Journal of Machine Learning Research 21 (2020) 1-40 Submitted 4/20; Revised 7/20; Published 8/20

Topology of Deep Neural Networks

Gregory Naitzat GREGN@UCHICAGO.EDU
Department of Statistics
University of Chicago
Chicago, IL, 60637, USA

Andrey Zhitnikov ANDREYZ@TECHNION.AC.IL
Faculty of Electrical Engineering
Technion – Israel Institute of Technology
Haifa 32000, Israel

Lek-Heng Lim LEKHENG@UCHICAGO.EDU
Computational and Applied Mathematics Initiative
University of Chicago
Chicago, IL, 60637, USA

Editor: Sayan Mukherjee

Abstract

We study how the topology of a data set $M = M_a \cup M_b \subseteq \mathbb{R}^d$, representing two classes a and b in a binary classification problem, changes as it passes through the layers of a well-trained neural network, i.e., one with perfect accuracy on training set and near-zero generalization error ($\approx 0.01\%$). The goal is to shed light on two mysteries in deep neural networks: (i) a nonsmooth activation function like ReLU outperforms a smooth one like hyperbolic tangent; (ii) successful neural network architectures rely on having many layers, even though a shallow network can approximate any function arbitrarily well. We performed extensive experiments on the persistent homology of a wide range of point cloud data sets, both real and simulated. The results consistently demonstrate the following: ① Neural networks operate by changing topology, transforming a topologically complicated data set into a topologically simple one as it passes through the layers. No matter how complicated the topology of M we begin with, when passed through a well-trained neural network $f : \mathbb{R}^d \to \mathbb{R}^p$, there is a vast reduction in the Betti numbers of both components M_a and M_b; in fact they nearly always reduce to their lowest possible values: $\beta_k\big(f(M_i)\big) = 0$ for $k \geq 1$ and $\beta_0\big(f(M_i)\big) = 1$, $i = a, b$. ② The reduction in Betti numbers is significantly faster for ReLU activation than for hyperbolic tangent activation as the former defines nonhomeomorphic maps that change topology, whereas the latter defines homeomorphic maps that preserve topology. ③ Shallow and deep networks transform data sets differently — a shallow network operates mainly through changing geometry and changes topology only in its final layers, a deep one spreads topological changes more evenly across all layers.

Keywords: neural networks, topology change, Betti numbers, topological complexity, persistent homology

1. Overview

A key insight of topological data analysis is that "*data has shape*" (Carlsson, 2013, 2014). That data sets often have nontrivial topologies, which may be exploited in their analysis, is

©2020 Gregory Naitzat, Andrey Zhitnikov, Lek-Heng Lim.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v21/20-345.html.

now a widely accepted tenet with abundant examples across multiple disciplines: dynamical systems (Khasawneh et al., 2018), medicine (Li et al., 2015; Nielson et al., 2015), genomics (Perea et al., 2015), neuroscience (Giusti et al., 2015), time series (Perea and Harer, 2015), etc. An early striking example came from computer vision, where Carlsson et al. (2008) showed that naturally occurring image patches reside on an embedded three-dimensional manifold that has the topology of a Klein bottle.

We will study how modern deep neural networks transform topologies of data sets, with the goal of shedding light on their breathtaking yet somewhat mysterious effectiveness. Indeed, we seek to show that neural networks operate by changing the topology (i.e., shape) of data. The relative efficacy of ReLU activation over traditional sigmoidal activations can be explained by the different speeds with which they change topology — a ReLU-activated neural network (which is not a homeomorphism) is able to sharply reduce Betti numbers but not a sigmoidal-activated one (which is a homeomorphism). Also, the higher the topological complexity of the data, the greater the depth of the network required to reduce it, explaining the need to have an adequate number of layers.

We would like to point out that the idea of changing the topology of a space to facilitate a machine learning goal is not as esoteric as one might imagine. For example, it is implicit in kernel methods (Schölkopf and Smola, 2002) — a data set with two components inseparable by a hyperplane is embedded in a higher-dimensional space where the embedded images of the components are separable by a hyperplane. *Dimension* is a topological invariant, changing dimension is changing topology. We will see that a ReLU-activated neural network with many layers effects topological changes primarily through changing Betti numbers, another topological invariant.

Our study differs from current approaches in two important ways. Many existing studies either (i) analyze neural networks in an asymptotic regime, where the number of neurons in each layer or the number of layers becomes unbounded or infinite, leading to conclusions that pertain to neural networks of somewhat unrealistic architectures; or (ii) they focus on what a neural network does to a single object, e.g., an image of a cat, and examine how that object changes as it passes through the layers. While we do not dispute the value of such approaches, we would like to contrast them with ours: We study what a neural network with a realistic architecture does to an entire class of objects. It is common to find expositions (especially in the mass media) of deep neural networks that purport to show their workings by showing how a cat image is transformed as it passes through the layers. We think this is misguided — one should be looking at the *entire manifold* of cat images, not a single point on that manifold (i.e., a single cat image). This is the approach we undertake in our article.

Figure 1 illustrates what we mean by 'changing topology'. The two subfigures are caricatures of real results (see Figures 2, 11, 12, 13, for the true versions obtained via actual Betti numbers computations and projections to principal components.) In both subfigures, we begin with a three-dimensional manifold $M = M_a \cup M_b$, comprising two disjoint embedded submanifolds M_a (green) and M_b (red) entangled in a topologically nontrivial manner, and track its progressive transformation into a topologically simple manifold comprising a green ball and a red ball. In the left box, M is initially the union of the two green solid tori M_a, interlocked with the red solid figure-eight M_b. In the right box, M is initially a union of M_a, a green solid ball with three voids, and M_b, three red balls each placed within one of the three voids. The topological simplification in both boxes are achieved via a reduction

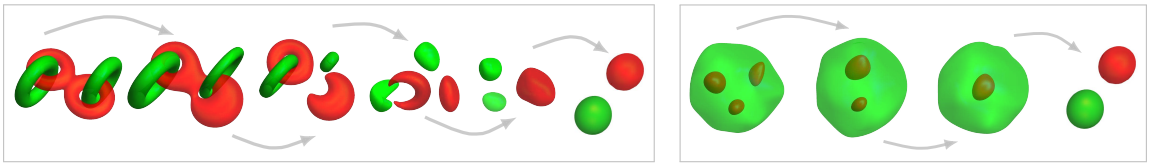


Figure 1: Progression of Betti numbers $\beta(X) = (\beta_0(X), \beta_1(X), \beta_2(X))$. *Left*: β(red): $(1,2,0) \to (1,2,0) \to (2,1,0) \to (2,0,0) \to (1,0,0) \to (1,0,0)$; β(green): $(2,2,0) \to (2,2,0) \to (2,1,0) \to (2,0,0) \to (2,0,0) \to (1,0,0)$. *Right*: β(red): $(3,0,0) \to (2,0,0) \to (1,0,0) \to (1,0,0)$; β(green): $(1,0,3) \to (1,0,2) \to (1,0,1) \to (1,0,0)$.

in the Betti numbers of both M_a and M_b so that eventually we have $\beta_k(M_i) = 0$ for $k \geq 1$ and $\beta_0(M_i) = 1$, $i = a, b$. Our goal is to provide (what we hope is) incontrovertible evidence that this picture captures the actual workings of a *well-trained*[1] neural network in a binary classification problem where M_a and M_b represent the two classes.

In reality, the manifold $M = M_a \cup M_b$ would have to be replaced by a point cloud data set, i.e., a finite set of points sampled with noise from M. The notion of *persistent homology* allows us to give meaning to the topology of point cloud data and estimate the Betti numbers of its underlying manifold.

Key findings: This work is primarily an empirical study — we have performed more than 10,000 experiments on real and simulated data sets of varying topological complexities and have made our codes available for the reader's further experimentations.[2] We summarize our most salient observations and discuss their implications:

(i) For a fixed data set and architecture, topological changes effected by a well-trained network are robust across different training instances and follow a similar profile.

(ii) Using smooth activations like hyperbolic tangent results in a slow down of topological simplification compared to nonsmooth activations like ReLU or Leaky ReLU.

(iii) The initial layers mostly induce only *geometric* changes, it is in the deeper layers that *topological* changes take place. As network depth is reduced, the burden of producing topological simplification is not spread uniformly across layers but remains concentrated in the last layers.

Observation (ii) provides a plausible answer to a widely asked question (Nair and Hinton, 2010; Maas et al., 2013; Glorot et al., 2011): What makes rectified activations such as ReLU and its variants perform better than smooth sigmoidal activations? We posit that it is not a matter of smooth versus nonsmooth but that a neural network with sigmoid activation is a *homeomorphic* map that preserves topology whereas one with ReLU activation is a *nonhomeomorphic* map that can change topology. It is much harder to change topology with homeomorphisms — in fact, mathematically it is impossible — but maps like the hyperbolic tangent achieve it in practice via rounding errors. Note that in IEEE finite-

1. One with near-zero generalization error.
2. `https://github.com/topnn/topnn_framework`.

precision arithmetic, the hyperbolic tangent is effectively a piecewise linear step function:

$$\tanh_\delta(x) = \begin{cases} +1 & \text{if } \mathrm{fl}(\tanh(x)) > 1-\delta, \\ \mathrm{fl}(\tanh(x)) & \text{if } -1+\delta \le \mathrm{fl}(\tanh(x)) \le 1-\delta, \\ -1 & \text{if } \mathrm{fl}(\tanh(x)) < -1+\delta, \end{cases}$$

where $\mathrm{fl}(x)$ denotes floating point representation of $x \in \mathbb{R}$, and $\delta > 0$ is the *unit roundoff*, i.e., $\delta = \epsilon/2$ with $\epsilon = \inf\{x > 0 : \mathrm{fl}(1+x) \neq 1\}$ the *machine epsilon* (Overton, 2001). Applied coordinatewise to a vector, $\tanh : \mathbb{R}^n \to (-1,1)^n$ is a homeomorphism of $\mathbb{R}^n$ to $(-1,1)^n$ and necessarily preserves topology; but $\tanh_\delta : \mathbb{R}^n \to [-1,1]^n$ is not a homeomorphism and thus has the ability to change topology. We also observe that lowering the floating point precision increases the value of δ (e.g., for double precision $\delta = 2^{-54}$, for half precision[3] $\delta = 2^{-9}$), which has the effect of coarsening $\tanh_\delta$, making it even further from a homeomorphism and thus more effective at changing topology. We suspect that this may account for the paradoxical superior performance of lower precision arithmetic in deep neural networks (Courbariaux et al., 2014; Gupta et al., 2015; Hubara et al., 2017).

The ReLU activation, on the other hand, is far from a homeomorphism (for starters, it is not injective) even in exact arithmetic. Indeed, if changing topology is the goal, then a composition of an affine map with ReLU activation, $\nu : \mathbb{R}^n \to \mathbb{R}^n$, $x \mapsto \max(Ax+b, 0)$, is a quintessential tool for achieving it — any topologically complicated part of $M \subseteq \mathbb{R}^n$ can be affinely moved outside the nonnegative orthant and collapsed to a single point by the rectifier. We see this in action in Figure 2, which unlike Figure 1, is a genuine example of a ReLU neural network trained to perfect accuracy on a two-dimensional manifold data set, where M_a comprises five red disks in a square M and $M_b = M \setminus M_a$ is the remaining green portion with the five disks removed. The 'folding' transformations in Figure 2 clearly require many-to-one maps and can never be achieved by any homeomorphism.

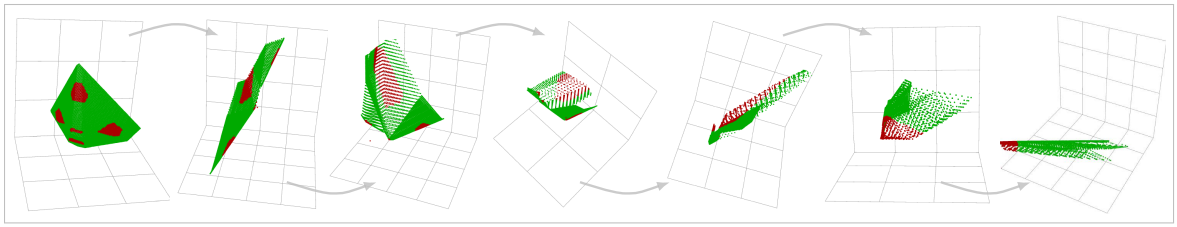


Figure 2: We see how the data set is transformed after passing through layers $2, 3, \ldots, 8$ of a ReLU network with three neurons in each layer, well-trained to detect five disks in a square. β(red): $(5,0) \to (4,0) \to (4,0) \to (4,0) \to (2,0) \to (1,0) \to (1,0)$.

The effectiveness of ReLU over sigmoidal activations is often attributed to the former's avoidance of vanishing/exploding gradient. Our results in Section 7 indicate that this does not give the full explanation. Leaky ReLU and ReLU both avoid vanishing/exploding gradients, yet they transform data sets in markedly different manners — for one, ReLU reduces topology faster than Leaky ReLU. The sharpness of the gradients is clearly not what matters most; on the other hand, the topological perspective perfectly explains why.

3. We assume the BFloat16 floating-point format used in TensorFlow.

Observation (iii) addresses another perennial paradox (Krizhevsky et al., 2017; Eigen et al., 2014; Yu et al., 2012): Why does a neural network with more layers work better, despite the well-known universal approximation property that any function can be approximated arbitrarily well by a two-layer one? We posit that the traditional approximation-theoretic view of neural networks is insufficient here; instead, the proper perspective is that of a topologically complicated input getting progressively simplified as it passes through the layers of a neural network. Observation (iii) accounts for the role of the additional layers — topological changes are minor in the first few layers and occur mainly in the later layers, thus a complicated data set requires many more layers to simplify.

We emphasize that our goal is to explain the mechanics of what happens from one layer to the next, and to see what role each attribute of the network's architecture — depth, width, activation — serves in changing topology. Note that we are not merely stating that a neural network is a blackbox that collapses each class to a component but how that is achieved, i.e., what goes on inside the blackbox.

Relations with and departures from prior works: As in topological data analysis, we make use of persistent homology and quantify topological complexity in terms of Betti numbers; we track how these numbers change as a point cloud data set passes through the layers of a neural network. But that is the full extent of any similarity with topological data analysis. In fact, from our perspective, topological data analysis and neural networks have opposite goals — the former is largely concerned with *reading* the shape of data, whereas the latter is about *writing* the shape of data; not unlike the relation between computer vision and computer graphics, wherein one is interested the inverse problems of the other. Incidentally, this shows that a well-trained neural network applied in reverse can be used as a tool for labeling components of a complex data set and their interrelation, serving a role similar to *mapper* (Singh et al., 2007) in topological data analysis. This idea has been explored in Pearson 2013; Naitzat et al. 2018.

To the best of our knowledge, our approach towards elucidating the inner workings of a neural network by studying how the topology, as quantified by persistent Betti numbers, of a point cloud data set changes as it passes through the layers, has never been done before. The key conclusion of these studies, namely, that the role of a neural network is primarily as a topology-changing map, is also novel as far as we know. Nevertheless, we would like to acknowledge a Google Brain blog post (Olah, 2014) that inspired our work — it speculated on how neural networks may act as homeomorphisms that distort *geometry*, but stopped short of making the leap to topology-changing maps.

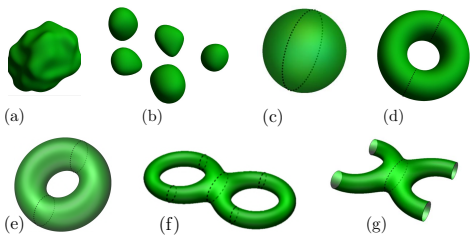
There are other works that employ Betti numbers in the analysis of neural networks. Bianchini and Scarselli (2014) did a purely theoretical study of upper bounds on the topological complexity (i.e., sum of Betti numbers) of the decision boundaries of neural networks with smooth sigmoidal activations; Ramamurthy et al. (2019) did a similar study with a different measure of topological complexity. Guss and Salakhutdinov (2018) studied the empirical relation between the topological complexity of a data set and the minimal network capacity required to classify it. Rieck et al. (2019) used persistent homology to monitor changes in the weights of neural network during training and proposed an early stopping criteria based on persistent homology.

Outline: In Section 2 we introduce, in an informal way, the main topological concepts used throughout this article. This is supplemented by a more careful treatment in Section 3,

which provides a self-contained exposition of simplicial homology and persistent homology tailored to our needs. Section 4 contains a precise formulation of the problem we study, specifies what is tracked empirically, and addresses some caveats. Section 6 introduces our methodology for tracking topological changes and implementation details. We present the results from our empirical studies with discussions in Section 7, verified our findings on real-world data in Section 8, and conclude with some speculative discussions in Section 9.

2. Quantifying Topology

In this article, we rely entirely on *Betti numbers* $\beta_k(M)$ to quantify topology as they are the simplest topological invariants that capture the shape of a space $M \subseteq \mathbb{R}^d$, have intuitive interpretations, and are readily computable within the framework of persistent homology for a point cloud data set sampled from M. The zeroth Betti number, $\beta_0(M)$, counts the number of connected components in M; the kth Betti number, $\beta_k(M)$, $k \geq 1$, is informally the number of k-dimensional holes in M. In particular, $\beta_k(M) = 0$ when $k \geq d$ as there are no holes of dimension d or higher in d-dimensional space. So for $M \subseteq \mathbb{R}^d$, we write $\beta(M) \coloneqq (\beta_0(M), \beta_1(M), \ldots, \beta_{d-1}(M))$ — these numbers capture the shape or topology of M, as one can surmise from Figure 3. So whenever we refer to 'topology' in this article, we implicitly mean $\beta(M)$.



	Manifold $M \subseteq \mathbb{R}^3$	$\beta(M)$
(a)	Single contractible manifold	$(1, 0, 0)$
(b)	Five contractible manifolds	$(5, 0, 0)$
(c)	Sphere	$(1, 0, 1)$
(d)	Solid torus (filled)	$(1, 1, 0)$
(e)	Surface of torus (hollow)	$(1, 2, 1)$
(f)	Genus two surface (hollow)	$(1, 4, 1)$
(g)	Torso surface (hollow)	$(1, 3, 0)$

Figure 3: Manifolds in $\mathbb{R}^3$ and their Betti numbers.

If M has no holes and can be continuously (i.e., without tearing) deformed to a point, then $\beta_0(M) = 1$ and $\beta_k(M) = 0$ for all $k \geq 1$; such a space is called *contractible*. The simplest noncontractible connected space is a circle $S^1 \subseteq \mathbb{R}^2$, which has a single connected component and a single one-dimensional hole, so $\beta_0(S^1) = 1 = \beta_1(S^1)$ and $\beta_k(S^1) = 0$ for all $k \geq 2$. Figure 3 contains a few more examples.

Intuitively, the more holes a space has, the more complex its topology. In other words, the larger the numbers in $\beta(M)$, the more complicated the topology of M. As such, we define its *topological complexity* by

$$\omega(M) \coloneqq \beta_0(M) + \beta_1(M) + \cdots + \beta_{d-1}(M). \tag{2.1}$$

While not as well-known as the *Euler characteristic* (which is an alternating signed sum of the Betti numbers), the topological complexity is also a classical notion in topology, appearing notably in Morse theory (Milnor, 1963); one of its best known result is that the topological complexity of M gives a lower bound for the number of critical points of a function $f : M \to \mathbb{R}$ with nondegenerate Hessians. Gromov's Betti Number Theorem (Gromov, 1981), another celebrated result in differential geometry, bounds $\omega(M)$ in terms of

the dimension and sectional curvature of a Riemannian manifold M. Topological complexity also appears in many other contexts (Milnor, 1964; Basu and Rizzie, 2018), including neural networks. We highlight the work of Bianchini and Scarselli (2014) that we mentioned earlier, which studies the topological complexity of the decision boundary of neural networks with activations that are Pfaffian functions (Zell, 2003; Gabrielov et al., 2004). These include sigmoidal activations but not the ReLU nor leaky ReLU activations studied in this article.

For piecewise linear activations like ReLU and leaky ReLU, the most appropriate theoretical upper bounds for topological complexity of decision boundaries are likely the number of linear regions (Montúfar et al., 2014; Zhang et al., 2018). The goal of our article is different, we are interested not in the shape of the decision boundary of an l-layer neural network $\nu_l : \mathbb{R}^d \to \mathbb{R}^p$ but in the shapes of the input $M \subseteq \mathbb{R}^d$, output $\nu_l(M) \subseteq \mathbb{R}^q$, and all intermediate $\nu_k(M)$, $k = 1, \dots, l-1$. By so doing, we may observe how the shape of M is transformed as it passes through the layers of a well-trained neural network, thereby elucidating the inner workings. In other words, we would like to track the Betti numbers

$$\beta(M) \to \beta(\nu_1(M)) \to \beta(\nu_2(M)) \to \cdots \to \beta(\nu_{l-1}(M)) \to \beta(\nu_l(M)).$$

To do this in reality, we will have to estimate $\beta(M)$ from a point cloud data set, i.e., a finite set of points $T \subseteq M$ sampled from M, possibly with noise. The next section will describe the procedure to do this via persistent homology, which is by now a standard tool in topological data analysis. Readers who do not want to be bothered with the details just need to know that one may reliably estimate $\beta(M)$ by sampling points from M. The main idea is that the Betti numbers of M may be estimated by constructing a simplicial complex from T in one of several ways that depend on a 'persistent parameter,' and then using simplicial homology to compute the Betti numbers of this simplicial complex. Roughly speaking, the 'persistent parameter' allows one to pick the right scale at which the point cloud T should be sampled so as to give a faithful estimation of $\beta(M)$. Henceforth whenever we speak of $\beta(M)$, we mean the Betti numbers estimated in this fashion.

For simplicity of the preceding discussion, we have used M as a placeholder for any manifold. Take say a handwritten digits classification problem (see Section 8), then $M = M_0 \cup M_1 \cup \cdots \cup M_9$ has ten components, with M_i the manifold of all possible handwritten digits $i \in \{0, 1, \dots, 9\}$. Here we are not interested in $\beta(M)$ per se but in $\beta(\nu_k(M_i))$ for all $k = 0, 1, \dots, l$ and $i = 0, 1, \dots, 9$ — so that we may see how each component is transformed as M passes through the layers, i.e., we will need to sample points from each of $\nu_k(M_i)$ to estimate its Betti numbers, for each component i and at each layer k.

3. Algebraic Topology and Persistent Homology Background

This section may be skipped by readers who are already familiar with persistent homology or are willing to take on faith what we wrote in the last two paragraphs of the previous section. Here we will introduce background knowledge in algebraic topology — simplicial complex, homology, simplicial homology — and provide a brief exposition on selected aspects of topological data analysis — Vietoris–Rips complex, persistent homology, practical homology computations — that we need for our purposes.

3.1. Simplicial Complexes

A k-dimensional *simplex*, or k-simplex, σ in $\mathbb{R}^d$, is the convex hull of $k+1$ affinely independent points $v_0, \ldots, v_k \in \mathbb{R}^d$. A 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron. A k-simplex is represented by listing the set of its $k+1$ *vertices* and denoted $\sigma = [v_0, \ldots, v_k]$. The *faces* of a k-simplex are simplices of dimensions 0 to $k-1$ formed by convex hulls of proper subsets of its *vertex set* $\{v_0, \ldots, v_k\}$. For example, the faces of a line segment/1-simplex are its end points, which are 0-simplices; the faces of a triangle/2-simplex are its three sides, which are 1-simplices, and its three vertices, which are 0-simplices.

An m-dimensional *geometrical simplicial complex* K in $\mathbb{R}^d$ is a finite collection of simplices in $\mathbb{R}^d$ of dimensions at most m that (i) are glued together along faces, i.e., any intersection between two simplices in K is necessary a face of both of them; and (ii) include all faces of all its simplices, e.g., if the simplex $\sigma_1 = [v_0, v_1, v_2]$ is in K, then the simplices $[v_0, v_1], [v_1, v_2], [v_0, v_2], [v_0], [v_1], [v_2]$ must all also belong to K. Behind each geometrical simplicial complex is an *abstract simplicial complex* — a list of simplices $K = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ with the property that if $\tau \subseteq \sigma \in K$, then $\tau \in K$. This combinatorial description of an abstract simplicial complex is exactly how we describe a graph, i.e., a 1-dimensional simplicial complex, as an abstract collection of edges, i.e., 1-simplices, comprising pairs of vertices. Conversely, any abstract simplicial complex can be realized as a geometrical simplicial complex in $\mathbb{R}^d$ like in Figure 4, an example of a 3-dimensional simplicial complex in $\mathbb{R}^3$. The abstract description of a simplicial complex allows us to treat its simplices as elements in a vector space, a key to defining simplicial homology, as we will see in Section 3.3.

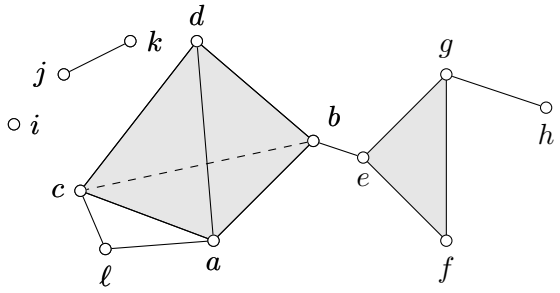


Figure 4: A *geometrical simplicial complex* in $\mathbb{R}^3$ that is a geometrical realization of an *abstract simplicial complex* $K = \{[a, b, c, d], [e, f, g], \ldots, [e, b], \ldots, [a], [b], \ldots, [\ell]\}$ comprising 32 simplices: a single 3-simplex $[a, b, c, d]$, five 2-simplices such as $[a, c, d]$ and $[e, f, g]$, fourteen 1-simplices such as $[e, b]$ and $[g, h]$, twelve 0-simplices $[a], [b], \ldots, [\ell]$. Note that in the geometrical simplicial complex, the simplices intersect along faces.

3.2. Homology and Betti Numbers

Homology is an abstract way to encode the topology of a space by means of a chain of vector spaces and linear maps. We refer readers to Lim (2020) for an elementary treatment requiring nothing more than linear algebra and graph theory. Here we will give an

even simpler treatment restricted to $\mathbb{F}_2 = \{0, 1\}$, the field of two elements with arithmetic performed modulo 2, which is enough for this article.

Let $C_0, C_1, \dots, C_d$ be vector spaces over $\mathbb{F}_2$. Let $\partial_k : C_k \to C_{k-1}$ be linear maps called *boundary operators* that satisfy the condition that "a boundary of a boundary is trivial," i.e.,

$$\partial_k \circ \partial_{k+1} = 0 \tag{3.1}$$

for all $k = 0, \dots, d$. A *chain complex* refers to the sequence

$$0 \xrightarrow{\partial_{d+1}} C_d \xrightarrow{\partial_d} C_{d-1} \xrightarrow{\partial_{d-1}} \cdots \xrightarrow{\partial_{k+1}} C_k \xrightarrow{\partial_k} C_{k-1} \xrightarrow{\partial_{k-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0,$$

where we set $C_{d+1} = C_{-1} = 0$, the trivial subspace. The elements in the image of ∂_k are called *boundaries* and elements in the kernel of ∂_{k-1} are called *cycles*. Clearly $\ker(\partial_k)$ and $\operatorname{im}(\partial_{k+1})$ are both subspaces of C_k and by (3.1),

$$B_k \coloneqq \operatorname{im}(\partial_{k+1}) \subseteq \ker(\partial_k) =: Z_k.$$

We may form the quotient vector space

$$H_k \coloneqq Z_k / B_k = \ker(\partial_k) / \operatorname{im}(\partial_{k+1}), \qquad k = 0, 1, \dots, d,$$

and we will call it the kth *homology group* — the 'group' here refers to the structure of H_k as an abelian group under addition. The elements of H_k are called *homology classes*; note that these are cosets or equivalence classes of the form

$$[\![z]\!] = z + B_k = \{z + b \in Z_k : b \in B_k\}. \tag{3.2}$$

In particular $[\![z+b]\!] = [\![z]\!]$ for any $b \in B_k$. The dimension of H_k as a vector space is denoted

$$\beta_k \coloneqq \dim(H_k), \qquad k = 0, 1, \dots, d.$$

This has special topological significance when H_k is the homology group of a topological space like a simplicial complex K and is called the kth *Betti number* of K. Intuitively β_k counts the number of k-dimensional holes in K. Note that by definition, H_k has a *basis* comprising homology classes $[\![z_1]\!], \dots, [\![z_{\beta_k}]\!]$ for some $z_1, \dots, z_{\beta_k} \in Z_k \subseteq C_k$.

3.3. Simplicial Homology

We present a very simple exposition of simplicial homology tailored to our purposes. The simplification stems partly from our working over a field of two elements $\mathbb{F}_2 \coloneqq \{0, 1\}$. In particular $-1 = +1$ and we do not need to be concerned with signs.

Given an abstract simplicial complex K, we define an $\mathbb{F}_2$-vector space $C_k(K)$ in the following way: Let $K^{(k)} = \{\sigma_1, \dots, \sigma_m\}$ be the set of all k-dimensional simplices in K. Then an element of $C_k(K)$ is a formal linear combination:

$$\sum_{j=1}^{m} n_j \sigma_j, \qquad n_j = 0 \text{ or } 1.$$

In other words, $C_k(K)$ is a vector space over $\mathbb{F}_2$ with $K^{(k)}$ as a basis.

The boundary operators $\partial_k : C_k(K) \to C_{k-1}(K)$ are defined on a k-simplex $\sigma = [v_0, \dots, v_k]$ by

$$\partial_k \sigma \coloneqq \sum_{j=0}^{k} [v_0, \dots, \hat{v}_j, \dots, v_k], \tag{3.3}$$

where $\hat{v}_j$ indicates that v_j is omitted, and extended linearly to all of $C_k(K)$, i.e.,

$$\partial_k \Big(\sum_{j=1}^{m} n_j \sigma_j \Big) \coloneqq \sum_{j=1}^{m} n_j \partial_k \sigma_j.$$

For example, $\partial_1[a,b] = a + b$, $\partial_2[a,b,c] = [a,b] + [b,c] + [c,a]$, $\partial_2([a,b,c] + [d,e,f]) = \partial_2[a,b,c] + \partial_2[d,e,f]$.

Working over $\mathbb{F}_2$ simplifies calculations enormously. In particular, it is easy to check that $\partial_k \circ \partial_{k+1} = 0$ for all $k = 0, \dots, d$, as each $(k-2)$-simplex appears twice in the resulting sum and $2 = 0$ in $\mathbb{F}_2$. Thus (3.1) holds and $\partial_k : C_k(K) \to C_{k-1}(K)$, $k = 0, \dots, d+1$ form a chain complex. The kth homology of the simplicial complex K is then $H_k(K) = \ker(\partial_k)/\operatorname{im}(\partial_{k+1})$ with ∂_k as defined in (3.3). Working over $\mathbb{F}_2$ also guarantees that $H_k(K) \cong \mathbb{F}_2^{\beta_k}$ where β_k is the kth Betti number, i.e.,

$$\beta_k(K) = \dim\big(H_k(K)\big) = \operatorname{nullity}(\partial_k) - \operatorname{rank}(\partial_{k+1}), \qquad k = 0, 1, \dots, d. \tag{3.4}$$

Let $m_k \coloneqq |K^{(k)}|$, the number of k-simplices in K. To compute $\beta_k(K)$, note that with the k-simplices in $K^{(k)}$ as basis, ∂_k is an $m_{k-1} \times m_k$ matrix with entries in $\mathbb{F}_2$ and the problem in (3.4) reduces to linear algebra over $\{0,1\}$ with modulo 2 arithmetic. While this seems innocuous, the cost of computing Betti numbers becomes prohibitive when the size of the simplicial complex $|K| = m_0 + m_1 + \cdots + m_d$ is large. The number of simplices in a d-dimensional simplicial complex K is bounded above by

$$|K| \leq \sum_{i=0}^{d} \binom{m_0}{i+1} \tag{3.5}$$

where m_0 is the size of the vertex set, and the bound is obtained by summing over the maximal number of simplices of each dimension. The cost of computing $\beta(K)$ is about $O(|K|^{2.38})$ (Storjohann, 1996).

We conclude with a discussion of simplicial maps, which we will need in persistent homology. Let K_1 and K_2 be two abstract simplicial complexes. A *simplicial map* is a map defined on their vertex sets $f : K_1^{(0)} \to K_2^{(0)}$ so that for each simplex $\sigma = [v_0, \dots, v_k] \in K_1$, we have that $[f(v_0), \dots, f(v_k)]$ is a simplex in K_2. Such a map induces a map between chain complexes that we will also denote by f, slightly abusing notation, defined by

$$f : C_k(K_1) \to C_k(K_2), \qquad \sum_{j=1}^{m} n_j \sigma_j \mapsto \sum_{j=1}^{m} n_j f(\sigma_j),$$

that in turn induces a map between homologies

$$H_k(f) : H_k(K_1) \to H_k(K_2), \qquad \Big[\!\Big[\sum_{j=1}^{m} n_j \sigma_j \Big]\!\Big] \mapsto \Big[\!\Big[\sum_{j=1}^{m} n_j f(\sigma_j) \Big]\!\Big] \tag{3.6}$$

for all $k = 0, 1, \ldots, d+1$. Recall that $[\![z]\!] \in H_k$ is a shorthand for homology class (3.2).

The composition of two simplicial maps $f : K_1^{(0)} \to K_2^{(0)}$ and $g : K_2^{(0)} \to K_3^{(0)}$ is again a simplicial map $g \circ f : K_1^{(0)} \to K_3^{(0)}$ and thus induces a map between homologies $H_k(g \circ f) : H_k(K_1) \to H_k(K_3)$ for any $k = 0, 1, \ldots, d+1$. For simplicial complexes and simplicial maps, we have that $H_k(g \circ f) = H_k(g) \circ H_k(f)$, a property known as functoriality.

3.4. Vietoris–Rips Complex

There are several ways to obtain a simplicial complex from a point cloud data set but one stands out for its simplicity and widespread adoption in topological data analysis. Note that a point cloud data set is simply a finite set of n points $X \subseteq \mathbb{R}^d$. We will build an abstract simplicial complex K with vertex set $K^{(0)} = X$.

Let δ be a metric on $\mathbb{R}^d$. The *Vietoris–Rips complex* at scale $\varepsilon \geq 0$ on X is denoted by $\mathrm{VR}_\varepsilon(X)$ and defined to be the simplicial complex whose vertex set is X and whose k-simplices comprise all $[x_0, \ldots, x_k]$ satisfying $\delta(x_i, x_j) \leq 2\varepsilon$, $i, j = 0, 1, \ldots, k$. In other words,

$$\mathrm{VR}_\varepsilon(X) \coloneqq \big\{[x_0, \ldots, x_k] : \delta(x_i, x_j) \leq 2\varepsilon,\ x_0, \ldots, x_k \in X,\ k = 0, 1, \ldots, n\big\}.$$

It follows immediately from definition that $\mathrm{VR}_\varepsilon(X)$ is an abstract simplicial complex. Note that it depends on two things — the scale ε and the choice of metric δ. Figure 5 shows an example of Vietoris–Rips complex constructed from a point cloud data set of ten points in $\mathbb{R}^2$ at three different scales $\varepsilon = 0.15$, 0.4, 0.6 and with δ given by the Euclidean norm.

For a point cloud $X \subseteq M \subseteq \mathbb{R}^d$ sampled from a manifold M embedded in $\mathbb{R}^d$, the most appropriate metric δ is the geodesic distance on M and not the Euclidean norm on $\mathbb{R}^d$. This is usually estimated from X using the graph geodesic distance as we will see in Section 6.3.

When X is sampled from a manifold $M \subseteq \mathbb{R}^d$, then for a dense enough sample, and at sufficiently small scale, the topology of $\mathrm{VR}_\varepsilon(X)$ recovers the true topology M in an appropriate sense, made precise in the following result, proved for the Čech complex in Niyogi et al. (2008) and extended to the Vietoris–Rips complex in Attali et al. (2013); Latschev (2001):

Proposition 3.1 *Let $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ be $(\varepsilon/2)$-dense in a compact Riemannian manifold $M \subseteq \mathbb{R}^d$, i.e., for every $p \in M$, there exists $x \in X$ such that $\|p - x\| < \varepsilon/2$. Let τ be the condition number of M. Then for any $\varepsilon < \sqrt{3\tau/5}$, the union of balls $V = \bigcup_{i=1}^n B_\varepsilon(x_i)$ deformation retracts to M. In particular, the homology of V equals the homology of M.*

Roughly speaking the condition number of a manifold embedded in $\mathbb{R}^d$ encodes its local and global curvature properties but the details are too technical for us to go into here.

3.5. Persistent Homology

The Vietoris–Rips complex $\mathrm{VR}_\varepsilon(X)$ of a point cloud data set involves a parameter ε. Here we will discuss how this may be determined.

Homology classes can be very sensitive to small changes. For example, punching even a very small hole in a sphere kills the H_2 homology class, turning a sphere into a topological disk. This affects the estimation of Betti numbers of a manifold from a subset of sampled

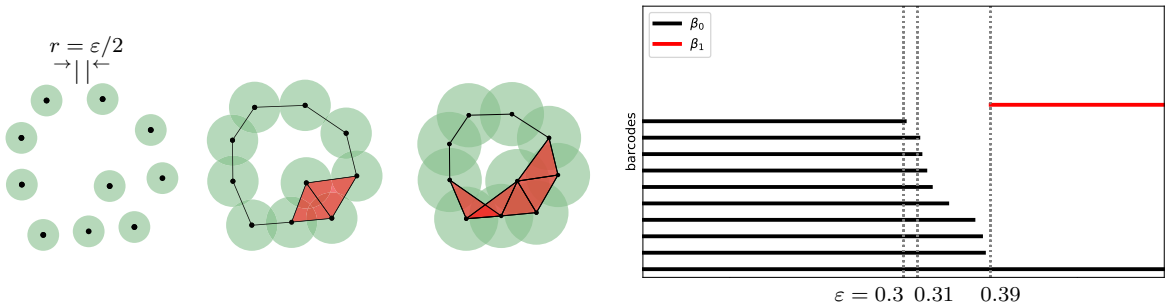


Figure 5: *Left*: Vietoris–Rips complex on ten points in $\mathbb{R}^2$ at scales $\varepsilon = 0.15$, 0.4, 0.6. *Right*: Persistence barcodes obtained from filtration of the Vietoris–Rips complex with scale ε varying from 0 to 0.6. Barcodes show two most prominent topological features of the point cloud, the long black line at the bottom and the long red line near the top, revealing the topology of a circle, i.e., $\beta_0 = \beta_1 = 1$. A 0-homology class dies at times $\varepsilon = 0.3$, 0.31, and 0.39; a 1-homology class is simultaneously born at time $\varepsilon = 0.39$.

point cloud data: there are many scenarios where moving a single point can significantly change the homology estimates. *Persistent homology* (Edelsbrunner et al., 2000) addresses this problem by blending geometry and topology. It allows one to reliably estimate the Betti numbers of a manifold from a point cloud data set, and to a large extent avoids the problem of extreme sensitivity to perturbations. In machine learning lingo, Betti numbers are features associated with the point cloud, and persistent homology enables one to identify features that are robust to noise.

Informally, the idea of persistent homology is to incorporate the scale ε, which varies from 0 to ∞, into homology calculations. At a scale of zero, $\mathrm{VR}_0(X) = \{[x] : x \in X\}$ is a collection of 0-dimensional simplices with $\beta_0 = |X|$ and all other Betti numbers zero. In machine learning lingo the simplicial $\mathrm{VR}_0(X)$ 'overfits' the data X, giving us a discrete topological space. As ε increases, more and more distant points come together to form higher and higher dimensional simplices in $\mathrm{VR}_\varepsilon(X)$ and its topology becomes richer. But as $\varepsilon \to \infty$, eventually all points in X become vertices of a single $|X|$-dimensional simplex, giving us a contractible topological space. So at the extreme ends $\varepsilon = 0$ and $\varepsilon \to \infty$, we have trivial (discrete and indiscrete) topologies and the true answer we seek lies somewhere in between — to obtain a 'right' scale ε_*, we use the so-called *persistence barcodes*. Figure 5 shows an example of a persistence barcode. This is the standard output of persistent homology calculations and it provides a summary of the evolution of topology across all scales.

Generally speaking, a persistence barcode is an interval $[\varepsilon, \varepsilon')$ where its left-end point ε is the scale at which the new feature appears or *born*, and its right-end point ε' is the scale at which that feature disappears or *die*. The length of the interval $\varepsilon' - \varepsilon$ is the *persistence* of that feature. Features that are non-robust to perturbations will produce short intervals; conversely, features that persist long enough, i.e., produce long intervals, are thought to be prominent features of the underlying manifold. For our purpose, the feature in question will always be a homology class in the kth homology group.

If we sample a point cloud satisfying Proposition 3.1 from a sphere with a small punctured hole, we expect to see a single prominent interval corresponding to $\beta_2 = 1$, and a short interval corresponding to the small hole. The persistence barcode would allow us to identify a scale ε_* at which all prominent topological features of M are represented, assuming that such a scale exists. In the following we will assume that we are interested in selecting ε_* from a list of finitely many scales $\varepsilon_0 < \varepsilon_1 < \cdots < \varepsilon_m$ but that they could go as fine as we want. For our purposes, the simplicial complexes below are taken to be $K_j = \mathrm{VR}(X, \varepsilon_j)$, $j = 0, 1, \ldots, m$, but the following discussion holds more generally.

We provide the details for computing persistence barcodes for homology groups, or *persistent homology* in short. This essentially tracks the evolution of homology in a *filtration* of simplicial complexes, which is chain of a nested simplicial complexes

$$K_0 \subseteq K_1 \subseteq K_2 \subseteq \cdots \subseteq K_m. \tag{3.7}$$

We let $f_j : K_j \hookrightarrow K_{j+1}$, $j = 0, 1, \ldots, m-1$ denote the inclusion maps where each simplex of K_j is sent to the same simplex in $K_j \subseteq K_{j+1}$ and regarded as a simplex in K_{j+1}. As f_j is obviously a simplicial map and induces a linear map $H_k(f_j)$ between the homologies of $H_k(K_j)$ and $H_k(K_{j+1})$ as discussed in Section 3.3, composing inclusions $f_{j+p-1} \circ \cdots \circ f_j$ gives us a linear map between any two complexes in a filtration $H_k(K_j)$ and $H_k(K_{j+p})$, $j = 0, 1, \ldots, m-1$, $p = 0, 1, \ldots, m-j-1$. The index j is often referred to as 'time' in this context. As such, for any $i < j$, one can tell whether two simplices belonging to two different homology classes in $H_k(K_i)$ are mapped to the same homology class in $H_k(K_j)$ — if this happens, one of the homology class is said to have *died* while the other has *persisted* from time i to j. If a homology class in $H_k(K_{j+1})$ is not in the image of $H_k(f_j)$, we say that its homology class is *born* at time $j+1$. The persistence barcodes simply keep track of the birth and death times of the homology classes.

To be completely formal, we have the two-dimensional complex called a *persistent complex* shown in Figure 6 with horizontal maps given by boundary maps $\partial_k : C_k(K_j) \to C_{k-1}(K_j)$ and vertical maps given by simplicial maps $f_j : C_k(K_j) \to C_k(K_{j+1})$. Thanks to a celebrated structure theorem (Zomorodian and Carlsson, 2005) which guarantees that a persistent barcode completely describes the structure of a persistent complex in an appropriate sense, we may avoid persistent complexes like Figure 6 and work entirely with persistence barcodes like the one on the right of Figure 5.

Henceforth we let $K_j = \mathrm{VR}(X, \varepsilon_j)$, $j = 0, 1, \ldots, m$, be the Vietoris–Rips complex of our point cloud data at scales $\varepsilon_0 < \varepsilon_1 < \cdots < \varepsilon_m$. An important fact to note is that persistence barcodes may be computed without having to compute homology at every scale ε_j, or, equivalently, at every time j. To identify the homology classes in $H_k(K_j)$ that persist from time j to time $j+p$, there is no need to compute $H_k(K_{j+1}), \ldots, H_k(K_{j+p})$ individually as one might think. Rather, one considers the *p-persistent kth homology group*

$$H_k^{j,p} = Z_k^j / (B_k^{j+p} \cap Z_k^j),$$

where $p = 1, 2 \ldots, m-j$. This captures the cycles in $C_k(K_j)$ that contribute to homology in $C_k(K_{j+p})$. One may consistently choose a basis for each $H_k^{j,p}$ so that the basis elements are compatible for homologies across $H_k(K_{j+1}), \ldots, H_k(K_{j+p})$ for all possible values of k and p. This allows one to track the persistence of each homology class throughout the

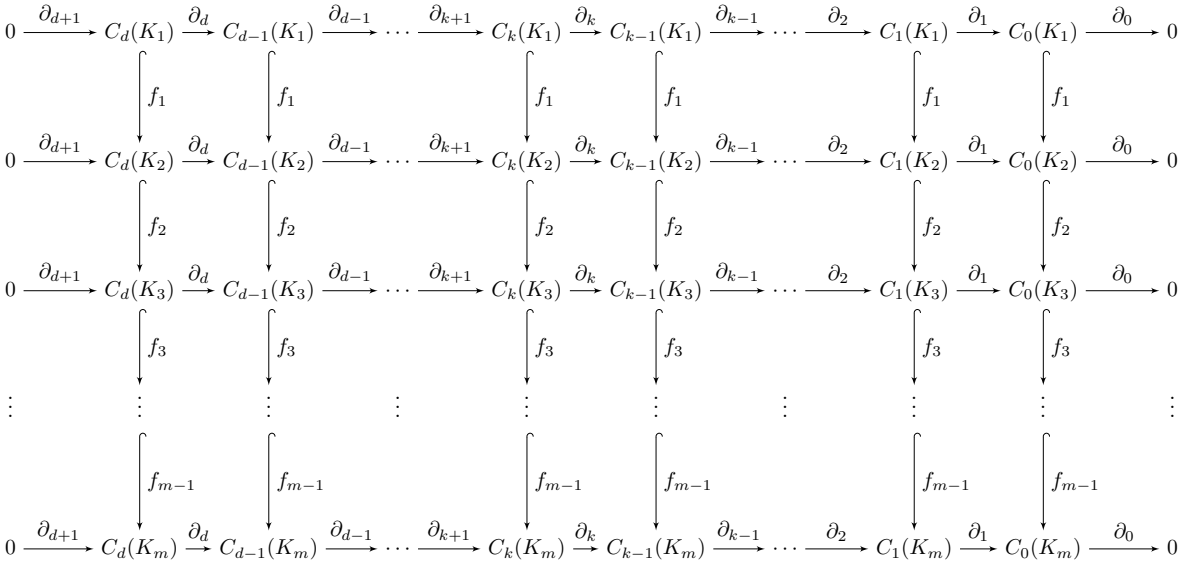


Figure 6: Persistence complex of the filtration $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m$.

filtration (3.7) and thus obtain the persistence barcodes: roughly speaking, with the right basis, we may simultaneously represent the boundary maps on $C_k(K_j)$ as matrices in a column-echelon form and read off the dimension of $H_k^{j,p}$, known as the *p-persistent kth Betti number* $\beta_k^{j,p}$, from the pivot entries in these matrices. For details we refer readers to Edelsbrunner et al. 2000; Zomorodian and Carlsson 2005.

3.6. Homology Computations in Practice

Actual computation of homology from a point cloud data set is more involved than what one might surmise from the description in the last few sections. We will briefly discuss some of the issues involved.

Before we even begin to compute the homology of the point cloud data $X \subseteq M \subseteq \mathbb{R}^d$, we will need to perform a few preprocessing steps, as depicted in Figure 7. These steps are standard practice in topological data analysis: (i) We smooth out X and discard outliers to reduce noise. (ii) We then select the scale ε and construct the corresponding Vietoris–Rips complex $\mathrm{VR}_\varepsilon(X)$. (iii) We simplify $\mathrm{VR}_\varepsilon(X)$ in a way that reduces its size but leaves its topology unchanged. All preprocessing operations that can have an effect on the homology are in steps (i) and (ii), the simplification in step (iii) is done to accelerate computations without altering homology. The homology of the preprocessed simplicial complex $\mathrm{VR}_\varepsilon(X)$ is assumed to closely approximate that of the underlying manifold M.

Note that the size of the final simplicial complex on which we perform homology calculations is the most important factor in computational cost. While increasing the number of points sampled from a manifold, i.e., the size of X, up to the point in Proposition 3.1 improves the accuracy of our homology estimates, it also results in a simplicial complex $\mathrm{VR}_\varepsilon(X)$ that is prohibitively large for carrying out computations, as we saw from (3.5).

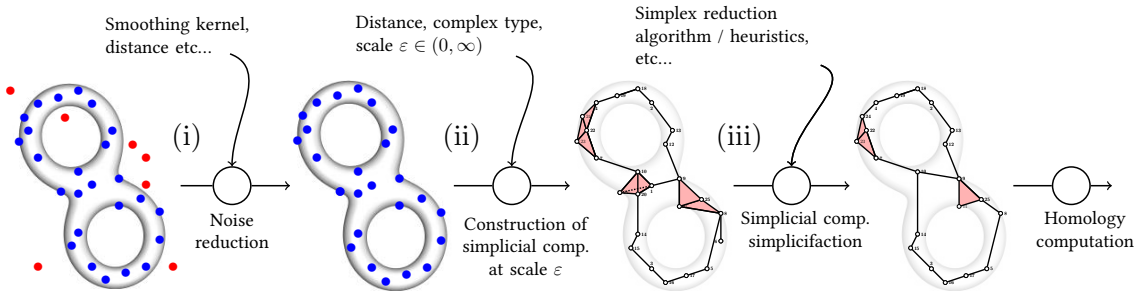


Figure 7: Pipeline for computation of homology from a point cloud data.

But since we are not concerned with the geometry of the underlaying manifold, only its topology, it is desirable to construct a small simplicial complex with minimal topology distortion. A well-known simplification is the *Witness complex* (De Silva and Carlsson, 2004), which gives results of nearly the same quality with a simplicial complex of smaller size constructed from so-called landmark points. Another cost-saving strategy is to stop computations early at a smaller ε as $\mathrm{VR}_\varepsilon(X)$ grows exponentially with ε. Many other tactics have been proposed (Boissonnat et al., 2018; Dlotko and Wagner, 2014; Mischaikow and Nanda, 2013) and we take advantage of them in our calculations implicitly.

The takeaway is that persistence barcodes are considerably more expensive to compute than homology at a single fixed scale ε. Therefore, running full persistent homology in the context of deep neural network poses some steep challenges: modern deep neural networks operate on very high-dimensional and very large data sets, a setting in which persistent homology cannot be used directly due to the overwhelming computational and memory costs. This situation is exacerbated by the fact that neural networks are randomly trained, with potentially big variation in the learned decision boundaries, and one needs to repeat the same computations many times to ensure reliability. As such, a direct analysis of the persistent homology of the best known deep learning data sets, e.g., `SVHN` of Netzer et al. 2011, `CIFAR-10` of Krizhevsky et al. 2009, `ImageNet` of Deng et al. 2009, is largely beyond the reach of current technology. We will return to this point later when we introduce our methodology for monitoring topology transformations in a neural network. In particular, we will see in Section 6.3 that our experiments are designed in such a way that although we will compute homology at every layer, we only need to compute persistence barcodes once, before the data set is passed through the layers.

4. Overview of Problem and Methodology

We will use *binary classification*, the most basic and fundamental problem in supervised learning, as our platform for studying how neural networks change topology. More precisely, we seek to classify two different probability distributions supported on two disjoint manifolds $M_a, M_b \subseteq \mathbb{R}^d$. We assume that the distance $\inf\{\|x-y\| : x \in M_a,\ y \in M_b\}$ can be arbitrarily small but not zero. So there exists an ideal classifier with zero prediction error. Here and henceforth, $\|\cdot\|$ will denote the Euclidean norm.

We sample a large but finite set of points $T \subseteq M_a \cup M_b$ uniformly and densely, so that the Betti numbers of M_a and M_b can be faithfully obtained from the point cloud data sets $T \cap M_a$ and $T \cap M_b$ as described in Section 3. Our training set is a labeled point cloud data set, i.e., $x \in T$ is labeled to indicate whether $x \in M_a$ or M_b. We will use $T_a \coloneqq T \cap M_a$ and $T_b \coloneqq T \cap M_b$, or rather, their Vietoris–Rips complexes as described in Section 3.4, as finite proxies for M_a and M_b.

Our feedforward neural network $\nu : \mathbb{R}^d \to [0, 1]$ is given by the usual composition

$$\nu = s \circ f_l \circ f_{l-1} \circ \cdots \circ f_2 \circ f_1, \tag{4.1}$$

where each *layer* of the network $f_j : \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}$, $j = 1, \dots, l$, is the composition of an affine map $\rho_j : \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}$, $x \mapsto A_j x + b_j$, with an *activation* function $\sigma : \mathbb{R}^{n_{j+1}} \to \mathbb{R}^{n_{j+1}}$; and $s : \mathbb{R}^{n_{l+1}} \to [0, 1]$ is the *score* function. The *width* n_j is the number of nodes in the jth layer and we set $n_1 = d$ and $n_{l+1} = p$. For $j = 1, \dots, l$, the composition of the first through jth layers is denoted

$$\nu_j \coloneqq f_j \circ \cdots \circ f_2 \circ f_1 \quad \text{and} \quad \nu = s \circ \nu_l.$$

We assume that s is a *linear classifier* and thus the decision boundary of s is a hyperplane in $\mathbb{R}^p$. For notational convenience later, we define the '$(l + 1)$th layer' $\nu_{l+1} \coloneqq s$ to be the score function and the '0th layer' ν_0 to be the identity function on $\mathbb{R}^d$; in particular, $\beta(M) = \beta\big(\nu_0(M)\big)$.

We train an l-layer neural network $\nu : \mathbb{R}^d \to [0, 1]$ on a training set $T \subseteq M_a \cup M_b$ to classify samples into class a or b. As usual, the network's output for a sample $x \in T$ is interpreted to be the probability of the event '$x \in M_a$.' In all our experiments, we train ν until it correctly classifies all $x \in T$ — we will call such a network ν *well-trained*. In fact, we sampled T so densely that in reality ν also has near zero misclassification error on any test set $S \subseteq (M_a \cup M_b) \setminus T$; and we trained ν so thoroughly that its output is concentrated near 0 and 1. For all intents and purposes, we may treat ν as an ideal classifier.

We deliberately choose $M_a \cup M_b$ to have doubly complicated topologies in the following sense:

(i) For each $i = a, b$, the component M_i itself will have complicated topologies, with multiple components, i.e., large $\beta_0(M_i)$, as well as multiple k-dimensional holes, i.e., large $\beta_k(M_i)$.

(ii) In addition, M_a and M_b will be entangled in a topologically complicated manner. See Figures 8 and 9 for example. They not only cannot be separated by a hyperplane but any decision boundary $D \subseteq \mathbb{R}^d$ that separates them will necessarily have complicated topology itself.

In terms of the topological complexity in (2.1), $\omega(M_a)$, $\omega(M_b)$, $\omega(D)$ are all large.

Our experiments are intended to reveal how the topologies of $\nu_j(M_a)$ and $\nu_j(M_b)$ evolve as j runs from 1 through l, for different manifolds M_a, M_b entangled in different ways, for different number of layers l and choices of widths $n_1, \dots, n_l$, and different activations σ. Getting ahead of ourselves, the results will show that a well-trained neural network $\nu : \mathbb{R}^d \to [0, 1]$ reduces the topological complexity of M_a and M_b on a layer-by-layer basis until, at the output, we see a simple disentangled arrangement where the point cloud T

gets mapped into two clusters of points $\nu(T_a)$ and $\nu(T_b)$ on opposite ends of $[0, 1]$. This indicates that an initial decision boundary $D \subseteq \mathbb{R}^d$ of complicated topology ultimately gets transformed into a hyperplane in $\mathbb{R}^p$ by the time it reaches the final layer. We measure and track the topologies of $\nu_j(M_a)$ and $\nu_j(M_b)$ directly, but our approach only permits us to indirectly observe the topology of the decision boundary separating them.

5. Real Versus Simulated Data

We perform our experiments on a range of both real-world and simulated data sets to validate our premise that a neural network operates by simplifying topology. We explain why each is indispensable to our goal.

Unlike real-world data, simulated data may be generated in a controlled manner with well-defined topological features that are known in advance (crucial for finding a single scale for all homology computations). Moreover, with simulated data we have clean samples and may skip the denoising step mentioned in the previous section. We can generate samples that are uniformly distributed on the underlaying manifold, and ensure that the assumptions of Section 4 are satisfied. In addition, we may always simulate a data set with a perfect classifier, whereas such a classifier for a real-wold data set may not exist. For convincing results, we train our neural network to perfect accuracy on training set and near-zero generalization error — this is often impossible for real-world data. Evidently, if there is no complete separation of one category M_a from the other M_b, i.e., $M_a \cap M_b \neq \varnothing$, the manifold $M = M_a \cup M_b$ will be impossible to disentangle. Such is often the case with real-world data sets, which means that they may not fit our required setup in Section 4.

Nevertheless, the biggest issue with real-world data sets is that they have *vastly* more complicated topologies that are nearly impossible to determine. Even for the Mumford data set (Lee et al., 2003), a mere collection of 3×3-pixels of high contrast patches of natural images, it took many years to discover its topology (Carlsson et al., 2008), and whether the conclusion (that it has the topology type of a Klein bottle) is correct is still a matter of some debate. Figuring out, say, the topology of the manifold of cat images within the space of all possible images is well-beyond our capabilities for the foreseeable future.

Since our experiments on simulated data allow us to pick the right scale to compute homology, we only need to compute homology at one single scale. On the other hand, for real data we will need to find the persistence barcodes, i.e., determine homology over a full range of scales. Consequently, our experiments on simulated data are extensive — we repeat our experiments for each simulated data set over a large number of neural networks of different architectures to examine their effects on topology changes. In all we ran more than 10,000 homology computations on our simulated data sets since these computations are cheap and reliable. In comparison, our experiments on real-world data are more limited in scope as it is significantly more expensive to compute persistence barcodes then to compute homology at a single scale.

As such, we use simulated data to fully explore and investigate the effects of depth, width, shapes, activation functions, and various combinations of these factors on the topology-changing power of neural networks. Thereafter we use real-world data to validate the findings we draw from the simulated data sets.

6. Methodology

In this section, we describe the full details of our methodology for (i) simulating topologically nontrivial data sets in a binary classification problem; (ii) training a variety of neural networks to near-perfect accuracy for such a problem; (iii) determining the homology of the data set as it passes through the layers of such a neural network. For real data sets, step (i) is of course irrelevant, but steps (ii) and (iii) will apply with minor modifications; these discussions will be deferred to Section 8.

The key underlying reason for designing our experiments in the way we did is relative computational costs:

- multiparameter persistent homology is much more costly than persistent homology;
- persistent homology is much more costly than homology;
- homology is much more costly than training neural networks.

As such, we train tens of thousands of neural networks to near zero generalization error; for each neural network, we compute homology at every layer but we compute persistent homology only once; and we avoid multiparameter persistent homology altogether.

6.1. Generating Data Sets

We generate three point cloud data sets D-I, D-II, D-III in a controlled manner to have complicated but manageable topologies that we *know in advance*.

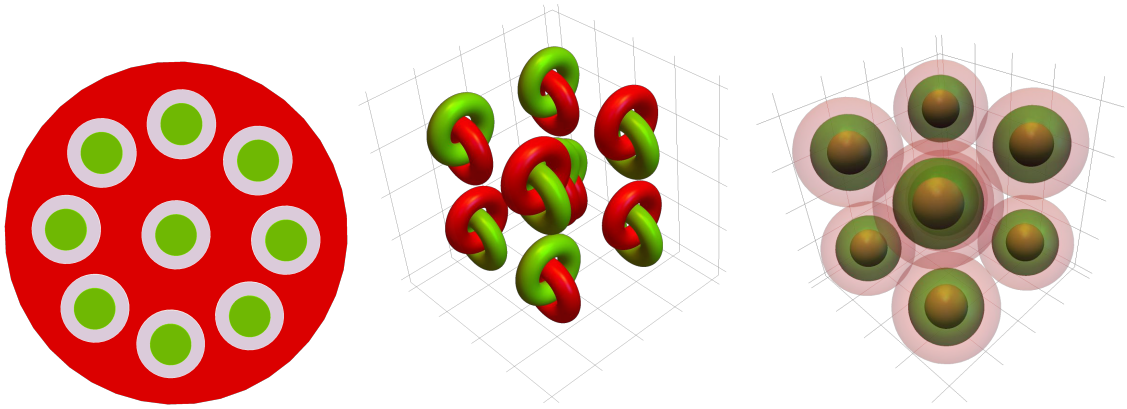


Figure 8: The manifolds underlying data sets D-I, D-II, D-III (*left to right*). The green M_a represents category a; the red M_b represents category b.

D-I is sampled from a two-dimensional manifold comprising M_a, nine green disks, positioned in M_b, a larger disk with nine holes, as on the left of Figure 8. We clearly have $\beta(M_a) = (9, 0)$ and $\beta(M_b) = (1, 9)$ (one connected component, nine holes). **D-II** is sampled from a three-dimensional manifold comprising nine disjoint pairs of a red solid torus linked with a green solid torus (a single pair is shown in Figure 9). M_a (resp. M_b) is the union of all nine green (resp. red) solid tori. So $\beta(M_a) = \beta(M_b) = (9, 9, 0)$. **D-III** is sampled from a three-dimensional manifold comprising nine disjoint units of the following — a large

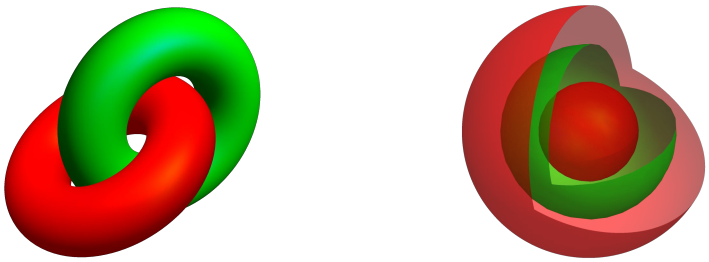


Figure 9: *Left*: D-II comprises nine pairs of such linked rings. *Right*: D-III comprises nine units of such doubly concentric spherical shells.

red spherical shell enclosing a smaller green spherical shell enclosing a red ball; the green spherical shell is trapped between the red spherical shell and the red ball. M_a is the union of all nine green spherical shells and M_b is the union of the nine red spherical shells and the nine red balls. So we have $\beta(M_a) = (9, 0, 9)$ and $\beta(M_b) = (18, 0, 9)$ (see Figures 8 and 9 for more details; in Figure 9 the spherical shells are shown with portions omitted to reveal the objects enclosed within). In all cases, M_a and M_b are entangled in such a way that any decision boundary separating them necessarily has highly complex topology.

The point cloud data sets D-I and D-III are sampled on a grid whereas D-II is sampled uniformly from the solid tori. The difference in sampling schemes is inconsequential, as we sample so densely that there is no difference in training and testing behaviors.

6.2. Training Neural Networks

Our goal is to examine the topology changing effects of (i) different *activations*: tanh, leaky ReLU set to be $\max(x, 0.2x)$, and ReLU; (ii) different *depths* of four to ten layers; and (iii) different *widths* of six to fifty neurons. So for any given data set (D-I, D-II, D-III) and any given architecture (depth, width, activation), we tracked the Betti numbers through all layers for at least 30 well-trained neural networks. The repetition is necessary — given that neural network training involves a fair amount of randomization in initialization, batching, optimization, etc — to ensure that what we observe is not a fluke.

To train these neural networks to our requirements — recall that this means zero training error and a near-zero ($\approx 0.01\%$) generalization error — we relied on TensorFlow (version 1.12.0 on Ubuntu 16.04.1). Training is done on cross-entropy categorical loss with standard Adam optimizer (Kingma and Ba, 2015) for up to 18,000 training epochs. Learning rate is set to 0.02–0.04 with an exponential decay, i.e., $\eta^{t/d}$ where t is the training epoch normalized by $d = 2500$. For the 'bottleneck architectures' where the widths narrow down in the middle layers (see Table 1), the decay is set to 4000 and $\eta = 0.5$. We use the softmax function as the score function in all of our networks, i.e., $s : \mathbb{R}^p \to \mathbb{R}^p$ whose ith coordinate is

$$s_i(x) = e^{x_i}/(e^{x_1} + \cdots + e^{x_p}), \qquad i = 1, \ldots, p,$$

where p is the number of categories. In our case, $p = 2$ and $i = a, b$; the image of s is then a 1-simplex, i.e., isomorphic to the unit interval $[0, 1]$, and we may regard the score function as $s : \mathbb{R}^p \to [0, 1]$, consistent with our choice in Section 4.

Table 1 summarizes our experiments: the data set used, the activation type, the width of each layer, and the number of successfully trained neural networks of that architecture obtained. The first number in the sequence of the third column gives the dimension of the input, which is two for the two-dimensional D-I and three for the three-dimensional D-II and D-III. The last number in that sequence is always two since they are all binary classification problems. To give readers an idea, training any one of these neural networks to near zero generalization error takes at most 10 minutes, often much less.

data set	activation	neurons in each layer	#
D-I	tanh	2-15-15-15-15-15-15-15-15-2	30
D-I	leaky ReLU	2-15-15-15-15-15-15-15-15-2	30
D-I	leaky ReLU	2-05-05-05-05-03-05-05-05-2	30
D-I	leaky ReLU	2-15-15-15-03-15-15-15-15-2	30
D-I	leaky ReLU	2-50-50-50-50-50-50-50-50-2	30
D-I	ReLU	2-15-15-15-15-15-15-15-15-2	30
D-II	tanh	3-15-15-15-15-15-15-15-15-2	32
D-II	leaky ReLU	3-15-15-15-15-15-15-15-15-2	36
D-II	ReLU	3-15-15-15-15-15-15-15-15-2	31
D-II	tanh	3-25-25-25-25-25-25-25-25-2	30
D-II	leaky ReLU	3-25-25-25-25-25-25-25-25-2	30
D-II	ReLU	3-25-25-25-25-25-25-25-25-2	30
D-III	tanh	3-15-15-15-15-15-15-15-15-2	30
D-III	leaky ReLU	3-15-15-15-15-15-15-15-15-2	46
D-III	ReLU	3-15-15-15-15-15-15-15-15-2	30
D-III	tanh	3-50-50-50-50-50-50-50-50-2	30
D-III	leaky ReLU	3-50-50-50-50-50-50-50-50-2	30
D-III	ReLU	3-50-50-50-50-50-50-50-50-2	34
D-I	tanh	2-15-15-15-15-2	30
D-I	tanh	2-15-15-15-15-15-15-15-2	30
D-I	leaky ReLU	2-15-15-15-2	30
D-I	leaky ReLU	2-15-15-15-15-15-15-15-2	30
D-I	ReLU	2-15-15-15-2	30
D-I	ReLU	2-15-15-15-15-15-15-15-2	30
D-II	tanh	3-15-15-15-2	31
D-II	tanh	3-15-15-15-15-2	31
D-II	tanh	3-15-15-15-15-15-15-2	30
D-II	leaky ReLU	3-15-15-15-2	31
D-II	leaky ReLU	3-15-15-15-15-2	30
D-II	leaky ReLU	3-15-15-15-15-15-2	30
D-II	leaky ReLU	3-15-15-15-15-15-15-2	31
D-II	leaky ReLU	3-15-15-15-15-15-15-15-2	42
D-II	ReLU	3-15-15-15-2	32
D-II	ReLU	3-15-15-15-15-2	32
D-II	ReLU	3-15-15-15-15-15-15-15-2	31
D-III	tanh	3-15-15-15-15-15-2	30
D-III	tanh	3-15-15-15-15-15-15-15-2	31
D-III	leaky ReLU	3-15-15-15-15-15-2	30
D-III	leaky ReLU	3-15-15-15-15-15-15-15-2	30
D-III	ReLU	3-15-15-15-15-15-2	33
D-III	ReLU	3-15-15-15-15-15-15-15-2	32

Table 1: First column specifies the data set on which we train the networks. Next two columns give the activation used and a sequence giving the number of neurons in each layer. Last column gives the number of well-trained networks obtained.

6.3. Computing Homology

For each of the neural networks obtained in Section 6.2, we track the topology of the respective point cloud data set as it passes through the layers. This represents the bulk of the computational effort, way beyond that required for training neural networks in Section 6.2. With simulated data, we are essentially assured of a perfectly clean data set and the preprocessing step in Figure 7 may be omitted. We describe the rest of the work below.

The metric δ used to form our Vietoris–Rips complex is given by the graph geodesic distance on the k-nearest neighbors graph determined by the point cloud $X \subseteq \mathbb{R}^d$. As this depends on k, a positive integer specifying the number of neighbors used in the graph construction, we denote the metric by δ_k. In other words, the Euclidean distance on $\mathbb{R}^d$ is used only to form the k-nearest neighbors graph and do not play a role thereafter. For any $x_i, x_j \in X$, the distance $\delta_k(x_i, x_j)$ is given by the minimal number of edges between them in the k-nearest neighbors graph. Each edge, regardless of its Euclidean length, has the same length of one when measured in δ_k.

The metric δ_k has the effect of normalizing distances across layers of a neural network while preserving connectivity of nearest neighbors. This is important for us as the local densities of a point cloud can vary enormously as it passes through a layer of a well-trained neural network — each layer stretches and shrinks different regions, dramatically altering geometry as one can see in the bottom halves of Figures 11, 12, and 13. Using an intrinsic metric like δ_k ameliorates this variation in densities; it is robust to geometric changes and yet reveals topological ones. Furthermore, our choice of δ_k allows for comparison across layers with different numbers of neurons. Note that if $d \neq p$, the Euclidean norms on $\mathbb{R}^d$ and $\mathbb{R}^p$ are two different metrics on two different spaces with no basis for comparison. Had we used Euclidean norms, two Vietoris–Rips complexes of the same scale ε in two different layers cannot be directly compared — the scale needs to be calibrated somehow to reflect that they live in different spaces. Using δ_k avoids this problem.

This leaves us with two parameters to set: k, the number of neighbors in the nearest neighbors graph and ε, the scale at which to build our Vietoris–Rips complex. This is where persistent homology, described at length in Section 3.5, comes into play. Informed readers may think that we should use *multiparameter persistence* since there are two parameters but this is prohibitively expensive as the problem is EXPSPACE-complete (Carlsson et al., 2010) and its results are not quite what we need; for one, there is no multiparameter analogue of persistence barcodes (Carlsson and Zomorodian, 2009).

To choose an appropriate (k_*, ε_*) for a point cloud $X \subseteq M \subseteq \mathbb{R}^d$, we construct a filtered complex over the two parameters: Let $\mathrm{VR}_{k,\varepsilon}(X)$ be the Vietoris–Rips complex of X with respect to the metric δ_k at scale ε. In our case, we know the topology of the underlying manifold M completely as we generated it in Section 6.1 as part of our data sets. Thus we may ascertain whether our chosen value (k_*, ε_*) gives a Vietoris–Rips complex $\mathrm{VR}_{k_*,\varepsilon_*}(X)$ with the same homology as M. Set $\varepsilon = 1$. We determine a value of k_* with persistent homology on the k-filtered complex in the metric δ_k with correct zeroth homology, i.e., k_* is chosen so that

$$\beta_0\big(\mathrm{VR}_{k_*,1}(X)\big) = \beta_0(M).$$

Set $k = k_*$. We determine a value of ε_* with persistent homology on the ε-filtered complex in the metric δ_{k_*} with correct first and second homologies, i.e., ε_* is chosen so that

$$\beta_1\big(\mathrm{VR}_{k_*,\varepsilon_*}(X)\big) = \beta_1(M) \qquad \text{and} \qquad \beta_2\big(\mathrm{VR}_{k_*,\varepsilon_*}(X)\big) = \beta_2(M).$$

If there is a range of parameters that all recover the correct homology we pick our (k_*, ε_*) closest to the middle of the range. Once these parameters are set, we keep them fixed for our homology computations across all layers of the network.

The parameters chosen via the aforementioned procedure for our data sets are as follows. For D-I, we have $k_* = 14$ neighbors, and scale is set at $\varepsilon_* = 2.5$; recall that this means that $x_0, x_1, \ldots, x_n \in X$ form an n-simplex in $\mathrm{VR}_{14,2.5}(X)$ whenever $\delta_{14}(x_i, x_j) \leq 2.5$ for all i, j. For both D-II and D-III, we have $k_* = 35$ and $\varepsilon_* = 2.5$. Figure 10 shows some (not all) of the Betti numbers for these three data sets over a range of values of (k, ε): green (resp. red) dots indicate integral points on the (k, ε)-plane with correct (resp. incorrect) Betti numbers and the blue dot marks the (k_*, ε_*) selected in each case.

Our homology and persistent homology computations are all performed using the Eirene package in Julia 0.6.4 (Henselman and Ghrist, 2016). To give readers an idea, the time required to compute a single Betti number from a point cloud X ranges from a few tens of seconds, if $X \subseteq \mathbb{R}^5$ is the output of a five-neuron-wide layer, to about 30 minutes, if $X \subseteq \mathbb{R}^{50}$ is the output of a 50-neuron wide layer. On the other hand, the time taken for the persistent homology computations to obtain a single (k_*, ε_*) is in excess of 80 minutes. These computations are run in parallel over 12 cores of an Intel i7-8750H-2.20GHz processor with 9,216KB cache and 32GB DDR4-2666MHz RAM. The jobs are fed in a queue, with each core limited to 9GB of memory.

6.4. Overview of Our Experiments

All our experiments on simulated data may be described at a high level as follows: (i) generate a manifold $M = M_a \cup M_b \subseteq \mathbb{R}^d$ with $M_a \cap M_b = \varnothing$; (ii) densely sample a point cloud $X \subseteq M$ and let $X_i := X \cap M_i$, $i = a, b$; (iii) train an l-layer neural network $\nu : \mathbb{R}^d \to [0, 1]$ on the labeled training set $X_a \cup X_b$ to classify points on M; (iv) compute homology of the output at the jth layer $\nu_j(X_i)$, $j = 0, 1, \ldots, l+1$ and $i = a, b$. This allows us to track the topology of M_i as it passes through the layers. Steps (i) and (ii) are described in Section 6.1, step (iii) in Section 6.2, and step (iv) in Section 6.3. The neural network notations are as in Section 4. Results will be described in Section 7. In reality,

data set	training neural networks	homology computations
D-I	7,800	2,600
D-II	45,000	11,250
D-III	37,800	9,450

Table 2: Comparison of sample sizes for computations in Sections 6.2 and 6.3.

the point cloud X_i used in step (iv) is not the same as the training set X_i used in step (iii) as it is considerably more expensive to compute homology (see Section 6.3) than to train a neural network to near zero generalization error (see Section 6.2). As such the size of a point cloud used in homology computations is a fraction (about 1/4) of that used to train a neural network.

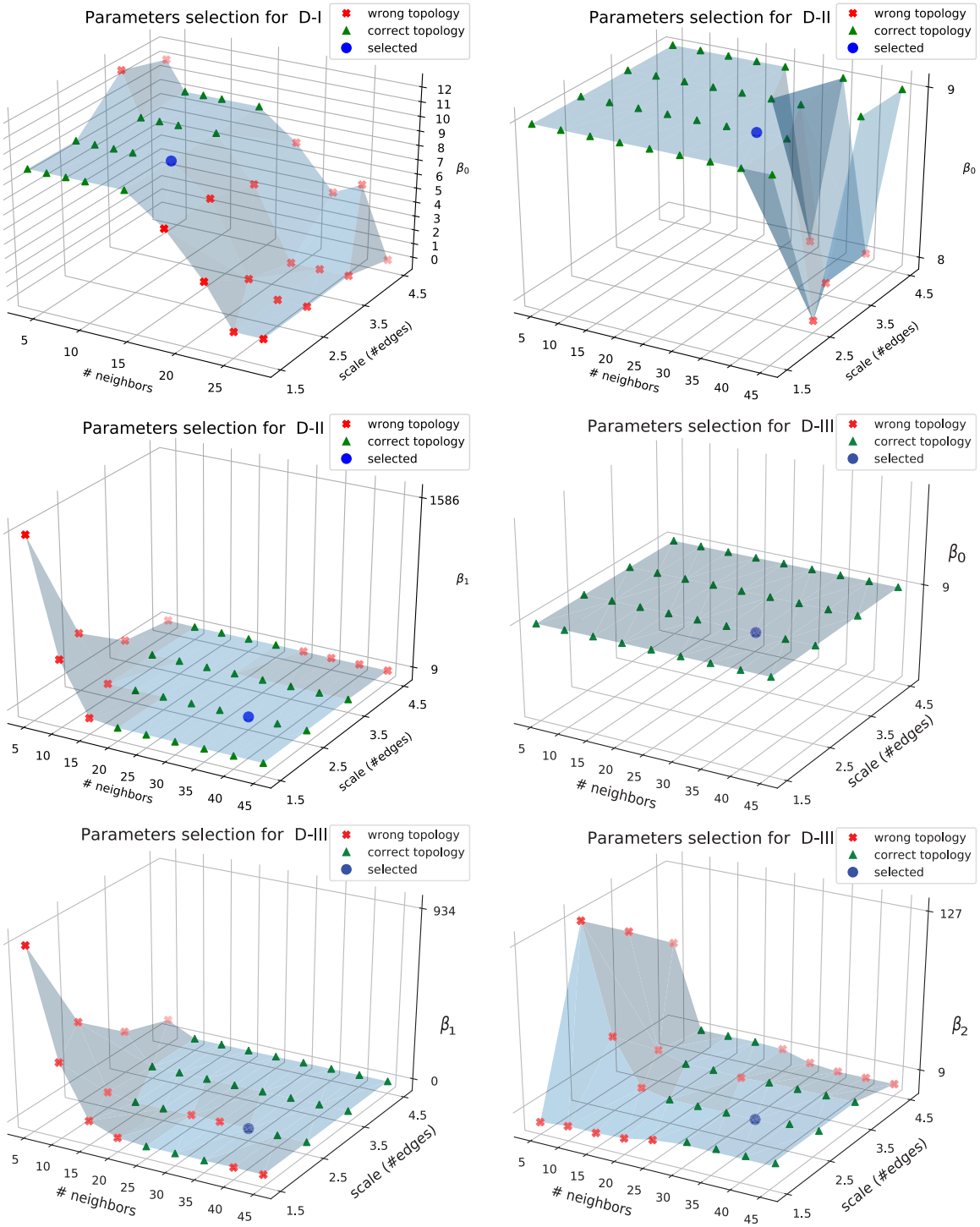


Figure 10: For each combination of parameters k and ε, we determine whether the homology of $\mathrm{VR}_{k,\varepsilon}(X)$ matches the homology of the manifold M from which X is sampled. We marked those values of (k, ε) with correct homology in green and those with incorrect homology in red. Our choice (k_*, ε_*) in each case is marked with a blue dot.

7. Results and Discussions

We present the results from our experiments to analyze how a well-trained neural network simplifies the topology of a data set, and discuss what one may surmise from these results. We start with the three simulated data sets D-I, D-II, D-III in Section 6.1 since the results are the most striking in this case. To validate that these observations indeed extend to real data, we repeat our experiments on four real-world data sets in Section 8.

Topological simplification evident across training instances: Figure 11 shows the result for our simplest data set D-I, where M_a comprises nine contractible components and so higher Betti numbers are irrelevant (all zero). Here we present every curve corresponding to every neural network trained on D-I. Recall that we do at least 30 runs for each experiment to account for the inherent randomness. They all show consistent profiles — a clear decay in β_0 across the layers although tanh activation (blue graph) shows larger variance in this decay than leaky ReLU (red graph) and ReLU (green graph). The profiles shown in Figure 11 are representative of other experiments on higher Betti numbers and on other data sets. To avoid clutter, in the corresponding figures for D-II and D-III (Figures 12 and 13), we omit curves corresponding to the individual runs and show only the curve of their means (dark curve in the middle) and the region of half standard deviation (shaded region). The bottom diagrams in Figure 11 show how M_a changes from layer to layer by projecting onto its first two principal components (note that the intervening layers are in $\mathbb{R}^{15}$ and so such a projection is necessary for visualization).

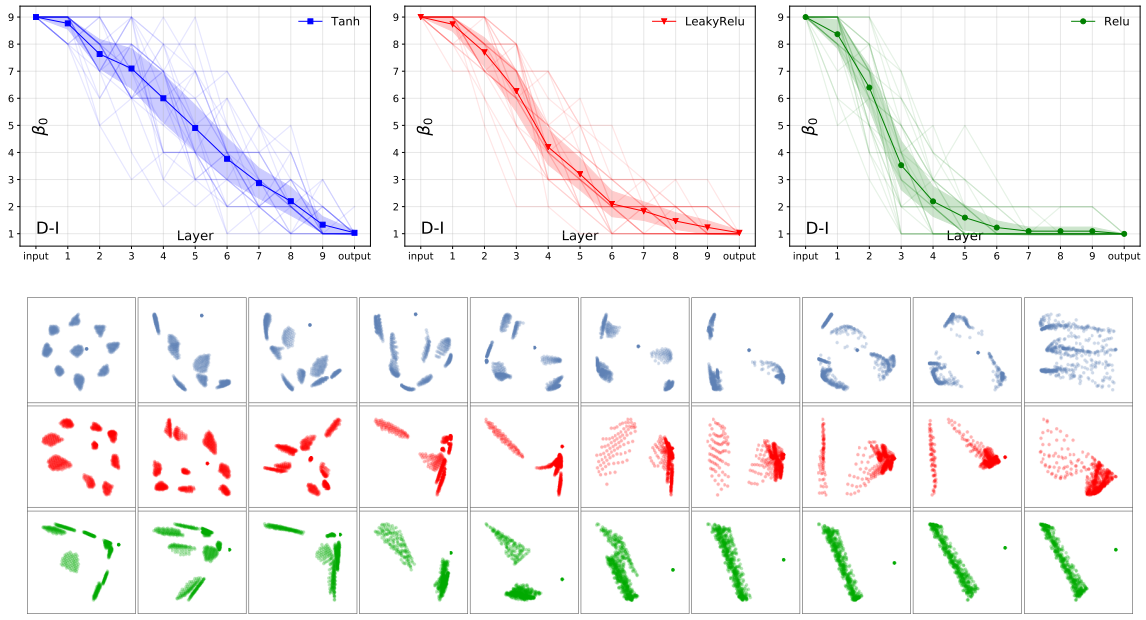
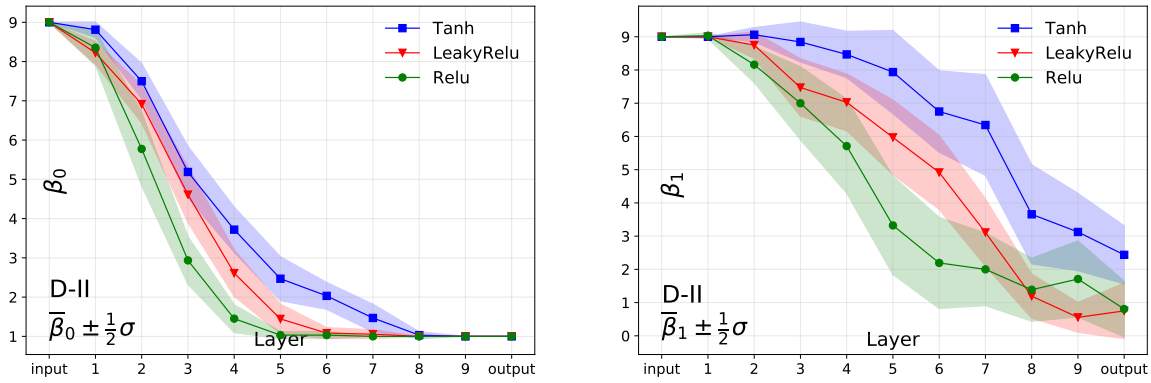


Figure 11: *Top*: Faint curves show individual profiles, dark curves show averaged profiles of $\beta_0\big(\nu_j(M_a)\big)$, $j = 0, 1, \ldots, 10$, in data set D-I. Shaded region is the region of $\pm$ half standard deviation about average curve. Networks have different activations — blue for tanh, red for leaky ReLU, green for ReLU; but same architecture — two-dimensional input and output, with fifteen neurons in each of the nine intervening layers. *Bottom*: Projections of $\nu_j(M_a)$, $j = 0, 1, \ldots, 10$, on the first two principal components, color-coded according to activations.

Nonhomeomorphic activations induce rapid topology changes: As the faint blue lines in Figure 11 reveal, tanh activation is less effective at reducing Betti numbers, occasionally even increasing them over layers. In all data sets, across all our experiments, the nonhomeomorphic activation ReLU exhibits the most rapid reductions in all Betti numbers. The top halves of Figures 11, 12, and 13 show the results for a ten-layer network (see captions for specifics). The different rates at which topological changes occur are also evident from the principal components projections in the bottom half of these figures.



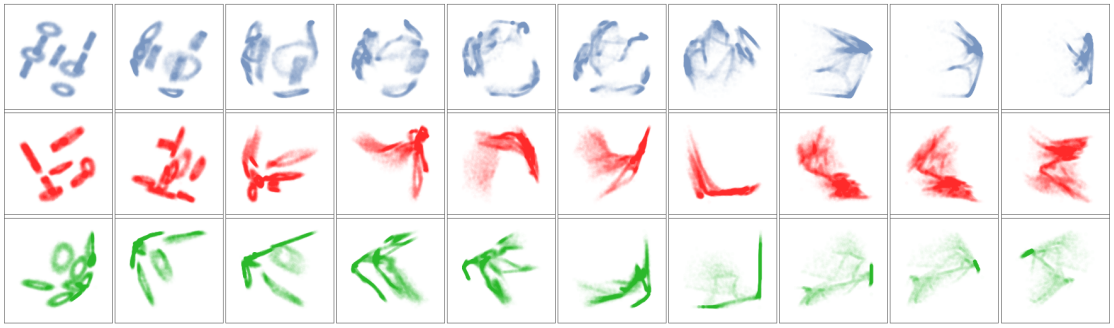


Figure 12: *Top:* Profiles of $\beta_0\big(\nu_j(M_a)\big)$ and $\beta_1\big(\nu_j(M_a)\big)$ for data set D-II, $j = 0, 1, \ldots, 10$. Network's architecture: three-dimensional input, two-dimensional output, and fifteen neurons in each of the nine intervening layers, with different activations. *Bottom*: Projections of $\nu_j(M_a)$, $j = 0, 1, \ldots, 10$, on the first two principal components.

Efforts depend on topological features: Some topological features evidently require more layers to simplify than others. The hardest one is the linked tori in the data set D-II. The profile of $\beta_1\big(\nu_l(M_a)\big)$ in the graph on the right of Figure 12 shows that some loops survive across many layers. This phenomenon is especially pronounced when the neural network is activated with tanh (blue): both the (blue) principal components projections and the (blue) profile show that the loops persist considerably longer than any other features in any of the three data sets.

Effects of width on topology change: For the data set D-I, we compare three sets of ten-layer networks: (i) narrow networks with six neurons in each layer; (ii) 'bottleneck' networks with 15, 15, 15, 3, 15, 15, 15, 15, 2 neurons respectively in layers one through nine — notice the three-neuron bottleneck layer; (iii) wide networks with fifty neurons in each layer. The left graph in Figure 14 suggests that a bottleneck layer forces large topo-

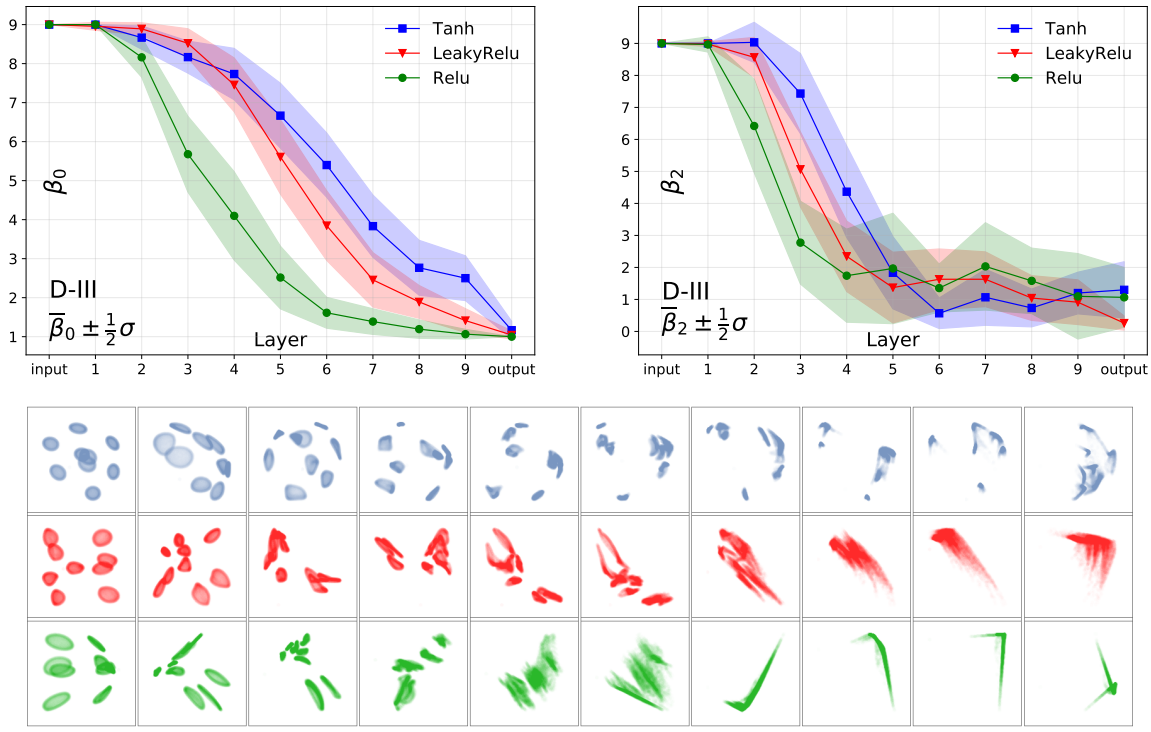


Figure 13: *Top:* Profiles of $\beta_0\big(\nu_j(M_a)\big)$ and $\beta_2\big(\nu_j(M_a)\big)$ for data set D-III, $j = 0, 1, \ldots, 10$. Network's architecture: three-dimensional input, two-dimensional output, and fifteen neurons in each of the nine intervening layers, with different activations. *Bottom*: Projections of $\nu_j(M_a)$, $j = 0, 1, \ldots, 10$, on first two principal components.

logical changes, and a narrow network changes topology faster than a wider one. The other two graphs compare a 15-neuron wide network with a 50-neuron wide one, both with ten layers, on data sets D-II and D-III respectively. However, for the same choice of activation, the difference between them is negligible. Also, reducing the width below fifteen neurons makes training to high accuracy increasingly difficult, i.e., the percentage of successfully trained networks starts to drop.

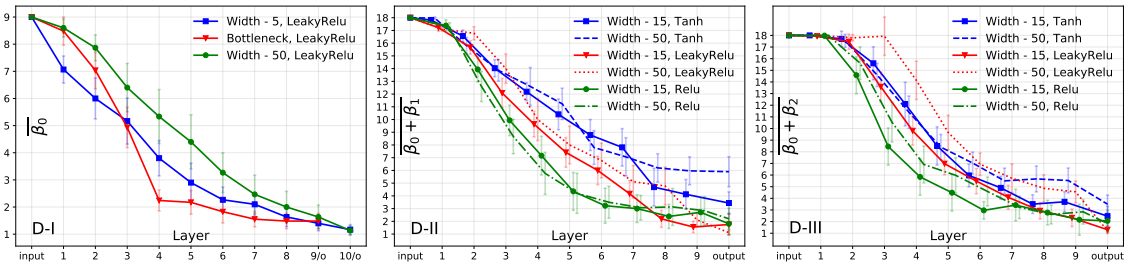


Figure 14: Mean values of topological complexity $\omega\big(\nu_j(M_a)\big)$, $j = 0, 1, \ldots, l$, for ten-layer networks of varying widths. Error bars indicate ± half standard deviation about the mean.

Effects of depth on topology change: Reducing the depth of a constant-width network beyond a certain threshold makes it increasingly difficult to train the network to high accuracy — the percentage of successfully trained networks drops noticeably. Moreover, as the depth is reduced, the burden of changing topology does not spread evenly across all layers but becomes concentrated in the final layers. The initial layers do not appear to play a big role in changing topology, reducing depth simply makes the final layers 'work harder' to produce larger reductions in Betti numbers. Figure 15 shows this effect.

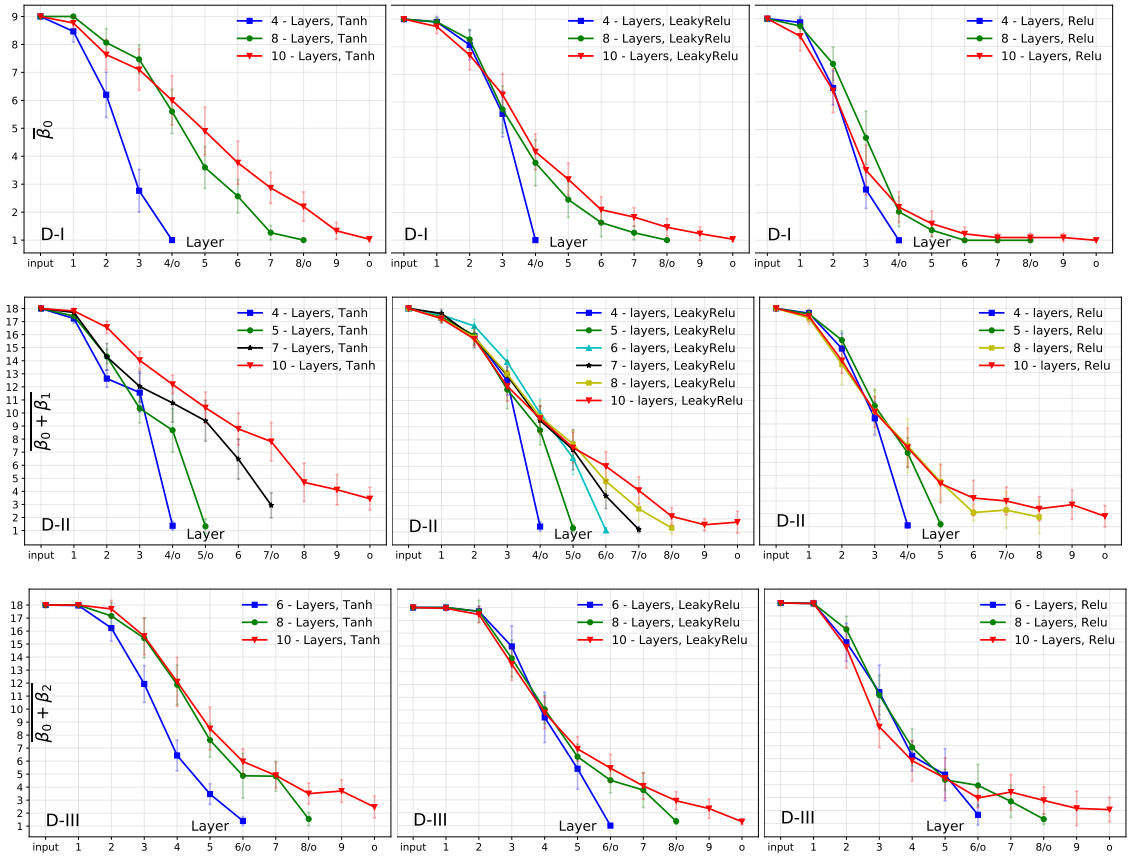


Figure 15: Mean values of topological complexity $\omega\big(\nu_j(M_a)\big)$, $j = 0, 1, \ldots, l$, for fifteen-neuron-wide networks of varying depths. Error bars indicate $\pm$ half standard deviation about the mean.

8. Consistency with Real-World Data

The results in Section 7 are deduced from experiments on the simulated data sets D-I, D-II, D-III generated in Section 6.1. It is natural to ask if these results remain valid on real data. In this section, we will see that they do, with some mild caveats. The key difference between real and simulated data is the amount of computational effort required to carry out our experiments — they are much more expensive for real data sets.

We will validate our results on four real-world data sets from (i) MNIST Handwritten Digits (LeCun et al., 1998), (ii) HTRU2 High Time-Resolution Universe Survey (Lyon et al., 2016), (iii) UCI Banknotes Authentication (Lohweg et al., 2013), (iv) UCI Sensorless Drive Diagnostic (Bayer et al., 2013). These data sets are chosen on the basis that they are real-valued and may be trained to high accuracy. The goal, as usual, is to observe how their topology changes as they pass through the layers of well-trained neural networks. To this end, our methodology in Section 6 applies to these data sets with some modifications:

- For real data, it is no longer possible to obtain the kind of near-perfectly trained neural networks in Section 6.2 that we could readily obtain with simulated data. As such, we adjust our expectations accordingly. By a well-trained neural network on a real data set, we mean one whose test accuracy ranges between 95 to 98% (recall that for simulated data, we require 99.99% or better).

- Unlike the simulated data sets in Section 6.1, we do not already know the topology of our real data sets and this has to be determined with persistent homology. More importantly, for real data, we cannot set a single scale for observing topological changes across different layers, as described in Section 6.3 — we have to compute persistent homology in every layer to track topological changes.

The complexity of real-world data and the need to calculate persistent homology at every layer limits the number of experiments that we could run. As it is prohibitively expensive to carry out extensive exploratory tests across a range of different architectures like what we did on simulated data (see Table 1), we will keep both width (ten neurons) and depth (ten layers) fixed in this section. In any case, our experiments on real data sets are not intended to be exploratory but to corroborate the findings in Section 7 that we deduced from simulated data. We seek confirmation on two findings in particular: the reduction in topological complexity through the layers and the relative effectiveness of ReLU over tanh activations in achieving this. Note that while we will only present the persistence barcodes at the output of the first, middle, and final layer, we computed persistence homology in every layer; interested readers may easily get the barcodes of other layers from our program.

MNIST Handwritten Digits (LeCun et al., 1998): Each of the 70,000 images in the MNIST handwritten digits data set is a 28×28-pixel image and collectively they form a point cloud on some manifold $M \subseteq \mathbb{R}^{784}$. Computing persistent homology for a 784-dimensional point cloud is way beyond what our computing resource could handle and we first reduce dimension by projecting onto its leading 50 principal components. Nevertheless, the dimension-reduced images remain to be of reasonably high-quality; we show a comparison of a few original digits alongside their principal component projections onto $\mathbb{R}^{50}$ in Figure 16. We will take the dimension-reduced point cloud X as the starting point for our experiments and will loosely refer to $X \subseteq M \subseteq \mathbb{R}^{50}$ as the MNIST data set.

Since we would like to have a binary classification problem for consistency with all our other experiments, instead of regarding the MNIST data as a classification problem with ten classes, we reduce it to a problem of classifying a chosen (any) digit a versus all non-a digits. So our manifold is $M = M_a \cup M_b$, where M_a is the 'manifold all handwritten a' and M_b is the union of the 'manifolds of all other non-a handwritten digits.' Following our methodology in Section 6, we train a neural network with ten layers and ten neuron in each

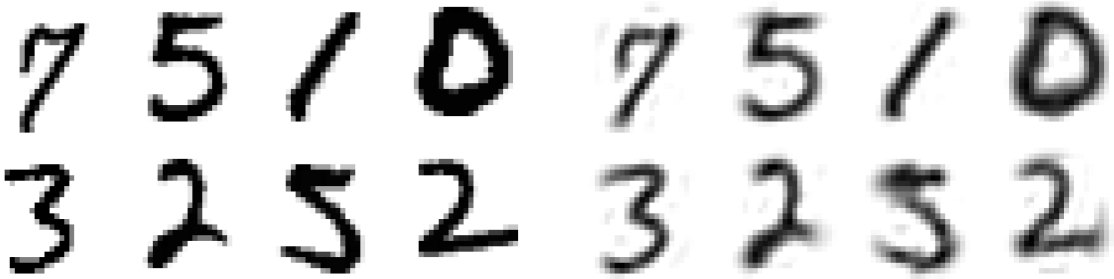


Figure 16: *Left*: Original MNIST handwritten digits. *Right*: MNIST handwritten digits projected onto the first fifty principal components.

layer on a labelled point cloud $X_a = X \cap M_a$ and $X_b = X \cap M_b$ to classify points in M_a and M_b; then with persistent homology we analyze how M_a is transformed after each layer in the network. While we fix the depth and widths, we vary the activation among tanh, leaky ReLU, and ReLU. We used 60,000 samples to train and the remaining 10,000 samples to test our neural networks; we use one third of the test set for our persistent homology calculations.

activation	scale ε	$j=0$	1	2	3	4	5	6	7	8	9	10
tanh	1.5	525	408	356	266	233	145	156	88	30	20	9
	2.5	6	5	2	1	3	14	12	8	1	4	4
	3.5	1	1	1	1	1	1	1	1	1	1	3
leaky ReLU	1.5	525	340	182	108	38	16	10	8	1	1	1
	2.5	6	6	6	5	1	1	2	1	1	1	1
	3.5	1	1	1	1	1	1	1	1	1	1	1
ReLU	1.5	525	199	106	27	13	6	1	1	1	1	1
	2.5	6	2	6	6	2	1	1	1	1	1	1
	3.5	1	1	1	1	1	1	1	1	1	1	1

Table 3: Topological complexity $\omega(M_a) = \beta_0\big(\nu_j(M_a)\big) + \beta_1\big(\nu_j(M_a)\big) + \beta_2\big(\nu_j(M_a)\big)$ at layers $j = 0, 1, \ldots, 10$ with M_a the 'manifold of handwritten a'. Network has 50-dimensional input, 2-dimensional output, and is 10-dimensional in the intermediate layers. For each of the three activation types, we show the homology at three scales $\varepsilon = 1.5$, 2.5, 3.5.

The results of our experiments for the digit $a = 0$ are shown in Table 3. What we see in the table corroborates our earlier findings on simulated data sets — an unmistakable reduction in topological complexity through the layers, with ReLU activation reducing topological complexity most rapidly when compared to the other two activations. With tanh activation, reduction in topological complexity is not only much slower but the network fails to reduce M_a to a topological disk, despite having ten layers. The persistence barcodes for the MNIST data set are much larger than those for the next three data sets but they

are not much more informative than our summary statistics in Table 3. As such we do not show the barcodes here although they can just as readily be generated from our program.

HTRU2 High Time-Resolution Universe Survey (Lyon et al., 2016): This data set consists of statistics of radio source signals from 17,898 stars, measured during the High Time-Resolution Universe Survey (HTRU2) experiment to identify pulsars. For our purpose, it suffices to know that pulsars are stars that produce radio emission measurable on earth. In the HTRU2 data set, each recorded radio emission is described by eight continuous variables: four are statistics of the radio signal called 'integrated profile' and the other four are statistics of the 'DM-SNR curve' that tracks frequency components of the signal versus its arrival time. The radio sources are labeled by a or b according to whether the source is a pulsar or not. We show a small portion of this data set in Table 4.

Star #	1	2	3	4	5
Mean (integral profile)	140.5625	102.5078	103.0156	136.7500	99.3672
Standard Deviation (integral profile)	55.6838	45.5499	39.3416	57.1784	41.5722
Excess Kurtosis (integral profile)	−0.2346	0.2829	0.3233	−0.0684	0.4653
Skewness (integral profile)	−0.6996	0.4199	1.0511	−0.6362	4.1541
Mean (DM-SNR)	3.1998	1.3587	3.1212	3.6423	1.6773
Standard Deviation (DM-SNR)	19.1104	13.0790	21.7447	20.9593	61.7190
Excess Kurtosis (DM-SNR)	7.9755	13.3121	7.7358	6.8965	2.2088
Skewness (DM-SNR)	74.2422	212.5970	63.1720	53.5937	127.3930
Pulsar 'a' or not 'b'	b	a	b	b	b

Table 4: Five entries from HTRU2. The first eight rows are statistics of the radio signal from that star. The last row indicates whether the respective star is a pulsar 'a' or not 'b'.

We take a 3,278 subsample of the HTRU2 data set so that we have an equal number of pulsars and non-pulsars. This is a point cloud $X \subseteq M \subseteq \mathbb{R}^8$ with $M = M_a \cup M_b$ a union of the 'pulsar manifold' M_a and 'non-pulsar manifold' M_b; the point clouds $X_a = X \cap M_a$ and $X_b = X \cap M_b$ each have 1,639 points. We use 80% of this balanced data X for training the neural networks and the remaining 20% for testing. For persistent homology computations, we use the test set but we first passed it through a local outlier removal algorithm of Breunig et al. (2000) for denoising. Again, our neural networks have ten layers with ten neurons in each layer and are activated with either ReLU or tanh.

In Figure 17, we show the persistence barcodes for $\nu_k(X_a)$ in the first, middle, and last layer, i.e., $k = 1, 5, 10$. The scatter plots below the barcodes show, for $k = 1, 5, 10$, the projections of $\nu_k(X_a)$ (red) and $\nu_k(X_b)$ (blue) onto the three leading principal components. These persistent barcodes tell the same story for the HTRU2 data as Table 3 does for the MNIST data and Figures 11, 12, 13 do for the simulated data D-I, D-II, D-III: Topology is simplified as the data passes through the layers; and ReLU does a better job than tanh activation at reducing topological complexity.

UCI Banknotes Authentication (Lohweg et al., 2013): This data set is derived from 400×400-pixel gray scale images of 1,372 genuine and forged banknotes; small patches ranging in sizes from 96×96 to 128×128-pixels are extracted from the images and wavelet-transformed. Figure 18 shows three of these small extracted patches.

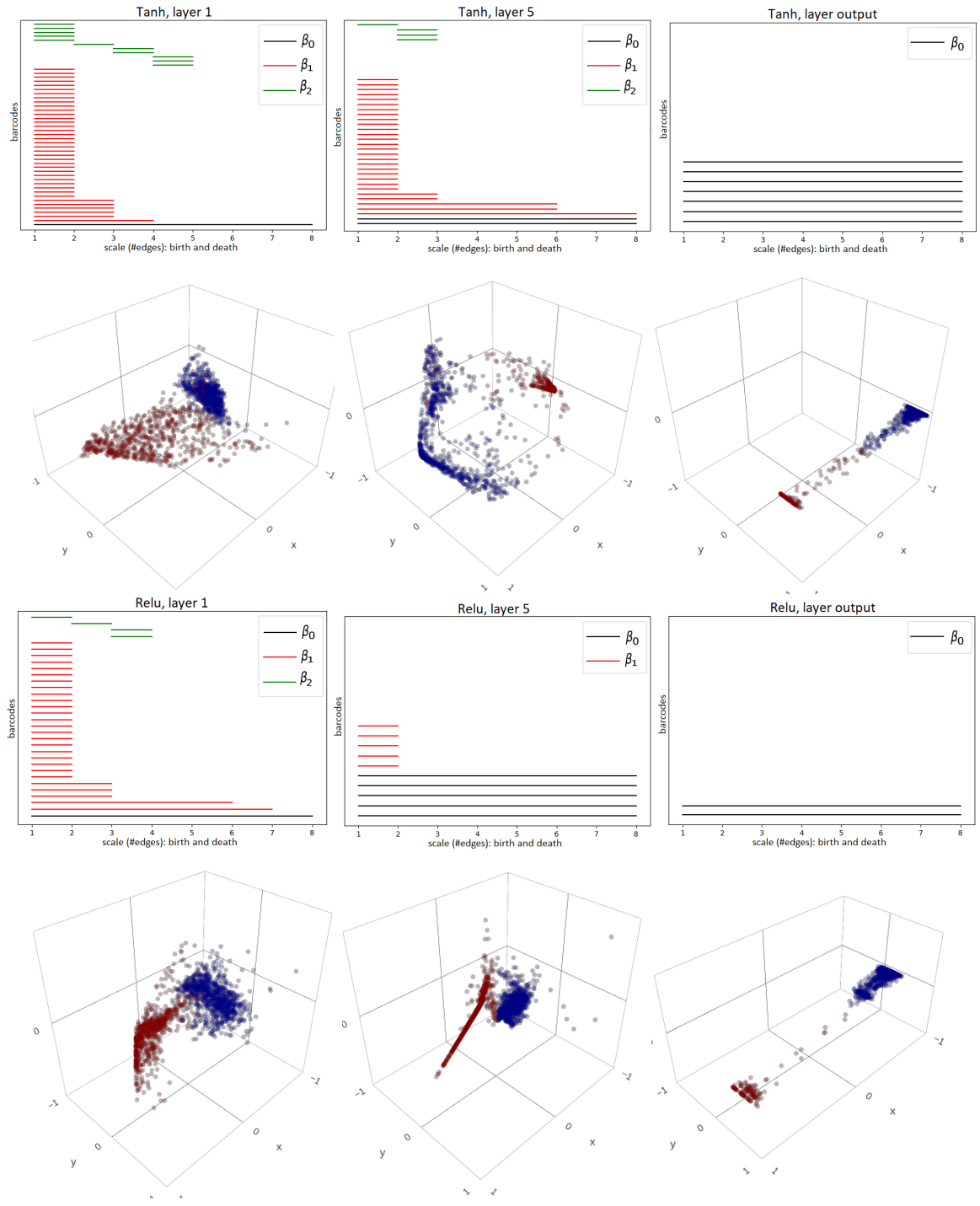


Figure 17: Persistence barcodes for neural networks trained on HTRU2 show topology changes in the 'pulsar manifold' M_a as it passes through the layers, activated with tanh (*top*) and ReLU (*bottom*). Scatter plots show principal component projections of M_a (red) and the 'non-pulsar manifold' M_b (blue) at the corresponding layers.

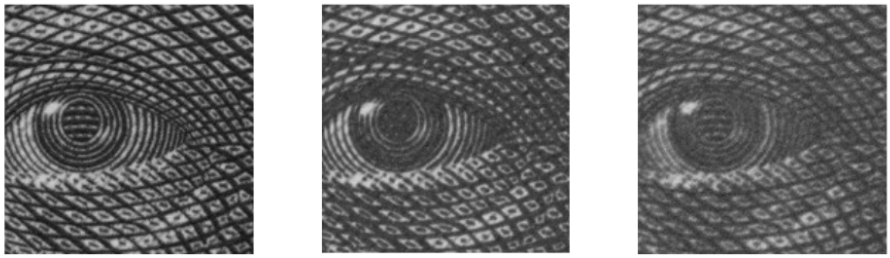


Figure 18: Texture sample for genuine banknote (*left*), high-quality forgery (*middle*) and low-quality forgery (*right*). The figures are taken from Lohweg et al. (2013).

The UCI Banknotes data set does not contain any images but is simply a list of four statistics computed from the wavelet coefficients of the image patches, together with a label indicating whether the patch is from a genuine 'a' or forged 'b' banknote. We show five of these entries in Table 5; the full data set contains 1,372 entries like these.

Banknote #	1	2	3	4	5	6
Variance (wavelet coef.)	−1.3971	4.5459	3.8660	3.4566	0.3292	0.3901
Skewness (wavelet coef.)	3.3191	8.1674	−2.6383	9.5228	−4.4552	−0.1428
Kurtosis (wavelet coef.)	−1.3927	−2.4586	1.9242	−4.0112	4.5718	−0.0319
Entropy (wavelet coef.)	−1.9948	−1.4621	0.1065	−3.5944	−0.9888	0.3508
Genuine 'a' or forged 'b'	a	b	b	b	b	a

Table 5: Five entries from the UCI Banknotes data set. The first four rows are statistics of the wavelet coefficients of the banknotes image patches. The last row indicates whether the respective banknote is genuine 'a' or forged 'b'.

As with the HTRU2 data set, we subsample 1,200 entries from the UCI Banknotes data set so that we have an equal number of genuine and forged samples; we use 80% of this data set for training and 20% for testing; and for persistent homology computations, we preprocess the data with the outliers removal algorithm in Breunig et al. (2000). For our purpose, the UCI Banknotes data set is a point cloud $X \subseteq M \subseteq \mathbb{R}^4$ with $M = M_a \cup M_b$ a union of the 'manifold of genuine banknotes' M_a and the 'manifold of forged banknotes' M_b; the point clouds $X_a = X \cap M_a$ and $X_b = X \cap M_b$ each has 600 points.

When X_a and X_b are passed through well-trained neural networks (ten layers, ten neurons in each layer, ReLU or tanh-activated), we obtained results consistent with all earlier experiments. The persistence barcodes in Figure 19 show that Betti numbers β_1 and β_2 are reduced to zero for both activations, β_0 successfully reduces to one when ReLU-activated but is stuck at two when tanh-activated. Also, the reduction of Betti numbers happens more rapidly with ReLU-activation. These observations are also reflected in the respective principal components scatter plot below each persistence barcode.

UCI Sensorless Drive Diagnostic (Bayer et al., 2013): This data set concerns a printed circuit board that operates a specific type of drive motor. The goal is to classify twelve types of common defects in the drive motor based on 49 measurements of electric

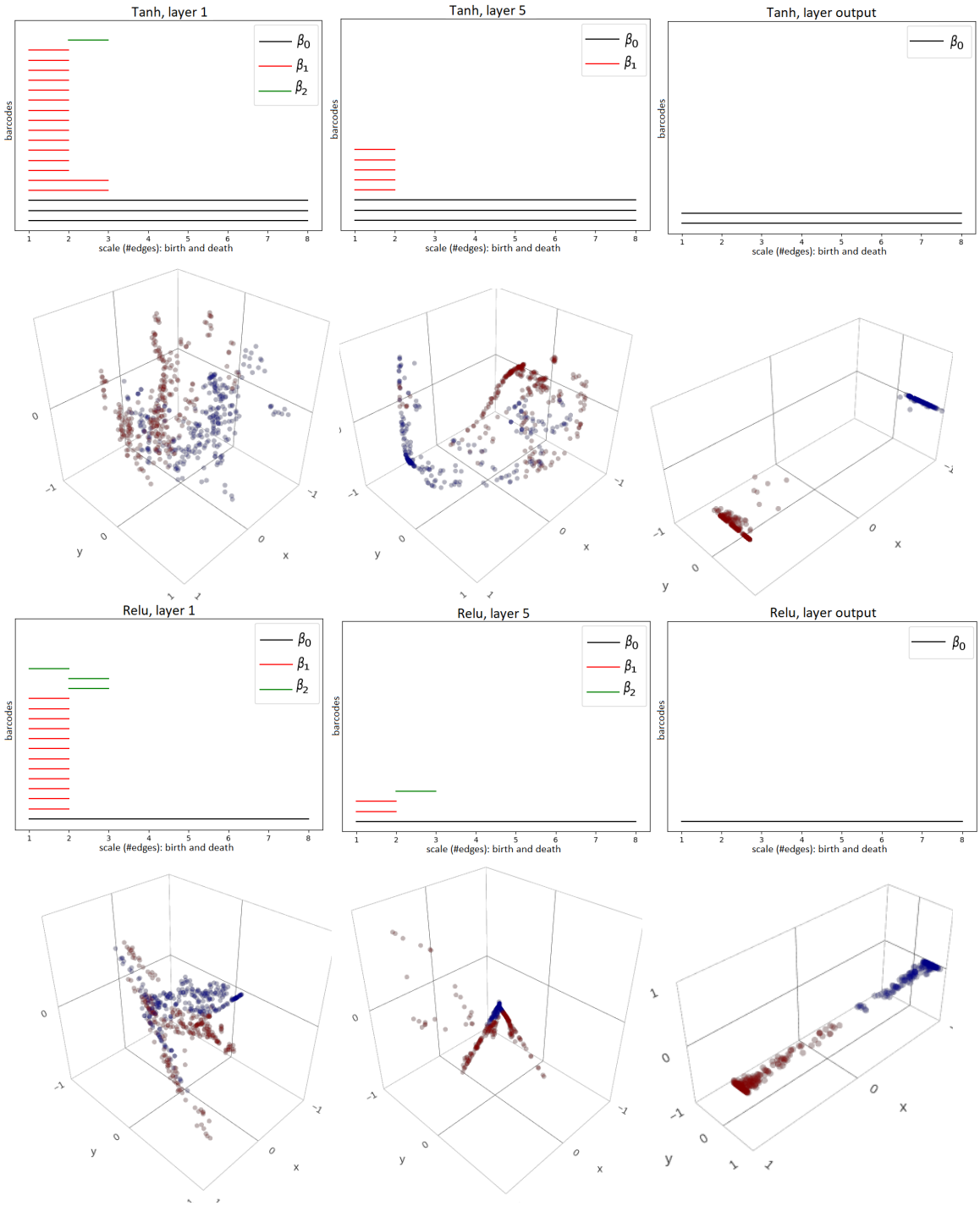


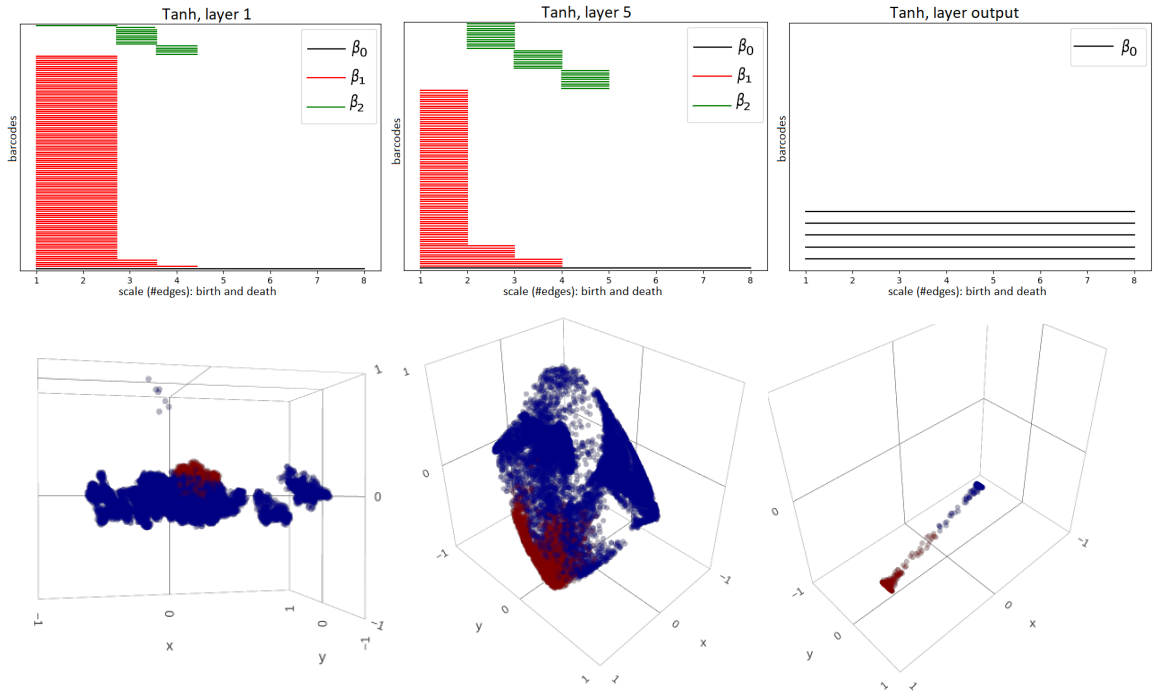
Figure 19: Persistence barcodes for neural networks trained on the UCI Banknotes data show topology changes in the 'manifold of genuine banknotes' M_a as it passes through layers activated with tanh (*top*) and ReLU (*bottom*). Scatter plots show principal component projections of M_a (red) and the 'manifold of forged banknotes' M_b (blue) at the corresponding layers.

currents at various locations on the printed circuit board. Table 6 shows five entries in this data set, which has a total of 58,509 such entries.

sample #	1	2	3	4	5
Elect. curr. 1	-3.015×10^{-7}	-2.952×10^{-6}	-2.952×10^{-6}	-4.961×10^{-6}	-6.501×10^{-6}
Elect. curr. 2	8.260×10^{-6}	-5.248×10^{-6}	-3.184×10^{-6}	-2.088×10^{-6}	-6.208×10^{-6}
Elect. curr. 3	-1.152×10^{-5}	3.342×10^{-6}	-1.592×10^{-5}	-1.366×10^{-5}	4.644×10^{-6}
Elect. curr. 4	-2.310×10^{-6}	-6.056×10^{-6}	-1.208×10^{-6}	4.661×10^{-7}	-2.749×10^{-6}
⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮
Elect. curr. 49	-1.500×10^{0}	-1.501×10^{0}	-1.496×10^{0}	-1.497×10^{0}	-1.500×10^{0}
Failure types	a	a	a	b	b

Table 6: A few entries from the UCI Drive data set. Last row indicates whether the failure is of type 'a' or one of the other eleven types, all of which are indicated as 'b'.

As in the case of the MNIST data set, instead of regarding the UCI Drive data as a classification problem with twelve classes, we reduce it to a binary classification problem of classifying a type a defect versus all other eleven types of defects. So our manifold is $M = M_a \cup M_b$ where M_a is the 'manifold of type a defects' and M_b is the union of the 'manifolds of all other types of defects.' Of the 58,509 entries in the UCI Drive data, we choose a random subset of 10,600 as our point cloud $X \subseteq M \subseteq \mathbb{R}^{49}$, divided equally into 5,300 points in $X_a = X \cap M_a$ and $X_b = X \cap M_b$ each. The rest of the experiment is as in the previous two cases (HTRU2 and UCI Banknotes data). The results are shown in Figure 20 and they are fully consistent with the results in Figures 17 and 19, supporting the same conclusions we drew from all previously examined data sets.



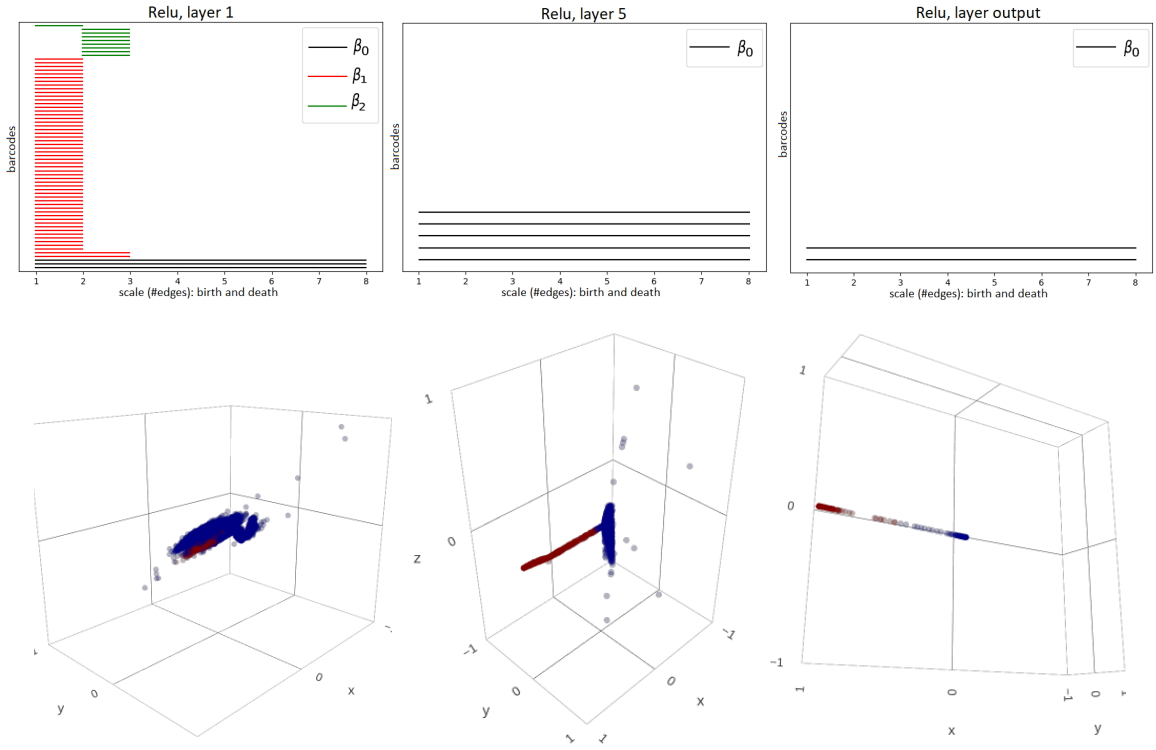


Figure 20: Persistence barcodes for neural networks trained on UCI Drive data show topology changes in the 'manifold of type a defects' M_a as it passes through layers activated with tanh (*previous page*) and ReLU (*above*). Scatter plots show principal component projections of M_a (red) and the 'manifold of non-type a defects' M_b (blue) at various layers.

9. Concluding Discussions

Our findings support the view that deep neural networks operate by transforming topology, gradually simplifying topologically entangled data in the input space until it becomes linearly separable in the output space. We proffered some insights on the roles of the deep layers and of rectified activations, namely, that they are mechanisms that aid topological changes. As this is an empirical study intended to provide evidence, we did not investigate the actual mechanics of how a ReLU-activated neural network carries out topological changes. We conclude our article with a few speculative words about the 'topology changing mechanism' of neural networks, mainly to serve as pointers for future work.

Consider the concentric red and blue circles on the left of Figure 21, two one-dimensional manifolds embedded in $\mathbb{R}^2$. By the Jordan Curve Theorem, there is no homeomorphism $\mathbb{R}^2 \to \mathbb{R}^2$ that will transform the two circles into two sets separable by a hyperplane in $\mathbb{R}^2$. Nevertheless it is easy to achieve this with a *many-to-one map* like $(x, y) \mapsto (|x|, |y|)$ that allows one to 'fold' a set, as shown on the left of Figure 21. An alternative way to achieve this is with an *embedding into higher dimensional space* like on the right of Figure 21 where a sequence of maps $\mathbb{R}^2 \to \mathbb{R}^3 \to \mathbb{R}^3 \to \mathbb{R}^3 \to \mathbb{R}^3 \to \mathbb{R}^2$ disentangles the red and blue circles in $\mathbb{R}^3$. We speculate that in a neural network, (i) the ReLU activation is a many-to-one map that can 'fold' a space; (ii) the excess width, i.e., width in excess of input dimension, of the

intermediate layers provides a higher dimensional *space* in which to transform the data; (iii) the depth plays the role of *time*, every additional layer affords additional time to transform the data. To elaborate on (iii), note that since we are limited to affine transformations and ReLU activation, a substantial change to the topology of a space may require a longer sequence of these operations, and by 'time', we simply mean the length of this sequence.

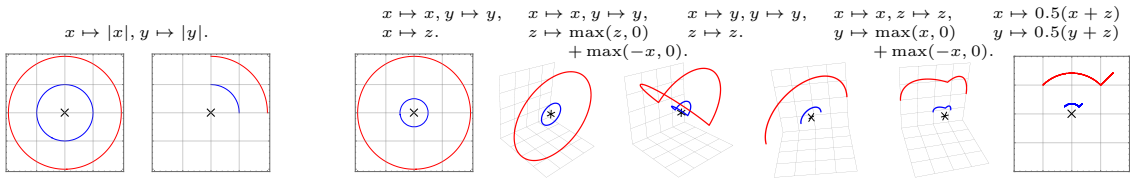


Figure 21: *Left*: Topology change with many-to-one maps: two neurons activated with the absolute value function can disentangle two concentric circles in a single step, transforming them into linearly separable sets. *Right*: Topology change by embedding into higher dimensions and performing the disentangling operations therein.

Adding to the first point (i), by singular value decomposition, an affine map takes the form $x \mapsto U\Sigma V x + b$, it provides the capability to translate (by the vector b), rotate/reflect (by the orthogonal matrices U and V), and stretch/shrink (by the nonnegative diagonal matrix Σ); ReLU-activation adds folding to the arsenal — an important capability. For example, to transform the surface of a donut (torus) into the surface of a croissant (sphere) as in Figure 22, the first two operations may be achieved with appropriate affine maps but the last one requires that we fold the doubly-pinched torus into a croissant surface.

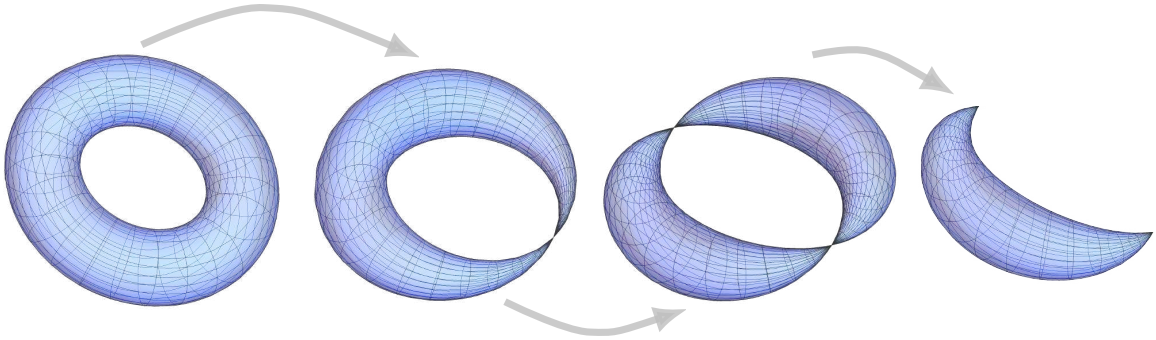


Figure 22: Donut to croissant: torus $\to$ pinched torus $\to$ doubly-pinched torus $\to$ sphere. Betti numbers: $(1, 2, 1) \to (1, 1, 1) \to (1, 1, 2) \to (1, 0, 1)$.

Acknowledgments

We gratefully acknowledge David Bindel for very helpful discussions on floating point arithmetic; Yali Amit, Peter McCullagh, and Brad Nelson for useful comments; and the two anonymous referees for their exceptionally thorough review. This work is supported by the National Science Foundation (NSF grant IIS 1546413), the Defense Advanced Research Projects Agency (DARPA D15AP00109 and a Director's Fellowship), and the University of Chicago (Chicago–Vienna Faculty Grant and Eckhardt Faculty Fund).

References

D. Attali, A. Lieutier, and D. Salinas. Vietoris–Rips complexes also provide topologically correct reconstructions of sampled shapes. *Comput. Geom.*, 46(4):448–465, 2013.

S. Basu and A. Rizzie. Multi-degree bounds on the Betti numbers of real varieties and semi-algebraic sets and applications. *Discrete Comput. Geom.*, 59(3):553–620, 2018.

C. Bayer, O. Enge-Rosenblatt, M. Bator, and U. Mönks. Sensorless drive diagnosis using automated feature extraction, significance ranking and reduction. In *IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–4, 2013.

M. Bianchini and F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Trans. Neur. Net. Lear.*, 25(8):1553–1565, 2014.

J.-D. Boissonnat, S. Pritam, and D. Pareek. Strong Collapse for Persistence. In *European Symposium on Algorithms (ESA)*, pages 67:1–67:13, 2018.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000.

G. Carlsson. The shape of data. In *Foundations of Computational Mathematics*, London Math. Soc. Lecture Note Ser., pages 16–44, 2013.

G. Carlsson. Topological pattern recognition for point cloud data. *Acta Numer.*, 23:289–368, 2014.

G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete Comput. Geom.*, 42(1):71–93, 2009.

G. Carlsson, T. Ishkhanov, V. De Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *Int. J. Comput. Vis.*, 76(1):1–12, 2008.

G. Carlsson, G. Singh, and A. Zomorodian. Computing multidimensional persistence. *J. Comput. Geom.*, 1(1):72–100, 2010.

M. Courbariaux, Y. Bengio, and J.-P. David. Training deep neural networks with low precision multiplications. *arXiv:1412.7024*, 2014.

V. De Silva and G. Carlsson. Topological estimation using witness complexes. In *Symposium on Point Based Graphics (SPBG)*, pages 157–166, 2004.

J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: a large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

P. Dlotko and H. Wagner. Simplification of complexes for persistent homology computations. *Homology Homotopy Appl.*, 16(1):49–63, 2014.

H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *Symposium on Foundations of Computer Science (FOCS)*, pages 454–463, 2000.

D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun. Understanding deep architectures using a recursive convolutional network. In *International Conference on Learning Representations (ICLR)*, 2014.

A. Gabrielov, N. Vorobjov, and T. Zell. Betti numbers of semialgebraic and sub-Pfaffian sets. *J. London Math. Soc.*, 69(1):27–43, 2004.

C. Giusti, E. Pastalkova, C. Curto, and V. Itskov. Clique topology reveals intrinsic geometric structure in neural correlations. *Proc. Natl. Acad. Sci.*, 112(44):13455–13460, 2015.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 315–323, 2011.

M. Gromov. Curvature, diameter and Betti numbers. *Comment. Math. Helv.*, 56(2):179–195, 1981.

S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning (ICML)*, pages 1737–1746, 2015.

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

G. Henselman and R. Ghrist. Matroid filtrations and computational persistent homology. *arXiv:1606.00199*, 2016.

I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 18(1):6869–6898, 2017.

F. A. Khasawneh, E. Munch, and J. A. Perea. Chatter classification in turning using machine learning and topological data analysis. *IFAC-PapersOnLine*, 51(14):195–200, 2018.

D. P. Kingma and J. Ba. ADAM: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

A. Krizhevsky, V. Nair, and G. Hinton. CIFAR-10, Canadian Institute for Advanced Research, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017.

J. Latschev. Vietoris-Rips complexes of metric spaces near a closed Riemannian manifold. *Arch. Math.*, 77(6):522–528, 2001.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

A. B. Lee, K. S. Pedersen, and D. Mumford. The nonlinear statistics of high-contrast patches in natural images. *Int. J. Comput. Vis.*, 54(1–3):83–103, 2003.

L. Li, W.-Y. Cheng, B. S. Glicksberg, O. Gottesman, R. Tamler, R. Chen, E. P. Bottinger, and J. T. Dudley. Identification of type 2 diabetes subgroups through topological analysis of patient similarity. *Sci. Transl. Med.*, 7(311):ra174-1–ra174-15, 2015.

L.-H. Lim. Hodge laplacians on graphs. *SIAM Rev.*, 63(3):685–715, 2020.

V. Lohweg, J. L. Hoffmann, H. Dörksen, R. Hildebrand, E. Gillich, J. Hofmann, and J. Schaede. Banknote authentication with mobile devices. In *Media Watermarking, Security, and Forensics (MWSF)*, 8665(07):1–14, 2013.

R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, and J. D. Knowles. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Mon. Notices Royal Astron. Soc.*, 459(1):1104–1123, 2016.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*, 2013.

J. Milnor. *Morse Theory*. Annals of Mathematics Studies. Princeton University Press, 1963.

J. Milnor. On the Betti numbers of real varieties. *Proc. Amer. Math. Soc.*, 15:275–280, 1964.

K. Mischaikow and V. Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete Comput. Geom.*, 50(2):330–353, 2013.

G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2924–2932, 2014.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

G. Naitzat, N. Lokare, J. Silva, and I. Kaynar-Kabul. M-Boost: Profiling and refining deep neural networks with topological data analysis. In *KDD Workshop on Interactive Data Exploration and Analytics*, 2018.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

J. L. Nielson, et al. Topological data analysis for discovery in preclinical spinal cord injury and traumatic brain injury. *Nat. Commun.*, 6(8581):1–12, 2015.

P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete Comput. Geom.*, 39(1–3):419–441, 2008.

C. Olah. Neural networks, manifolds, and topology. `http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/`, 2014.

M. L. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*. Society for Industrial and Applied Mathematics, 2001.

P. T. Pearson. Visualizing clusters in artificial neural networks using Morse theory. *Adv. Artificial Neural Systems*, 2013(486363):1–8, 2013.

J. A. Perea and J. Harer. Sliding windows and persistence: an application of topological methods to signal analysis. *Found. Comput. Math.*, 15(3):799–838, 2015.

J. A. Perea, A. Deckard, S. B. Haase, and J. Harer. Sw1pers: Sliding windows and 1-persistence scoring; discovering periodicity in gene expression time series data. *BMC Bioinformatics*, 16(257):1–12, 2015.

K. N. Ramamurthy, K. Varshney, and K. Mody. Topological data analysis of decision boundaries with application to model selection. In *International Conference on Machine Learning (ICML)*, pages 5351–5360, 2019.

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

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

G. Singh, F. Mémoli, and G. E. Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *Symposium on Point Based Graphics (SPBG)*, pages 91–100, 2007.

A. Storjohann. Near optimal algorithms for computing smith normal forms of integer matrices. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 267–274, 1996.

D. Yu, F. Seide, and G. Li. Conversational speech transcription using context-dependent deep neural networks. In *International Conference on Machine Learning (ICML)*, pages 1–2, 2012.

T. P. Zell. *Quantitative Study of Semi-Pfaffian Sets*. Ph.D. Thesis, Purdue University, 2003.

L. Zhang, G. Naitzat, and L.-H. Lim. Tropical geometry of deep neural networks. In *International Conference on Machine Learning (ICML)*, pages 5824–5832, 2018.

A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33(2):249–274, 2005.