TOPOLOGY OF DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study how the topology of a data set $M = M_a \cup M_b \subseteq \mathbb{R}^d$, representing two classes of objects 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 a generalization error of less than 1%. The goal is to shed light on two well-known 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, despite the fact that a shallow network is able to approximate any function arbitrary well. We performed extensive experiments on persistent homology of point cloud data sets, both simulated and real-world. 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 $f : \mathbb{R}^d \to \mathbb{R}^p$, the Betti numbers of both components M_a and M_b invariably reduce to their lowest possible values: $\beta_k\bigl(f(M_i)\bigr) = 0$ for $k \geq 1$ and $\beta_0\bigl(f(M_i)\bigr) = 1$, $i = a, b$. Furthermore, (2) reduction in Betti numbers is significantly faster for ReLU activation compared to hyperbolic tangent activation as the former defines nonhomeomorphic maps that change topology whereas the latter defines homeomorphic maps that preserve topology. Lastly, (3) shallow and deep networks transform the same data set differently — a shallow network operates mainly through changing geometry and changes topology only in its final layers, a deep one spreads topological changes evenly across all layers.

1 OVERVIEW

A key insight of topological data analysis is that "*data has shape*" Carlsson (2013; 2014). That data sets often have nontrivial topologies that may be exploit in their analysis is now a widely accepted principle with abundant examples across mutliple 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 & Harer (2015), etc. An early striking example came from computer vision Carlsson et al. (2008), where the authors showed that naturally occurring image patches reside on a low-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. Most existing approaches tend to focus on what a network does to a single object, e.g. an image of a cat; but we are interested in what it does to all objects in the same class, e.g. the set of all cats. As in topological data analysis, we employ persistent homology — a computational topology tool with proven stability, robust algorithms, and high-quality software — to track changes in the topology of a data set as it passes through the layers of neural network. Nevertheless we highlight a fundamental difference: topological data analysis is about *reading* the shape of data, deep neural network is about *writing* the shape of data.

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 activation can be explained by the different speeds they change topology — an 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 necessity of having sufficient number of layers.

Figure 1 illustrates what we mean by "changing topology." The two subfigures are caricatures of real results (see Figures 2, 6, 7 for the true versions) obtained via actual persistent homology computations and projections to principal components. In both subfigures, we begin with a three-dimensional

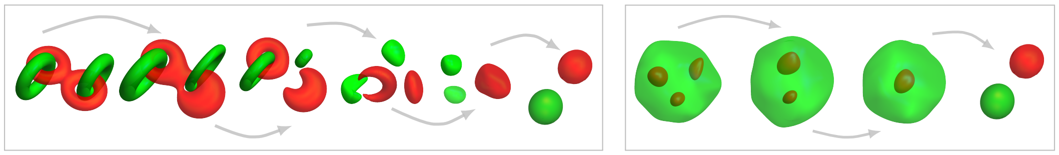


Figure 1: Progression of Betti numbers $\beta(X) = (\beta_0(X), \beta_1(X), \beta_2(X))$. *Left*: β(red): $(1,2,0) \to (1,2,0) \to (2,1,0) \to (2,0,0) \to (1,0,0) \to (1,0,0)$; β(green): $(2,2,0) \to (2,2,0) \to (2,1,0) \to (2,0,0) \to (2,0,0) \to (1,0,0)$. *Right*: β(red): $(3,0,0) \to (2,0,0) \to (1,0,0)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

manifold $M = M_a \cup M_b$ comprising two disjoint 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, the green solid ball with three voids inside, and M_b, three red balls each placed within one of the three voids of M_a. The topological simplification in both boxes are achieved via a reduction in the *Betti numbers* (see Section 4 for a definition) 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 main goal is to provide (what we hope are) 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.

2 Key findings

While we will provide some qualitative results in the supplement, this work is primarily an empirical study — we performed more than 10,000 experiments on data sets of varying topological complexities and have made our codes available for reader's further experimentations.[2] Before describing our methodologies and results, we summarize the most salient observations and discuss their implications:

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

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

(iii) The initial layers often effect only *geometric* changes, it is only in deeper layers that *topological* changes take place. Moreover, as we reduce network depth, the burden of producing topological simplification is not spread uniformly across layers but remains concentrated in the last layers. The last layers see much greater reduction in topological complexity than the initial layers.

Observation (ii) provides a plausible answer to the widely posed question Nair & 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. The merit of ReLU activation is often attributed to its avoidance of the vanishing gradient problem. Our experiments indicate that this does not give the full picture: Leaky ReLU, which also avoids vanishing gradient, has noticeably inferior results than ReLU. The topological perspective, on the other hand, perfectly explains why.

It is much harder to change topology with homeomorphisms; in fact, mathematically it is impossible; but maps like hyperbolic tangent achieves it in practice via rounding errors. Note that in IEEE

[1]One with perfect accuracy on training set and less than 1% generalization error.

[2]`https://github.com/topnn/topnn_framework` — fully anonymized for review purposes.

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

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

where $\mathrm{fl}(x)$ denotes floating point representation of x, 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 $\varphi_\delta : \mathbb{R}^n \to [-1,1]^n$ 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 φ_δ, 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 we 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.

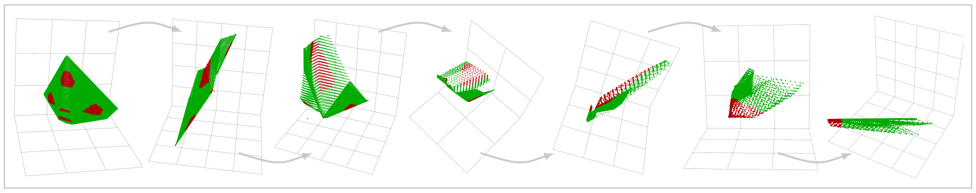


Figure 2: We see how the data set is transformed after passing through layers $2, 3, \ldots, 8$ of a ReLU network with three neurons in each layer, well-trained to detect five disks in a square. β(red): $(5,0) \to (4,0) \to (4,0) \to (4,0) \to (2,0) \to (1,0) \to (1,0)$.

Observation (iii) addresses another perennial paradox Krizhevsky et al. (2012); Eigen et al. (2014); Seide et al. (2011): 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 irrelevant 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) explains the role of additional layers — topological changes are minor in the first few layers and occur mainly in the later layers; moreover, a complicated data set requires many more layers to simplify.

We drew our inspiration partly from a Google Brain blog post Olah (2014) speculating how neural networks act as homeomorphisms that distort *geometry*, but it stopped short of taking the leap to topology-changing maps. The idea of changing 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 & 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. Note that *dimension* is a topological invariant, changing dimension is changing topology.

3 Problem formulation

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

[3]Assuming the BFloat16 floating-point format used in TensorFlow.

two different probability distributions supported on two disjoint manifolds $M_a,\ M_b \subseteq \mathbb{R}^d$. 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.

Our samples form a *point cloud data* set, a large but finite set of points on M_a and M_b. Our training set $T \subseteq M_a \cup M_b$ 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 sample T uniformly and densely, so that Betti numbers of M_a and M_b can be faithfully obtained from the persistent homology of T. We will use $T_a := T \cap M_a$ and $T_b := T \cap M_b$, or rather, their *Vietoris–Rips complex* (see Section 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

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

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

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

We assume that s is a *linear classifier* and thus the decision boundary of s is a hyperplane in $\mathbb{R}^p$.

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 $x \in M_a$. In all our experiments, we train ν until it correctly classifies all $x \in T$ — we will call such an 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 deliberate choose $M_a \cup M_b$ to have doubly complicated topologies in the following sense:

(a) 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 $\beta_k(M_i)$.

(b) In addition, M_a and M_b will be entangled in a topologically complicated manner. See Figures 4 and 5 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.

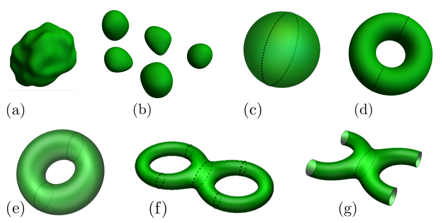
In our experiments, we analyze how the topologies of $\nu_j(M_a)$ and $\nu_j(M_b)$ evolve as j runs from 1 through l, for different manifolds M_a, M_b entangled in different ways, for different number of layers l and choices of widths $n_1, \dots, n_d$, and different activations σ. The results show a well-trained neural network $\nu : \mathbb{R}^d \to [0,1]$ will reduce 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 the initial decision boundary $D \subseteq \mathbb{R}^d$ of complicated topology ultimately gets transformed into a hyperplane in $\mathbb{R}^p$ by the final layer. Although we could measure and track the topologies of $\nu_j(M_a)$ and $\nu_j(M_b)$ directly, our approach only permits us to observe the topology of the decision boundary separating them indirectly. We have more to say about this in supplementary Section G.2.

4 Quantifying topological complexity

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 $M \subseteq \mathbb{R}^d$ and have intuitive interpretations. 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 $k > d$ as there is no $(d+1)$-dimensional holes in d-dimensional space. So for $M \subseteq \mathbb{R}^d$, we write $\beta(M) := (\beta_0(M), \beta_1(M), \dots, \beta_d(M))$ and define its *topological complexity* by

$$\omega(M) := \beta_0(M) + \beta_1(M) + \cdots + \beta_d(M). \tag{2}$$

If M has no holes and can be continuously (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 non contractible space is a circle $S^1 \subseteq \mathbb{R}^2$, which has one 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 gives a few more examples. For in-depth treatments, see Zomorodian (2009); Massey (1991); Hatcher (2002).



	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)	$(1,2,0)$
(e)	Surface of torus (hollow)	$(1,2,1)$
(f)	Genus two surface (hollow)	$(1,5,2)$
(g)	Torso surface (hollow)	$(1,5,0)$

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

The definition of topological complexity (2) as the sum of Betti numbers captures the idea that "the more holes a space has, the more complex its topology." It differs from the better known *Euler characteristic* in that it is an unsigned sum. Nevertheless it is a natural notion that has appeared in Milnor (1964); Basu & Rizzie (2018). In the context of neural networks, Bianchini & Scarselli (2014) studies topological complexity of decision boundaries of neural networks with nonlinearities that are Pfaffian functions Zell (2003); Gabrielov et al. (2004).

Given a point cloud data set $T \subseteq M_a \cup M_b$, i.e., a sample of finite points, we estimate the Betti numbers of the manifolds M_a, M_b by constructing simplicial complexes from $T_a = T \cap M_a$, $T_b = T \cap M_b$ and then computing the Betti numbers of these complexes (a matter of simple linear algebra). There are many ways to build these complexes Otter et al. (2017), but for dense enough point clouds and sufficiently small scales, their Betti numbers are all equal to those of the manifolds they approximate with high probability Niyogi et al. (2008). We choose the *Vietoris–Rips complex* for its simplicity: any collection of k points $x_1, \dots, x_k \in T_a$ forms a k-simplex iff $\|x_i - x_j\| \leq \varepsilon$ for all i, j. The appropriate scale $\varepsilon > 0$ is determined through *persistent barcodes*, which allow one to see the topologies at different scales all at once Otter et al. (2017); Oudot (2015).

By far the most expensive step is the persistent barcodes needed to determine scale ε. Ideally we want a single ε that works for all layers, but this is impossible — as we saw in Figure 2, a well-trained neural network stretches and shrinks different regions of the space differently; as a point cloud data set passes through the layers, its local densities vary enormously. Our side contribution is to provide a solution to this problem: Instead of the Euclidean norm, we use the geodesic distance to define our Vietoris–Rips complex, i.e., nearest neighboring points are distance one from each other. This metric effectively normalizes densities across layers while preserving connectivity of nearest neighbors, i.e., it only affects geometry but not topology, which is what we care about. With this approach, a single scale for construction of simplicial complexes across multiple layers can be determined by looking at a single barcode diagram of the input. The resulting reduction is computation time allows a thorough examination of topology changes in a neural network that we describe in the next section.

5 Experiments

We perform our experiments on three simulated data sets and one real data set. The simulated data set D-I, D-II, D-III are generated in a controlled manner as described in Section 3 to have complicated but manageable topologies that we *know in advance*. The main issue with using real data sets is that they have vastly more complicated topologies that are nearly impossible to determine in advance. For example, even something as basic as the Mumford data set, a mere collection of 3-by-3-pixels of high contrast patches of natural images, took many years to have its topology determined 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 intense debate. Figuring out, say, the topology of the manifold of cat images with the space of all possible images is well-beyond our capabilities for the foreseeable future. Nevertheless, we did limited experiments on one real data set, the MNIST handwritten digits LeCun & Cortes (2010) to show that the conclusions are consistent with those drawn from simulated data.

D-I is a two-dimensional manifold consisting of M_a, nine green disks, positioned in M_b, a larger disk with nine holes as in Figure 4. We clearly have $\beta(M_a) = (9, 0)$ and $\beta(M_b) = (1, 9)$ (one

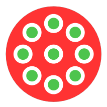
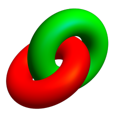


Figure 4: D-I resembles radial cross-section of lotus root.



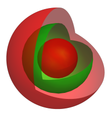


Figure 5: *Left*: D-II comprises nine pairs of interlocking rings. *Right*: D-III comprises nine units of this doubly concentric object.

connected component, nine holes). **D-II** is a three-dimensional manifold comprising nine disjoint pairs of red solid torus interlocked with a green solid torus (a single pair is shown in Figure 5). M_a (resp. M_b) is the union of all nine green (resp. red) tori. So $\beta(M_a) = \beta(M_b) = (9, 9, 0)$. **D-III** is a three-dimensional manifold comprising nine disjoint units of the following — a large red sphere enclosing a smaller green sphere enclosing a red ball; the green sphere is trapped between the red sphere and the red ball (see Figure 5 but note that the spheres are shown with portions omitted). M_a is the union of all nine green spheres and M_b is the union of the nine spheres and nine balls. So $\beta(M_a) = (9, 0, 9)$ and = $\beta(M_b) = (18, 0, 9)$. See Figure 10 in the supplement for more details.

To show our data sets visually, we have confined ourselves to two- and three-dimensional manifolds and thus only the first two or three Betti numbers (higher ones are all zero). Nevertheless, with enough computing power, the code that we provided in principle works for manifolds M of any dimension and, with sufficiently many data points sampled from M, also computes $\beta_k(M)$ for any $k = 0, 1, \dots, \dim M$. Also, space constraints mandates that we can only present select results (in Section 6). We will just show the change in $\beta_i\big(\nu_k(M_a)\big)$ in each of D-I, D-II, D-III for each layer k. We observe similar behavior for M_b but will not present them here. Instead we will examine the effects of having (i) different activations: hyperbolic tangent, leaky ReLU set to be $\max(x, 0.2x)$, and ReLU; (ii) different depths of four to ten layers; (iii) different widths of six to 50 neurons.

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[4] 30 well-trained neural networks. Description of the software and hardware used in our simulations may be found in Section H.

6 Results discussions

Computations are consistent across training instances. Figure 6 records our simplest data set D-I, where M_a comprises nine contractible components and so higher Betti numbers are irrelevant (all zero). Here we show every curve corresponding every neural network trained on D-I (recall that we do at least 30 runs for each experiment to account for the inherent randomness in TensorFlow) and they all show consistent profiles — a clear decay in β_0 across the layers even though the case of hyperbolic tangent activation (blue graph) show larger variance, a point we will elaborate in the next paragraph. The consistency in Figure 6 is representative of other experiments on higher Betti numbers and on other data sets. Henceforth, we will omit curves corresponding to individual runs and just show the curve of their means and the region of half standard deviation. The bottom diagrams show how topology changes occur from layer-to-layer using a two-dimensional projection onto the first two principal components (note that the intervening layers are in $\mathbb{R}^{15}$).

Nonhomeomorphic activations induce rapid topology changes. As the faint blue lines in Figure 6 reveal, hyperbolic tangent 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 more rapid reductions in all Betti numbers. The top half of Figure 7 shows selected results. The different rates at which topological changes occur are evident from the principal components projections in the bottom half.

Difference between topological features. Some topological features evidently require more layers to simplify than others. The hardest one is the interlocking tori in data set D-II. The profile of $\beta_1\big(\nu_l(M_a)\big)$ in the second graph in Figure 7 shows that the “loops” survive across many layers. This is especially so for networks with hyperbolic tangent activation (blue): both the principal components projections and the reduction profile show that the loops persist much longer compared with any other features in any of the three data sets.

[4]The repetition is necessary — given that TensorFlow involves a fair amount of randomization in initialization, batching, optimization, etc — to ensure that what we observe is not a fluke.

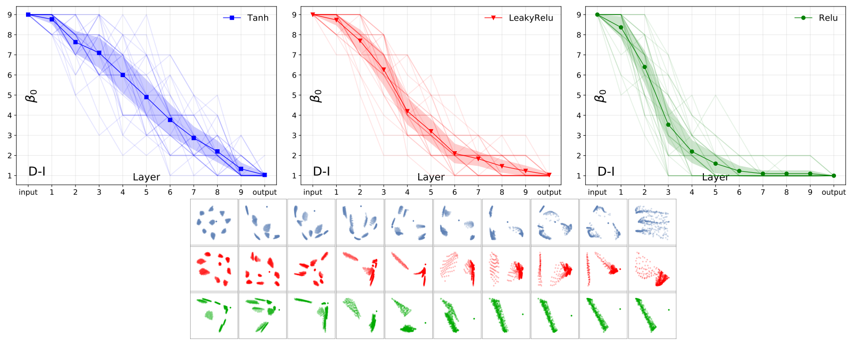


Figure 6: *Top*: Faint curves show individual profiles, dark curves show averaged profiles of $\beta_0(\nu_k(M_a))$, $k = 1, \dots, 10$, in data set D-I. Shaded regions show $\pm$ half standard deviations. Networks have different activations — blue for $\tanh$, red for $\mathrm{leakyrelu}$, green for relu; but same architecture — 10 layers, two neurons in first and last, 15 in intervening layers. *Bottom*: Projections of $\nu_k(M_a)$, $k = 1, \dots, 10$, on first two principal components, color-coded according to activations.

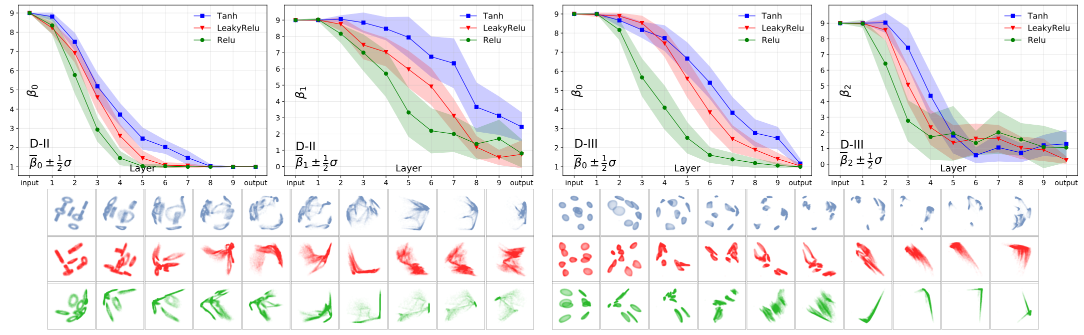


Figure 7: *Top*: Profiles of $\beta_0(\nu_k(M_a))$ and $\beta_1(\nu_k(M_a))$ for data set D-II (*left*), $\beta_0(\nu_k(M_a))$ and $\beta_2(\nu_k(M_a))$ for data set D-III (*right*), $k = 1, \dots, 10$. Networks share the same architecture: three-dimensional input, two-dimensional output, and 15 neurons in intervening layers 1 to 9, with different activations. *Bottom*: Projections of $\nu_k(M_a)$, $k = 1, \dots, 10$, on first two principal components.

Effects of depth on topology change. Reducing the depth of a constant-width network beyond a certain threshold makes it difficult to train it to high accuracy — the percentage of successfully trained networks drops noticeably. Moreover, as the depth is reduced, the burden of topology changing does not spread evenly across all layers but remains concentrated in the final layers — the initial layers do not appear to play a big role in changing topology, eliminating layers simply makes the final layers "work harder" to produce larger reductions in Betti numbers. Figure 8 shows this effect.

Effects of width on topology change. For the data set D-I, we compare three 10-layer networks: (i) a narrow network with six neurons in each layer; (ii) a "bottleneck" network with 15, 15, 15, 15, 3, 15, 15, 15, 15 neurons respectively in layers 1 through 9 — notice 3-neuron bottleneck layer; (iii) a wide network with 50 neurons in each layer. The left graph in Figure 9 suggests that a bottleneck layer forces large topological 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 10 layers, on data sets D-II and D-III respectively. The difference between them is negligible, for the same choice of activation. However, reducing the width to under 15 neurons makes training to high accuracy much more difficult, i.e., the percentage of successfully trained networks drops noticeably.

Consistency with real world data. We train a feedforward neural network to classify a handwritten digit x from the MNIST data set: $x \in M_a$ if it is zero and $x \in M_b$ otherwise. The digit zero is arbitrary and may be replaced by any other digits without affecting the conclusions. We analyze how M_a, the "manifold of handwritten zeros" is transformed by the network. We first project the data set onto its leading 50 principal components, reducing input dimension from 784 to 50 — this has negligible effect on the training and generalization errors but reduces computational costs significantly. We used a network with 10 layers and a width of 10 neurons in each hidden layer but we vary the choice of activation. The results for relu activation are in Table 1 and those for other activations

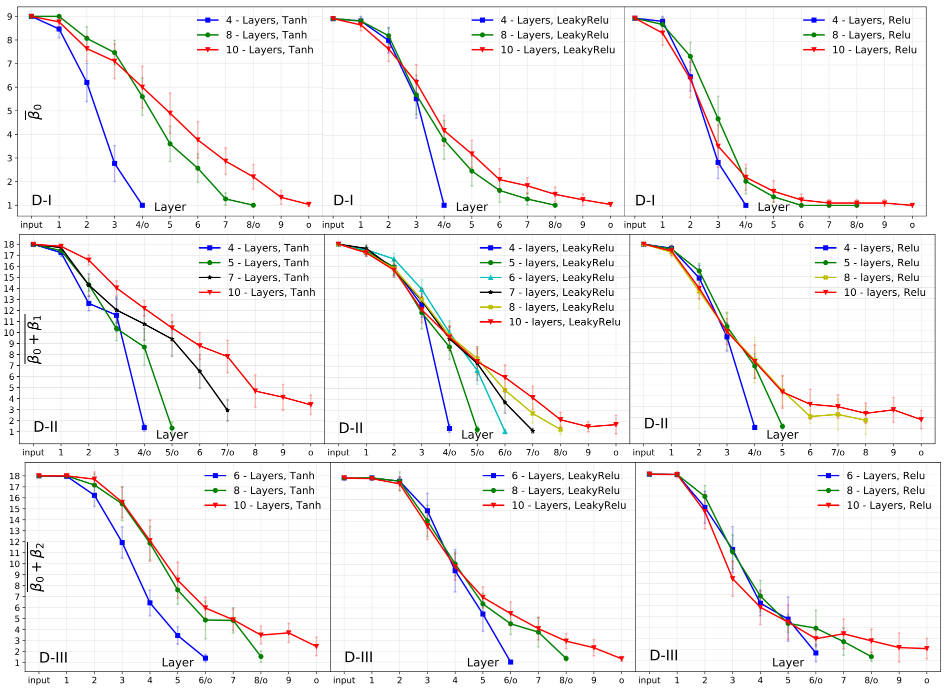


Figure 8: Mean values of topological complexity $\omega\big(\nu_k(M_a)\big)$, $k = 1, \ldots, l$, for 15-neuron wide networks of varying depths. Error bars indicate $\pm$ half standard deviation about the mean.

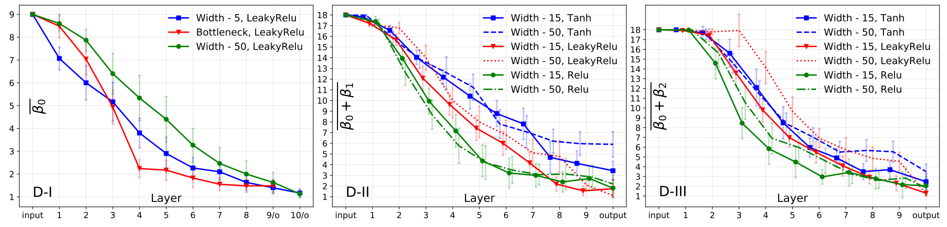


Figure 9: Mean values of topological complexity $\omega\big(\nu_k(M_a)\big)$, $k = 1, \ldots, l$, for 10-layer deep networks of varying widths. Error bars indicate $\pm$ half standard deviation about the mean.

in appendix Table 4. Our findings corroborate those in simulated data: topological complexity diminishes across layers, with relu showing the most rapid reduction and tanh the slowest.

	$k = 0$	1	2	3	4	5	6	7	8	9	10
$\varepsilon = 1.5$	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 1: The manifold of handwritten zeros M_a passing through a ReLU network with 50-dimensional input, two-dimensional output, and 10 neurons in intervening layers. Table shows values of $\beta_0\big(\nu_k(M_a)\big) + \beta_1\big(\nu_k(M_a)\big) + \beta_2\big(\nu_k(M_a)\big)$ at input ($k = 0$) and layers $k = 1, \ldots, 10$. Unlike with simulated data, we cannot fix ε in advance and so need to observe multiple ε's.

7 Conclusion

Our findings support the view that deep neural networks operate by transforming topology, gradually simplifying topologically complicated data shapes and arrangements in the input until it becomes linearly separable in the output. We proffered some new insights on the role of deep layers and of rectified activations — they are mechanisms that aid topological changes — suggesting that it may be fruitful to design neural networks with an eye towards effecting topological changes more efficiently.

REFERENCES

Saugata Basu and Anthony 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. ISSN 0179-5376. doi: 10.1007/s00454-017-9944-1. URL https://doi.org/10.1007/s00454-017-9944-1.

Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565, 2014.

Gunnar Carlsson. The shape of data. In *Foundations of computational mathematics, Budapest 2011*, volume 403 of *London Math. Soc. Lecture Note Ser.*, pp. 16–44. Cambridge Univ. Press, Cambridge, 2013.

Gunnar Carlsson. Topological pattern recognition for point cloud data. *Acta Numer.*, 23:289–368, 2014. ISSN 0962-4929. doi: 10.1017/S0962492914000051. URL https://doi.org/10.1017/S0962492914000051.

Gunnar Carlsson, Tigran Ishkhanov, Vin de Silva, and Afra Zomorodian. On the local behavior of spaces of natural images. *Int. J. Comput. Vis.*, 76(1):1–12, 2008. ISSN 0920-5691. doi: 10.1007/s11263-007-0056-x. URL https://doi.org/10.1007/s11263-007-0056-x.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*, 2014.

David Eigen, Jason Rolfe, Robert Fergus, and Yann Lecun. Understanding deep architectures using a recursive convolutional network. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

Andrei Gabrielov, Nicolai Vorobjov, and Thierry Zell. Betti numbers of semialgebraic and sub-Pfaffian sets. *J. London Math. Soc. (2)*, 69(1):27–43, 2004. ISSN 0024-6107. doi: 10.1112/S0024610703004939. URL https://doi.org/10.1112/S0024610703004939.

Chad Giusti, Eva Pastalkova, Carina Curto, and Vladimir Itskov. Clique topology reveals intrinsic geometric structure in neural correlations. *Proc. Natl. Acad. Sci. USA*, 112(44):13455–13460, 2015. ISSN 1091-6490. doi: 10.1073/pnas.1506407112. URL https://doi.org/10.1073/pnas.1506407112.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pp. 315–323, 2011. URL http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf.

Victor Guillemin and Alan Pollack. *Differential topology*, volume 370. American Mathematical Soc., 2010.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pp. 1737–1746, 2015.

Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002. ISBN 0-521-79160-X; 0-521-79540-0.

Gregory Henselman and Robert Ghrist. Matroid Filtrations and Computational Persistent Homology. *ArXiv e-prints*, June 2016.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 18:Paper No. 187, 30, 2017. ISSN 1532-4435.

Firas A. Khasawneh, Elizabeth Munch, and Jose A. Perea. Chatter classification in turning using machine learning and topological data analysis. *CoRR*, abs/1804.02261, 2018. URL http://arxiv.org/abs/1804.02261.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Li Li, Wei-Yi Cheng, Benjamin S. Glicksberg, Omri Gottesman, Ronald Tamler, Rong Chen, Erwin P. Bottinger, and Joel T. Dudley. Identification of type 2 diabetes subgroups through topological analysis of patient similarity. *Science translational medicine*, 7(311):311ra174–311ra174, 2015.

Elon L. Lima. The Jordan-Brouwer separation theorem for smooth hypersurfaces. *Amer. Math. Monthly*, 95(1):39–42, 1988. ISSN 0002-9890. doi: 10.2307/2323445. URL https://doi.org/10.2307/2323445.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in International Conference on Machine Learning Workshop on Deep Learning for Audio, Speech and Language Processing*, volume 30, pp. 3, 2013.

William S. Massey. *A basic course in algebraic topology*, volume 127 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1991. ISBN 0-387-97430-X.

John W. Milnor. On the Betti numbers of real varieties. *Proc. Amer. Math. Soc.*, 15:275–280, 1964. ISSN 0002-9939. doi: 10.2307/2034050. URL https://doi.org/10.2307/2034050.

Guido F. Montúfar, Razvan Pascanu, KyungHyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2924–2932, 2014. URL http://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 807–814, 2010. URL https://icml.cc/Conferences/2010/papers/432.pdf.

Jessica L. Nielson, Jesse Paquette, Aiwen W. Liu, Cristian F. Gu, C. Amy Tovar, Tomoo Inoue, A Irvine, John C. Gensel, Jennifer Kloke, Tanya C. Petrossian, et al. Topological data analysis for discovery in preclinical spinal cord injury and traumatic brain injury. *Nature communications*, 6: 8581, 2015.

Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete Comput. Geom.*, 39(1-3):419–441, 2008. ISSN 0179-5376. doi: 10.1007/s00454-008-9053-2. URL https://doi.org/10.1007/s00454-008-9053-2.

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

Nina Otter, Mason A. Porter, Ulrike Tillmann, Peter Grindrod, and Heather A. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Sci.*, 6(1):17, 2017. doi: 10.1140/epjds/s13688-017-0109-5. URL https://doi.org/10.1140/epjds/s13688-017-0109-5.

Steve Y. Oudot. *Persistence theory: from quiver representations to data analysis*, volume 209 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2015. ISBN 978-1-4704-2545-6. doi: 10.1090/surv/209. URL https://doi.org/10.1090/surv/209.

Michael L. Overton. *Numerical computing with IEEE floating point arithmetic*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. ISBN 0-89871-482-6. doi: 10.1137/1.9780898718072. URL https://doi.org/10.1137/1.9780898718072. Including one theorem, one rule of thumb, and one hundred and one exercises.

Jose A. Perea and John Harer. Sliding windows and persistence: an application of topological methods to signal analysis. *Found. Comput. Math.*, 15(3):799–838, 2015. ISSN 1615-3375. doi: 10.1007/s10208-014-9206-z. URL https://doi.org/10.1007/s10208-014-9206-z.

Jose A. Perea, Anastasia Deckard, Steven B. Haase, and John Harer. Sw1pers: Sliding windows and 1-persistence scoring; discovering periodicity in gene expression time series data. *BMC Bioinformatics*, 16:257:1–257:12, 2015. doi: 10.1186/s12859-015-0645-6. URL https://doi.org/10.1186/s12859-015-0645-6.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series. MIT Press, 2002. ISBN 9780262194754. URL http://www.worldcat.org/oclc/48970254.

Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Twelfth annual conference of the international speech communication association*, 2011.

Thierry P. Zell. *Quantitative study of semi-Pfaffian sets*. ProQuest LLC, Ann Arbor, MI, 2003. ISBN 978-0496-55984-8. URL http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqdiss&rft_dat=xri:pqdiss:3108439. Thesis (Ph.D.)–Purdue University.

Afra J. Zomorodian. *Topology for computing*, volume 16 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2009. ISBN 978-0-521-13609-9. Reprint of the 2005 original [MR2111929].

APPENDIX

In these pages we provide additional details on our simulations, more elaborate illustrations of our data sets, and further discussions as to how a neural network changes topology and its limitations in this regard. We also fill-in some relevant background material from topology and explain how we actually computed the Betti numbers in our experiments.

In the interest of reproducibility, we have made all our data sets and codes (fully anonymized for review purposes) available at:

> https://github.com/topnn/topnn_framework.

The static images in our article do not fully convey the dramatic topological changes a data set undergoes as it passes through the layers of a well-trained neural network. For a more complete picture, we have posted video clips generated from the same experiments and data sets at:

> https://github.com/topnn/topnn_framework/tree/master/videos.

A LIST OF EXPERIMENTS

Table 2 summarizes all our experiments on the three simulated data sets D-I, D-II, D-III (experiments on the MNIST data set are separately documented in Section F.1). For each data set, we state the type of architecture it is trained on and the number of well-trained networks obtained. For each well-trained[5] network, we track the Betti numbers through their layers — those results are presented in the main body of this article. We specify architecture with a sequence of numbers giving the widths of the layers in the network. The first number gives the dimension of the input, which is always 2 or 3 as we have limited ourselves to two- and three-dimensional manifolds for visualization purposes. The score function has two outputs that add up to 2 (one for category a and the other for category b) and for this reason the last number is always 2. We use $\operatorname{softmax}$ as the score function in all of our networks; recall that this is the function $s : \mathbb{R}^p \to \mathbb{R}^p$ whose ith coordinate is

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

where p is the number of categories. In our case, $p = 2$.

B SIMULATED DATA SETS

We have used three simulated point cloud data sets D-I, D-II, D-III in our experiments. Figure 10 shows the underlying manifolds $M = M_a \cup M_b$ (top) and the corresponding point cloud data sets $T_a = T \cap M_a$ and $T_b = T \cap M_b$ (bottom) sampled from the manifolds. The point cloud data set $T = T_a \cup T_b$ has each point labeled a or b to indicate its membership.

The samples for Betti numbers computations are obtained by further subsampling the training data set — a standard practice in computational topology and topological data analysis — without this step the Betti number computations will be prohibitively expensive. The subsampling is done by simply picking every kth sample: For D-I, $k = 3$; for D-II and D-III, $k = 4$.

While the Betti numbers are computed from a subsample of T_a, the neural networks are trained on the full sample T_a. Table 3 gives the number of samples used for training the neural networks and that used to run Betti numbers computations. To set hyperparameters (see Section E) we used holdout samples: Two for each data set, the first one is the data set itself and the other is the data set mapped by a randomly picked well-trained network.

C TOPOLOGICAL CHANGES IN A NEURAL NETWORK

Since our goal is to show how a neural network can simplify the topology of two entangled data sets and separate them, we first specify the most fundamental restrictions that topology places on

[5]Recall that this means zero training error and less than 1% generalization error.

data set	activation	architecture	#
D-I	tanh	2-15-15-15-15-15-15-15-15-15-15-15-2	30
D-I	leakyrelu	2-15-15-15-15-15-15-15-15-15-15-15-2	30
D-I	leakyrelu	2-05-05-05-05-03-05-05-05-05-05-2	30
D-I	leakyrelu	2-15-15-15-15-03-15-15-15-15-2	30
D-I	leakyrelu	2-50-50-50-50-50-50-50-50-50-50-2	30
D-I	relu	2-15-15-15-15-15-15-15-15-15-15-15-2	30
D-II	tanh	3-15-15-15-15-15-15-15-15-15-15-15-2	32
D-II	leakyrelu	3-15-15-15-15-15-15-15-15-15-15-15-2	36
D-II	relu	3-15-15-15-15-15-15-15-15-15-15-15-2	31
D-II	tanh	3-25-25-25-25-25-25-25-25-25-25-25-2	30
D-II	leakyrelu	3-25-25-25-25-25-25-25-25-25-25-25-2	30
D-II	relu	3-25-25-25-25-25-25-25-25-25-25-25-2	30
D-III	tanh	3-15-15-15-15-15-15-15-15-15-15-15-2	30
D-III	leakyrelu	3-15-15-15-15-15-15-15-15-15-15-15-2	46
D-III	relu	3-15-15-15-15-15-15-15-15-15-15-15-2	30
D-III	tanh	3-50-50-50-50-50-50-50-50-50-50-2	30
D-III	leakyrelu	3-50-50-50-50-50-50-50-50-50-50-2	30
D-III	relu	3-50-50-50-50-50-50-50-50-50-50-2	34

data set	activation	architecture	#
D-I	tanh	2-15-15-15-15-2	30
D-I	tanh	2-15-15-15-15-15-15-15-15-15-2	30
D-I	leakyrelu	2-15-15-15-15-2	30
D-I	leakyrelu	2-15-15-15-15-15-15-15-15-15-2	30
D-I	relu	2-15-15-15-15-2	30
D-I	relu	2-15-15-15-15-15-15-15-15-15-2	30
D-II	tanh	3-15-15-15-15-2	31
D-II	tanh	3-15-15-15-15-15-2	31
D-II	tanh	3-15-15-15-15-15-15-15-15-2	30
D-II	leakyrelu	3-15-15-15-15-2	31
D-II	leakyrelu	3-15-15-15-15-15-2	30
D-II	leakyrelu	3-15-15-15-15-15-15-2	30
D-II	leakyrelu	3-15-15-15-15-15-15-15-2	31
D-II	leakyrelu	3-15-15-15-15-15-15-15-15-15-2	42
D-II	relu	3-15-15-15-15-2	32
D-II	relu	3-15-15-15-15-15-2	32
D-II	relu	3-15-15-15-15-15-15-15-15-15-2	31
D-III	tanh	3-15-15-15-15-15-15-2	30
D-III	tanh	3-15-15-15-15-15-15-15-15-15-2	31
D-III	leakyrelu	3-15-15-15-15-15-15-2	30
D-III	leakyrelu	3-15-15-15-15-15-15-15-15-15-2	30
D-III	relu	3-15-15-15-15-15-15-2	33
D-III	relu	3-15-15-15-15-15-15-15-15-15-2	32

Table 2: *List of experiments*: Each row in the table describes one experiment. First column specifies the data set on which we have trained the networks. The next two columns describe the architecture, respectively the activation used and the number of neurons of each layer. The last column gives the number of well-trained networks obtained.

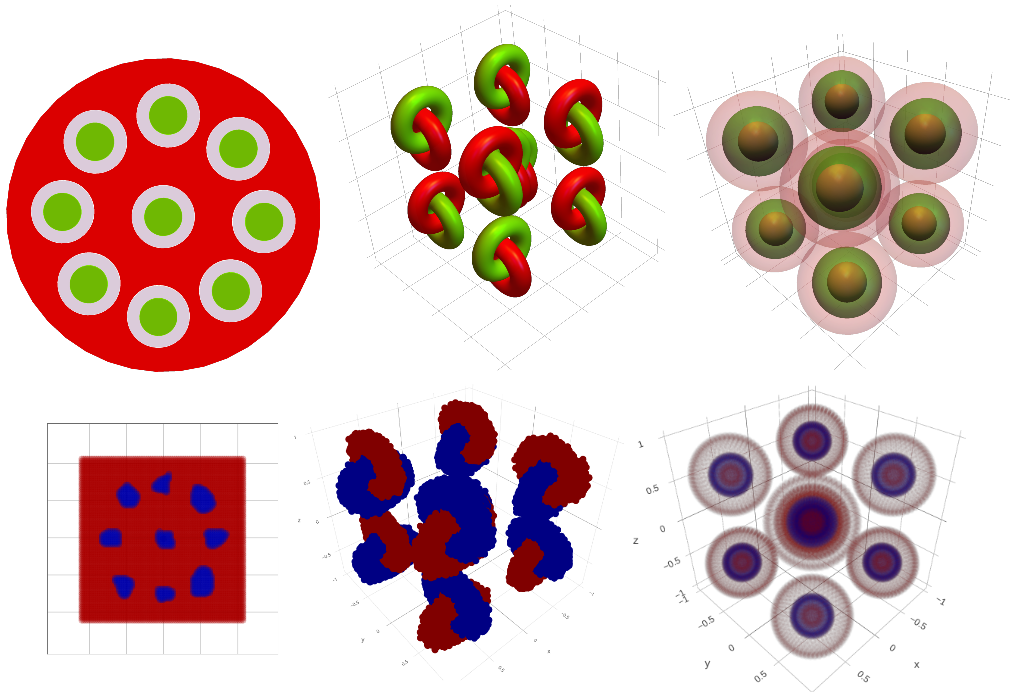


Figure 10: *Top*: The manifolds underlying data sets D-I, D-II, D-III. The green M_a represents category a, red M_b represents category b. *Bottom*: Actual training set sampled from the manifolds. D-I and D-III are sampled on a grid and D-II is sampled uniform randomly from the tori. The difference in sampling schemes is inconsequential as the samples are sufficiently dense that there is no difference in training and testing behaviors.

the ability of maps to separate two entangled sets. They allow us to identify situations when these restrictions do not apply, and in which case we know the data sets may potentially be disentangled.

Data set	Training set size	Subsample for Betti calculations
D-I	7,800	2,600
D-II	45,000	11,250
D-III	37,800	9,450

Table 3: Size of training set and size of the subset used to run Betti numbers calculations.

We start with an elementary (but deep) well-known fact from topology, a consequence of the Jordan–Brouwer Separation Theorem (Guillemin & Pollack, 2010, Chapter 2), that formalizes our intuition that there is no diffeomorphism that can pull a set in the "inside" of a surface away from the surface itself.

Lemma C.1 *Let $M \subseteq \mathbb{R}^n$ be a compact connected hypersurface, i.e., a codimension-one smooth manifold*[6] *in $\mathbb{R}^n$. Let U be a set in the "inside" of M, i.e., the bounded connected set*[7] *in the complement of M in $\mathbb{R}^n$. Then U and M are not linearly separable and for any diffeomorphism*[8] *$f : \mathbb{R}^n \to \mathbb{R}^n$, the sets $f(M)$ and $f(U)$ are also not linearly separable.*

Lemma C.1 requires diffeomorphisms, but depending on the choice of activations, a neural network may not necessarily be a diffeomorphism, and thus Lemma C.1 may not apply directly. Fortunately, all of the standard activations can be slightly perturbed so that the respective neural network becomes a diffeomorphism.

Definition C.2 (Near-diffeomorphism) *A continuous map $g : M \to N \subseteq \mathbb{R}^n$ between two topological spaces is called a* near-diffeomorphism *on the compact subspace $U \subseteq M$ if for any $\varepsilon > 0$, there is a diffeomorphism $f_\varepsilon : U \to V \subset \mathbb{R}^n$ such that for all $x \in U$, $\|g(x) - f_\varepsilon(x)\|_2 < \varepsilon$.*

Definition C.2 implicitly assumes that $\dim(M) = \dim(N) = \dim(U) = n$ since the dimension of a topological space is invariant under diffeomorphism. We may however extend the definition to the case where $N \subseteq \mathbb{R}^m$ and $m < n$. To this end, notice that a constant map is a near-diffeomorphism on any compact domain — adding a linear map with a tiny gradient to the constant map turns it into a diffeomorphism. Therefore if g is a map to $N \subseteq \mathbb{R}^m$, by augmenting it with an additional $m - n$ zeros, we obtain

$$\widetilde{g}(x) \coloneqq (g_1(x), g_2(x), \dots, g_m(x), 0, \dots, 0) \quad \text{for any } x \in M,$$

and Definition C.2 now applies to $\widetilde{g}$ as it maps to $\mathbb{R}^n$.

Lemma C.1 carries over to near-diffeomorphisms. The implications of this for neural networks are encapsulated in the following theorem.

Theorem C.3 *Let $M_a \subseteq \mathbb{R}^d$ be a compact connected hypersurface representing category a and $M_b \subset \mathbb{R}^d$ be a set in the inside of M_a representing category b. Consider a feedforward neural network $\nu : \mathbb{R}^d \to \mathbb{R}$ given by the compositions of l layers and a score function as in Section 3, with activations* $\tanh$, relu, *or* $\mathrm{leakyrelu}$*, and whose widths satisfy*

$$d \geq n_1 \geq n_2 \geq \cdots \geq n_l. \tag{3}$$

Then for any set of parameters, i.e., the weights and biases in all layers and the score function, the network ν is a near-diffeomorphism (in the extended sense discussed after Definition C.2) and there exists a neighborhood $U \subseteq M_a$ or a neighborhood $V \subseteq M_b$ that is wrongly classified by ν.

The proof is elementary. Each layer in the network satisfying assumption of the theorem is a near-diffeomorphism, so we can approximate the network uniformly and arbitrary well by a diffeomorphism on a compact domain containing the manifold M_a. Meanwhile, by Lemma C.1, the manifolds cannot be linearly separated by any diffeomorphism, which completes the proof.

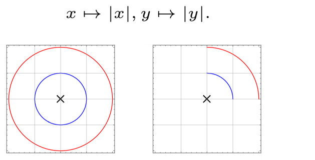
[6] All topological spaces that we consider here are subspaces of $\mathbb{R}^n$ for some n, with topological and smooth structures induced by the Euclidean metric.

[7] The existence of which follows from Jordan–Brouwer Separation Theorem Lima (1988). In addition, such a manifold is necessarily orientable.

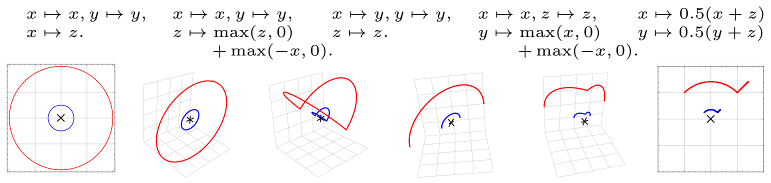
[8] It is enough to have a C^1-diffeomorphism and M a C^1-manifold Guillemin & Pollack (2010).

There are two ways in which the assumptions of the theorem can be violated, and each illustrates how a neural network transforms the topology of its input. The first is when one of the activations is not a near-diffeomorphism (e.g., absolute value $|\cdot|$) and the second is when the requirement on nonincreasing width, i.e., equation 3, is not satisfied. Note that if the entangled manifolds have dimension less than d, then equation 3 is already violated in the first layer. Figures 11a, 11b illustrate these two cases, showing that if the assumptions in Theorem C.3 are not satisfied, then M_a and M_b can be separated by some neural network.

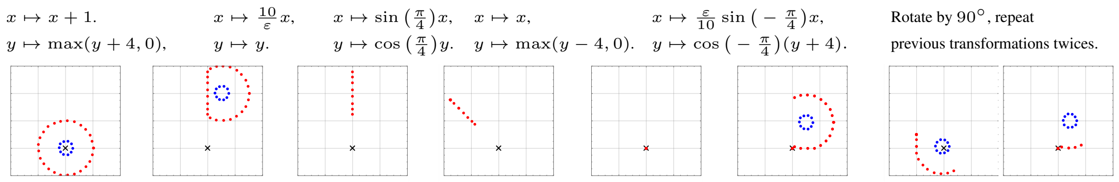
Standard activations always result in layers that are near-diffeomorphisms except if the layer maps input to a higher dimensional output. Therefore, among those two ways to separate entangled data sets, it is the later one that occurs in a standard feedforward network. Figure 2 shows actual topology transformation in a neural network, a two dimension data is mapped into three dimensional space where it undergoes topology transformations.[9]



(a) non near-diffeomorphism



(b) mapping to a higher dimension


$x \mapsto x + 1.$
$y \mapsto \max(y + 4, 0),$
$x \mapsto \frac{10}{\varepsilon}x,$
$y \mapsto y.$
$x \mapsto \sin\left(\frac{\pi}{4}\right)x,$
$y \mapsto \cos\left(\frac{\pi}{4}\right)y.$
$x \mapsto x,$
$y \mapsto \max(y - 4, 0).$
$x \mapsto \frac{\varepsilon}{10}\sin\left(-\frac{\pi}{4}\right)x,$
$y \mapsto \cos\left(-\frac{\pi}{4}\right)(y + 4).$
Rotate by 90°, repeat
previous transformations twices.

(c) discrete data set

Figure 11: Examples where Theorem C.3 does not hold: (a) two neurons activated with, $|\cdot|$, a non near-diffeomorphism that directly changes topology of entangled circles making them linearly separable (b) for each coordinate the map is a near-diffeomorphism, however overall, separation of the entangled circles happens by mapping to a higher dimension space (c) for a finite point clouds sampled off two entangled circles, a near-diffeomorphism network maps to linearly separable point clouds, yet the network fails to generalize: some areas on the red circle will be wrongly classified if new samples are observed.

Figure 11c shows another example of what is actually not a topology change but does lead to separation of two point cloud data sampled on an entangled manifolds despite the fact that all assumptions of the Theorem C.3 are satisfied. The figure shows samples of two categories (red and blue) sampled off two manifolds in $\mathbb{R}^2$, a larger circle and a smaller circles centered at the origin. Assume, we have a finite number of samples x_i, $i = 1, \ldots, N$ (N is the size of train and test sets combined). There exists a sequence of transformations realizable by deep enough 2-dimensional neural network (no more than two neurons in each layer), based on near-diffeomorphisms activations, that can make the point cloud linearly separable, regardless of how large N is. Here is how:

1. Collapse samples on the outer circle that are left of y-axis to an interval on the y-axis (by applying relu, $x \mapsto \mathrm{relu}(x)$)
2. Next, rescale x so that the gap between the left most sample on the right of y-axis and this interval becomes large enough (the rescaling is of order $o(1/\varepsilon)$ where ε, the closes distance to the y-axis of any of the samples of x_i, $i = 1, \ldots, N$ that are now to the right of y-axis)
3. Rotate the data set, so that the interval is entirely contained in the forth quadrant, where values of coordinates x and y are negative
4. Collapse the interval into a point at the origin (again, relu activation will do the trick)

[9]Strikingly, a simple two steps linear interpolation makes those topological changes stand out and appear as "foldings" of a space. A movie showing those foldings is available at (fully anonymized repository for review purposes) https://github.com/topnn/topnn_framework/tree/master/videos

5. Repeat similar transformations a few times over and over again, until enough of the outer circle is changed so that the inner circle is linearly separable

As a result, for all samples x_i, $i = 1, \dots, N$ no misclassification is observed, however Theorem C.3 implies that some neighborhood on the circles is still misclassified for new samples — the network fails to generalize properly, leading to existence of "adversarial" examples.

We finish this section by noting that in Montúfar et al. (2014) it was suggested that existence of "foldings" allows deep network to replicated their behavior across many regions in the inputs which might explain the efficacy of deep relu networks, our results further support this view.

Looking at a deep neural network through the lens of topology, prescribed different roles for shallow vs deep layers. Our results in the main text provided evidence that training a neural networks yields solutions with bottom layers changing geometry (i.e., the co-domain is merely a stretched or twisted version of the domain) they produce little to no "foldings", and that it is in the deeper layers that the "foldings" start to occur, changing topology and allowing for classification of complex data. Figure 11b illustrates such archetypal "folding".

D Background from topology

Here we briefly overview background material from topology, with the goal of providing a short exposition to the parts of simplicial homology theory relevant for our purposes. Simplicial homology studies topology of *simplicial complex* so our motivation for introducing it is clear: as we mentioned in the main text, computation of Betti numbers of a point cloud data proceeds through construction of simplicial complex on a point cloud data and then computation of Betti numbers of this complex. Since topology is not often encountered within machine learning and neural networks, we feel including this material might be of benefit. We start by recalling definition of simplicial complex and proceed to recall definition of *Vietoris–Rips complex* that was briefly mentioned in the main text. Vietoris–Rips complex is the simplicial complex that we use in our computations.

D.1 Simplicial complex

A k-dimensional *simplex* σ in $\mathbb{R}^d$ (or a k-simplex), is a convex hull of $(k+1), k = 0, \dots, d$, affinely independent points in $\mathbb{R}^d$. Namely, a simplex is either a point (0-simplex), a line segment (1-simplex), a triangle (2-simplex) or their higher dimensional counterparts. A single k-dimensional simplex is represented by listing the set of its $(k+1)$ vertices $\sigma = \{v_1, \dots, v_{k+1}\}$. Facets of a k-simplex form simplices of dimension 0 to $(k-1)$ (e.g. the two edges of 1-simplex are 0-simplices, the three sides of 2-simplex are 1-simplices and its three vertices are 0-simplices). An m-dimensional *simplicial complex* S in $\mathbb{R}^d$, is a finite collection of simplices in $\mathbb{R}^d$ of dimension at most m that are attached together in a "nice way", so that any intersection between two simplices in S is necessary a facet of both of them. Simplicial complex must also include all facets of all its simplices as a separate simplices in their own right, e.g. if simplex $\sigma_1 = \{v_1, v_2, v_3\}$ is included in S, then the simplices $\{v_1, v_2\}, \{v_2, v_3\}, \{v_1, v_3\}, \{v_1\}, \{v_2\}, \{v_3\}$ also belong to it. A simplicial complex is represented by listing all of the simplices that comprise it, e.g. $S = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$. See Figure 12 for illustration of 3-dimensional simplicial complex in $\mathbb{R}^3$.

Next we recall definition of the simplicial complex that we use for computation of Betti numbers of point cloud data.

Let N points $X = x_1, x_2, \dots, x_N$ on $\mathbb{R}^d$ and a distance $d(\cdot, \cdot)$ on $\mathbb{R}^d$, define $\mathrm{VR}_\varepsilon\big(X\big)$, *Vietoris–Rips complex* at scale $\varepsilon \geq 0$ on X, to be the simplicial complex whose 0-simplices are points X, and whose higher dimensional simplices are all possible simplices built on X in which the maximal pairwise distance between any two vertices is at most 2ε:

$$\mathrm{VR}_\varepsilon(X) := \big\{\text{all possible simplices } \sigma_k = \{x_{i_1}, \dots, x_{i_{k+1}}\}, x_{i_j} \in X, \\ \text{such that } d(x_{i_j}, x_{i_l}) \leq 2\varepsilon \text{ for all } x_{i_j}, x_{i_l} \in \sigma_k\big\}.$$

It is a routine to verify that $\mathrm{VR}_\varepsilon(X)$ satisfies all the requirements of simplicial complex.

In the next section we start our exposition of homology theory, in the subsequent section we show how one can construct homology theory for a simplicial complex.

D.2 Homology groups and Betti numbers

Homology theory, roughly speaking, is an abstract way to encode topology of a space by means of chain of abelian groups and their homeomorphisms.

Let a collection of abelian groups C_i, $i = 0, \dots, d$ and a special kind of group homeomorphisms $\partial_i : C_i \longrightarrow C_{i-1}$ called *boundary operators*. A *chain complex* composed of those abelian groups is an abstract construction where domains an codomains of boundary operators form a chain:

$$\cdots \to C_d \xrightarrow{\partial_d} C_{d-1} \xrightarrow{\partial_{d-1}} \cdots \xrightarrow{\partial_{i+1}} C_i \xrightarrow{\partial_i} C_{i-1} \xrightarrow{\partial_{i-1}} \cdots \xrightarrow{\partial_{\partial_2}} C_1 \xrightarrow{\partial_{\partial_1}} C_0 \xrightarrow{\partial_{\partial_0}} 0.$$

Boundary operators, ∂_i, are required to satisfy

$$\partial_i \circ \partial_{i+1} = 0 \tag{4}$$

for all $i = 0, \dots, d$. The elements in the image of ∂_i (a subgroup of C_{i-1}) are called *boundaries*, while elements in the kernel of ∂_{i-1} (also a subgroup of C_{i-1}) are called *cycles*. One of the immediate consequences of equation 4 is that for all $i = 0, \dots, d$

$$B_i \coloneqq \operatorname{im}(\partial_{i+1}) \subset Z_i \coloneqq \ker(\partial_i).$$

A subgroup of abelian group is normal, so $\operatorname{im}(\partial_{i+1})$ is a normal subgroup of $\ker(\partial_i)$ therefore we can take quotient

$$H_i \coloneqq Z_i / B_i = \ker(\partial_i) / \operatorname{im}(\partial_{i+1}), \quad i = 0, \dots, d.$$

H_i is called the ith *homology group*. *Betti numbers* are defined to be the ranks of the homology groups i.e.,

$$\beta_i \coloneqq \operatorname{rank}(H_i), \quad i = 0, \dots, d.$$

Different types of chain complexes and different types of boundary operators give rise to different homology theories. What do chain complexes and the boundary operators look like in simpilicial homology is describe in the next section.

D.3 Simplicial homology

Given k simplex and its set of vertices $\sigma = \{v_0, \dots, v_k\}$, one can attach negative or positive orientation to the representation of this simplex (according to the ordering of the vertices in the representation. An orderings corresponds to one of the two equivalence classes, with two ordering belonging to the same category if and only if one ordering can be obtained from the other using even number of permutations). Orientation of a simplex induces orientation on all it facets. We call a simplicial complex *consistently oriented* if all simplexes in the complex agree on the orientation on their intersecting facets. Given, S, a consistently oriented simplicial complex, we formally define

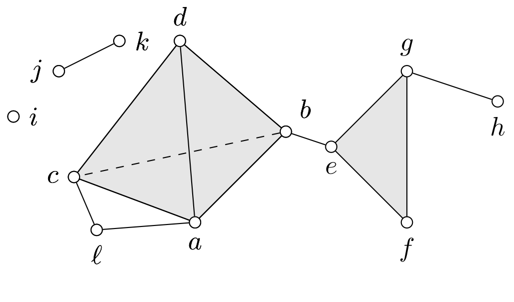


Figure 12: A 3-dimensional simplicial complex in $\mathbb{R}^3$ that consists of thirty two simplices $S = \big\{\{a, b, c, d\}, \{e, f, g\}, \dots, \{e, b\}, \dots, \{a\}, \{b\}, \dots, \{\ell\}\big\}$. It has single 3-simplex $\{a, b, c, d\}$, five 2-simplices, e.g. $\{e, f, g\}$ is 2-simplex, eighteen 1-simplices, e.g. $\{e, b\}$, $\{g, h\}$, $\{c, \ell\}$, $\{\ell, a\}$ and fourteen 0-simplices $\{a\}, \{b\}, \dots, \{\ell\}$. Within simplicial complex the simplices are attached (intersect) along their facets.

abelian groups, $C_i(S)$, on S in the following way: let S_i be the set of i dimensional simpleces in S, then an element of $C_i(S)$ is a formal sum (i.e., a free abelian group)

$$\sum_j n_j \sigma_j \big(-1^{[\sigma_j]}\big), \quad n_j \in \mathbb{F}_2 \text{ and } \sigma_j \in S_i,$$

where $-1^{[\sigma_j]}$ is either 1 or -1 depending on the orientation of the simplex σ_j and $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$ (i.e., $\mathbb{F}_2$ is a field with two elements only). The boundary homeomorphisms $\partial_i : C_i(S) \to C_{i-1}(S)$ are defined as the following:

$$\partial_i \sigma_i = \sum_j (-1)^j \{v_0, v_1, \ldots, \hat{v}_j, \ldots, v_i\}$$

where $\hat{v}_j$ indicates that vertex v_j is deleted from the sequence. For example

$$\partial_1\{a, b\} = b - a \text{ and } \partial_2\{a, b, c\} = \{b, c\} - \{a, c\} + \{a, b\} = \{b, c\} + \{c, a\} + \{a, b\}.$$

It can be verified that we have $\partial_i \circ \partial_{i-1} \equiv 0$ for all $i = 0, \ldots, n$, therefore the requirement of equation 4 on the boundary operators is satisfied. The abelian groups defined in this way and boundary homeomorphism form a chain complex. Therefore in order to compute k homology of the simplicial complex S we take quotient of a cycle group by a boundary group to obtain

$$H_i(S) = Z_i(S)/B_i(S).$$

Yielding

$$\beta_i(S) = \operatorname{rank}\big(H_i(S)\big) = \operatorname{rank}\big(Z_i(S)\big) - \operatorname{rank}\big(B_i(S)\big).$$

For a given simplicial complex, an element of abelian group $C_i(S)$ is vector with values in $\mathbb{F}_2$ and with length of $|S_i|$ (the number of i-simplices in the complex S), the boundary operators are realized as matrices D_{∂_i} of dimensions $|S_{i-1}| \times |S_i|$ which map each i dimensional simplex represented as a vector of length $d_i = |S_i|$ to a vector of length $d_{i-1} = |S_i|$ and so rank nullity theorem of linear algebra prescribes:

$$\begin{aligned}\beta_i(S) &= \operatorname{rank}(H_i(S)) \\ &= \operatorname{rank}(Z_i(S)) - \operatorname{rank}(B_i(S)) \\ &= d_{i-1} - \operatorname{rank}(D_{\partial_i}) - \operatorname{rank}(D_{\partial_{i-1}}).\end{aligned}$$

Since, in principle, the size of the simplicial complex $|S|$ can be very large (e.g. exponential in the number of vertices in a complex), the naive construction of matrices D_{∂_i} is often problematic so different approach needs to be developed to compute the rank of those matrices without ever obtaining the complete matrix. The study of techniques that can render those computations and accelerate them are within the scope of computational topology.

Let N points $X = x_1, x_2, \ldots, x_N$, we are now at the position to define homology of this point cloud data at scale $\varepsilon > 0$ to be the simplicial homology of $\mathrm{VR}_\varepsilon(X)$, Vietoris–Rips complex at scale ε constructed on X.

E COMPUTING BETTI NUMBERS FOR NEURAL NETWORKS

E.1 PERSISTENT HOMOLOGY

Topological features, the Betti numbers, are inherently non-robust to small local changes. For example punching a small hole in a sphere has little influence on geometry of the shape but has a large consequence from the point of view of topology. Even a very small hole "kills" β_2 making a sphere into a topological disk. The problem becomes even bigger when we are trying to estimate Betti numbers of a manifold form point cloud data sampled of this manifold. Persistent homology blends geometry and topology in an attempt to compute Betti number of a manifold from point cloud data while at the same time avoiding the problem of extreme sensitivity of topology to local perturbations. The idea of persistent homology is to introduce "geometric scale" into topology computation. At the beginning the scale is set to zero, this is when we "over-fit" our point cloud data. Each point contributes separate component to the overall shape. So we have a discrete topological space. Then as the scale gradually increases we fuse together more and more distant points and topological space becomes richer. If we fuse all the points together, we get a single topological ball and the space becomes simple again. The output of persistent homology is a set of intervals (barcodes), a single interval for each topological feature. The left end of the interval records when the feature has appeared (its " birth") and the right end of the interval records when it has disappeared (its "death"). The length of the interval is called *persistence* of the corresponding feature. Topological features that are nonrobust to perturbations will produce short intervals in the persistent barcodes, conversely topological feature that persist long enough, i.e., produce long intervals are the prominent resilient features of the underlaying manifold. This way persistent homology gives a way to identify prominent topological features in the underlaying manifold from point cloud data (e.g. point cloud sampled of a sphere with a small punched hole will have single β_2 as its prominent topological feature).

As we already mentioned in the main text, running persistent homology is not computationally feasible when it comes to massive number of experiments. In addition, while barcodes avoid the problem of sensitivity to local perturbation, they complicate interpretation of the results. Automated statistical analysis of many barcodes is still an active area of research (and often requires additional large computation effort). For those reasons we resort to single homology computation at a given scale, where persistent homology allows us to pick the right scale.

E.2 Computation of homology

Let us now take a step back and describe a general pipeline for homology computation. Practical homology computation involve a number of steps: the preprocessing step which consist of smoothing out point cloud data, or throwing away outliers. This step is generally called noise reduction step. After noise reduction the next step is to construct simplicial complex from the point cloud, a number of complexes exists. In general, when a complex is associated to a point cloud data we are not concerned to preserve the underlaying geometry, only the topology. While keeping the geometry intact is not our concern, what is our concern is the size of the final simplicial complex. The size of the simplicial complex is related to the number of points, this leads to the following trade-off: on the one hand, as the number of points sampled increases we are better positioned to recover the underlaying topology. On the other hand, the resulting simplicial complex might turn out to be prohibitively large for caring out computations. So the goal is to construct optimal (small) simplicial complex with minimal topology distortion.[10]

We use Vietoris–Rips complex, it comes with good theoretical guarantees Oudot (2015), is simple, suitable for high-dimensional data and supported on many computational topology packages, in particular the latest and arguable the most advance package, EIRENE Henselman & Ghrist (2016) provides support for Vietoris–Rips complex.

Once the simplicial complex is constructed, to further accelerate computations the complex is usually simplified in a way that reduces its size but does not change its topology. The different between this step, and the previous two steps is clear: all the reductions that have effect on the topology of the point cloud have been performed in the first two steps and the obtained simplicial complex (before its simplification) is assumed to represent the correct topology of the underlaying domain, the simplification is done to accelerate computations but mast not alter the topology. Over the years many method have been proposed to simplify the simplicial complex and to accelerate computations examining those in some details would take us to far into computational topology and is beyond the scope of our work. Figure 13 summarizes homology computations steps. Next we examine how each of the steps is adopted to our case and discuss how we use persistent homology.

E.3 Our method for homology computation

Our data sets are simulated, so we have the luxury of having a clean data with no need to run de-noising before Betti numbers computation. Nevertheless, a few simple steps to improve Betti numbers computations are done prior to construction of the simplicial complex. For D-I and D-III

[10]The worst case performance of topological features extraction is $O(m^3)$ where m is the number of simplices in the simplicial complex (the topology is extracted by computing Smith normal form of boundary matrix of the complex). The number of simplices, in principle, can be as large as 2^N (N is the number of points in point cloud). This is avoided in practice, but nevertheless the size of the simplicial complex is a major problem, for larger Betti number the size of the complex increases since it now contains more simplexes of a higher dimension.

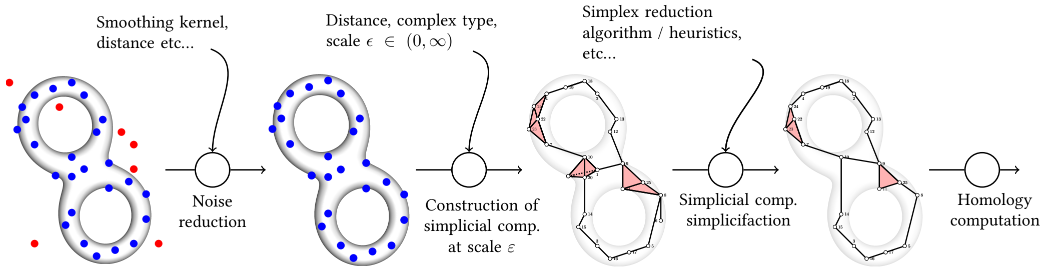

Smoothing kernel, distance etc...
Noise reduction
Distance, complex type, scale $\epsilon \in (0,\infty)$
Construction of simplicial comp. at scale ε
Simplex reduction algorithm / heuristics, etc...
Simplicial comp. simplicifaction
Homology computation

Figure 13: Steps towards computation of homology of point cloud data.

data sets, we sample on a grid rather than randomly sampling on the manifold (for D-I the grid is two dimensional square and for D-III the samples are the grid is a spherical Fibonacci lattice see Figure 10). While for D-I this choice is not very critical and data could be randomly sampled of D-I, for D-III this is more important. D-III is the most complex data set and having evenly spread samples on a spherical lattice makes computations more stable (we can have less samples and run computations faster). For D-II we randomly subsample the data. D-II has easier topology than D-III and random sampling worked well. Next, we select a unique subset of the samples, namely if a few points overlay we throw away all but one of those points. This way we prevent degenerate simplices in which two points happen to be at the same coordinate (this can potentially happen in the intermediate layers if some points are mapped to the same point in the output). To construct simplicial complex we use *unweighted* geodesic distance (graph distance) induced by k-nearest neighbors graph. The distance is called unweighted since when we compute the distance we normalize each edge in the k-nearest neighbors graph to have length one. As explained in the main text, Section 4, this is done to address the problem of varying local densities. Exact computation of nearest neighbors graph is too costly, we use a well known ball tree algorithm to construct approximate k-nearest neighbors graph. Ball tree algorithm is especially suitable here, since our samples comes from a low dimensional manifold embedded in a higher dimensional space (a wide hidden layer in neural network maps a low dimensional data to a higher dimensional space). Once the graph is built, normalized and geodesic distance is computed from the graph, we use Eirene package to construct Vietoris–Rips simplicial complex at a *fixed* (for all subsequent calculation on the same data set) scale and to run homology computation. Since all edges are normalized, the geodesic distance is a discrete distance and consequently the scale is an integer number specifying the number of edges. Next we describe how we tune the two parameters: number of neighbors used for graph construction and the scale at which we construct Vietoris–Rips simplicial complex, this is where persistent homology is used.

To tune the parameters we run persistent homology to select the scale at which the topology of category a (this is the category we monitor through the layers) in the input and in the output is recovered. For this purpose we randomly and uniformly sample data from D-I D-II, D-III data sets, this gives us samples in the input of the network. In addition we select one well trained network for each data sets and look how its input, those are the samples at the output of the network.

Picking the number of neighbors for nearest neighbors graph construction is done as the following: for data points p_1 and p_2 define $d_{p_1}(p_2)$ to be ∞ if p_2 is not one of the k neighbors of p_1 (assume that k is large $k \gg 10$), or set it to be $q \in \{1,\dots,k\}$ if p_2 is the qth nearest neighbor of p_1. Then define a *nearest neighbors distance* $d_k(p_1,p_2)$ to be:

$$d_k(p_1,p_2) \coloneqq \min\left\{d_{p_1}(p_2), d_{p_2}(p_1)\right\}.$$

With this distance we run persistent homology. Example of the barcode diagrams obtained for category a of D-III data set in the input of a network, are in Figure 15, in red is the selected number of neighbors k^* for nearest neighbor graph construction. Persistent homology allows us to see what is the right scale at which we recover correct topology. Our next step is to use this distance and to find the right scale (in this distance) at which we detect all topological features of this manifold. The barcode obtained at this step are show in Figure 14. Let us note, in principle those two steps can be combined (after all the same topological features are used to pick both of those parameters) in which case we run a single persistent homology and only pick the number of neighbors in the nearest neighbors graph and then recover topology directly from the resulting unweighted nearest neighbors graph. But splitting the process into two steps is more flexible for future variations. Indeed, we think

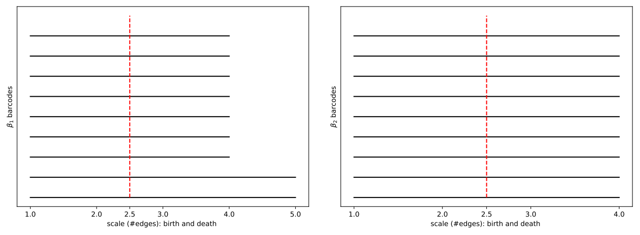


Figure 14: Once the k^* is set, we compute nearest neighbors graph, and obtain distance $d_{k^*}(\cdot, \cdot)$. *Left*: barcodes diagrams for β_1 for D-II category a (the input to the network). *Right*: barcodes diagrams for β_1 for D-III category a (the input to the network). In both cases the selected scale is set to 2.5 and is marked in red.

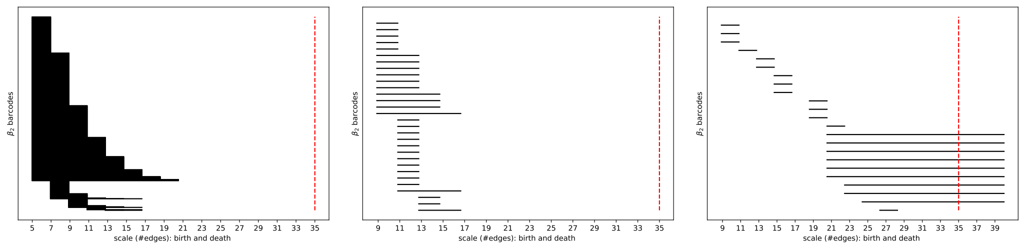


Figure 15: Here we illustrate some of the persistence barcodes obtained when tuning the number of neighbors for D-III, category a input. The selected value of k^* is set to 35 and is marked by red punctured line, at this scale the topology of category a in D-III is correctly recovered. *Left*: The full persistence barcodes diagram for β_1. Below scale 21 (in the distance $d_k(\cdot, \cdot)$ there are many intervals (in fact so many that it is impossible to distinguish between individual intervals, they look like as filled area), while we expect it to be a set of nine sphere $\beta_1 = 0$, this means that we need to set number of neighbors larger than 21 to have correct scale). *Center*: zoom in on the bottom 32 persistence barcodes in barcodes diagram of β_1. *Right*: the full persistence barcodes diagram for β_2.

of the first step as recovering geodesic distance on the underlaying manifold and think of the next step as building simplicial complex at a given scale using this geodesic distance. The second step can be naturally converted into persistent homology.[11]

The selected parameters are: for D-I the nearest neighbors graph is constructed using 14 neighbors, and then Vietoris–Rips simplicial complex is computed at scale $\varepsilon = 2.5$ (since our distance is discrete this is the same as $d_{14}(\cdot, \cdot) \leq 2$). For D-II and D-III the nearest neighbors graph is constructed using 35 neighbors, and then Vietoris–Rips simplicial complex is computed at scale $\varepsilon = 2.5$ (same scale as before). For MNIST experiments we've adopted same distance as for D-I data set, i.e., we are using $d_{14}(\cdot, \cdot) \leq 2$) distance.

F ADDITIONAL PLOTS AND EXPERIMENTS

Here we show some additional plots not presented in the main part. Figure 16 shows individual Betti numbers for shallow networks rather than the total complexity $\omega(M_a)$ (the two extreme cases: one is homeomorphism and the other is non-homeomorphism).

In Figure 17, PCA projection shown for first few layers in a shallow network with $\tanh$ activation, the data set D-II (category b), is bent but no topology changes are observed.

Finally Figure 18 shows, comparison of deep and shallow networks. In Figure 18, the left plot shows the average change in topology for shallow network (averaged over all simulation D-I, D-II, D-III, taking most shallow network), and the average change in topology for deep networks (averaged over

[11]If persistent homology is used in the second step then we need not to be very precise at the first step, e.g. all we might want to recover is the number of connected components, then in the second step we investigate topology of each component by running persistent homology.

all simulation D-I, D-II, D-III, taking 10 layers deep network). The difference is zero for the output of first layer it increases for the second layer, and is largest for the forth layer. In other words, first layer in short and deep networks produce (on average) similar changes in topology, and the change increases in the subsequent layers, deep layers now "work harder" than before. The right plot in In Figure 18, compares average change in topology for relu networks and for tanh networks, the difference is heavily skewed to the left, showing that relu reduces topology much faster compared with tanh .

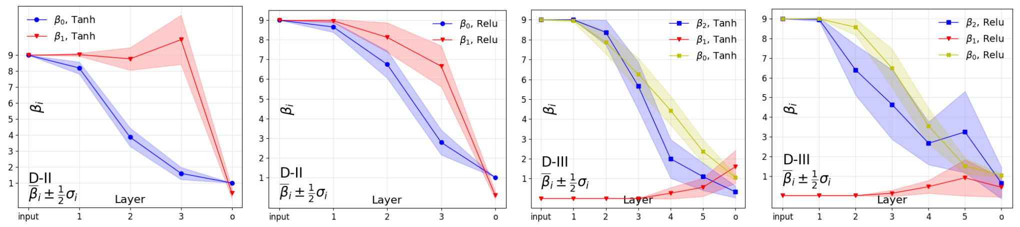


Figure 16: Shallow networks with relu and tanh activation, changes of individual Betti numbers. For D-II data set the network has architecture (3-15-15-15-15-2), for D-III data set the network has architecture (3-15-15-15-15-15-15-2)

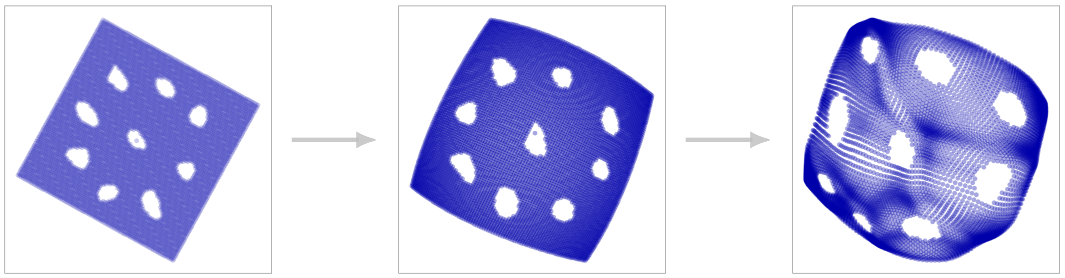


Figure 17: From left to right: Initial data sets, output of the first and second layers. PCA projections of the first 3 layers in a shallow smooth network (tanh activation, 2-15-15-15-15-2).

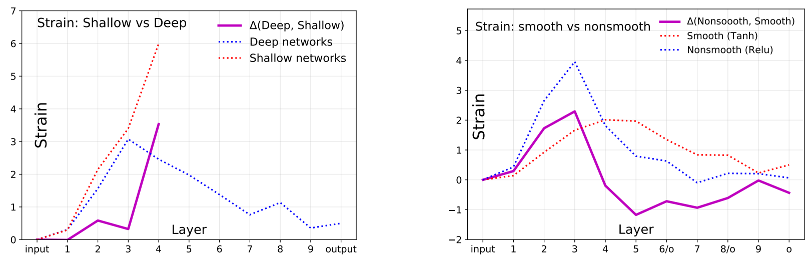


Figure 18: Summary of the effects of depth and smoothness of activation function on topology changes within a neural network. The "Strain" is the relative reduction of the topological complexity at a given layer compared with the previous layer $\omega\big(\nu_l(M_a)\big) - \omega\big(\nu_{l-1}(M_a)\big)$. The purple continues line is the difference between the average strains for each kind of network: *Left* deep vs shallow networks, ***Right***: tanh vs relu networks

F.1 Experiments on MNIST handwritten digits

We complete our analysis of change of topology by neural networks by examining how the topology of real world data is changing within a well trained neural network. Given current technological limitations on the computation of Betti numbers for complex data, we carry out this analysis on a simplified version of MNIST hand written data set: We project the data set (70k samples) into first 50 principle component, reducing the dimension of each sample form 748 to 50. The reduced dimension of the data enable us to run homology computations while at the same time we still retain a relatively high quality images, so can still to train a neural network to high accuracy. A few examples of original digits and their projections on the first 50 dimensions are shown in Figure 19

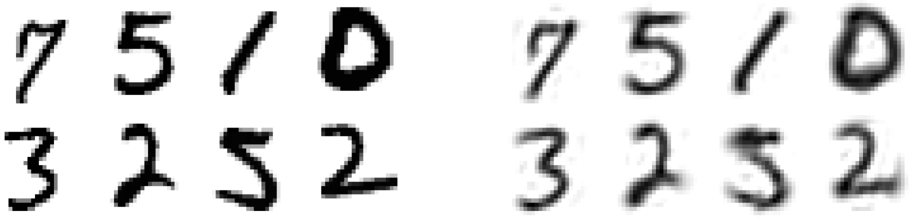


Figure 19: *left*: original MNIST handwritten digits, *right*: MNIST handwritten digits projected on the fifty principal components.

We use feedforward neural network with 10 layers, the network has 50 dimensional input and 10 neurons in the interleaving layers. The network is trained to classify digits into two categories: category a if a digit is zero or category b if it is not zero (zero digit was arbitrary picked to have binary classification rather than multi-categorical classification). We then analyze how the manifold of zeros is transformed within the network (for faster computation, we subsample $1/6$ of all zero digits to further reduce the size of the point cloud that we analyze). We compare different activation types: $\tanh$, $\mathrm{leakyrelu}$ and relu. Unlike experiments with simulated data where the topology is known, here we have no way to set the proper scale, and have to observe multiple scales. The results of those computations are given in Table 4. The results verify what we've seen in our simulated experiments: we observe topology reduction through the layers, with relu activation the topology is reduces most rapidly compared with $\mathrm{leakyrelu}$ and $\tanh$ activations. For $\tanh$ we clearly see much slower reduction in topological complexity, in fact we see that in this case the network has not reduced the topological complexity all the way down to topological disk (this might be due to a small samples size that we use in our Betti numbers estimate for MNIST data set).

activation	scale	k=0	1	2	3	4	5	6	7	8	9	10
tanh	$\varepsilon = 1.5$	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	$\varepsilon = 1.5$	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	$\varepsilon = 1.5$	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 4: Profile of $\beta_0(\nu_k(M_a)) + \beta_1(\nu_k(M_a)) + \beta_2(\nu_k(M_a)))$, at input $(k = 0)$ and layers $k = 1, \dots, 10$ with M_a the manifold of handwritten zeros. Network has fifty dimensional input, two dimensional output, and 10 neurons in intervening layers. The table is split into three parts corresponding to three activation type. For each activation 3 scales (using geodesic on a graph distance metric) are examined $\varepsilon = 1.5, 2.5, 3.5$.

G Additional topics

G.1 Visualization of topological changes in 3 dimensional networks.

In 3-dimensional setting an easy data interpolation makes those transformations standout. Let $T \subset R^3$, a cube with category domains M_a, M_b where $M_a = T \setminus M_b$. First run the affine transformation on T:

$$(1 - t_1)\nu^{(l-1)}(T) + t_1\big(W\nu^{(l-1)}(T) + b\big), \tag{5}$$

for t_1 going from 0 to 1, where $b \in \mathbb{R}^{n_l}$ and $W \in \mathbb{R}^{n_{l-1} \times n_l}$ are the bias and weight matrix of layer l, then run transformations that bend the space

$$(1 - t_2)(W\nu^{(l-1)}(T) + b) + t_2\nu^{(l)}(T) \tag{6}$$

for t_2 going from 0 to 1. For a well trained network, running this interpolation for each layer shows consecutive "foldings" of the space that lead to topology simplification. Figure 2 illustrates such foldings. A movie showing similar foldings is available at https://github.com/topnn/topnn_framework/tree/master/videos.

G.2 Complexity of arrangement

As was mentioned in the main text, observing Betti numbers for each category separately is not the whole story of what happens when data is simplified by a neural network, since this only monitors the change in complexity of each set, and does not directly measure how the complexity of their *arrangement* is changing. Here we introduce an additional measure of arrangement complexity, one that also takes the geometry into account and is able to detect when two sets are linearly separable.

At this point, this definition serves us primarily as a reference for what we kind of measure we would, ideally, like to compute, and it remains a (hard) future question of how to compute it.

Definition G.1 (complexity of arrangement of sets) *Let two sets $M, W \subset \mathbb{R}^n$ such that there exists compact connected hypersurface $D \subset \mathbb{R}^n$, such that M_a is in the inside of D, and M_B is in the outside of D and D does not intersect with either. Let $\mathcal{D}$ be the family of all such hypersurfaces. Topological complexity of arrangement of sets M_A and M_B, $\mathrm{AC}(M, W)$, is defined to be*

$$\mathrm{AC}(M, W) = \min_{D \in \mathcal{D}} \left\{ \sum_{i=1}^{n} \beta_i \left(D \cap \mathrm{Conv}(M)\right), \sum_{i=1}^{n} \beta_i \left(D \cap \mathrm{Conv}(W)\right) \right\} \tag{7}$$

Where $\mathrm{Conv}(X)$ is convex hull of the set X. In other words, the complexity is given by "the best cut" D between M and W. Where a good cut is such that $\beta_i \left(D \cap \mathrm{Conv}(M)\right)$ and $\beta_i \left(D \cap \mathrm{Conv}(W)\right)$ are small.

Observation: $\mathrm{AC}(M, W) = 0$ if and only if M and W are linearly separable.

In a situation where M_b is roughly the complement of M_a in a hypercube $T \subset \mathbb{R}^n$ so that the boundary of the cube T belongs to M_b. Then the boundary of M_a is the only separating manifold between M_a and M_b in T, and so it is the decision boundary between M_a and M_b. If we follow the prescription of equation 7 and take intersection between the decision boundary and T, the convex hull of M_b, this yields the boundary of ∂M_a:

$$\partial M_a = D \cap \mathrm{Conv}(M_b) = \partial M_a \cap T.$$

A reasonable assumption is that $\sum_i \beta_i(\partial M_a \cup \mathrm{Conv}(M_a)) \le \sum_i \beta_i(\partial M_a)$. And so under those assumptions $\mathrm{AC}(M_a, M_b)$ is exactly equals to the topological complexity of ∂M_a

$$\mathrm{AC}(M_a, M_b) = \omega(\partial M_a).$$

Therefore, in this scenario, knowing the topology of M_a gives information about topological complexity of arrangement of M_a and M_b.

H Hardware and software specifications

The framework is coded in TensorFlow version 1.12.0 on Ubuntu 16.04.1. Training is done on cross entropy categorical loss with standard Adam Kingma & Ba (2015) optimizer of TensorFlow, for up to 18000 training epochs. Learning rate set to $0.02 - 0.04$ with and exponential learning rate decay: $\eta^{t/d}$ where t is the training epoch normalized by $d = 2500$ (for training bottleneck architecture the decay was change to 4000) and $\eta = 0.5$. Topology computation were done using EIRENE package in Julia 0.6.4 Henselman & Ghrist (2016), the time required for a single computation of Betti numbers (i.e., construction of Vietoris–Rips complex on point cloud at the output of a single layer) was between a few tens of seconds (for 5 neurons wide networks of D-I) to about half an hour (50 neurons wide networks of D-III). The calculations were run in parallel over 12 cores, Intel i7-8750H@2.20GHz, Cache 9216kb, sharing 32Gb DDR4-2666MHz RAM. The jobs were fed in queue, with a single core limited to use up to 9Gb memory. For MNIST data set the computation were run using 32GB, the longest run took about 6 hours with this hardware setup.