# A Topological Understanding of Representational Power and Generalization in Deep Learning

Anonymous
University of Massachusetts Amherst
anonymous@cs.umass.edu

## Abstract

*In recent years deep learning has become incredibly popular and effective at solving a wide variety of challenging problems. However, as a community we lack full understandings of fundamental key problems in deep learning, such as knowing why modern neural network architectures generalize so effectively and understanding thoroughly the representational powers of complex network architectures, without resorting to a blowup in the number of neurons. In this paper we argue that (algebraic) topology is a key and under-researched ingredient to fundamental mathematical knowledge in deep learning by demonstrating its effectiveness at analyzing representational power with respect to activation function choice and at characterizing overfitting in neural networks.*

## 1. Introduction

Machine learning, and in particular deep learning, has advanced rapidly in recent years. Recent models contain a massive number of parameters, and are somewhat surprisingly able to generalize well to unseen testing data. Despite these practical successes though, there is a lag in the foundational theory for deep learning. [10] shows surprising results suggesting that we do not have a full view of the nature of generalization in deep learning. Meanwhile, the somewhat arbitrary art of choosing activation functions has lead researchers to automate discovery of better activation functions [7], without fully understanding what makes for a good activation function.

In this paper, we show that the field of topology, specifically algebraic topology and topological data analysis, provides interesting insights into the two key questions about representational power and generalization in deep learning. This paper does not claim to definitively answer these two fundamental questions, but rather introduces an alternative theoretical viewpoint from the perspective of topology and presents preliminary evidence that topology, specifically computational topological data analysis (TDA), may be an effective tool for understanding such fundamental questions in deep learning.

**Our contributions.** This paper presents a topological viewpoint through which to understand the ability of deep networks to classify input data, and argues for the further use of topology in deep learning research. Our primary contribution is to bring topology into the mainstream, standard way of understanding representational power and generalization. In order to motivate the use of topology, we present *two sub-contributions* as case studies in how topology can help to understand deep neural networks:

(1) Previous theoretical approaches to the representational power of neural networks are primarily concerned with growing the width or depth of a fully-connected network so as to arbitrarily accurately approximate a given function. We show that it is possible to increase the representational power of a neural network by changing the activation functions, while keeping width and depth fixed, and that this change is explained by topology.

(2) Motivated by the topological tools from the first case study, we show that using similar techniques we can characterize neural networks which are generalizing well compared to neural networks which are memorizing the training set (*i.e.* maximally overfitting). Based on this we suggest future ideas for topologically-inspired regularization techniques which are not currently computationally feasible, but are interesting future directions.

This paper is structured as follows. In Section 2 we give a high-level intuitive introduction to topology, homology theory, and topological data analysis so that this paper is self-contained for readers that are not otherwise familiar with topology. In addition, we compare our paper with related work. Section 3 discusses theoretical and experimental results in support of case study (1) outlined above. Section 4 introduces one new tool of topological data analysis and presents experimental results in support of case study (2). Section 5 provides concluding remarks on what these two case studies alone show, and future directions for incorporating topology in deep learning.

## 2. Background and Related Work

### 2.1. A Primer in Algebraic Topology

In this section we give a high-level, brief overview of key pieces of algebraic topology for readers who are unfamiliar with the concepts of homology. For a thorough introduction to topology, see a standard textbook on Algebraic Topology, such as by Hatcher [6]. In particular, we avoid complete mathematical precision for the benefit of clarity. Readers already familiar with homology may skip to Section 2.2.

**Manifolds.** The study of topology is primarily concerned with understanding the abstract "shape" of objects. For the purpose of this paper, we will restrict the "objects" we study to the case of *manifolds*. A $d$-dimensional manifold $M$ is a space which is locally indistinguishable from $d$-dimensional Euclidean space. For example, consider the *hollow* donut $M \subseteq \mathbb{R}^3$. For any point $x$ on the skin of the donut, there is a small 2-dimensional disk surrounding $x$ that lies on the skin of the donut. Thus, the hollow donut is a 2-dimensional manifold.

**Homeomorphisms.** Topology is not concerned with the precise scale, rotation, translation, *etc.* of a manifold, but only with the fundamental properties of its shape. For instance, topology views a circle and an ellipse as equivalent. This equivalence is made precise by *homeomorphisms*:

**Definition 1** *A homeomorphism between two manifolds $M_1$ and $M_2$ is a function $f : M_1 \to M_2$ such that:*

*(1) $f$ is continuous.*

*(2) $f$ is a bijection, and $f^{-1}$ is also continuous.*

If any homeomorphism $f : M_1 \to M_2$ exists, then $M_1$ and $M_2$ are *homeomorphic*, that is they are topologically equivalent. Unfortunately it is very difficult to prove directly if two manifolds are non-homeomorphic.

**Homology groups and Betti numbers.** In order to characterize which manifolds are equivalent and which are not, we introduce the powerful tool of homology groups.

Any $d$-dimensional manifold $M$ has associated with it a sequence of $d + 1$ commutative *homology groups*, denoted as $H_0(M), H_1(M), \cdots, H_d(M)$, which here will be products of either the group of integers, $\mathbb{Z}$, or the group of integers modulo some $m$, $\mathbb{Z}_m$. Roughly, the group $H_k(M)$ describes the number of $k$-dimensional "holes" present in $M$. $H_0(M)$ describes the number of connected components of $M$; $H_1(M)$ describes the 1 dimensional holes (loops) of $M$; $H_2(M)$ describes the 2 dimensional holes (enclosed volumes) of $M$; *etc.*

Table 1: Example Homology Groups

| Manifold | $H_0(M)$ | $H_1(M)$ | $H_2(M)$ | Betti #s |
|---|---|---|---|---|
| Circle | $\mathbb{Z}$ | $\mathbb{Z}$ | $0$ | $1, 1, 0$ |
| Sphere | $\mathbb{Z}$ | $0$ | $\mathbb{Z}$ | $1, 0, 1$ |
| 2 Disjoint Spheres | $\mathbb{Z} \times \mathbb{Z}$ | $0$ | $\mathbb{Z} \times \mathbb{Z}$ | $2, 0, 2$ |
| Donut | $\mathbb{Z}$ | $\mathbb{Z} \times \mathbb{Z}$ | $\mathbb{Z}$ | $1, 2, 1$ |
| Solid Donut | $\mathbb{Z}$ | $\mathbb{Z}$ | $0$ | $1, 1, 0$ |
| Klein Bottle | $\mathbb{Z}$ | $\mathbb{Z} \times \mathbb{Z}_2$ | $0$ | $1, 2, 0$ |

To work through one example, consider the Sphere: it has only one connected component, so $H_0(M) = \mathbb{Z}$, and it has 1 enclosed volume (the hollow inside of the sphere), so $H_2(M) = \mathbb{Z}$. One might be tempted to think that the sphere has a loop that goes around the sphere (the equator), but this loop can be continuously slid along the surface of the sphere until it collapses to a single point, so the sphere does not have a proper loop, and thus $H_1(M) = 0$. To build further intuition, Table 1 provides a variety of examples of the first 3 homology groups. A simplified view of the homology groups are the *Betti numbers*, which are simply the number of factors in each homology group, also shown in Table 1.

**Homology invariance.** The key result and application of homology theory is that homology groups are invariant across homeomorphisms. As a theorem:

**Theorem 1** *Suppose $M_1$ and $M_2$ are both manifolds. If $f : M_1 \to M_2$ is a homeomorphism, then $M_1$ and $M_2$ have identical homology groups.*

### 2.2. Persistent Homology

In machine learning we may often naturally think of a dataset as being drawn probabilistically from a manifold, possibly with noise. For example, we may consider the manifold of MNIST digits, which consists of the union of a sub-manifold for each of the 10 digit classes. Homology provides a powerful tool for testing if two manifolds are non-homeomorphic. However, homology relies on having complete *analytic* descriptions of manifolds. In practice we only have finitely many data samples from the manifold of *e.g.* MNIST digits, so we have no analytic description of such a manifold and cannot directly apply homology.

The basic idea is to take the finitely many sampled points and connect points that are within distance $\varepsilon$ of each other to form a triangulated mesh manifold, called a *simplicial* *complex*. For instance, Figure 1 shows the construction of simplicial complexes from a set of points sampled from two disjoint circles of different sizes for three different values of $\varepsilon$. For $\varepsilon = 0$, $H_1(M) = 0$ since neither of the circles have been fully connected. At $\varepsilon = 0.08$, the smaller circle has been connected, but the larger one has not so we get $H_1(M) = \mathbb{Z}$. Finally, $\varepsilon = 0.32$ connects the larger circle but completely fills in the interior of the smaller circle again giving $H_1(M) = \mathbb{Z}$. The key observation is that no single choice of $\varepsilon$ can give the correct result of $H_1(M) = \mathbb{Z} \times \mathbb{Z}$, but each one highlighted a different component of the correct homology.

Motivated by this example, the strategy of *persistent homology* is to compute the homology across all varying values of $\varepsilon$, and observe how the homology evolves. For some values of $\varepsilon$, such as $0.08$ and $0.32$ in Figure 1, new features are *born* (when the circles become connected), and for larger values of $\varepsilon$ features will die, like $0.32$ in Figure 1 when the smaller circle was completely filled in.

The persistent homology of a dataset is often summarized in a *persistence diagram*, an example of which for the circles from Figure 1 is shown in Figure 2. We read a persistence diagram by interpreting blue dots as features of $H_0(M)$ and orange dots as features of $H_1(M)$, with the axes representing the birth and death values of $\varepsilon$ for that feature. A dot close to the diagonal line indicates a feature that was born and then died very quickly, *i.e.* probably noise. The orange dot near the bottom left indicates an $H_1(M)$ feature was born very early, and died a short (but not very short) time later. This is the feature corresponding to the smaller of the two circles. The second orange dot is not born until later, but once it is born it takes a long time to die. This is the feature corresponding to the larger of the two circles. The diagram also shows (in blue dots) there are two fairly strong connected components, and a number of weaker connected components, which may or may not be interpreted as noise.

Note that throughout this paper, we use the Ripser.py [9] implementation to compute persistent homology.

## 2.3. Related Work

**Topological Data Analysis (TDA).** The concept of persistent homology and the closely related Mapper algorithm for dimensionality reduction and visualization are not new ideas, and were popularized largely by Carlsson in [2]. In addition to foundational theory for topological data analysis, previous work has also applied TDA to a variety of machine learning-adjacent topics. For instance, in [3] it is shown using TDA that the manifold of 3x3 patches of natural images has the topological structure of a Klein bottle.

**TDA in deep learning applications.** Generally, current TDA in machine learning is split into two purposes. The first purpose is to apply TDA to improve existing machine learning methods. One such approach focuses on defining a loss function over a persistence diagram, and then backpropagating the loss function through the persistence diagram itself back to the data [5], which allows for applications such as encouraging GANs to produce topologically coherent outputs, and regularizing based on topology of weight vectors.

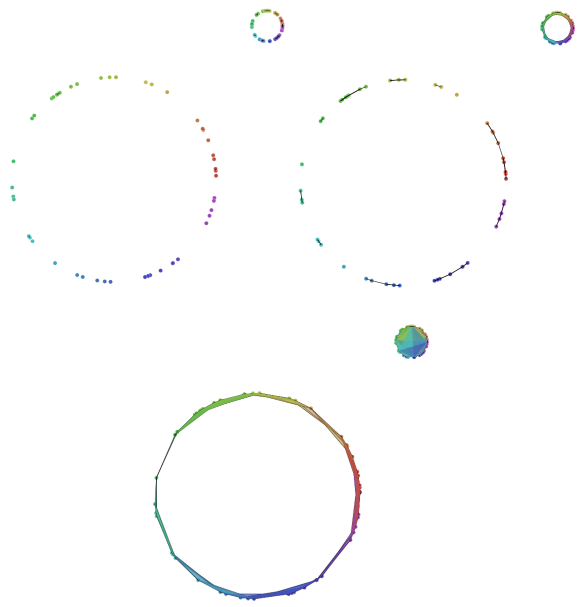
Figure 1: Simplicial Complexes using, from left to right, top to bottom, $\varepsilon = 0, 0.08, 0.32$

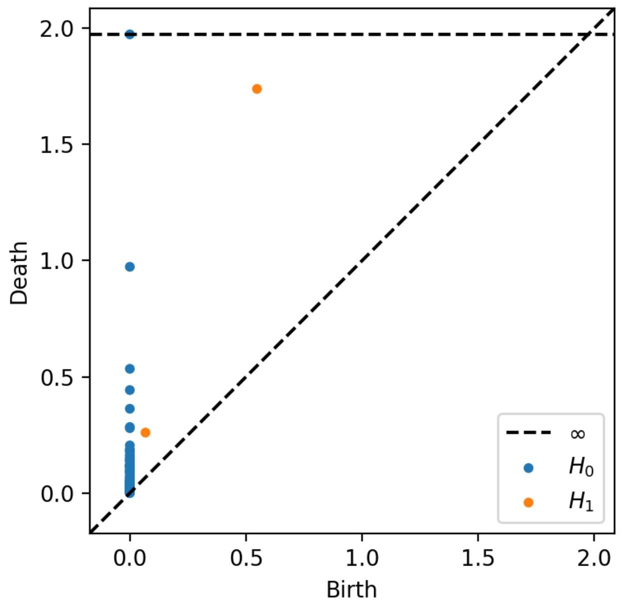

Figure 2: Persistence Diagram for circles in Figure 1

**TDA to understand deep learning.** Another purpose of TDA, and the purpose of this paper, is to use TDA to attempt to understand some fundamental questions in deep learning. One approach is to analyze the topology of the manifold of learned weights. In [4] the authors train CNNs on MNIST (and other experiments), and take all of the 3x3 filters from the trained first layer, reshape these into points in $\mathbb{R}^9$, and use TDA to analyze the topology. The authors claim to find the topology of a single circle in the first layer trained on MNIST, and more complex topologies involving three intersecting circles in CIFAR-10 experiments. However, we have carefully attempted to reproduce the MNIST results of [4], and failed to find results confirming their conclusions.

An alternative approach, and the approach taken in this paper, is to analyze the topology of the input data (mini-batch) and the topology of the subsequent hidden activations as the data is fed forward through the network. Concurrently to the development of this paper, [1] was submitted to ICRL 2020 and is currently under review (unlikely to be accepted), and presents many of the same concepts presented here. Hence, in this paper we reference some of their definitions and theorems. However, there are some important differences between the two papers. First, the authors of [1] look only at comparing ReLU activation functions to smooth activation functions such as sigmoid and tanh, whereas in this work we also analyze additional unconventional activation functions. Second, [1] only considers fully-connected networks on toy examples and MNIST, which does not necessarily extend to practically important CNN architectures. In this work we conduct some preliminary experiments on CNN architectures. Third, [1] conducts extensive experiments related to the rate of change of topological complexity progressively through multiple layers, which is a valuable contribution which we have not yet performed. Fourth, in our work we borrow an intriguing experimental technique of randomizing target labels from [10] to analyze how topology is related to generalization, rather than just looking at representational power as in [1].

## 3. Topology and Representational Power

A key theoretical question in deep learning is understanding the representational power of deep neural networks. The universal approximation properties of neural networks tell us that arbitrary functions can be approximated arbitrary well by neural networks, but that increasing the approximation accuracy may require an exponential number of neurons. We want to design neural network architectures which have sufficient representational power for the problem at hand without causing a blow up in the number of neurons. In this section we argue that a missing component to the discussion of representational power of neural networks is understanding the topology of the input data, and the topological transformations performed by the network.

Topological arguments will show that the choice of activation functions can actually have a large impact on the representational power of the neural network, which is contrary to the conventional wisdom of preferring *e.g.* ReLU over sigmoid functions for the more mundane reasons of ease of training issues such as vanishing gradients.

### 3.1. Approach

A feed-forward neural network $N$ with softmax output which has been successfully trained on a classification problem with $k$ classes is a function $N : \mathbb{R}^D \to \mathbb{R}^K$ such that an input $x \in \mathbb{R}^D$ of class $k$ is almost always mapped approximately to $N(x) \approx e_k$, where $e_k$ is the $k$-th basis vector $(0, \cdots, 1, \cdots, 0)$. A basic neural network is typically the composition of alternating affine functions and point-wise nonlinearities:

$$N = (f_n \circ L_n) \circ (f_{n-1} \circ L_{n-1}) \circ \cdots \circ (f_1 \circ L_1) \quad (1)$$

For a fully connected network, $L_i$ is given by matrix multiplication, while for CNNs $L_i$ is given by a convolution operation (which is also linear with respect to the input). The last activation function $f_n$ we assume to be softmax.

In a topological approach, we assume that our input dataset has a richer structure than just the vector space $\mathbb{R}^D$. We suppose that the input dataset is a manifold $M$ consisting of manifolds for each class: $M = M_1 \cup M_2 \cup \cdots \cup M_K$. For instance, we suppose that the MNIST dataset is a manifold consisting of 10 manifolds, one for each digit class.

In this topological view, we can now rephrase what it means for $N$ to be well-trained. We now view $N$ as a function $N : M_1 \cup \cdots \cup M_K \to \mathbb{R}^K$, and say that $N$ is well-trained if $N$ maps the manifold $M_k$ for the $k$-th class to $\{e_k\}$ (approximately).

In this section we fix the affine transformations to be fully-connected layers (so matrix multiplications plus a bias), fix the number of layers and neurons per layer, and ask how the choice of activation functions $\{f_i\}$ affects the representational power of the network.

### 3.2. Theoretical Results

Given a manifold $M = M_1 \cup M_2 \cup \cdots \cup M_K$ with non-intersecting $M_k$ we wish to see if there exists a neural network $N$ that can perfectly classify $M$, that is $N(M_k) = e_k$ In other words, if $N$ is perfectly trained and we apply $N$ to the entire input manifold we expect to obtain $N(M) = \{e_1, \cdots, e_K\}$, a cluster of many points about each basis vector. Computing the homology of $N(M)$ is trivial since it is a collection of $K$ distinct points: $H_0(N(M)) = K$, and $H_p(N(M)) = 0$ for $p \geq 1$.

Practically, $M$ has a complex homology structure with $H_p(M) \neq 0$ for some $p \geq 1$ (if $M$ has a trivial homology we could simply apply a clustering algorithm rather than a neural network), and it is not necessarily the case that $H_0(M) = K$. Thus, the homology of $M$ is distinct from the (desired) homology of $N(M)$. Combining this result with Theorem 1 gives the following theorem as a result:

**Theorem 2** *Suppose $N : M_1 \cup M_2 \cup \cdots \cup M_K \to \mathbb{R}^K$ is a neural network with the architecture from Equation 1 which perfectly classifies $M = M_1 \cup M_2 \cup \cdots \cup M_K$, and $M$ does not have the homology $H_0(M) = K, H_{p \geq 1}(M) = 0$. Then $N$ is not a homeomorphism. Furthermore, there exists at least one layer of $N$, $(f_i \circ L_i)$ that is not a homeomorphism.*

Note that Theorem 2 is essentially a slightly less specific version of Theorem C.2 presented in [1]. Theorem 2 says that if we want to built a neural network which has the representational power to perfectly classify $M$, we need to make sure at least one layer is not a homeomorphism. Referencing Definition 1, this means that for at least one layer $(f_i \circ L_i)$ we need to ensure that either the layer is discontinuous, which practically we cannot do as it would make optimization intractable, or we make the layer non-bijective. There are a few ways this can occur. We could make the affine map $L_i$ non-surjective by mapping to a higher-dimensional space, *i.e.* making the width of the layer larger. However, we are investigating how we can affect the representational power of the network *while keeping architecture fixed*, so we ignore this standard way of improving representational power. An alternative approach is to consider activation function choices for $f_i$ which are not homeomorphisms by being non-bijective. Let us analyze some choices for $f_i$.

First, we have classic choices such as sigmoid and tanh. Since both sigmoid and tanh are continuous bijections [1], in a theoretical view they do not help to increase the representational power of the network to be able to alter the topology of the input manifold. In practice by using sigmoid / tanh the network could stretch the manifold so extremely so as to be able to classify very well, but not perfectly. Such extreme stretching quickly leads to vanishing gradient problems.

Second, we have the extremely popular choice of ReLU. Unlike sigmoid / tanh, ReLU is continuous but not injective (since all non-positive inputs are mapped to 0). This allows networks that use ReLU to effectively collapse topologically complex manifolds into simpler structures. However, ReLU is only barely a non-homemorphism: it is a so called *near-diffeomorphism* [1] as (roughly) it can be approximated arbitrarily well with a homeomorphism (leaky ReLU). Practically in large networks, ReLU can be a large improvement over sigmoid / tanh as it can effectively collapse manifolds, but certain entangled manifolds it cannot (perfectly) disentangle.

[1]Formally, we need to be careful with regards to the domain and codomain

Finally, we present a very non-conventional activation function, the absolute value. Unlike ReLU, absolute value is so highly injective that it is not a *near-diffeomorphism*. For this reason we conjecture the following:

**Conjecture 1** *The absolute value activation function provides fundamentally greater representational power to the neural network to disentangle and classify manifolds.*

### 3.3. Experiment

In a toy experiment we show by demonstration that at least in a simple example Conjecture 1 holds. We show this by first generating a synthetic dataset consisting of entangled manifolds, and then training two networks, one with only ReLU, one with absolute value, and observing how their behavior differs.

**Dataset.** We generate a dataset in $\mathbb{R}^3$ by sampling 2000 points each from two hollow donuts, each of central radius 15 and tube radius 2, and rotating and translating one donut so that it is linked with the other. We assign target class 0 to the first donut and target class 1 to the second donut. Lastly, we randomly split the 4000 total points into a training set of 3000 points and a test set of 1000 points. In order to properly classify the donuts, the network will have to unlink them, which is impossible using homeomorphisms or *near-diffeomorphism* such as ReLU, as discussed above. [2]

**Models and training.** We consider two nearly identical networks. The first network is defined as:

$$\begin{aligned} N_{\mathrm{ReLU}} =& \text{softmax} \circ L_{3\to 2} \circ \\ & \text{ReLU} \circ L_{3\to 3} \circ \text{ReLU} \circ L_{3\to 3} \circ \text{ReLU} \circ L_{3\to 3} \end{aligned} \tag{2}$$

where $L_{m \to n}$ denotes a fully-connected layer from $m$ to $n$ dimensions. The second network is identical except for the first activation function:

$$\begin{aligned} N_{|\cdot|} =& \text{softmax} \circ L_{3\to 2} \circ \\ & \text{ReLU} \circ L_{3\to 3} \circ \text{ReLU} \circ L_{3\to 3} \circ |\cdot| \circ L_{3\to 3} \end{aligned} \tag{3}$$

where $|\cdot|$ denotes the (pointwise) absolute value function.

For training, all weights were randomly initialized using the default PyTorch random initialization. Both networks were trained using the entire training dataset in a single batch (*i.e.* non-stochastic), using the Adam optimizer with a learning rate of 0.03, no weight decay, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, for 800 iterations.

[2]*Homology* is not actually sufficient to show that homeomorphisms can not unlink the donuts. A more complex tool called *homotopy theory* is technically required to show that homeomorphisms can not unlink.

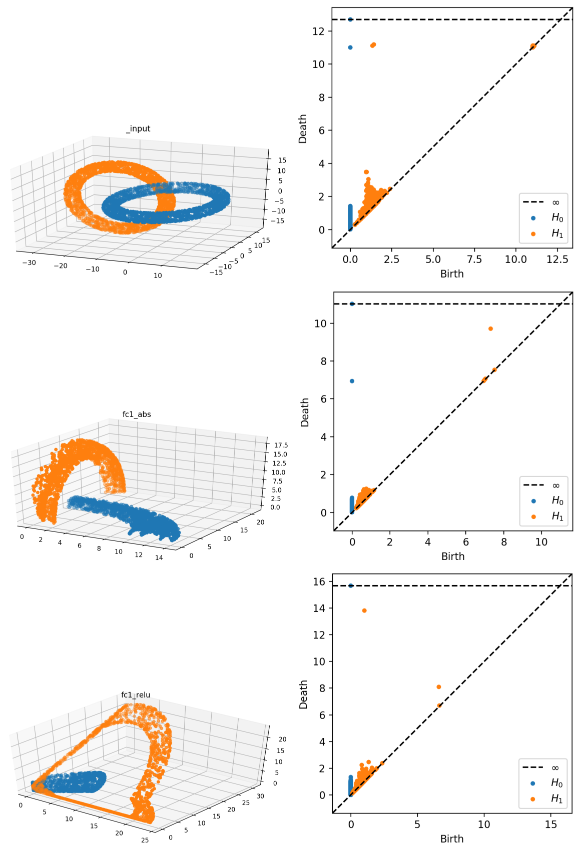

Figure 3: From top to bottom we have plots and persistence diagrams of: the input data, the first hidden activations of $N_{|\cdot|}$, and the first hidden activations of $N_{\mathrm{ReLU}}$

**Results.** $N_{\mathrm{ReLU}}$ obtains 98% accuracy on both the training and test sets. $N_{|...|}$ obtains perfect 100% accuracy on both the training and test sets (though is sensitive to random initialization, sometimes getting stuck in local optima). This suggests that introducing an absolute value activation function did indeed increase the representational power of the network such that it can disentangle the donuts. To fully understand how $N_{\mathrm{ReLU}}$ fails, we look at plots of manifolds of hidden activations of both networks, to compare. We also look at persistence diagrams of each.

The left column of Figure 3 shows a plot of the manifold of the input data, and plots of the activations of the first hidden layer (including the activation function) of $N_{|\cdot|}$ and $N_{\mathrm{ReLU}}$. The right column is the persistence diagram associated with the plot. The second row clearly shows that $N_{|\cdot|}$ is able to use the absolute value non-homeomorphism to fold the donuts such that they become disentangled. After this classification is trivial for the network. The third row shows that $N_{\mathrm{ReLU}}$ is unable to disentangle the donuts, and instead must take the strategy of stretching them as to minimize the volume that overlaps after the ReLU (the overlap can be seen in the bottom left of the plot). In addition, the persistence diagrams confirm the overlapping: in the first two rows the persistence diagrams clearly indicate that there are two connected components. However, the persistence diagram for the third row only shows one connected component, which confirms that $N_{\mathrm{ReLU}}$ squished the two donuts such that they intersect.

**Discussion.** The conclusion from this small toy experiment is *not* that absolute value should be used prevalently as an activation function. In fact, the instability of $N_{|\cdot|}$ with respect to random initialization highlights the downside that absolute value greatly increases the non-convexity of the loss function, and makes learning difficult.

Rather, the main points are that: a) *nonlinearity is not the only important aspect of an activation function when considering representational power*, and b) *the representational power of a network can be increased purely by changing activation functions*.

Of potential relevance is the recently introduced Swish activation function , which significantly outperforms ReLU on a variety of tasks [7]. However, it is unknown exactly why Swish is able to offer performance improvements. The authors hint that experimentally, “the non-monotonic bump is an important aspect of Swish” [7]. The non-monotonic bump actually means that Swish is, like absolute value, not a *near-diffeomorphism* (though certainly not as drastic as absolute value), and thus a plausible conjecture is that the performance improvements are at least partially due to its ability to increase the power of neural networks to modify the topology of activation manifolds. Evaluating this conjecture is far beyond the scope of this work.

## 4. Topology and Overfitting

In the previous section we looked at how topology can be used to shed theoretical and experimental light on the fundamental topic of representational power in deep learning. In this section we turn to a different case study of considering how topology can help to inform our understanding of generalization of neural networks.

[10] introduced a fascinating experimental approach to try to understand generalization. The authors started with a standard dataset $M$, say MNIST, and easily trained a standard deep CNN model on the dataset which achieves very high training and test set performance. However, they also produced an alternate dataset $M'$ which consisted of the same input images of $M$, but each image was assigned an entirely random target label. The authors then trained the exact same model on $M'$, and found that the model is able to nearly perfectly memorize the training set, while of course performing no better than random on the test set. From these experiments, [10] showed that modern neural

networks easily have the ability to massively overfit a training set, but that well-designed and trained models are able to avoid the problem of overfitting, and that regularization techniques such as $l_2$ regularization and dropout account for only a small piece of this effect.

Understanding the nature of overfitting and how to prevent it is a necessary avenue for improving the field of deep learning. Here, we aim to see if topology can be used to characterize a neural network which is overfit from a neural network which generalizes well.

## 4.1. Approach

We follow the experimental approach of [10] with MNIST. Specifically, we take the original MNIST dataset $M$, and produce another dataset $M'$ by randomly permuting all of the target labels. We then train two identical models $N$ and $N'$ on datasets $M$ and $M'$, respectively, such that both $N$ and $N'$ achieve high performance on the training set, but only $N$ achieves high performance on the test set. Of course $N'$ can do no better than random on the test set, because the test is truly random labels. In addition, we use the same training procedure for both networks

Once these networks are fully trained, we then proceed to analyze the topological properties of the networks, and report interesting results.

## 4.2. Experiment

**Dataset.** The dataset $M$ consists of the standard MNIST dataset available with PyTorch with the default PyTorch train / test split of 60000 / 10000. The dataset $M'$ consists of exactly the images in $M$, but with all 70000 target labels randomly permuted.

**Models and training.** The primary model we look at is a standard CNN architecture of the following form:

$$\begin{aligned} \text{softmax} \circ L_{256\to 10} \circ \text{ReLU} \circ L_{3200\to 256} \\ \circ \text{MP}^2 \circ \text{ReLU} \circ C^3_{256\to 128} \\ \circ \text{MP}^2 \circ \text{ReLU} \circ C^3_{1\to 256} \end{aligned} \tag{4}$$

where $C^k_{n\to m}$ denotes a convolution layer of kernel size $k \times k$ with $n$ input channels and $m$ output channels, and MP$^k$ denotes a max pooling layer with kernel size $k \times k$.

In both of the training regimes (normal and randomized labels), the models are trained with identical optimization routines. Specifically, all weights were randomly initialized with the default PyTorch random initialization, and were trained with a batch size of 64 using the Adam optimizer with a learning rate of $5 \times 10^{-5}$, no weight decay, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. For the normal label regime trained succeeded fast enough that we stopped training at 20 epochs, and for the randomized label regime we trained for 100 epochs.

**Training results.** The model $N$ trained on the normal data $M$ obtained a training accuracy of 99.57% and a test accuracy of $98.77\%$. In other words, $N$ trains and generalizes very effectively. The model $N'$ trained on the data $M'$ with random labels obtained a training accuracy of 88.25% and a test accuracy of 9.62%, showing that it manages to memorize most of the training set, while predictably performing no better than random on the test set.

**Topology analysis.** We now wish to observe the evolution of the topology of the training set of $M$ as it is fed through the layers of each network. Unfortunately, we cannot directly plot the hidden layer activations as we did for the toy example in Figure 3, as they are extremely high-dimensional. Instead, we must rely on the tools of topological data analysis. Persistence diagrams are a viable tool, but with noisy data such as this they can become hard to interpret. We will instead focus on the Mapper algorithm [8] which essentially allows for visualization of topological structures by reducing a high-dimensional manifold to a graph representation. In addition, to further simplify the visualization process, we only visualize digits that are assigned labels (either real or random) of 1 and 2, though this technique generalizes to more classes.

We analyze the topology of the final two layer activations of $N$ on the normal data, working backwards. The top of Figure 4 shows a visualization of the final softmax outputs of $N$. There are three main clusters, which consist of 1s, 2s with loops, and 2s without loops. Then if we move back one layer, the bottom of Figure 4 shows a visualization of the activations of the final hidden ReLU layer of $N$. There are two main components, which are barely connected. The long component consists of 1s, while the blob component consists of 2s. This shows that the final layer sitting between these two activations needs to do very little work in $N$, as the two classes are already mostly topologically separated.

Moving to $N'$, we look at the same two layers in Figure 5. For the softmax outputs, there are two components (two long tails) of two (randomly assigned classes) that are loosely connected by misclassified inputs. However, for the hidden ReLU layer activations, we see that the randomly assigned labels *do not* topologically separate the activations, and instead a point's location in the manifold is determined by high level digit features, such as loops and rotation. This implies that the earlier convolution layers are not responsible for performing the overfit memorization of the training data, but that the responsibility of the memorization lies in the later fully-connected layers.

Critically, this suggests convolutional layers are capable of expressing the fairly mild topological transformations necessary to classify by digits, but that they are not so capable of the massive topological contortions necessary in ex-

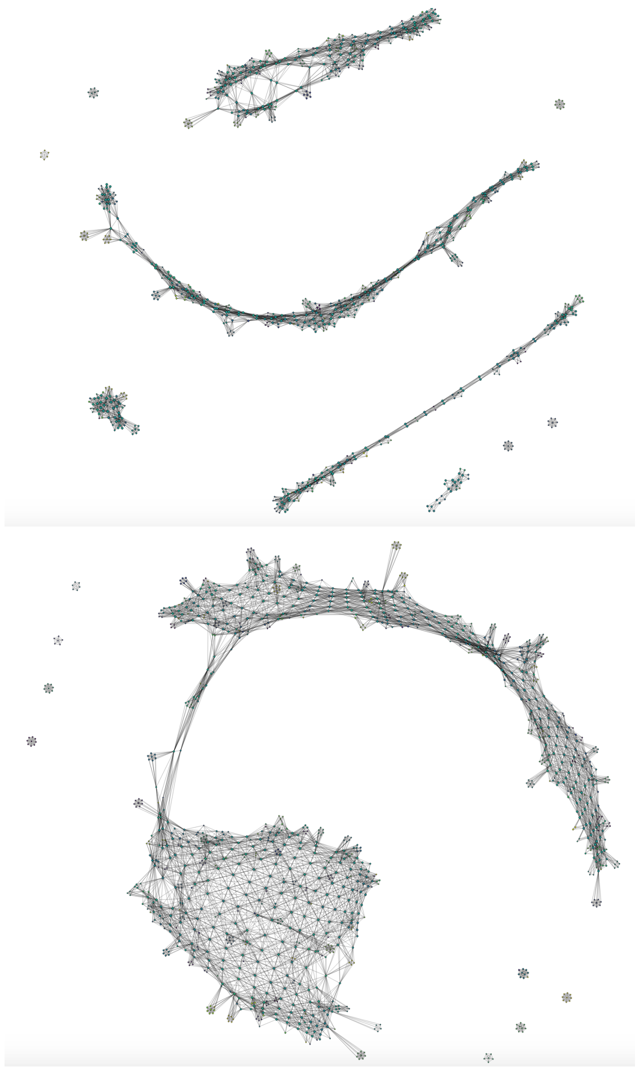

Figure 4: From top to bottom we have Mapper visualizations of the final softmax output of $N$, and the activations of the final (fully-connected) layer of $N$

treme cases of overfitting. In future work we can quantify this distinction by considering a distance function between persistence diagrams: there should be a fairly small distance between the persistence of these two layers of $N$, but quite a large distance for $N'$. By backpropagating through this loss as in [5] it may be possible to use this as a regularizer, however currently it will be computationally intractable.

## 5. Conclusion

In this paper we have argued for a greater use of topology in understanding fundamental questions in deep learning. To this end, we presented two case studies of how topology can inform both results in representational power and in generalization capabilities. The first case study suggested that incorporating topological information, namely whether activation functions are homeomorphisms, is an important aspect to the field of theoretical results in representational power of neural networks. The second case study suggested that overfitting may be able to be characterized by topological information extracted from hidden activations. Neither of these case studies are conclusive results themselves, but point to a bright future of topology in deep learning.

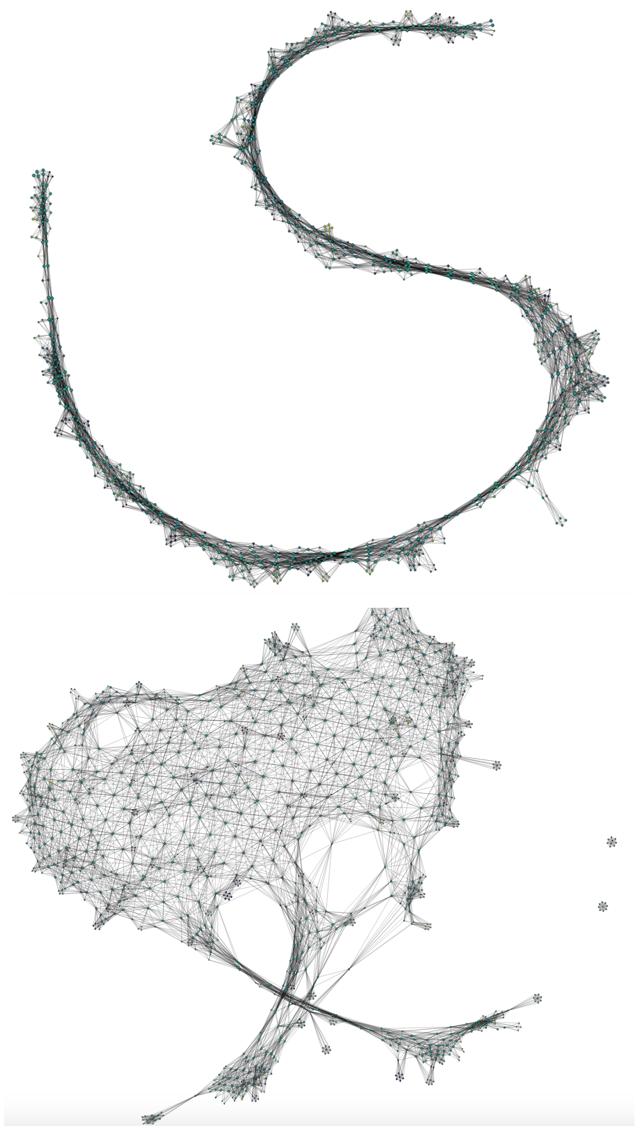

Figure 5: These are Mapper visualizations of the same layers as in Figure 4, but for $N'$ instead.

## References

[1] Anonymous. Topology of deep neural networks. *ICLR*, 2020.

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

[3] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76(1):1–12, Jan 2008.

[4] R. B. Gabrielsson and G. E. Carlsson. A look at the topology of convolutional neural networks. *CoRR*, abs/1810.03234, 2018.

[5] R. B. Gabrielsson, B. J. Nelson, A. Dwaraknath, P. Skraba, L. J. Guibas, and G. E. Carlsson. A topology layer for machine learning. *CoRR*, abs/1905.12200, 2019.

[6] A. Hatcher. *Algebraic Topology*.

[7] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions, 2017.

[8] G. Singh, F. Mémoli, and G. Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. *Eurographics Symposium on Point-Based Graphics*, 2007.

[9] C. Tralie, N. Saul, and R. Bar-On. Ripser.py: A lean persistent homology library for python. *The Journal of Open Source Software*, 3(29):925, Sep 2018.

[10] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization, 2016.