TopoAct: Visually Exploring the Shape of Activations in Deep Learning

Archit Rathore[1], Nithin Chalapathi[1], Sourabh Palande[1], Bei Wang[1]

[1] School of Computing, Scientific Computing and Imaging (SCI) Institute, University of Utah, USA

Abstract

Deep neural networks such as GoogLeNet, ResNet, and BERT have achieved impressive performance in tasks such as image and text classification. To understand how such performance is achieved, we probe a trained deep neural network by studying neuron activations, i.e., combinations of neuron firings, at various layers of the network in response to a particular input. With a large number of inputs, we aim to obtain a global view of what neurons detect by studying their activations. In particular, we develop visualizations that show the shape of the activation space, the organizational principle behind neuron activations, and the relationships of these activations within a layer. Applying tools from topological data analysis, we present ***TopoAct****, a visual exploration system to study topological summaries of activation vectors. We present exploration scenarios using* ***TopoAct*** *that provide valuable insights into learned representations of neural networks. We expect* ***TopoAct*** *to give a topological perspective that enriches the current toolbox of neural network analysis, and to provide a basis for network architecture diagnosis and data anomaly detection.*

CCS Concepts

• ***Human-centered computing*** $\rightarrow$ ***Visualization toolkits;*** • ***Computing methodologies*** $\rightarrow$ *Learning latent representations;*

1. Introduction

Deep convolutional neural networks (CNNs) have become ubiquitous in image classification tasks thanks to architectures such as GoogLeNet and ResNet. Meanwhile, transformer-based models such as BERT are now the state-of-the-art language model for text classification. However, we do not quite understand how these networks achieve their impressive performance. One main challenge in deep learning is *interpretability*: How can we make the representations learned by these networks interpretable to humans?

Given a trained deep neural network, we address the interpretability issue by probing neuron activations, *i.e.*, the combinations of neurons firings, in response to a particular input image. With a large number of input images for a CNN, we can obtain a global view of what the neurons have learned by studying neuron activations in a particular layer. We aim to address the following questions: What is the shape of the activation space? What is the organizational principle behind neuron activations? And how are the activations related within a layer and across layers? We propose to leverage tools from topological data analysis (TDA) to capture global and local patterns of how a trained network reacts to a large number of input images. In this work:

- We present **TopoAct**, an interactive visual analytics system that uses topological summaries to explore the space of activations in deep learning classifiers for a fixed layer of the network. **TopoAct** leverages the mapper construction [SMC07] from TDA to capture the overall shape of activation vectors for interactive exploration.
- We present exploration scenarios where **TopoAct** helps us discover valuable, sometimes surprising, insights into learned representations of image classifiers such as InceptionV1 [SLJ*15] and ResNet [HZRS16].
- We observe structures in the topological summaries, specifically branches and loops, that correspond to evolving activation patterns that help us understand how a neural network reacts to a large group of images. In particular, we find a correlation between semantically meaningful distinctions and topological separations among images from different classes.
- We further demonstrate the generality and utility of **TopoAct** by applying it to activation vectors obtained from text classifiers such as the BERT family of models. Via a collaboration with a machine learning expert, we provide concrete use cases in the wild where **TopoAct** reveals syntactic and semantic regularities within layers of BERT that help with hypothesis generation in natural language processing (NLP).

Finally, we release an open-source, web-based implementation of the exploration interface on Github: `https://github.com/tdavislab/TopoAct/`; the current system is also available via a public demo link: `https://tdavislab.github.io/TopoAct/`.

We expect **TopoAct** to benefit the analysis and visualization of neural networks by providing researchers and practitioners the ability to probe black box neural networks from a novel topological perspective.

- To the best of our knowledge, **TopoAct** is the first tool that focuses on exploring complex topological structures – branches and loops – within the space of activation vectors. Its exploratory nature helps to inform the global and local organizational principles of activation vectors across different scales.
- **TopoAct** detects if and when activations from different classes become separated, via branches in a fixed layer (see section 6, and in particular, the deer-horse example in section 7), which may be used to inform diagnostic or corrective actions such as selective data augmentation for misclassified inputs or network layer modification for increasing the separation between confounding classes (see section 9).

2. Related Work

We review visual analytics systems for deep learning interpretability as well as various notions of topological summaries.

Visual analytics systems. Visual analytics systems have been used to support model explanation, interpretation, debugging, and improvement for deep learning in recent years; see [HKPC18] for a survey. Here we focus on approaches based on neuron activations for interpretability in deep learning. This line of research attempts to explain the internal operations and the behavior of deep neural networks by visualizing the features learned by hidden units of the network. Erhan *et al.* proposed *activation maximization* [EBCV09], which uses gradient ascent to find the input image that maximizes the activation of the neuron under investigation. It has been used to visualize the hidden layers of a deep belief network [EBCV09] and deep auto-encoders [LRM*13]. Simonyan *et al.* [SVZ14] used a similar gradient-based approach to obtain salience maps by projecting neuron activations from the fully connected layers of the CNN back on to the input space. Building on the idea of activation maximization, Zieler *et al.* [ZF14] proposed a deconvolutional network that reconstructs the input of convolutional layers from its output. Yosinski *et al.* [YCN*15] introduced the *DeepVis* framework that visualizes the live activations produced on each layer of a CNN as it processes images/videos. Their framework also enabled visualizing features in each layer via regularized optimization.

These methods assume that each neuron specializes in learning one specific type of feature. However, the same neuron can be activated in response to vastly different types of input images. Reconstructing a single feature visualization, in such cases, leads to an unintelligible mix of color, scales or parts of objects. To address this issue, Nguyen *et al.* [NYC16] proposed *multifaceted feature visualization*, which synthesizes a visualization of each type of input image that activates a neuron. Another problem with these visualization approaches is the assumption that neurons operate in isolation. This problem is addressed by the *model inversion* method proposed by Mahendran *et al.* [MV15, MV16]. Model inversion looks at the representations learned by the fully connected layers of a CNN, and reconstructs the input from these representations. Kim *et al.* introduced the *TCAV* (Testing with Concept Activation Vectors) framework, which uses directional derivatives of activations to quantify the sensitivity of model predictions to an underlying high-level concept [KWG*18]. All these techniques can help us understand how a single input or a single class of inputs is “seen" by the network, but visualizing activations of neurons alone is somewhat limited in explaining the global behavior of the network. To obtain a global picture of the network, Karpathy [Kar14] used t-SNE to arrange input images that have similar CNN codes (i.e., fc7 CNN features) nearby in the embedding. Nguyen *et al.* [NYC16] projected the training set images that maximally activate a neuron into a low-dimensional space, also via t-SNE. They clustered the images using k-means in the embedded space, and computed a mean image by averaging the images nearest to the cluster centroid.

Carter *et al.* recently proposed the *activation atlas* [CAS*19], which combines feature visualization with dimensionality reduction (DR) to visualize averaged activation vectors with respect to millions of input images. For a fixed layer, the *activation atlas* obtains a high-dimensional activation vector corresponding to each input image. These high-dimensional vectors are then projected onto low-dimensional space via UMAP [MHM18, MHSG18] or t-SNE [MH08]. Finally, feature visualization is applied to averaged activation vectors from small patches of the low-dimensional embedding that allow users to intuitively understand how a particular layer reacts to millions of input images. Hohman *et al.* proposed *SUMMIT* [HPRC20], another framework that summarizes neuron activations of an entire layer of a deep CNN using DR. In addition to aggregated activations, *SUMMIT* also computes neuron influences to construct an *attribution graph*, which captures relationships between neurons across layers.

Activation atlas computes average activation vectors in a low-dimensional embedding, which may introduce errors due to neighborhood distortions. In comparison, our approach aggregates activation vectors in a different manner. Using the *mapper* construction, a tool from TDA, we obtain a topological summary of a particular layer by preserving the clusters as well as relationships between the clusters in the original high-dimensional activation space. Our approach preserves more neighborhood structures since the topological summary is obtained within the high-dimensional activation space. We then study how a particular layer of the neural network reacts to a large number of images through the lens of this topological summary.

Various notions of topological summaries. In TDA, various notions of topological summaries have been proposed to understand and characterize the structure of a scalar function $f : \mathbb{X} \to \mathbb{R}$ defined on some topological space $\mathbb{X}$. Some of these, such as merge trees, contour trees [CSA03], and Reeb graphs [Ree46], capture the behavior of the (sub)level sets of a function. Others, including Morse complexes and the Morse-Smale complexes [EHZ03, EHNP03], focus on the behavior of the gradients of a function. Fewer topological summaries are applicable for a vector-valued function, including Jacobi sets [EH02, BWN*15], Reeb spaces [EHP08, MW16], and their discrete variant, the mapper construction [SMC07]. In this paper, we apply the mapper construction to the study of the space of activations to generate topological summaries suitable for interactive visualization. The mapper construction introduced by Singh *et al.* [SMC07] has seen widespread applications in data science, including cancer research [NLC11, MNL*19], sports analytics [Ala12], gene expression analysis [JCR*19], micro-epidemiology [KGCFR*20], genomic profiling [CZJ*19], and neuroscience [GSPS19, SSGC*18], to name a few; see [PVP17] for an overview. In visualization, topological approaches such as persistent homology and mapper have recently been applied in graph visualization [HWR18, HWSR18, SHW*20].

3. Comparison with t-SNE and UMAP

Various dimensionality reduction (DR) techniques have been proposed to analyze and visualize high-dimensional point cloud data [Cay08, LMW*17]. Among these, t-SNE [MH08] and UMAP [MHM18] are most relevant to our proposed work as they have been used previously for exploring neuron activations [Kar14, NYC16, CAS*19]. In particular, Carter *et al.* [CAS*19] employ both t-SNE and UMAP to project high-dimensional activation vectors to low-dimensions for visual exploration. In comparison with t-SNE and UMAP, the benefit of **TopoAct** is two-fold.

- **TopoAct** provides a global, graph-based representation of the space of activations, which explicitly summarizes the organizational principles (clusters and cluster relations) behind neuron activations. t-SNE and UMAP detect structures visible in the *low-dimensional* embedding, whereas **TopoAct** captures complex topological structures – loops and branches – in the original *high-dimensional* space.
- Whereas t-SNE and UMAP focus on preserving proximities within local neighborhoods, **TopoAct** explicitly reveals branches and loops that are not necessarily visible via t-SNE/UMAP. These topological structures can be used to guide refined, local structural analysis (section 6).

The kNN (k-nearest neighbor) graph constructed by UMAP can be considered as a topological representation of the high-dimensional data [MHM18]. However, it only approximately preserves the *connectivity* among points within local patches of the manifold, and does not capture structures such as loops or branches. **TopoAct** addresses this important challenge by utilizing the mapper construction.

In addition, several DR techniques have been developed recently [SMVJ09, WSPVJ11, YZR*18] that explicitly preserve loops and branches; however, none of them give a global summary of all such structures in a single visualization. In studying the shape of the space of images, Lee *et al.* [LPM03], Carlsson *et al.* [CISZ08], and Xia [Xia16] studied the global topological structures of patches from natural images; they used different topological tools (such as persistence homology) and focused on different data problems (such as studying the global structure of natural images from a database) in comparison to **TopoAct**. Finally, a few recent works applied topological techniques in the study of activations. Gebhart *et al.* [GSH19] computed persistent homology over the activation structure of neural networks. In particular, they characterized the topological structure of the neural network architecture when viewed as a graph with edge weights provided by activations. Gabella *et al.* [GAES19] used both persistent homology and the mapper construction to study the parameter space of neural networks (i.e., weight matrices) during training.

4. Technical Background

We review technical background on the mapper construction and neural network architecture. We delay the discussions on activation vectors and feature visualization until section 5.

Mapper graph on point cloud data. We give a high-level description of the framework by Singh *et al.* [SMC07] in a point cloud setting. Given a high-dimensional point cloud $\mathbb{X} \subset \mathbb{R}^d$ equipped with a function f on $\mathbb{X}$, $f: \mathbb{X} \to \mathbb{R}$, the mapper construction provides a topological summary of the data for compact representation and exploration. It utilizes the topological concept known as the *nerve of a covering* [Ale28].

An *open cover* of $\mathbb{X}$ is a collection $\mathcal{U} = \{U_i\}_{i\in I}$ of open sets in $\mathbb{R}^d$ with an index set I such that $\mathbb{X} \subset \bigcup_{i\in I} U_i$. Given a cover $\mathcal{U}$ of $\mathbb{X}$, the 1-dimensional *nerve* of $\mathcal{U}$, denoted as $\mathcal{N}_1(\mathcal{U})$, is constructed as follows: A finite set $\{i, j\} \subset I$ (i.e., an edge) belongs to $\mathcal{N}_1(\mathcal{U})$ if and only if the intersection of U_i and U_j is nonempty; if the set $\{i, j\}$ belongs to $\mathcal{N}_1(\mathcal{U})$, then any of its subsets (i.e., the point i and the point j) is also in $\mathcal{N}_1(\mathcal{U})$. See Figure 1 for an example. A cover $\mathcal{U} = \{U_1, U_2, U_3, U_4\}$ that contains open rectangles is given for a 2-dimesional point cloud $\mathbb{X}$ in (a). The 1-dimensional nerve of $\mathcal{U}$, $\mathcal{N}_1(\mathcal{U})$, is shown in (c). For instance, there is an edge $\{1,2\}$ that belongs to $\mathcal{N}_1(\mathcal{U})$ since $U_1 \cap U_2 \neq \emptyset$.

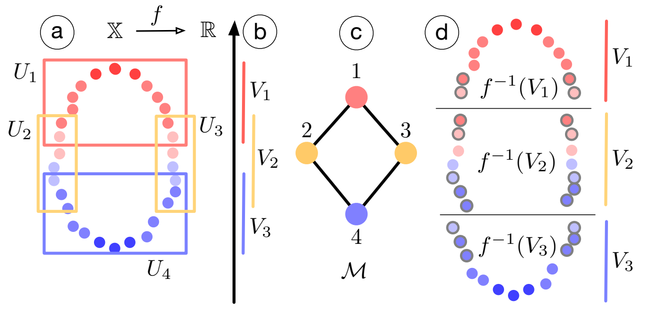


Figure 1: *A simple example of a mapper graph on a point cloud.*

For the mapper construction, we start with a finite cover $\mathcal{V} = \{V_j\}_{j\in J}$ (J being an index set) of the image $f(\mathbb{X}) \subset \mathbb{R}$ of f, such that $f(\mathbb{X}) \subseteq \bigcup_j V_j$, see Figure 1(b). Since f is a scalar function, V_i is an open interval in $\mathbb{R}$. Let $\mathcal{U}$ denote the cover of $\mathbb{X}$ obtained by considering the clusters of points induced by points in $f^{-1}(V_j)$ for each j, see (d). The 1-dimensional nerve of $\mathcal{U}$, denoted as $\mathcal{M} := \mathcal{N}_1(\mathcal{U})$, is called the *mapper graph* of $(\mathbb{X}, f)$. $\mathcal{M}$ is a multiscale representation that serves as a topological summary of of $(\mathbb{X}, f)$, *i.e.*, the point cloud $\mathbb{X}$ equipped with a function f. Its construction relies on three parameters: the function f, the cover $\mathcal{V}$, and the clustering algorithm.

The function f plays the role of a *lens*, through which we look at the data, and different lenses provide different insights [BGSF08, SMC07]. An interesting open problem for the mapper construction is how to define topological lenses beyond best practices or a rule of thumb [BGSF08, BMMP03]. In practice, functions such as height, distance from the barycenter, surface curvature, integral geodesic distances, and geodesic distances from a source point have all been used as lenses [BGSF08]. In this paper, we use the L_2 norm of the activation vectors as the lens, although other options are possible (see the supplementary material for a discussion of L_2-norm as a lens function).

The cover $\mathcal{V}$ of $f(\mathbb{X})$ consists of a finite number of open intervals as cover elements, $\mathcal{V} = \{V_j\}_{j \in J}$. A common strategy is to use uniformly sized overlapping intervals. Let n be the number of intervals and p the amount of overlap between adjacent intervals. Adjusting these parameters increases or decreases the amount of aggregation $\mathcal{M}$ provides.

We compute the clustering of the points lying within $f^{-1}(V_i)$ and connect the clusters whenever they have nonempty intersection. A typical algorithm to use is DBSCAN [EKSX96], a density-based clustering algorithm; it groups points in high-density regions together and makes points that lie alone in low-density regions outliers. The algorithm requires two input parameters: *minPts* (the number of samples in a neighborhood for a point to be considered as a core point), and ε (the maximum distance between two samples for one to be considered in the neighborhood of the other).

Figure 1 illustrates a mapper graph construction for a dataset $\mathbb{X}$ sampled from a noisy circle. The function (lens) used is $f(x) = ||x - p||_2$, where p is the lowest point in the data. $\mathbb{X}$ is colored by the value of the function. We divide the range of the f into three intervals $\{V_1, V_2, V_3\}$ with a 30% overlap. For each interval, we compute the clustering of the points lying within the domain of the lens function restricted to the interval $f^{-1}(V_i)$, and connect the clusters whenever they have a nonempty intersection. $\mathcal{M}$ is the mapper graph whose nodes are colored by the index set, and it preserves the shape of the point cloud data – a loop.

InceptionV1 architecture. We give a high-level overview of InceptionV1 [SLJ*15] (GoogLeNet), the neural network architecture employed in this paper. However, our framework is not restricted to the specific architecture of a neural network. InceptionV1 is a CNN that won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) for image classification in 2014. It was trained on ImageNet ILSVRC [DDS*09]. ImageNet consists of over 15 million labeled high-resolution images with roughly $22K$ classes/categories. ILSVRC takes a subset of ImageNet of around $1K$ images in each of $1K$ classes, for a total of 1 million training images, $50K$ validation images, and $100K$ testing images. The highlights of InceptionV1 architecture include the use of 1×1 convolutions, inception modules, and global average pooling. The 1×1 convolution from NIN (networks in networks) [LCY14] is used to reduce dimensionality (and computation) prior to expensive convolutions with larger image patches. A new level of organization is introduced in the form of the *inception module*, which combines different types of convolutions for the same input and stacking all the outputs on top of each other. InceptionV1 contains nine *inception modules*, each composed of multiple convolution layers.

The demo version of **TopoAct** explores the activations of the last layer of each inception module. The module names such as *mixed3a*, *mixed3b* are shortened as *3a*, *3b*, etc. This choice is well aligned with previous literature on visual exploration of InceptionV1 [OMS17, OSJ*18, HPRC20, CAS*19].

ResNet architecture. To demonstrate the generality of topological structures observed across different neural network architectures, we also apply **TopoAct** to activation vectors from a ResNet model. Residual Network or ResNet [HZRS16] was one of first neural network architectures that enabled training extremely deep neural networks with up to $1K$ layers. A neural network N_D of depth D is a subnetwork of any network N_{D+K} of depth $D + k, k > 0$. Theoretically, N_{D+K} should be capable of learning any function that N_D can learn by setting the extra k layers to an identity mapping, and thus perform at least as well as the smaller network. In practice, however, increasing layers beyond a certain depth leads to a sharp degradation in performance (higher training error and lower classification accuracy on the test set) even when normalization schemes are used for both initialization and intermediate representations. ResNet overcomes this problem by adding "shortcut" connections to a layer that adds the output from layer k to the input of layer $k+i$ where i is usually 2.

In our experiments, we have implemented ResNet-18, a residual network with 18 layers following the layer specifications in [HZRS16]. With 200 training epochs, it achieves a classification accuracy of 93.24% on CIFAR-10 and 91.87% on CIFAR-100 datasets [KH09].

5. Methods

We describe data analytic components of **TopoAct**. First, for a chosen layer of a neural network model (such as InceptionV1), we obtain activation vectors as high-dimensional point clouds for topological data analysis. Second, we construct mapper graphs from these point clouds to support interactive exploration. The nodes in the mapper graphs correspond to clusters of activation vectors in high-dimensional space, and the edges capture relationships between these clusters. Third, for each node (cluster) in the mapper graph, we apply feature visualization to individual activation vectors in the cluster and to the averaged activation vector.

Obtaining activation vectors as point clouds. The activation of a neuron is a nonlinear transformation of its input. To start, we fix a trained model (*i.e.*, InceptionV1) and a particular layer (*e.g.*, *4c*) of interest. We feed each input image to the network and collect the activations, *i.e.*, the numerical values of how much each neuron has fired with respect to the input, at a chosen layer. Since InceptionV1 is a CNN, a single neuron does not produce a single activation for an input image, but instead a collection of activations from a number of overlapping spatial patches of the image. When an entire image is passed through the network, a neuron will be evaluated multiple times, once for each patch of the image. For example, a neuron within layer *4c* outputs 14×14 activations per image (for 14×14 patches). To simplify the construction, in our setting, we randomly sample a single spatial activation from the 14×14 patches, excluding the edges to prevent boundary effects. For $300K$ images, this gives us $300K$ activation vectors for a given layer. Figure 2 illustrates what we mean by an activation vector. The dimension of an activation vector depends on the number of neurons in the layer. For instance, layers *3a*, *3b*, and *4a* have 256, 480, and 512 neurons, respectively, producing point clouds of corresponding dimensions.

Constructing mapper graphs from activation vectors. Given a point cloud of activation vectors, we now compute a mapper graph as its topological summary. Each node in the mapper graph represents a cluster of activation vectors, and an edge connects two nodes

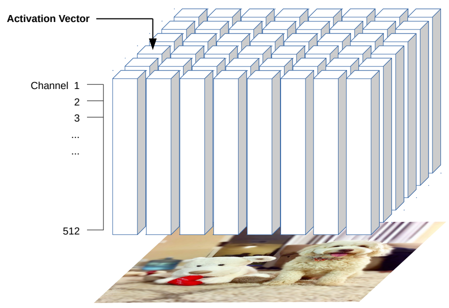


Figure 2: *Each activation vector is a vector of spatial activations across all channels.*

if their corresponding clusters have a nonempty intersection. Our mapper graphs use the L_2-norm of the activation vector as the lens function. We set $n = 70$ cover elements with $p = 20\%, 30\%$, and 50% as the amount of overlap. We use DBSCAN [EKSX96] as the clustering algorithm, with minimum points per cluster $minPts = 5$. For the ε parameter of the DBSCAN algorithm, which defines core points, we use two variations in our experiments. In the first variation, we use a fixed $\varepsilon = 1,000$ estimated using the distribution of pairwise distances at the middle layer. In the second variation, we set ε adaptively for each layer, employing the procedure proposed in [EKSX96]. Specifically, we generate an approximate kNN graph, sort the distances to the 5-th nearest neighbor, and select an ε based on the location of a “elbow” when these distances are plotted [EKSX96]. This way, the ε value is more adaptive to the activation space of each layer. Our *adaptive* ε values are 830 for *3a*, 1070 for *3b*, 1750 for *4a*, 1630 for *4b*, 1330 for *4c*, 980 for *4d*, 775 for *4e*, 790 for *5a*, and 260 for *5b*.

These parameter configurations give rise to six datasets currently deployed in our live demo. Each dataset contains nine mapper graphs (across nine layers of InceptionV1) constructed by a particular set of parameters associated with the mapper construction. The mapper graphs are named according to these parameters. In particular, each name starts with "overlap-*x*" where *x* is 20, 30, or 50 to denote 20%, 30%, or 50% overlap, respectively. The second half of the name consists of "epsilon-*x*" where *x* is either "fixed" or "adaptive", indicating whether ε was fixed ($= 1000$) or set adaptively for each layer. For example, *overlap-50-epsilon-fixed* is the dataset containing mapper graphs of nine layers generated using $p = 50\%$ with fixed $\varepsilon = 1000$.

Applying feature visualization to activation vectors. Activation vectors are high-dimensional abstract vectors. We employ *feature visualization* to transform them into a more meaningful semantic representation using techniques proposed by Olah *et al.* [OMS17, OSJ*18]. Whereas the neural network transforms an input image into activation vectors, feature visualization goes in the opposite direction.

Given an activation vector $h_{x,y}$ at a spatial position (x,y), feature visualization synthesizes an idealized image that would have produced $h_{x,y}$ via an iterative optimization process. Normally, this synthesis is achieved by maximizing the dot product $h_{x,y} \cdot v$ of the vector $h_{x,y}$ with the direction v. However, the vector v that maximizes the dot product can have a large orthogonal component. To counter this, following [CAS*19], the dot product is multiplied with a cosine similarity, putting greater emphasis on the angle between vectors. The optimization process, which is similar to back propagation, begins with a random noise image. Using gradient descent, this image is iteratively tweaked to maximize the following objective: $(h_{x,y} \cdot v)^{n+1}/(||h_{x,y}||\,||v||)^n$. Subsequently, a *transformation robustness* regularizer [OMS17] is used, which applies small stochastic transformation (jitter, rotate or scale) to the image before applying the optimization step. Max-pooling can introduce high frequencies in the gradients. To tackle this problem, the gradient descent is performed in Fourier basis with frequencies scaled to have equal energy, which is equivalent to whitening and de-correlating the data.

Applying feature visualization to all 300K activation vectors results in corresponding images that are likely to produce such activations, which we call *activation images*. Once we obtain a mapper graph, we also apply feature visualization to the averaged activation vector per cluster to obtain an *averaged activation image* for each cluster. However, feature visualization is not without drawbacks; due to the optimization process and the size of each cluster, it can generate abstract images that remain hard to interpret.

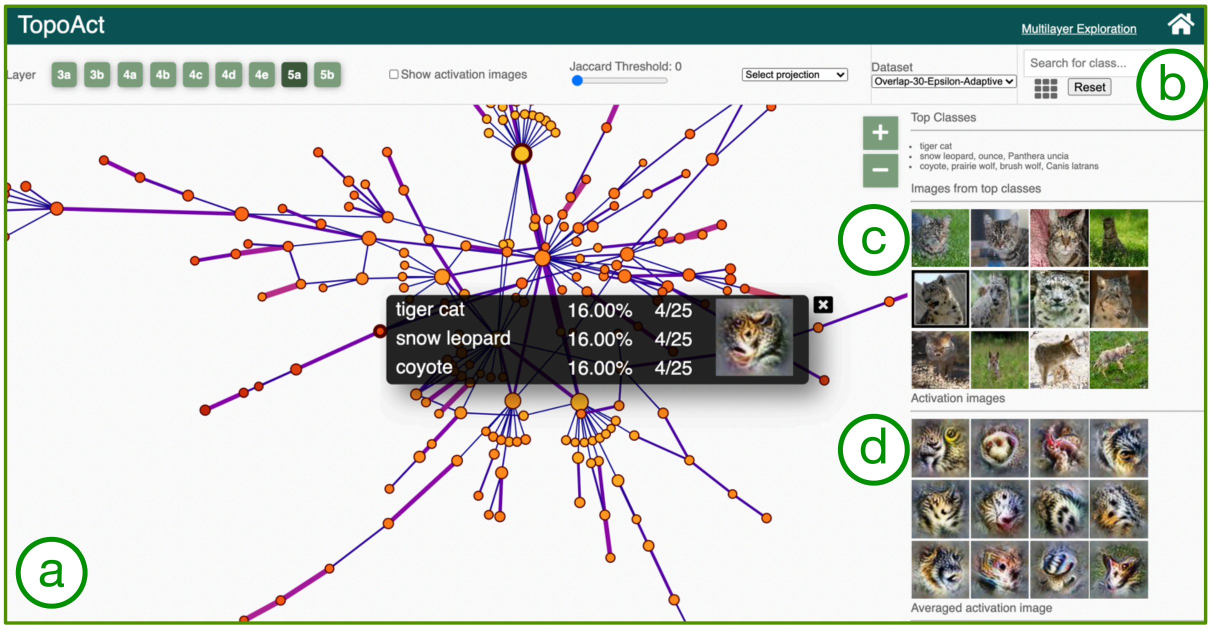
6. Exploring the Shape of Activations from InceptionV1

We present the user interface of **TopoAct**, an interactive system used to explore the organizational principles of neuron activations in deep learning image classifiers. We use InceptionV1 trained on 1 million ImageNet images across 1K classes. We obtain activation vectors of 300K images (300 images per class) via the trained model. The **TopoAct** user interface, Figure 3, contains two exploration modes: single-layer exploration and multilayer exploration. We present various exploration scenarios using **TopoAct** under the single-layer exploration mode that provide valuable insights into learned representations of InceptionV1. See the supplementary materials for a detailed description on the interface, implementation, and multilayer exploration mode.

For single-layer exploration, the main takeaway from these scenarios is that **TopoAct** captures specific topological structures, in particular, branches and loops, in the space of activations that are hard to detect via classic DR techniques; such features offer new perspectives on how a neural network “sees" the input images. The topological features identified by **TopoAct** can also be used to guide refined, local shape analysis of the space of activations.

6.1. Discovering Branches from the Space of Activations

We provide several examples of interesting topological structures that capture relationships between activations during single-layer exploration. Two main types of topological structures unique to our framework, branches and loops, differentiate **TopoAct** from prior work (*e.g.*, [HPRC20, CAS*19]). Topologically, branches in a graph represent bifurcations, thus separations, among image classes. Although we observe variations of similar features along a specific branch, different branches may capture distinct, sometimes unrelated, features. In order to illustrate the insights gained through



- tiger cat
- snow leopard, ounce, Panthera uncia
- coyote, prairie wolf, brush wolf, Canis latrans

tiger cat	16.00%	4/25
snow leopard	16.00%	4/25
coyote	16.00%	4/25

Figure 3: *With **TopoAct**, users can interactively explore topological summaries of activations in a neural network for a single layer and across multiple layers. Users can investigate activations at a particular layer under the single-layer exploration mode. The mapper graph panel (a) provides a graph-based topological summary of the activation vectors from* 300*K images across* 1*K classes, where each node of the mapper graph represents a cluster of activation vectors, and each edge encodes the relationships between the clusters. The size of a node is proportional to the number of activations, and the color is mapped to the average* L_2 *norm of activations in the cluster. The edge thickness is proportional to the Jaccard index between two nodes. The control panel (b) supports the selection of parameters for the mapper construction and visual encoding. For a chosen cluster in the mapper graph, the data example panel (c) gives textual description of the top three classes within the cluster together with (up to) five image examples from each top class. The feature visualization panel (d) applies feature inversion to generate idealized images, called activation images, for individual activation vectors (obtained from data examples) and for an averaged activation vector within a chosen cluster.*

user interactions and views between different parts of the system, figures in this section have average activation images (computed from the average of all activations in a node) overlaid on the nodes of the mapper graph.

Leg-face bifurcation. Our first example of a bifurcation comes from the layer *4c* of the ImageNet dataset (*overlap-30-epsilon-adaptive*). Figure 4 shows two branches emerging from node (d) in the mapper graph; we refer to such a node as the *branching node*. Node (d) is composed of 381 activation vectors. The top three classes within node (d) are **rugby ball**, **Indian elephant**, and **wig**. Although this clustering of class labels appears to be random, the mapper graph coupled with averaged activation images reveals interesting insights.

As illustrated in Figure 4, the left branch appears to capture the leg of an animal. The top three classes represented in all the clusters within this branch include various breeds of dogs and horses (a). The right branch appears to capture features that resemble (distorted) human faces. Although the class names associated with clusters along the right branch may not suggest a relation to humans faces, the data examples associated with these clusters reveal that all the top classes in the right branch contain images with humans, most of which include close-ups of faces (b, c). Returning to the branching node (d), upon closer inspection, we see that it contains images of **rugby** players and **elephants** that include both leg and face features, whereas **wig** images also include human faces. Therefore, the activation space bifurcates at the branching node to further differentiate between leg and face features.

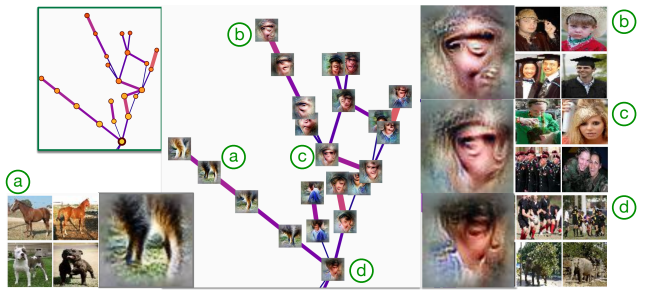


Figure 4: *Leg-face bifurcation. Configuration: layer 4c, Euclidean norm,* 70 *intervals,* 30% *overlap, adaptive* ε *for DBSCAN.*

We further compare **TopoAct** against t-SNE and UMAP projections. For t-SNE, we use the Multicore-TSNE [Uly16] Python library and set the perplexity parameter to be 50 following the parameter choice used in the *activation atlas* [CAS*19]. The

UMAP projection is performed using its official Python implementation [MHSG18] with 20 nearest neighbors and a minimum distance of 0.01. As illustrated in Figure 5, we select nodes that are involved in the leg-face bifurcation and highlight their corresponding activation vectors in the t-SNE and UMAP projections. In particular, neither t-SNE nor UMAP reveals a bifurcation as the activation vectors in the projection are scattered over the entire projection.

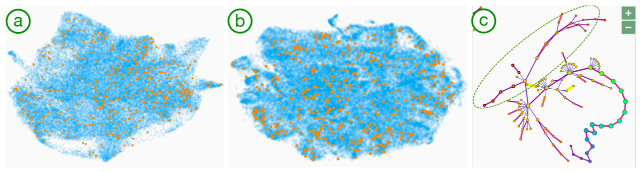


Figure 5: *Highlighting activation vectors that belong to the leg-face bifurcation (c) as orange points in the UMAP (a) and t-SNE projection (b).*

Bird-mammal bifurcation. Our second example of a branch comes from the layer *5a* of the ImageNet dataset (*overlap-30-epsilon-fixed*). Figure 6 shows two branches emerging from the branching node (a), which is composed of 398 activation vectors. It contains images of both birds and dogs such as **oystercatcher** and **Brittany spaniel**, and the averaged activation image of the branching node appears to be a combination of the left profile of bird faces and right profile of the dog-like faces. Upon further investigation, the bottom branch that contains nodes (d), (e), and (f) focuses on the features of bird faces: profile views composed of the left eye and beak, with variations of color and textures as we move along the branch. The clusters in this branch include mainly bird species such as **bee eater**, **robin**, and **lorikeet**. The variations in the captured features and corresponding data samples can be seen in nodes (d), (e), and (f). The clusters in the top branch, on the other hand, appear to capture features of mammalian faces: eyes and snouts, with variations in color and texture. This branch primarily consists of images from classes of mammals, including various dog breeds, **wolves**, and **foxes**.

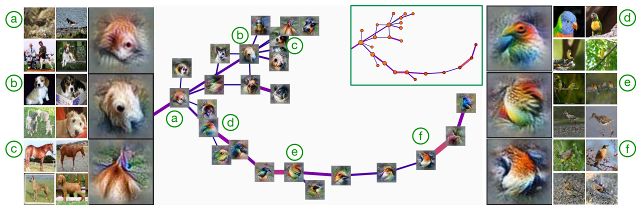


Figure 6: *Bird-mammal bifurcation. Configuration: layer 5a, Euclidean norm,* 70 *intervals,* 30% *overlap, fixed* ε *for DBSCAN.*

Wheel-tread bifurcation. Our third example comes from the layer *4c* of the ImageNet dataset (*overlap-30-epsilon-adaptive*). As illustrated in Figure 7, the branching node (a) is a small cluster of size 136. All the clusters in this example contain images of various types of automobiles, for example **minibus**, **police van**, **fire engine**, **limousine**, etc. The branching node (a) appears to capture what looks like the wheel of a vehicle - a dark round shape with tread-like pattern. The two branches appear to focus on one of these two features. Whereas the left branch focuses on the dark round swirling patterns of automobile wheels (b, c, d), the right branch appears to focus more on the tread-like patterns and textures (e, f).

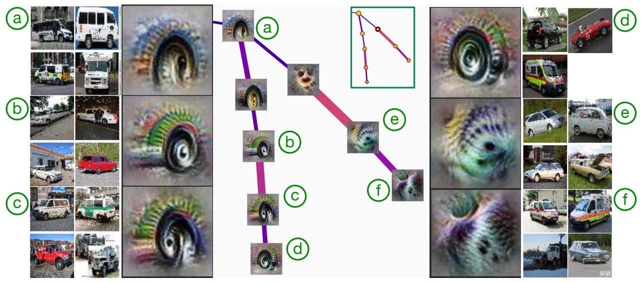


Figure 7: *Wheel-tread bifurcation. Configuration: layer 4c, Euclidean norm, 70 intervals, 30 % overlap, adaptive* ε *for DBSAN.*

6.2. Exploring Loops from the Space of Activations

Branches capture bifurcations in the types of features across different objects, but some loops seem to capture different aspects of the same underlying object.

Fur-nose-ear-eye loop of mammals. Our first example comes from layer *4d* of the ImageNet dataset (*overlap-30-epsilon-fixed*). Figure 8 shows a loop created by six clusters. The top classes in all six clusters include various dog breeds and a variety of foxes. All these clusters seem to capture different features (*i.e.*, body parts) related to these animals. Based on feature visualization, the leftmost cluster appears to capture the color patterns and the texture of the fur from the body (a). Going clockwise, the next cluster also captures the color and texture of the fur but from a different body part, possibly the fur and hair surrounding the nose, suggested by the dark spot and the swirling pattern (b). The next two clusters (c, d) appear to capture animal ears. The averaged activation image captured by the cluster (e) is not as clearly attributable to a specific part of an animal's body. As can be observed in (e), this cluster consists of images from a larger variety of animals, from **foxes** to **Siamese cats** and **hogs**. As a result, the corresponding averaged activation image is a mixture of various colors and slightly different textures. The last cluster (f) appears to capture the eyes and noses of the animals. We can observe in (f) that the cluster contains front and side views of dog heads.

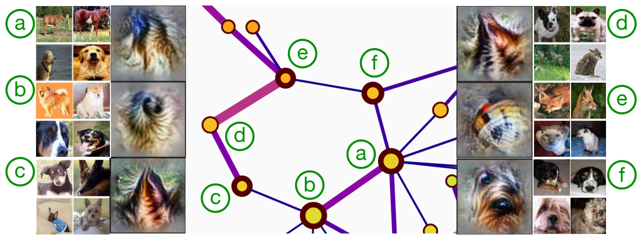


Figure 8: *Fur-nose-ear-eye loop. Configuration: layer 4d,* 70 *intervals,* 30% *overlap, fixed* ε *for DBSCAN.*

Face-body-leg loop of birds. Our next example originates from layer *5a* of the ImageNet dataset (*overlap-30-epsilon-adaptive*).

Figure 9 shows six clusters creating the loop. The top three classes of all the clusters in the loop consist of bird species, and similar to the previous example, the averaged activation images show us different features (body parts) of the birds captured by these clusters. Clusters (c, d, e) on the top of the loop appear to capture the left profile views of the bird faces with the left eye and the beak identifiable in the averaged activation images. These clusters are, in fact, composed of images of birds. The variation in the color of birds (from red to blue, and to brown) is reflected in the corresponding activation images (b, c, d, e). Clusters (a, b, f) on the bottom of the loop appear to capture the body and legs along with a feathered texture, although not as clearly as the other three clusters. As can be seen, cluster (f) also includes images of **leopard** and **jaguar** mixed with images of birds (**partridge** and **ruffed grouse**) for the representation of texture.

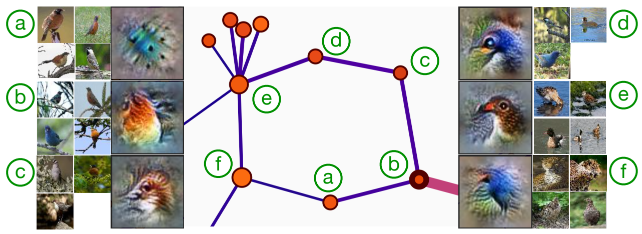


Figure 9: *Face-body-leg loop of birds. Configuration: layer 5a,* 70 *intervals,* 30% *overlap, adaptive* ε *for DBSCAN.*

6.3. Studying Global Views of Activation Spaces

We now explore the global view of an activation space using the single-layer exploration mode. Instead of focusing on a single type of topological structure such as loops or branches, we investigate the distribution of topological structures. As illustrated in Figure 10, we investigate the distribution of branches within the largest connected component of a mapper graph, at layer *5b* of the ImageNet dataset (*overlap-30-epsilon-adaptive*). We pay special attention to branching nodes with high degrees (a, b). These branching nodes, in some sense, serve as "anchors" or "hubs" of the underlying space of activations. We make a few interesting, though speculative, observations. For each of the two branching nodes in

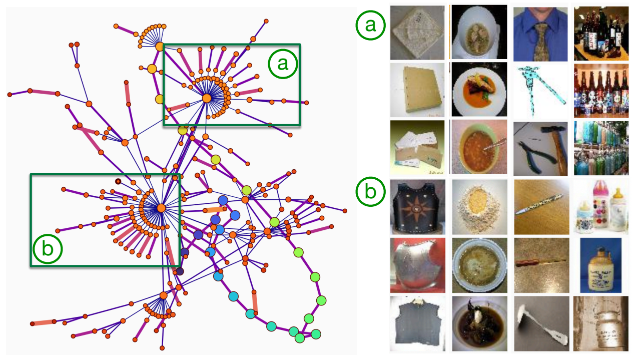


Figure 10: *A global view of a mapper graph for a fixed layer. Configuration: layer 5b,* 70 *intervals,* 30% *overlap, adaptive* ε *for DBSCAN.*

(a) and (b), a mixture of geometric and texture-based images contributes to the representation of the node. Nodes immediately adjacent to the branching node (a), *i.e.*, those that form branches that merge at node (a), contain geometric objects that are square-shaped (**envelop**, **bath towel**), circle-shaped (**bowl**, **pasta**), pointy-shaped (**tie**, **hammer**), and bottle-shaped (**beer**). Nodes immediately adjacent to the branching node (b) have other objects that serve similar purposes, including square-shaped (**vest**, **cuirass**), circle-shaped (**dough**, **mashed potato**), pointy-shaped (**ladle**, **ball point**), and bottle-shaped (**milk cans**, **whiskey jug**). However, (a) and (b) seem to draw these geometric shapes from (almost completely) different classes of images, which may indicate a level of self-similarity within the space of activations that requires further investigation.

6.4. Refined Analysis of Topological Structures

We can utilize interesting topological structures identified by **TopoAct** – branches and loops – to obtain topologically meaningful subsets of the activation vectors for further analysis. We present some examples of **TopoAct**-guided principal component analysis (PCA) with the following procedure. We first identify all nodes that form a branch or a loop within a mapper graph. We then extract activation vectors (as high-dimensional points) that map to these nodes. Next, we apply PCA to these points and project them to a 2-dimensional plane.

Consider the leg-face bifurcation from Figure 4. Figure 11(a) shows the PCA projection of all points that participate in the bifurcation. The red points belong to the activation vectors from the "face" branch and the blue points belong to the "leg" branch. Similarly, for the wheel-tread bifurcation from Figure 7, Figure 11(b) illustrates the PCA projection of its associated points. For both examples, we could easily observe that points from the two branches lie along two distinct directions.

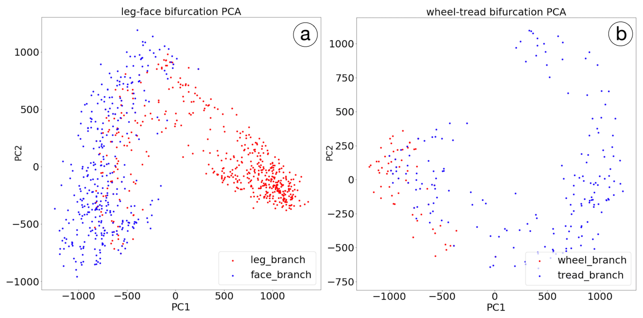


Figure 11: *PCA of the activation vectors that belong to (a) the leg-face bifurcation and (b) the wheel-tread bifurcation.*

For comparative purposes, we apply PCA to all 300K points from layer *4c* and highlight those from the leg-face bifurcation. As shown in Figure 12(a), there are no observable branching or clustering structure within this global projection. To verify that the leg-face bifurcation is not spurious, we construct a minimal bounding box and identify around 86K neighboring points in the space of activations. We apply PCA to these 86K points and observe that points from the leg-face bifurcation form clusters that are separable from their neighboring points; see Figure 12(b), which confirms that the leg-face bifurcation detected by **TopoAct** is not spurious.

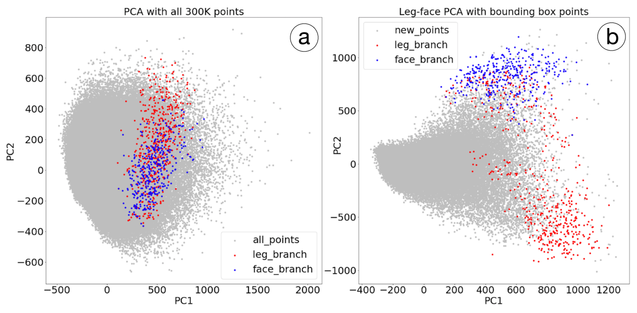


Figure 12: *PCA applied to the entire set of activation vectors from layer* 4c *(a), and to the activation vectors in the neighborhood of leg-face bifurcation (b). We do not see any branching or clustering structure in (a), while (b) reveals the leg-face bifurcation with respect to its neighboring points.*

This example demonstrates the advantage of using **TopoAct** in combination with classic DR techniques (such as PCA, t-SNE, and UMAP) to perform refined shape analysis of the space of activations. Although the total number of activation vectors in our dataset is large, the number of significant branches and loops is relatively small, which leads us to hypothesize that a layer is particularly well-trained to identify certain directions in the activation space. As shown here, **TopoAct** can help us identify these directions.

7. Applying TopoAct to ResNet Trained on CIFAR

To demonstrate the generality of our framework, we provide additional experiments using ResNet trained on the CIFAR-10 and CIFAR-100 datasets [KH09]. Both datasets consist of the same set of 60K color images of dimension 32×32, with 50K training images and 10K test images. CIFAR-10 has 10 images classes with 6K images per class, and CIFAR-100 has 100 image classes with 600 images per class. The class labels in CIFAR-10 are coarser, such as **automobiles** and **mammals**; whereas classes in CIFAR-100 are finer, such as **bicycle**, **bus**, **beaver**, and **hamster**. We demonstrate that the insights provided by **TopoAct** are not specific to a particular dataset or a particular network architecture. We give a few exploration scenarios involving branches by applying **TopoAct** to ResNet-18 trained on the CIFAR-10 dataset; such examples are similar to those described in section 6. We encourage readers to explore further with our open-source online demo.

Horse-deer bifurcation. Our first example is a horse-deer bifurcation from the last layer *4.1.bn2* of the CIFAR-10 dataset, as illustrated in Figure 13. The left branch that contains nodes (b) and (c) corresponds to images of deer, whereas the right branch with nodes (d), (e) and (f) corresponds to images of horses. The branching node (a) contains images of both horses and deer. In addition, none of the earlier layers show such a clear bifurcation between the horse and deer classes. **TopoAct** reveals the layer at which the network first begins to differentiate between these two classes. Such insights would make **TopoAct** a useful diagnostic tool for deep learning researchers (see section 9).

Frog-cat bifurcation. Similarly, our second example is a frog-cat bifurcation from the last layer *4.1.bn2* of the CIFAR-10 dataset, as illustrated in Figure 14. Here, the branching node (a) contains images of frogs and cats. It then bifurcates into a left branch (with nodes (b) and (c)) that contains only images of cats, and a right branch (with nodes (d), (e), and (f)) that contains only images of frogs. Even though these are very different types of animals (mammals vs. amphibians), they share similar postures.

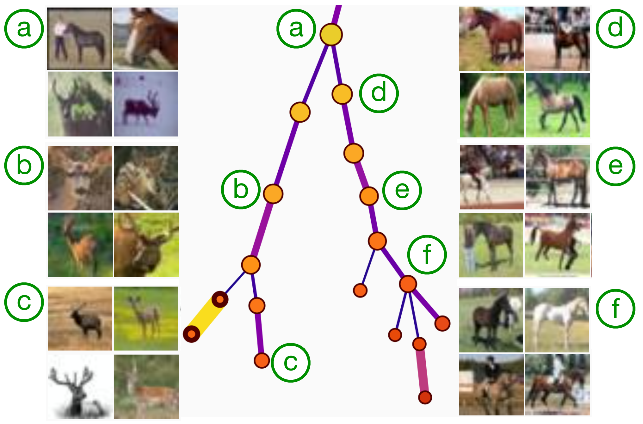


Figure 13: *Horse-deer bifurcation. Configuration: layer 4.1.bn2,* 40 *intervals,* 20% *overlap.*

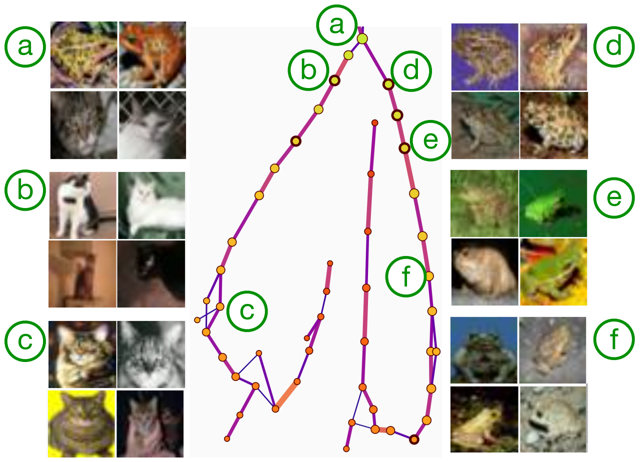


Figure 14: *Frog-cat bifurcation. Configuration: layer 4.1.bn2,* 100 *intervals,* 40% *overlap.*

8. Applying TopoAct to BERT Neural Network for NLP

To demonstrate the utility of **TopoAct** with concrete use cases in the wild, we have collaborated with a machine learning (ML) expert in Natural Language Processing (NLP), Dr. Vivek Srikumar from the University of Utah. We apply **TopoAct** to activation vectors obtained from the BERT (Bidirectional Encoder Representations from Transformers) family of models, which is the default representation of text for a variety of NLP tasks.

Although BERT and similar models are undoubtedly helpful in improving predictive accuracy, it is not immediately clear why it helps. This problem arises because BERT embeddings (*i.e.*, activation vectors) consist of a collection of high-dimensional vectors for every sentence, and this high-dimensional space is not easy to explore. The highlight of our collaboration is that **TopoAct** presents an opportunity in this context by revealing the various syntactic and semantic regularities that each layer of BERT captures, some of which are surprising but interpretable for the ML expert.

In addition, **TopoAct** shows potential directions for improving these representations, for instance, targeting a specific downstream NLP task. For example, in addition to some clear patterns of concepts captured via topological branches, we also see clusters and relationships between them that appear noisy, yet stable. The existence of these suggests that there may be opportunities for developing topologically aware regularization techniques for training BERT-like models by imposing additional constraints such that the mapper graphs resulting from **TopoAct** should exhibit certain structural properties, which is left for future work.

8.1. BERT and Activation Datasets

BERT and other transformers-based language models have recently found widespread application as the go-to methods for many tasks in NLP such as sentiment analysis, sentence classification, and domain-specific language modeling [BLC19, LYK*20]. Jawahar *et al.* [JSS19] explored and established that contextual embeddings from BERT do indeed encode syntactic structures in earlier layers and compositional structures in later layers.

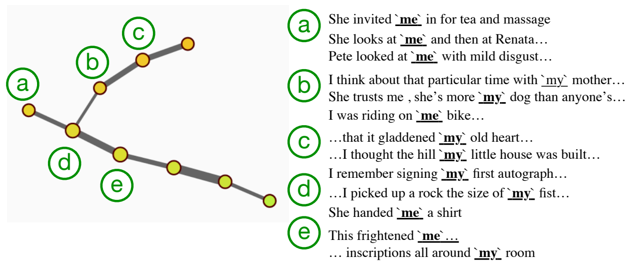
For our experiments, we use the "*bert-base-uncased*" trained network from the *Huggingface's transformers* library [WDS*19] with 12 layers, each with 768 neurons. We collect activation vectors from BERT on the training set of the Georgetown University Multilayer (GUM) corpus [Zel17]. The data contains 4780 sentences and 81,857 tokens. Tokens are individual words, numbers, or punctuation marks that form sentences. Among the 80K tokens, about 11K are punctuations.

We collect the activations by passing each sentence through the trained BERT model, collecting per-token activations, and applying **TopoAct** to the activations for all 12 layers. That is, for each of the 12 layers of the BERT neural network, we compute mapper graphs for point clouds of the token's activations in 768 dimensions across various parameter settings, similar to our earlier examples involving InceptionV1 and ResNet-18.

Our experiments applying **TopoAct** to BERT activations confirm and expand upon the earlier results in NLP [BLC19, LYK*20, JSS19]. In the following sections, we present some use cases of structures found in the BERT activations through our tool that highlight both local and global structures in both syntactic and semantic regimes.

8.2. Pronoun Differentiation

For layer 12 (the last layer) of BERT, we notice a branching structure that highlights the differentiation among pronouns. Recall that a pronoun is "a word that can function by itself as a noun phrase and that refers either to the participants in the discourse (*e.g.*, **I**,



a She invited `**me**` in for tea and massage
She looks at `**me**` and then at Renata…
Pete looked at `**me**` with mild disgust…

b I think about that particular time with `my` mother…
She trusts me , she's more `**my**` dog than anyone's…
I was riding on `**me**` bike…

c …that it gladdened `**my**` old heart…
…I thought the hill `**my**` little house was built…

I remember signing `**my**` first autograph…
d …I picked up a rock the size of `**my**` fist…
She handed `**me**` a shirt

e This frightened `**me**`…
… inscriptions all around `**my**` room

Figure 15: *Pronoun differentiation. Configuration: layer 12, Euclidean norm, 80 intervals, 30% overlap.*

you) or to someone or something mentioned elsewhere in the discourse (*e.g.*, **she**, **it**, **this**)", according to Oxford English Dictionary.

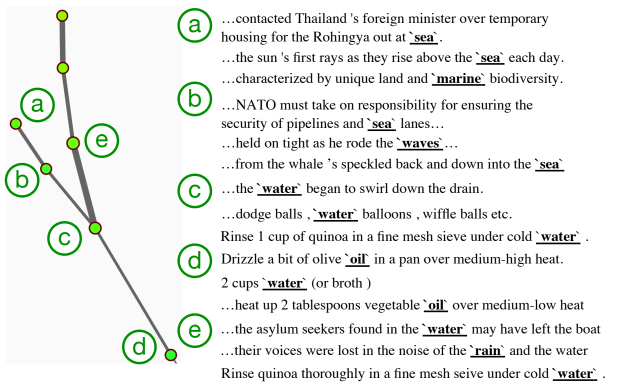
As illustrated in Figure 15, the chain of nodes with (d) and (e) represents a mixture of the first person singular personal pronoun (i.e, **me**) and the first person singular personal possessive form (*i.e.*, **my**). The chain continues along (b) and (c) with the possessive form (**my**), but the personal pronoun splits into its own branch (a) (**me**). Node (a) consists of sentences which employ the word **me** in phrases such as *"She invited **me** in for tea"*,*"She looks at **me** and then at Renata"*, *"He soon came back and gave **me** the good news"*, etc. Node (b) consists of sentences that employ the word **my** in phrases such as *"since I was in **my** twenties"*, *"I wished **my** da would come home"*. In particular, it also contains a sentence *"I was riding on **me** bike and I thought I'd swallowed an insect"*; here the word **me** is used instead of **my** likely due to a local dialect. This indicates that the differentiation between the pronouns goes beyond just the word-level and instead captures semantic differences as well.

This structure is interesting to our ML expert for a couple of reasons. First, we have a cluster of only the first person forms (**me** and **my**), and not the second (**you**) and third person forms (**he, she, they**). Second, although the pronoun (specifically, the object pronoun **me**) and the possessive form (**my**) are related in one sense (i.e., first person singular), they are distinct both in terms of their meanings and grammatical roles. The branching structure highlights this similarity and their divergence.

8.3. Contextual Differentiation

We give two examples involving how branches from **TopoAct** capture contextual differentiations. As shown in Figure 16, the structure of nodes about water highlights the different roles that water may play. Nodes (a) and (b) reflect oceanic usages (*e.g.*, **sea**, **marine**, **waves**). Node (d) reflects culinary usages (*"2 cups **water** (or broth)"*) and starts to connect with other liquids used in cooking (e.g., *"olive **oil** to taste"*). And other labeled nodes (*e.g.*, node (e)) reflect meteorological usages (*"the noise of the **rain**"*), culinary usages (*"rinse...under cold **water**"*), and oceanic usages.

We see similar fine-grained distinctions between **photographs**, **photography**, and **art** in Figure 17. The structure starts with **artworks**, **museum**, and **art** in node (a), moves on to **display**, **exhibition**, and **exhibits** in node (b), which then gets further refined into



(a) …contacted Thailand 's foreign minister over temporary housing for the Rohingya out at **`sea`**.
…the sun 's first rays as they rise above the **`sea`** each day.
…characterized by unique land and **`marine`** biodiversity.

(b) …NATO must take on responsibility for ensuring the security of pipelines and **`sea`** lanes…
…held on tight as he rode the **`waves`**…
…from the whale 's speckled back and down into the **`sea`**

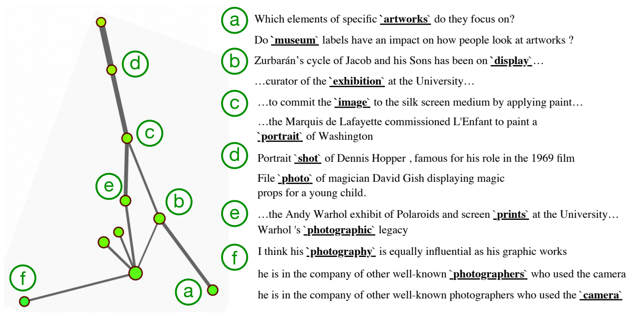
(c) …the **`water`** began to swirl down the drain.
…dodge balls , **`water`** balloons , wiffle balls etc.
Rinse 1 cup of quinoa in a fine mesh sieve under cold **`water`** .

(d) Drizzle a bit of olive **`oil`** in a pan over medium-high heat.
2 cups **`water`** (or broth)
…heat up 2 tablespoons vegetable **`oil`** over medium-low heat

(e) …the asylum seekers found in the **`water`** may have left the boat
…their voices were lost in the noise of the **`rain`** and the water
Rinse quinoa thoroughly in a fine mesh seive under cold **`water`** .

Figure 16: *Contextual differentiation. Configuration: layer 9, Euclidean norm, 80 intervals, 30% overlap.*

portrait and **painting** in node (c), as well as **shot** and **photo** in node (d).



(a) Which elements of specific **`artworks`** do they focus on?
Do **`museum`** labels have an impact on how people look at artworks ?

(b) Zurbarán's cycle of Jacob and his Sons has been on **`display`**…
…curator of the **`exhibition`** at the University…

(c) …to commit the **`image`** to the silk screen medium by applying paint…
…the Marquis de Lafayette commissioned L'Enfant to paint a **`portrait`** of Washington

(d) Portrait **`shot`** of Dennis Hopper , famous for his role in the 1969 film
File **`photo`** of magician David Gish displaying magic props for a young child.

(e) …the Andy Warhol exhibit of Polaroids and screen **`prints`** at the University…
Warhol 's **`photographic`** legacy

(f) I think his **`photography`** is equally influential as his graphic works
he is in the company of other well-known **`photographers`** who used the camera
he is in the company of other well-known photographers who used the **`camera`**

Figure 17: *Contextual differentiation. Configuration: layer 9, Euclidean distance, 80 intervals, 30% overlap, Jaccard* $= 0.01$.

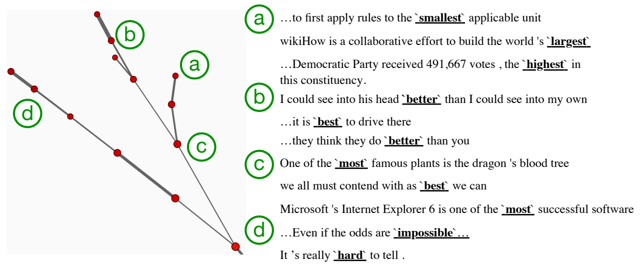
The fact that BERT (without any fine-tuning for any task) captures these differences is surprising for our ML expert. Such an insight could point toward an explanation for how transformer models like BERT and its many variants seem to capture world knowledge and even common-sense knowledge [BRS*19].

8.4. Local and Global Syntax

It is known that the lower layers of BERT characterize more local syntax and non-contextual lexical semantics, whereas the later layers capture global sentential structure and semantics [TDP19]. We see this by comparing the structures we see in layer 3 in (Figure 18) with the ones in layer 9 (Figure 15, Figure 16, Figure 17).

In lower layers, we observe groupings of words with similar parts of speech that diverge into chains that contain only words with similar meaning, as shown in Figure 18. For example, a generic adjective node (c) bifurcates into a set of nodes that describe size (e.g., **bigger**, **largest**, **highest** in node (a)) and a set of nodes that describe goodness (e.g., **best** and **better** in node (b)).

The other examples from later layers – such as layer 9 (Figure 19) – show that the mapper graph encodes complex relational abstractions that go beyond simply the dictionary meaning of the word.



(a) …to first apply rules to the **`smallest`** applicable unit
wikiHow is a collaborative effort to build the world 's **`largest`**
…Democratic Party received 491,667 votes , the **`highest`** in this constituency.

(b) I could see into his head **`better`** than I could see into my own
…it is **`best`** to drive there
…they think they do **`better`** than you

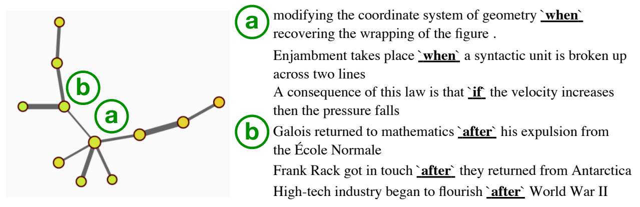
(c) One of the **`most`** famous plants is the dragon 's blood tree
we all must contend with as **`best`** we can
Microsoft 's Internet Explorer 6 is one of the **`most`** successful software

(d) …Even if the odds are **`impossible`**…
It 's really **`hard`** to tell .

Figure 18: *Local syntax at an earlier layer. Configuration: layer 3, cosine distance, 80 intervals, 30% overlap.*

8.5. After-When Separation

The branching structure in Figure 19 represents a surprising and difficult to characterize dichotomy in the usage of words like "after" and "when". Both these words are used to convey temporal meaning, but, the latter is also used to introduce discourse relationships such as explanations or sometimes even causations. The fact that these two usages are distinctly represented in the BERT embeddings shows that BERT does indeed characterize this subtle difference and, as a result, could serve as a basis for developing future parsers for discourse or rhetorical structure of text.



(a) modifying the coordinate system of geometry **`when`** recovering the wrapping of the figure .
Enjambment takes place **`when`** a syntactic unit is broken up across two lines
A consequence of this law is that **`if`** the velocity increases then the pressure falls

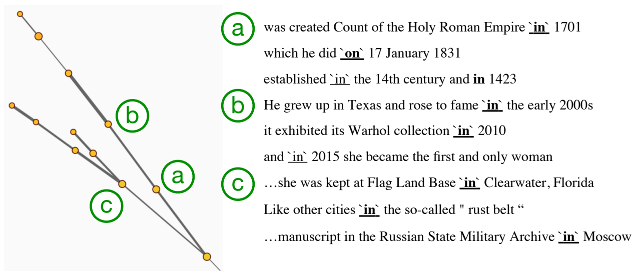
(b) Galois returned to mathematics **`after`** his expulsion from the École Normale
Frank Rack got in touch **`after`** they returned from Antarctica
High-tech industry began to flourish **`after`** World War II

Figure 19: *After-When separation. Configuration: layer 9, Euclidean distance, 80 intervals, 40% overlap.*

8.6. Temporal and Locative Prepositions

The difference between temporal and locative usages of prepositions is well studied, and forms the basis of the Preposition Supersense Project that Dr. Srikumar is part of [SSHP15]. As illustrated in Figure 20, both nodes (a) and (b), and their parent node, represent clusters of prepositions, but the two branches capture distinct meanings rather than mere surface level differences. To see this, note that both branches include the preposition word **in**, but the branch (a) represents its temporal usage, whereas (b) represents its locative usage. This branching structure is reminiscent of the Supersense Hierarchy developed via linguistic analysis [SHB*20].

These, and the previous observations, suggest that we can discover linguistic structures from BERT activations using **TopoAct**. Furthermore, **TopoAct** also provides investigative directions for why these embeddings have been successful at a wide variety of linguistic tasks.



(a) was created Count of the Holy Roman Empire **`in`** 1701
which he did **`on`** 17 January 1831
established `in` the 14th century and **in** 1423
(b) He grew up in Texas and rose to fame **`in`** the early 2000s
it exhibited its Warhol collection **`in`** 2010
and `in` 2015 she became the first and only woman
(c) …she was kept at Flag Land Base **`in`** Clearwater, Florida
Like other cities **`in`** the so-called " rust belt "
…manuscript in the Russian State Military Archive **`in`** Moscow

Figure 20: *Temporal and locative prepositions. Configuration: layer 9, Euclidean distance, 100 intervals, 30% overlap.*

9. Discussion

TopoAct supports exploratory analysis of numerous interesting topological structures, locally and globally, in the space of activation vectors. We encourage readers to utilize the live demo of datasets for InceptionV1 and ResNet for such an exploration. Our approach is not without its limitations. The exploration scenarios presented here are specific to the choice of input images as well as the choice of activation vectors. Further analysis is required to determine how stable the results are with respect to these choices. However, some of these limitations are common to other recent approaches (*e.g.*, [CAS*19, HPRC20]). We offer some topics for discussion and future work.

Generality. We focus on CNNs, specifically, InceptionV1 and ResNet-18, in this paper. However, our approach is not restricted to a particular network architecture. Mapper graphs could be generated and used as a vehicle for visual exploration whenever neuron activations are present. **TopoAct** can be generalized to explore new datasets coupled with other trained neural network architectures, such as ZFNet [ZF12], AlexNet [KSH17], and VGGNet [SZ15]. We have showed that our approach is generalizable beyond image classifiers to include textual embedding networks such as BERT.

Parameter tuning. Practical and automatic parameter tuning for the mapper construction remains a challenging open problem for the broad TDA community. Carriere *et al.* [CMO18] provided the state-of-the-art, albeit theoretical, results on mapper parameter selection under restrictive settings. Their framework assumed that a point cloud sample taken from the underlying space has a well-behaved, parameterizable probability distribution (formally, an (a,b)-standard distribution) and that the sample is sufficiently large, so that the Hausdorff distance between the sample and the underlying space is small. However, upon careful investigation, these assumptions are not applicable in our setting. Although we may assume that the activation space is a compact subset of the Euclidean space, we cannot verify that the activation vectors we sample follow the generative model of an (a,b)-standard distribution, and that $300K$ vectors form a sufficiently large sample for approximating or possibly reconstructing the underlying space.

On the other hand, the mapper construction comes with "best practices" in terms of parameter tuning, which rely on a grid search in the parameter space where good parameter combinations are those that produce stable structures. Finding a theoretically sound and yet practical parameter tuning strategy for our mapper graph construction remains open; see the supplementary materials for more discussion on this topic. For the current version of the **TopoAct**, we focus on exploring various mapper graphs with predetermined sets of parameter combinations following the best practices. Additionally, confirmation bias is a risk when **TopoAct** is utilized in practice because users may simply tune the parameters until they see what they want to see in the visualization. However, confirmation bias cannot be resolved without automatic parameter tuning, which remains an open problem.

Stability. Additional theoretical results regarding the stability of mapper construction are available in [BBMW20] and the references therein. The investigation is on-going into how stable the mapper graphs are with respect to different sampling techniques. Under some assumptions on the sampling condition, Brown *et al.* [BBMW20] showed that a pair of mapper graphs is close if their underlying point clouds are sampled from the same probability density function concentrated on the underlying topological space. However, similar to the situation of parameter tuning, the gap between theory and practice is still large. Filling such a gap is beyond the scope of this paper.

Adversarial attacks. An important aspect in understanding the effectiveness of adversarial attacks on neural networks is how an attack alters the intermediate representations, *i.e.*, the activations. **TopoAct** visualizes these representations from a topological perspective and hence might be useful in analyzing the effect of adversarial attacks at different layers of the network.

Corrective actions during training. A branching point (a bifurcation) in the space of activations at a particular layer may indicate the point where the network starts distinguishing a pair of classes. This knowledge can be useful to inform corrective actions for inputs in the test data that are being misclassified. For example, if two classes that bifurcate at a particular layer in **TopoAct** are still being misclassified as each other, an expert can choose to increase the network width at subsequent layers, or to selectively augment the training data for these classes to encourage better separation.

10. Conclusion

In this paper, we present **TopoAct**, a framework to explore the topology of the activation spaces of neural networks. We obtain topological summaries of the activation spaces via mapper graphs that capture the organizational principal behind neuron activations. We apply **TopoAct** to trained neural networks such as ResNet and InceptionV1 for image classification, and BERT for contextual word embeddings. In each case, we present exploration scenarios that provide valuable insights into the image representations or word embeddings learned by different layers of these networks. This paper is the first step toward understanding the topological structure of the activation spaces in deep neural networks.

Acknowledgments

This research was partially funded by NSF IIS 1910733, DBI 1661375, and IIS-1513616. We would like to thank Vivek Srikumar for his extremely valuable feedbacks regarding applying **TopoAct** to the BERT neural networks. We also would like to thank Jeff Phillips who forwarded a Twitter message from Chris Olah on March 7, 2019, which inspired this project.

References

[Ala12] Alagappan M.: From 5 to 13: Redefining the positions in basketball. *MIT Sloan Sports Analytics Conference* (2012). 3

[Ale28] Aleksandroff P. S.: Über den allgemeinen dimensionsbegriff und seine beziehungen zur elementaren geometrischen anschauung. *Mathematische Annalen 98*, 1 (1928), 617–635. 3

[BBMW20] Brown A., Bobrowski O., Munch E., Wang B.: Probabilistic convergence and stability of random mapper graphs. *Journal of Applied and Computational Topology* (2020). 12

[BGSF08] Biasotti S., Giorgi D., Spagnuolo M., Falcidieno B.: Reeb graphs for shape analysis and applications. *Theoretical Computer Science 392* (2008), 5–22. 3

[BLC19] Beltagy I., Lo K., Cohan A.: SciBERT: A pretrained language model for scientific text. arXiv:1903.10676, 2019. 10

[BMMP03] Biasotti S., Marini S., Mortara M., Patane G.: An overview on properties and efficacy of topological skeletons in shape modelling. *Shape Modeling International* (2003). 3

[BRS*19] Bosselut A., Rashkin H., Sap M., Malaviya C., Celikyilmaz A., Choi Y.: COMET: Commonsense transformers for automatic knowledge graph construction. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), 4762–4779. 11

[BWN*15] Bhatia H., Wang B., Norgard G., Pascucci V., Bremer P.-T.: Local, smooth, and consistent Jacobi set simplification. *Computational Geometry: Theory and Applications (CGTA) 48*, 4 (2015), 311–332. 2

[CAS*19] Carter S., Armstrong Z., Schubert L., Johnson I., Olah C.: Activation Atlas. *Distill 4*, 3 (2019), e15. 2, 3, 4, 5, 6, 12

[Cay08] Cayton L.: *Algorithms for manifold learning*. Tech. Rep. CS2008-0923, University of California at San Diego, 2008. 3

[CISZ08] Carlsson G., Ishkhanov T., Silva V. D., Zomorodian A.: On the local behavior of spaces of natural images. *International journal of computer vision 76*, 1 (2008), 1–12. 3

[CMO18] Carrière M., Michel B., Oudot S.: Statistical analysis and parameter selection for mapper. *Journal of Machine Learning Research 19*, 12 (2018), 1–39. 12

[CSA03] Carr H., Snoeyink J., Axen U.: Computing contour trees in all dimensions. *Computational Geometry 24*, 2 (2003), 75–94. 2

[CZJ*19] Cho H. J., Zhao J., Jung S. W., Ladewig E., Kong D.-S., Suh Y.-L., Lee Y., Kim D., Ahn S. H., Bordyuh M., Kang H. J., Sa J. K., Seo Y. J., Kim S. T., Lim D. H., Dho Y.-S., Lee J.-I., Seol H. J., Choi J. W., Park W.-Y., Park C.-K., Rabadan R., Nam D.-H.: Distinct genomic profile and specific targeted drug responses in adult cerebellar glioblastoma. *Neuro-Oncology 21*, 1 (2019), 47–58. 3

[DDS*09] Deng J., Dong W., Socher R., Li L.-J., Li K., Fei-Fei L.: ImageNet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition* (2009). 4

[EBCV09] Erhan D., Bengio Y., Courville A., Vincent P.: *Visualizing Higher-Layer Features of a Deep Network*. Tech. rep., Univeristy of Montreal, 2009. 2

[EH02] Edelsbrunner H., Harer J.: Jacobi sets of multiple Morse functions. In *Foundations of Computational Mathematics, Minneapolis 2002*, Cucker F., DeVore R., Olver P., Süli E., (Eds.). Cambridge University Press, 2002, pp. 37–57. 2

[EHNP03] Edelsbrunner H., Harer J., Natarajan V., Pascucci V.: Morse-Smale complexes for piece-wise linear 3-manifolds. *Proceedings of the 19th Annual symposium on Computational geometry* (2003), 361–370. 2

[EHP08] Edelsbrunner H., Harer J., Patel A.: Reeb spaces of piecewise linear mappings. *Proceedings of the 24th annual symposium on Computational geometry* (2008), 242–250. 2

[EHZ03] Edelsbrunner H., Harer J., Zomorodian A. J.: Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete and Computational Geometry 30*, 87-107 (2003). 2

[EKSX96] Ester M., Kriegel H.-P., Sander J., Xu X.: A density-based algorithm for discovering clusters in large spatial databases with noise. *International Conference on Knowledge Discovery and Data Mining* (1996), 226–231. 4, 5

[GAES19] Gabella M., Afambo N., Ebli S., Spreemann G.: Topology of learning in artificial neural networks. arXiv:1902.08160, 2019. 3

[GSH19] Gebhart T., Schrater P., Hylton A.: Characterizing the shape of activation space in deep neural networks. *Proceedings of the 18th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2019). 3

[GSPS19] Geniesse C., Sporns O., Petri G., Saggar M.: Generating dynamical neuroimaging spatiotemporal representations (DyNeuSR) using topological data analysis. *Network Neuroscience 3*, 3 (2019). 3

[HKPC18] Hohman F., Kahng M., Pienta R., Chau D. H.: Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2018). 2

[HPRC20] Hohman F., Park H., Robinson C., Chau D. H. P.: Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 1096–1106. 2, 4, 5, 12

[HWR18] Hajij M., Wang B., Rosen P.: Mapper on graphs for network visualization. arXiv:1804.11242, 2018. 3

[HWSR18] Hajij M., Wang B., Scheidegger C., Rosen P.: Visual detection of structural changes in time-varying graphs using persistent homology. *IEEE Pacific Visualization Symposium (PacificVis)* (2018). 3

[HZRS16] He K., Zhang X., Ren S., Sun J.: Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778. 1, 4

[JCR*19] Jeitziner R., Carriére M., Rougemont J., Oudot S., Hess K., Brisken C.: Two-tier mapper, an unbiased topology-based clustering method for enhanced global gene expression analysis. *Bioinformatics 35*, 18 (2019), 3339–3347. 3

[JSS19] Jawahar G., Sagot B., Seddah D.: What does BERT learn about the structure of language? *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), 3651–3657. 10

[Kar14] Karpathy A.: t-SNE visualization of CNN codes. https://cs.stanford.edu/people/karpathy/cnnembed/, 2014. 2, 3

[KGCFR*20] Knudson A., González-Casabianca F., Feged-Rivadeneira A., Pedreros M. F., Aponte S., Olaya A., Castillo C. F., Mancilla E., Piamba-Dorado A., Sanchez-Pedraza R., Salazar-Terreros M. J., Lucchi N., Udhayakumar V., Jacob C., Pance A., Carrasquilla M., Apráez G., Angel J. A., Rayner J. C., Corredor V.: Spatio-temporal dynamics of plasmodium falciparum transmission within a spatial unit on the colombian pacific coast. *Scientific Reports 10*, 3756 (2020). 3

[KH09] Krizhevsky A., Hinton G.: *Learning multiple layers of features from tiny images*. Tech. rep., University of Toronto, 2009. 4, 9

[KSH17] Krizhevsky A., Sutskever I., Hinton G. E.: ImageNet classification with deep convolutional neural networks. *Communications of the ACM 60*, 6 (2017), 84–90. 12

[KWG*18] Kim B., Wattenberg M., Gilmer J., Cai C., Wexler J., Viegas F., Sayres R.: Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). *International Conference on Machine Learning* (2018). 2

[LCY14] Lin M., Chen Q., Yan S.: Network in network. *International Conference on Learning Representations (ICIR)* (2014). 4

[LMW*17] LIU S., MALJOVEC D., WANG B., BREMER P.-T., PASCUCCI V.: Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 23*, 3 (2017), 1249–1268. 3

[LPM03] LEE A. B., PEDERSEN K. S., MUMFORD D.: The non-linear statistics of high-contrast patches in natural images. *International Journal of Computer Vision 54* (2003), 83–103. 3

[LRM*13] LE Q. V., RANZATO M., MONGA R., DEVIN M., CHEN K., CORRADO G. S., DEAN J., NG A. Y.: Building high-level features using large scale unsupervised learning. *IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), 8595–8598. 2

[LYK*20] LEE J., YOON W., KIM S., KIM D., KIM S., SO C. H., KANG J.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics 36*, 4 (2020), 1234–1240. 10

[MH08] MAATEN L. V. D., HINTON G.: Visualizing data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605. 2, 3

[MHM18] MCINNES L., HEALY J., MELVILLE J.: UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv:1802.03426, 2018. 2, 3

[MHSG18] MCINNES L., HEALY J., SAUL N., GROSSBERGER L.: UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software 3*, 29 (2018), 861. 2, 7

[MNL*19] MATHEWS J. C., NADEEM S., LEVINE A. J., POURYAHYA M., DEASY J. O., TANNENBAUM A.: Robust and interpretable PAM50 reclassification exhibits survival advantage for myoepithelial and immune phenotypes. *NPJ Breast Cancer 5*, 30 (2019). 2

[MV15] MAHENDRAN A., VEDALDI A.: Understanding deep image representations by inverting them. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 5188–5196. 2

[MV16] MAHENDRAN A., VEDALDI A.: Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision 120*, 3 (2016), 233–255. 2

[MW16] MUNCH E., WANG B.: Convergence between categorical representations of Reeb space and mapper. *International Symposium on Computational Geometry* (2016). 2

[NLC11] NICOLAU M., LEVINE A. J., CARLSSON G.: 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 (2011), 7265–7270. 2

[NYC16] NGUYEN A., YOSINSKI J., CLUNE J.: Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. arXiv:1602.03616, 2016. 2, 3

[OMS17] OLAH C., MORDVINTSEV A., SCHUBERT L.: Feature visualization. *Distill 2*, 11 (2017), e7. 4, 5

[OSJ*18] OLAH C., SATYANARAYAN A., JOHNSON I., CARTER S., SCHUBERT L., YE K., MORDVINTSEV A.: The building blocks of interpretability. *Distill 3*, 3 (2018), e10. 4, 5

[PVP17] PATANIA A., VACCARINO F., PETRI G.: Topological analysis of data. *EPJ Data Science 6*, 7 (2017). 3

[Ree46] REEB G.: Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique (on the singular points of a complete integral pfaff form or of a numerical function). *Comptes Rendus Acad.Science Paris 222* (1946), 847–849. 2

[SHB*20] SCHNEIDER N., HWANG J. D., BHATIA A., SRIKUMAR V., HAN N.-R., O'GORMAN T., MOELLER S. R., ABEND O., SHALEV A., BLODGETT A., PRANGE J.: Adposition and case supersenses v2.5: Guidelines for english. arXiv:1704.02134, 2020. 11

[SHW*20] SUH A., HAJIJ M., WANG B., SCHEIDEGGER C., ROSEN P.: Persistent homology guided force-directed graph layouts. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 697–707. 3

[SLJ*15] SZEGEDY C., LIU W., JIA Y., SERMANET P., REED S., ANGUELOV D., ERHAN D., VANHOUCKE V., RABINOVICH A.: Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015). 1, 4

[SMC07] SINGH G., MÉMOLI F., CARLSSON G.: Topological methods for the analysis of high dimensional data sets and 3D object recognition. *Eurographics Symposium on Point-Based Graphics 22* (2007). 1, 2, 3

[SMVJ09] SILVA V. D., MOROZOV D., VEJDEMO-JOHANSSON M.: Persistent cohomology and circular coordinates. *Proceedings of the 25th Annual Symposium on Computational Geometry* (2009), 227–236. 3

[SSGC*18] SAGGAR M., SPORNS O., GONZALEZ-CASTILLO J., BANDETTINI P. A., CARLSSON G., GLOVER G., REISS A. L.: Towards a new approach to reveal dynamical organization of the brain using topological data analysis. *Nature Communications 9*, 1399 (2018). 3

[SSHP15] SCHNEIDER N., SRIKUMAR V., HWANG J. D., PALMER M.: A hierarchy with, of, and for preposition supersenses. *Proceedings of the 9th Linguistic Annotation Workshop* (2015), 112–123. 11

[SVZ14] SIMONYAN K., VEDALDI A., ZISSERMAN A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. *Workshop at International Conference on Learning Representations,* (2014). 2

[SZ15] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)* (2015). 12

[TDP19] TENNEY I., DAS D., PAVLICK E.: BERT rediscovers the classical NLP pipeline. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), 4593–4601. 11

[Uly16] ULYANOV D.: Multicore-TSNE. `https://github.com/DmitryUlyanov/Multicore-TSNE`, 2016. 6

[WDS*19] WOLF T., DEBUT L., SANH V., CHAUMOND J., DELANGUE C., MOI A., CISTAC P., RAULT T., LOUF R., FUNTOWICZ M., DAVISON J., SHLEIFER S., VON PLATEN P., MA C., JERNITE Y., PLU J., XU C., SCAO T. L., GUGGER S., DRAME M., LHOEST Q., RUSH A. M.: HuggingFace's transformers: State-of-the-art natural language processing. ArXiv:1910.03771, 2019. 10

[WSPVJ11] WANG B., SUMMA B., PASCUCCI V., VEJDEMO-JOHANSSON M.: Branching and circular features in high dimensional data. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1902–1911. 3

[Xia16] XIA S.: A topological analysis of high-contrast patches in natural images. *Journal of Nonlinear Sciences and Applications 9* (2016), 126–138. 3

[YCN*15] YOSINSKI J., CLUNE J., NGUYEN A., FUCHS T., LIPSON H.: Understanding neural networks through deep visualization. *Deep Learning Workshop at the 31st International Conference on Machine Learning* (2015). 2

[YZR*18] YAN L., ZHAO Y., ROSEN P., SCHEIDEGGER C., WANG B.: Homology-preserving dimensionality reduction via manifold landmarking and tearing. *Symposium on Visualization in Data Science (VDS) at IEEE VIS* (2018). 3

[Zel17] ZELDES A.: The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation 51*, 3 (2017), 581–612. 10

[ZF12] ZEILER M. D., FERGUS R.: Visualizing and understanding convolutional networks. *The 26th Annual Conference on Neural Information Processing Systems* (2012). 12

[ZF14] ZEILER M. D., FERGUS R.: Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014, Lecture Notes in Computer Science*, Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), vol. 8689. Springer, Cham, 2014, pp. 818–833. 2