hyperedges, respectively, and ω, ω' to be the weighted shortest path based on Jaccard distance. For normalized vertex degree, we set $\mu(x) := \deg(x) / \sum_{x' \in X} \deg(x')$. For normalized "hyperedge degree", we use the normalized sum of vertex degree, that is, $\nu(y) := \widehat{\nu}(y) / \sum_{y' \in Y} \widehat{\nu}(y')$, where $\widehat{\nu}(y) := \sum_{x \in y} \deg(x)$. See Appendix B for other parameter details.

We perform 20 instances in optimizing the GW distance between $\mathcal{H}$ and $\mathcal{H}'$ with random initialization. The best (and smallest) GW

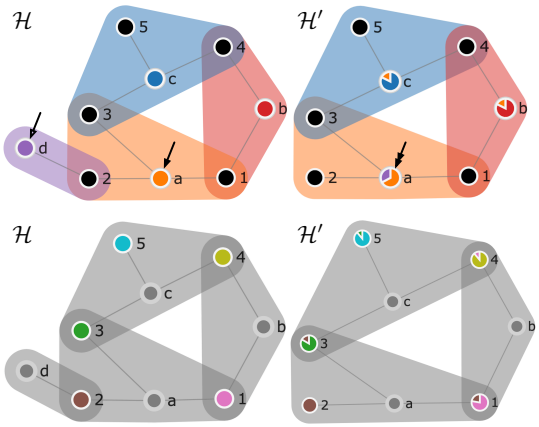

Fig. 6. An example of GW-based coupling of hyperedges and vertices based on the best GW matching.

distance (valued at 0.1079) gives rise to a hyperedge coupling matrix π and a vertex coupling matrix ξ, which are, respectively,

$$\xi = \begin{pmatrix} 0.2395 & 0.0504 & 0.0630 \\ 0 & 0.2353 & 0 \\ 0 & 0 & 0.2941 \\ 0.1176 & 0 & 0 \end{pmatrix}$$

$$\pi = \begin{pmatrix} 0.1944 & 0 & 0 & 0.0278 & 0 \\ 0.0556 & 0.1250 & 0.0417 & 0 & 0 \\ 0 & 0 & 0.2083 & 0 & 0.0139 \\ 0 & 0 & 0 & 0.2222 & 0 \\ 0 & 0 & 0 & 0 & 0.1111 \end{pmatrix}$$

ξ and π are visualized in Figure 6 using a color transfer. Fix a categorical colormap for rows of ξ (i.e., hyperedges in $\mathcal{H}$), we encode the columns of ξ (i.e., hyperedges in $\mathcal{H}'$) using a pie chart. For example, the 1st column of ξ encodes the probabilistic matching between hyperedges a and d in $\mathcal{H}$ and the 1st hyperedge a in $\mathcal{H}'$. Intuitively speaking, $a \in Y'$ (double filled arrow) is matched to $a \in Y$ (single filled arrow) with twice the probability as $a \in Y'$ being matched to $d \in Y$ (single hollow arrow); this gives rise to a pie chart at $a \in Y'$ with $2/3$ orange and $1/3$ purple. π is visualized similarly using a color transfer, with diagonal entries with the highest probability.

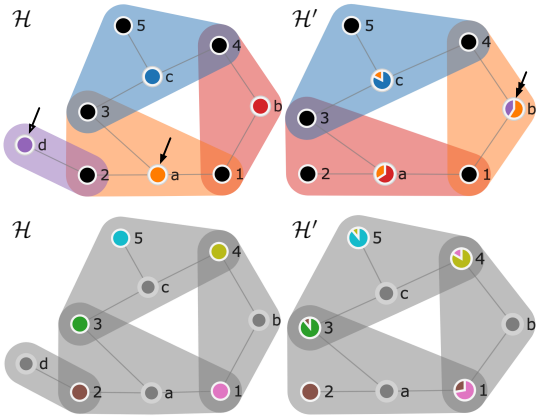

Fig. 7. Another example of GW-based coupling of hyperedges and vertices based on the second best GW matching.

We further investigate the second best GW matching with a GW dis-

tance valued at 0.1372, the coupling matrices ξ and π are, respectively,

$$\xi = \begin{pmatrix} 0.1218 & 0.1681 & 0.063 \\ 0.2353 & 0 & 0 \\ 0 & 0 & 0.2941 \\ 0 & 0.1176 & 0 \end{pmatrix}$$

$$\pi = \begin{pmatrix} 0.1806 & 0 & 0 & 0.0417 & 0 \\ 0.0694 & 0.125 & 0.0278 & 0 & 0 \\ 0 & 0 & 0.2222 & 0 & 0 \\ 0 & 0 & 0 & 0.2083 & 0.0139 \\ 0 & 0 & 0 & 0 & 0.1111 \end{pmatrix}$$

ξ and π are visualized in Figure 7 using a color transfer. Comparing with the best GW matching in Figure 6, the second best GW distance matches $b \in Y'$ (double filled arrow) to $a, d \in Y$ (single filled arrow and single hollow arrow). This is possible due to near symmetric structure between a and b in $\mathcal{H}'$.

However, if the vertex sets between the two hypernetworks are identical, π becomes an identity matrix (scaled by a constant). Then, we can compute the Wasserstein distance instead, which focuses only on optimizing the coupling between hyperedges, that is,

$$d_W(\mathcal{H}, \mathcal{H}')^2 = \min_{\xi \in C(\nu, \nu')} \sum_{y \in Y, y' \in Y'} \text{cost}(y, y')\xi(y, y'), \quad (4)$$

where $\text{cost}(y, y') = \sum_{x \in X, x' \in X'} (\omega(x, y) - \omega'(x', y'))^2$ is a cost function between y and y'.

Using the same pair of hypernetworks $\mathcal{H}$ and $\mathcal{H}'$ of Figure 6, we perform 20 instances in optimizing their Wasserstein distance. The best Wasserstein distance (valued at 0.0303) gives rise to a hyperedge coupling ξ,

$$\xi = \begin{pmatrix} 0.2395 & 0.0504 & 0.0630 \\ 0 & 0.2353 & 0 \\ 0 & 0 & 0.2941 \\ 0.1176 & 0 & 0 \end{pmatrix}$$

This hyperedge coupling matrix happens to be the same as the best hyperedge coupling matrix in the GW setting. This is not too surprising as $\mathcal{H}$ and $\mathcal{H}'$ share very similar vertex degree distributions and optimizing the Wasserstein distance obtains the same local optimal as the GW setting.

Implementation details. Following [22], we use the Python3 implementation of CO-Optimal Transport (COOT) due to Redko et al. [59], which contains an efficient algorithm for approximating the solution to the optimization problems involving the GW distance and Wasserstein distance. With a slight abuse of notations, the computational complexity is $O(\min((n + n')dd' + n'^2 n, (d + d')nn' + d'^2 d))$ [59], where $n = |X|$, $n' = |X'|$, $d = |Y|$ and $d' = |Y'|$. The COOT implementation uses *Numpy*, *Matplotlib*, and *Python Optimal Transport (POT)* [30] library.

To compute an optimal transport distance, we would need to initialize the coupling matrices ξ and π. We use random initialization as follow. First, we randomly generate the elements in the matrix ξ from a gamma distribution. Then we apply a Sinkhorn optimization to ensure that the marginal distributions of the coupling are consistent with the distributions of hyperedges ν and ν'. π is initialized similarly. See [23] for details on the Sinkhorn algorithm.

5 Comparing Mapper Graphs of Neuron Activations

Our primary objective is to study the evolution of neuron activations across the (convolutional) layers of CNNs. To do so, given a set of input images, a mapper graph is constructed from the neuron activations from each layer, and we compare mapper graphs across layers to study the topological changes in the activation space, even though the underlying activation spaces are of different dimensions between layers.

Experimental setup. We experiment with CIFAR-10 [41] training images on a ResNet-18 [35] model (see Section 3). We construct mapper graphs on both the random activations and the foreground activations.

To quantify the changes in mapper graphs across layers, for each type of activations, we compute optimal transport distances between each convolutional layer (e.g. layers 1 to 16) and the last convolutional layer (i.e., layer 16). Given that the activations at each layer correspond to the same set of input images, the vertex sets remain the same across layers. Therefore, we compute the Wasserstein distances between the mapper graphs.

Experimental details. To compute the Wasserstein distances, we first convert each mapper graph to its corresponding hypergraph H. Each mapper graph node becomes a hyperedge in H, and each activation vector (data point in the mapper graph) becomes a vertex in H. Figure 8 and Figure 9 show the mapper graphs and their corresponding hypergraphs from random and foreground activations, respectively. For all mapper graphs, we set $n = 40$, $p = 25\%$. For DBSCAN, $minPts = 5$, and the size of the neighborhood ϵ is determined using the elbow method for each layer; see [77] for details on the elbow method. We model each hypergraph as a hypernetwork by setting μ and ν to be proportional to vertex and hyperedge degrees, respectively. For ω, we experiment with all three configurations, with ω being incidence relation, weighted shortest path relation based on Jaccard distance, and the inverse of intersection size, respectively.

For each type of activation, we plot the Wasserstein distances between the mapper graph from the last convolutional layer (layer 16) and the mapper graphs from earlier layers (e.g., layers 4, 8, 12, 13, 14, 15, and 16).

For each pair of hypergraphs modeled as hypernetworks, we start with randomly initialized coupling matrices and repeat the optimization process (during optimal transport) 10 times. At each layer, we plot the box plot of the 10 obtained Wasserstein distances, and we also plot the minimum distance at each layer as a line. Figure 10 shows the Wasserstein distances between various layers (4, 8, 12, 13, 14, 15, 16) and layer 16 with ω chosen among three configurations, for mapper graphs generated using random and foreground activations, respectively.

Structural correspondence via color transfer. We use a color transfer to highlight the structural correspondences between mapper graphs across layers. Since a hyperedge coupling matrix ξ provides a probabilistic matching between hyperedges, we can obtain a structural correspondence from $\mathcal{H}$ to $\mathcal{H}'$. To perform a color transfer in the simplest form, we declare that a hyperedge x in $\mathcal{H}$ is matched to a hyperedge y in $\mathcal{H}'$, if $y = \text{argmax}_{y'} \xi(x, y')$. In other words, x is matched with a hyperedge with the highest probability.

Given a pair of mapper graphs obtained at two different layers, we could study their structural correspondence via a color transfer. In Fig. 11 (top), we construct mapper graphs of random activations at layer 15 and layer 16, respectively. We color each mapper node by the category of its associated images (considered to be the ground truth label). A mapper node that comprises images from a single category is considered to be *pure* and marked by a single color; otherwise, it is visualized by a pie chart. In Fig. 11 (bottom), we study the color transfer from pure nodes at layer 15 to nodes at layer 16. We observe that a significant number of nodes at layer 16 (bottom right) have colors that align with their ground truth labels (top right). Such an alignment suggests that the nodes from layer 15 are mapped to the appropriate branches at layer 16 via optimal transport. Notably, the pink branch (pointed by a single arrow) from layer 15 (bottom left) is mapped to the pink branch from layer 16 (bottom right), which in turn, matches exactly to the pink branch with the ground truth label (top right). We observe a similar behavior involving the purple branch (pointed by a double arrow).

By observing how node color transfers from one layer to another, we obtain an explicit structural correspondence that helps us better understand how the mapper graphs evolve across layers. It would be interesting to explore further how the images within the mapper nodes are transported across layers, potentially yielding additional insights into the evolution of activation spaces.

Takeaway. Based on qualitative observations from Figure 8 and Figure 9, the mapper graphs of both random and foreground activations show more noticeable branching structures that indicate class separa-

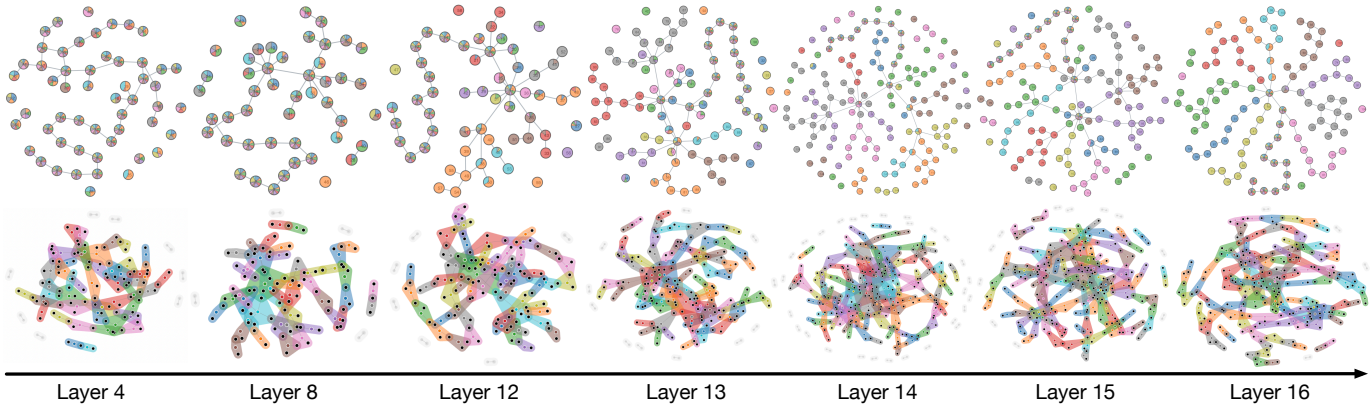

Fig. 8. Mapper graphs (top) and their corresponding hypergraphs (bottom) from random activations using CIFAR-10 training images and ResNet-18. For mapper graphs, mapper nodes are colored using the same categorical colormap from Figure 1.

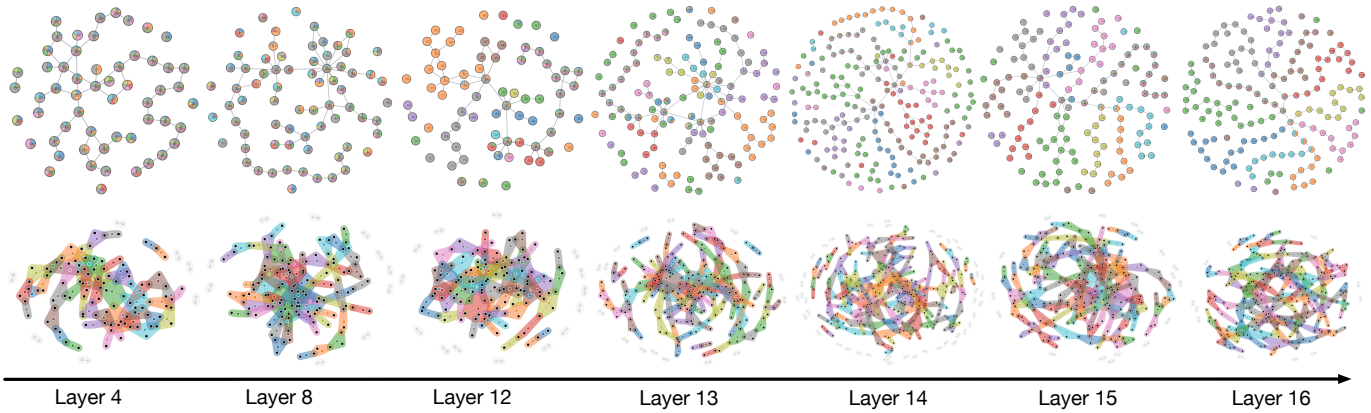

Fig. 9. Mapper graphs (top) and their corresponding hypergraphs (bottom) from foreground activations using CIFAR-10 training images and ResNet-18. For mapper graphs, mapper nodes are colored using the same categorical colormap from Figure 1.

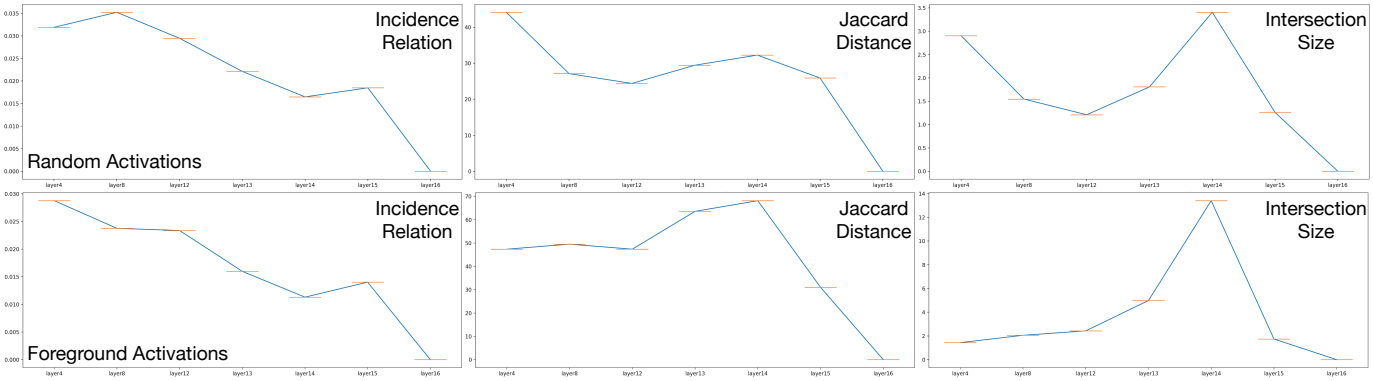

Fig. 10. Wasserstein distances between earlier layers (4, 8, 12, 13, 14, 15) and layer 16 with different configurations of ω for mapper graphs generated using random (top) and foreground activations (bottom), respectively. From left to right, ω is based on the incidence relation, Jaccard-distance-based shortest paths, and intersection-size-based shortest paths, respectively.

tions as we go deeper in the layers. Our hypothesis is that the distance between intermediate layers and the last layer decreases as they become closer to the last layer, as the model is getting better at differentiating classes. As shown in Figure 10, when ω is set to encode the incidence relation, the distance plots generally agree with our hypothesis, providing quantitative evidence that the topological structures of neuron activations are improving in later layers.

However, when ω is set to use the intersection size or Jaccard-distance-based shortest path relation, there is a sharp rise in distances at layer 14, which deserves further study. We argue that the incidence relation is a better choice for ω than the other two configurations, since the incidence relation (e.g., does the hyperedge y' containing x intersect with the hyperedge y?) matters more than the (normalized) size of intersection (e.g., what is the size of intersection between y' and y if they intersect?) In other words, the incidence relation plays a crucial role in determining the connectivity of a mapper graph, and thus is useful in capturing the topological similarities between pairs of mapper graphs.

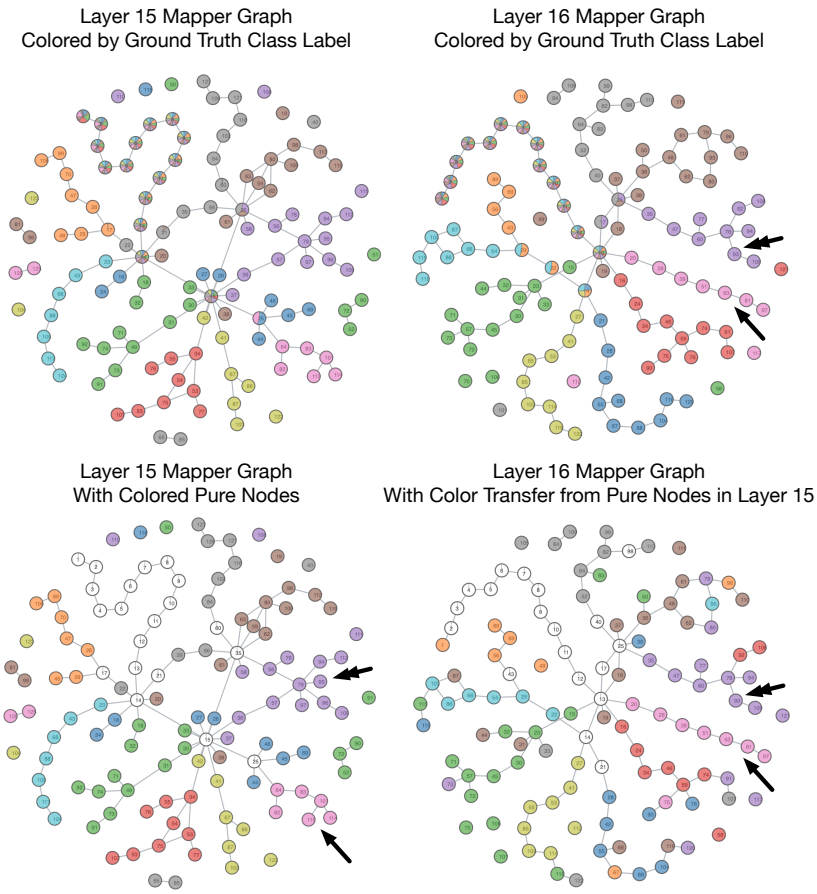

Fig. 11. Structural correspondence via Wasserstein distance between mapper graphs generated from random activations at adjacent layers. Top: mapper graphs at layer 15 and layer 16 colored by the ground truth class label (using a categorical colormap). Bottom: a color transfer from the pure nodes in layer 15 to nodes in layer 16 that highlights structural correspondences.

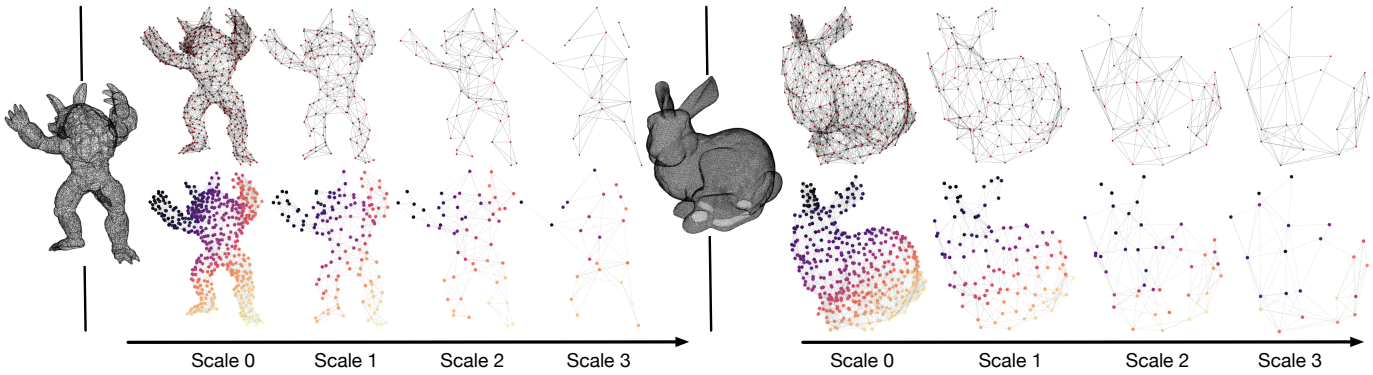

Fig. 12. The original graphs converted from the mesh of Armadillo (left) and Stanford Bunny (right), respectively, together with the mapper graphs across four scales. From left to right: mapper graphs from the finest to the coarsest scales. Top: mapper graphs before computing Wasserstein distances. Bottom: color transfer between the mapper graph at scale 0 and the mapper graphs at scales 1, 2, and 3, respectively, based on coupling matrices from Wasserstein distance computations.

6 Comparing Mapper Graphs Across Scales

Our framework may be used to compare mapper graphs that arise in other scenarios beyond deep learning. As demonstrated in this section, our framework may be used to compare a sequence of mapper graphs that arise from multiscale mapper [25].

Generating mapper graphs across scales. We obtained two meshes from the Stanford 3D scanning repository [1] called Armadillo and Stanford Bunny. Starting from the Armadillo mesh, we first convert it into an original graph G using the mesh vertices and edges, see Figure 12 (left).

We then compute a sequence of mapper graphs across scales following the approach in [22]. Given an original graph $G = (V, E)$, we first use a heat kernel based method to generate an overlapping cover $\mathcal{U}_0$ of the vertex set V, and compute its 1-dimensional nerve graph G_0 (this is the mapper graph at scale 0). We repeat such a process on the new nerve graph, generating coarser nerve graphs at each step of the iteration, G_1, G_2, G_3, and so on; see Appendix A for details. We choose heat kernels as they are known to capture shape signatures (e.g, [47, 60]). The size of the mapper graphs across scales are shown in Table 1.

For example, Figure 12 (left) shows mapper graphs of Armadillo

	Armadillo		Stanford Bunny	
	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$
Mesh (G)	17483	52443	34834	104288
G_0	506	3632	423	3047
G_1	147	731	114	502
G_2	54	224	44	183
G_3	23	76	22	85

Table 1. Size of mapper graphs across scales. G is the original graph from the mesh.

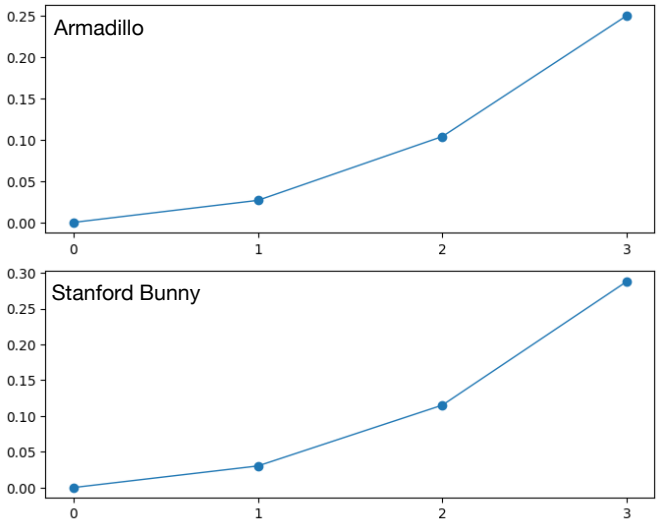

Fig. 13. Wasserstein distances between the mapper graphs of Armadillo (top) and Stanford Bunny (bottom) at coarser scales (1, 2, and 3) and the mapper graph at the finest scale (0).

across four scales. By construction, each node in a coarser mapper graph contains a subset of nodes in a finer mapper graph, and therefore corresponds to a subset of nodes in the original graph. The mapper graphs of Stanford Bunny across four scales are constructed similarly, shown in Figure 12 (right).

For each mapper graph, we construct its corresponding hypergraph by mapping each mapper node as a hyperedge, and each vertex in the original graph as a vertex in the hypergraph. Therefore, all mapper graphs share the same set of vertices, and we can compute the Wasserstein distances between them.

For each mesh, we compute the Wasserstein distances between the mapper graphs at increasingly coarser scales (scales 1, 2 and 3) and the mapper graph at the finest scale (scale 0). Figure 13 displays the Wasserstein distances for Armadillo and Stanford Bunny, respectively. **Takeaway.** A key takeaway is that the Wasserstein distances faithfully capture the increasing structural changes from the finest to the coarsest scales. In other words, an increasing distance captures the loss of structural information for mapper graphs at coarser scales. Furthermore, the color transfer from G_0 to G_1 (and respectively, G_2 and G_3) highlights the structural correspondences between mapper graphs across scales.

To connect the mapper graph convergence across scales with the convergence across convolutional layers, we observe that in the architecture of ResNet-18 (and indeed most convolutional neural networks) the size of the activation layers decreases as we go deeper in the model. In a way, the convolutions are performing a scale coarsening operation (albeit with weights and activation functions). Our observations of distances converging through convolutional layers in Section 5, and across scale refinement in this section can be seen as complementary.

7 Conclusion and Discussion

Mapper graphs have been used recently to visualize the topology of artificial neural activations from CNNs and large language models. This work addresses a key question on how to perform mapper graph comparisons. We show that distances based on optimal transport are well suited to compare mapper graphs (treated as hypergraphs) across convolutional layers (in deep learning) and across scales (in a multiscale mapper).

Purvine et al. [55] demonstrated that the mapper graph branching structures are present in a number of model-data pairs, such as ImageNet studied using ResNet-18, InceptionV1/V3, and AlexNet. For future work, we are interested in generalizing our experiments to other models and datasets. We would also like to extend our work to study the evolution of mapper graphs that arise from large language models, for instance, to study the mapper graphs during the fine-turning process of transformer-based models across multiple batch updates, following the work of Rathore et al. [58]. Understanding the convergence of mapper graphs would shed light on the behavior of the underlying model, and possibly serve as an indicator for developing early stopping or early exiting strategy during model inference [43].

Given the popularity of mapper graphs in data analysis and visualization [73], we expect our framework for comparing mapper graphs would have wide applicability beyond deep learning, such as shape analysis [9] and scientific visualization in material sciences.

Furthermore, multiresolution Reeb spaces were used recently to study multifield data (i.e., multiple scalar fields defined on the same domain) from computational physics and computational chemistry [56]. Since multiresolution Reeb spaces could be considered as variants of mapper graphs in higher dimensions, it would be interesting to explore extensions of our framework in studying multifield data.

Acknowledgments

BW and YZ were partially supported by grants from NSF IIS 1910733, DMS 2134223, and IIS 2205418.

A Generating Mapper Graphs Using Heat Kernels

For completeness, we review the approach in [22] that generates a mapper graph at a fixed scale using heat kernels. The key idea is to generate a partially overlapping cover of a given graph.

Fix a graph $G = (V, E)$ with a vertex set V. Let L denote the normalized graph Laplacian of G. Initialize a set of visited vertices $P = \emptyset$ and a cover $Q = \emptyset$.

First, starting with a fixed vertex $x \in V \setminus P$, we compute the eigendecomposition of L as $L = \Phi\Lambda\Phi^T$. For a given $t > 0$, we compute the graph heat kernel $K^t := \exp(-tL) = \Phi \exp(-t\Lambda)\Phi^T$. For a vertex $x \in V \setminus P$ and a Dirac delta vector δ_x on x, $v := K^t\delta_x$ encodes the diffusion of a unit mass of heat out from x within time t [22]. We mark the set $\{v \geq \max(v)/2\}$ as visited, and set $P \leftarrow P \cup \{v \geq \max(v)/2\}$. By setting $Q \leftarrow Q \cup \{v \geq \max(v)/4\}$, we have produced a cover element of V centered at x. We then select another vertex $x \in V \setminus P$ and iterate the procedure until $P = V$. Following previous work [22], setting $t := \log_{10}(|V|)$ produces good results, we therefore use $t = 5$ (see Table 1). Such a procedure produces covers that are not too dense but that have overlaps between cover elements. Its 1-dimensional nerve gives rise to a mapper graph.

B Examples on Couplings from Optimal Transport

For examples shown in Figure 6, we configure the parameters associated with hypernetworks $\mathcal{H} = (X, \mu, Y, \nu, \omega)$ and $\mathcal{H}' =$

$(X', \mu', Y', \nu', \omega')$ as follows:

$$\mu = [0.2222, 0.2222, 0.2222, 0.2222, 0.1111]$$
$$\nu = [0.3529, 0.2353, 0.2941, 0.1176]$$
$$\mu' = [0.25, 0.125, 0.25, 0.25, 0.125]$$
$$\nu' = [0.3571, 0.2857, 0.3571]$$
$$\omega = \begin{pmatrix} 0 & 0 & 0.75 & 0.6667 \\ 0 & 0.75 & 0.8 & 0 \\ 0 & 0.75 & 0 & 0.6667 \\ 0.75 & 0 & 0 & 1.4167 \\ 0.8 & 0.75 & 0 & 1.4667 \end{pmatrix}$$
$$\omega' = \begin{pmatrix} 0 & 0 & 0.75 \\ 0 & 0.75 & 0.8 \\ 0 & 0.75 & 0 \\ 0.75 & 0 & 0 \\ 0.8 & 0.75 & 0 \end{pmatrix}$$

REFERENCES

[1] Stanford 3D Scanning Repository. `http://graphics.stanford.edu/data/3Dscanrep/`.

[2] S. G. Aksoy, C. Joslyn, C. O. Marrero, B. Praggastis, and E. Purvine. Hypernetwork science via high-order hypergraph walks. *EPJ Data Science*, 9(1):16, 2020.

[3] U. Bauer, H. B. Bjerkevik, and B. Fluhr. Quasi-universality of Reeb graph distances. In *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[4] U. Bauer, B. Di Fabio, and C. Landi. An edit distance for Reeb graphs. In A. Ferreira, A. Giachetti, and D. Giorgi, editors, *Eurographics Workshop on 3D Object Retrieval*, Eindhoven, The Netherlands, 2016. The Eurographics Association.

[5] U. Bauer, X. Ge, and Y. Wang. Measuring distance between Reeb graphs. In *30th International Symposium on Computational Geometry (SoCG 2014)*, pages 464–474, 2014.

[6] U. Bauer, C. Landi, and F. Memoli. The Reeb graph edit distance is universal. *Foundations of Computational Mathematics*, 21(5):1441–1464, 2020.

[7] U. Bauer, E. Munch, and Y. Wang. Strong equivalence of the interleaving and functional distortion metrics for Reeb graphs. In *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34, pages 461–475, 2015.

[8] K. Beketayev, D. Yeliussizov, D. Morozov, G. Weber, and B. Hamann. Measuring the distance between merge trees. *Topological Methods in Data Analysis and Visualization III*, pages 151–165, 2014.

[9] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392:5–22, 2008.

[10] H. Bjerkevik and M. Botnan. Computational complexity of the interleaving distance. In B. Speckmann and C. D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, pages 13:1–13:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[11] B. Bollen, E. Chambers, J. A. Levine, and E. Munch. Reeb graph metrics from the ground up. *arXiv preprint arXiv:2110.05631*, 2022.

[12] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[13] A. Brown, O. Bobrowski, E. Munch, and B. Wang. Probabilistic convergence and stability of random mapper graphs. *Journal of Applied and Computational Topology*, 2020.

[14] P. Bubenik, V. de Silva, and J. Scott. Interleaving and Gromov-Hausdorff distance. arXiv preprint arXiv:1707.06288, 2018.

[15] G. Cardona, A. Mir, F. Rosselló, L. Rotger, and D. Sánchez. Cophenetic metrics for phylogenetic trees, after Sokal and Rohlf. *BMC Bioinformatics*, 14(1), 2013.

[16] G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[17] M. Carrière and S. Oudot. Local equivalence and intrinsic metrics between Reeb graphs. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77, pages 25:1–25:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[18] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah. Activation Atlas. *Distill*, 4(3):e15, 2019.

[19] N. Chalapathi, Y. Zhou, and B. Wang. Adaptive covers for mapper graphs using information criteria. *IEEE International Conference on Big Data (IEEE BigData)*, 2021.

[20] E. W. Chambers, E. Munch, and T. Ophelders. A family of metrics from the truncated smoothing of Reeb graphs. *37th International Symposium on Computational Geometry (SoCG 2021)*, 189:22:1–22:17, 2021.

[21] F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, and S. Y. Oudot. Proximity of persistence modules and their diagrams. In *25th Annual Symposium on Computational Geometry (SoCG 2009)*, pages 237–246, New York, NY, USA, 2009. Association for Computing Machinery.

[22] S. Chowdhury, T. Needham, E. Semrad, B. Wang, and Y. Zhou. Hypergraph co-optimal transport: Metric and categorical properties. *arXiv preprint arXiv:2112.03904*, 2021.

[23] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[24] V. De Silva, E. Munch, and A. Patel. Categorified Reeb graphs. *Discrete & Computational Geometry*, 55(4):854–906, 2016.

[25] T. K. Dey, F. Mémoli, and Y. Wang. Multiscale mapper: Topological summarization via codomain covers. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 997–1013, 2016.

[26] B. Di Fabio and C. Landi. The edit distance for Reeb graphs of surfaces. *Discrete & Computational Geometry*, 55(2):423–461, 2016.

[27] H. Edelsbrunner and J. Harer. Persistent homology - a survey. In *Surveys on Discrete and Computational Geometry: Twenty Years Later*. American Mathematical Society, 2007.

[28] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical report, Univeristy of Montreal, 2009.

[29] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[30] R. Flamary and N. Courty. POT: Python Optimal Transport library. `https://github.com/rflamary/POT`, 2017.

[31] R. B. Gabrielsson and G. Carlsson. Exposition and interpretation of the topology of neural networks. In *Proceedings of 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1069–1076, 2019.

[32] T. Gebhart, P. Schrater, and A. Hylton. Characterizing the shape of activation space in deep neural networks. *Proceedings of the 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1537–1542, 2019.

[33] C. Geniesse, O. Sporns, G. Petri, and M. Saggar. Generating dynamical neuroimaging spatiotemporal representations (DyNeuSR) using topological data analysis. *Network Neuroscience*, 3(3):763–778, 2019.

[34] W. H. Guss and R. Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. arXiv preprint arXiv:1802.04443, 2018.

[35] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[36] F. Hensel, M. Moor, and B. Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, 2021.

[37] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2018.

[38] F. Hohman, H. Park, C. Robinson, and D. H. P. Chau. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1096–1106, 2020.

[39] M. Kamruzzaman, A. Kalyanaraman, B. Krishnamoorthy, S. Hey, and P. Schnable. Hyppo-X: A scalable exploratory framework for analyzing complex phenomics data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2019.

[40] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). *International Conference on*

Machine Learning, 2018.

[41] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, 2009.

[42] T. Lacombe, Y. Ike, and Y. Umeda. Topological uncertainty: Monitoring trained neural networks through persistence of activation graphs. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 2666–2672, 2021.

[43] S. Laskaridis, A. Kouris, and N. D. Lane. Adaptive inference through early-exit networks: Design, challenges and directions. *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning (EMDL)*, pages 1–6, 2021.

[44] Y. Li, J. Wang, T. Fujiwara, and K.-L. Ma. Visual analytics of neuron vulnerability to adversarial atacks on convolutional neural networks. *ACM Transactions on Interactive Intelligent Systems: Special Issue on Human-Centered Explainable AI*, 2023.

[45] P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3, 2013.

[46] L.-Y. Ma, T. Feng, C. He, M. Li, K. Ren, and J. Tu. A progression analysis of motor features in Parkinson's disease based on the mapper algorithm. *Frontiers in Aging Neuroscience*, 15, 2023.

[47] F. Mémoli. A spectral notion of Gromov–Wasserstein distance and related methods. *Applied and Computational Harmonic Analysis*, 30(3):363–401, 2011.

[48] D. Morozov, K. Beketayev, and G. Weber. Interleaving distance between merge trees. *Proceedings of Topology-Based Methods in Visualization (TopoInVis)*, 2013.

[49] D. Müllner and A. Babu. Python Mapper: An open-source toolchain for data exploration, analysis and visualization. http://danifold.net/mapper, 2013.

[50] E. Munch and A. Stefanou. *The ℓ^∞-cophenetic metric for phylogenetic trees as an interleaving distance*, volume 17 of *Association for Women in Mathematics Series*, pages 109–127. Springer International Publishing, Cham, 2019.

[51] E. Munch and B. Wang. Convergence between categorical representations of Reeb space and mapper. *International Symposium on Computational Geometry*, 2016.

[52] A. Nguyen, J. Yosinski, and J. Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. arXiv:1602.03616, 2016.

[53] M. Nicolau, A. J. Levine, and G. Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011.

[54] G. Peyré, M. Cuturi, and J. Solomon. Gromov-Wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672, 2016.

[55] E. Purvine, D. Brown, B. Jefferson, C. Joslyn, B. Praggastis, A. Rathore, M. Shapiro, B. Wang, and Y. Zhou. Experimental observations of the topology of convolutional neural network activations. *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[56] Y. Ramarmurthi, T. Agarwal, and A. Chattopadhyay. A topological similarity measure between multi-field data using multi-resolution Reeb spaces. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 28:4360–4374, 2022.

[57] A. Rathore, N. Chalapathi, S. Palande, and B. Wang. TopoAct: Visually exploring the shape of activations in deep learning. *Computer Graphics Forum (CGF)*, 40(1):382–397, 2021.

[58] A. Rathore, Y. Zhou, V. Srikumar, and B. Wang. TopoBERT: Exploring the topology of fine-tuned word representations. *Information Visualization*, 2023.

[59] I. Redko, T. Vayer, R. Flamary, and N. Courty. Co-optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, pages 17559–17570, 2020.

[60] M. Reuter, F.-E. Wolter, and N. Peinecke. Laplace–Beltrami spectra as 'Shape-DNA' of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.

[61] B. A. Rieck, M. Togninalli, C. Bock, M. Moor, M. Horn, T. Gumbsch, and K. Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. In *International Conference on Learning Representations (ICLR 2019)*, 2019.

[62] D. Romano, M. Nicolau, E.-M. Quintin, P. K. Mazaika, A. A. Lightbody, H. C. Hazlett, J. Piven, G. Carlsson, and A. L. Reiss. Topological methods reveal high and low functioning neuro-phenotypes within fragile X syndrome. *Human Brain Mapping*, 35(9):4904–15, 2014.

[63] M. Saggar, J. M. Shine, R. Liégeois, N. U. F. Dosenbach, and D. Fair. Precision dynamical mapping using topological data analysis reveals a hub-like transition state at rest. *Nature Communications*, 13(4791), 2022.

[64] M. Saggar, O. Sporns, J. Gonzalez-Castillo, P. A. Bandettini, G. Carlsson, G. Glover, and A. L. Reiss. Towards a new approach to reveal dynamical organization of the brain using topological data analysis. *Nature Communications*, 9(1399), 2018.

[65] H. Saikia, H. P. Seidel, and T. Weinkauf. Extended branch decomposition graphs: Structural comparison of scalar data. *Computer Graphics Forum (CGF)*, 33(3):41–50, 2014.

[66] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *Workshop at International Conference on Learning Representations,*, 2014.

[67] G. Singh, F. Mémoli, and G. Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *Eurographics Symposium on Point-Based Graphics*, pages 91–100, 2007.

[68] R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan. Edit distance between merge trees. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 26(3):1518–1531, 2020.

[69] G. Tauzin, U. Lupo, L. Tunstall, J. B. Pérez, M. Caorsi, A. Medina-Mardones, A. Dassatti, and K. Hess. giotto-tda: A topological data analysis toolkit for machine learning and data exploration. *Journal of Machine Learning Research*, 22:1–6, 2020.

[70] The GUDHI Project. GUDHI User and Reference Manual. https://gudhi.inria.fr/doc/3.3.0/, 2020.

[71] E. F. Touli and Y. Wang. FPT-algorithms for computing Gromov-Hausdorff and interleaving distances between trees. *Proceedings of the 27th Annual European Symposium on Algorithms*, pages 83:1–83:14, 2019.

[72] H. J. van Veen, N. Saul, D. Eargle, and S. W. Mangham. Kepler Mapper: A flexible python implementation of themapper algorithm. *Journal of Open Source Software*, 4(42):1315, 2019.

[73] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum (CGF)*, 40(3):599–633, 2021.

[74] Y. Yao, J. Sun, X. Huang, G. R. Bowman, G. Singh, M. Lesnick, L. J. Guibas, V. S. Pande, and G. Carlsson. Topological methods for exploring low-density states in biomolecular folding pathways. *Journal of Chemical Physics*, 130(14):144115, 2009.

[75] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *Deep Learning Workshop at the 31st International Conference on Machine Learning*, 2015.

[76] Z. Zeng, A. K. Tung, J. Wang, J. Feng, and L. Zhou. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, 2009.

[77] Y. Zhou, N. Chalapathi, A. Rathore, Y. Zhao, and B. Wang. Mapper Interactive: A scalable, extendable, and interactive toolbox for the visual exploration of high-dimensional data. *Proceedings of IEEE 14th Pacific Visualization Symposium (PacificVis)*, 2021.

[78] Y. Zhou, M. Kamruzzaman, P. Schnable, B. Krishnamoorthy, A. Kalyanaraman, and B. Wang. Pheno-Mapper: an interactive toolbox for the visual exploration of phenomics data. *Proceedings of the 12th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB)*, pages 1–10, 2021.

[79] Y. Zhou, Y. Zhou, J. Ding, and B. Wang. Visualizing and analyzing the topology of neuron activations in deep adversarial training. *Topology, Algebra, and Geometry in Machine Learning (TAGML) Workshop at the 40th International Conference on Machine Learning*, 2023.