A Topology Layer for Machine Learning

Rickard Brüel-Gabrielsson
Stanford University, UnboxAI

Bradley J. Nelson
Stanford University

Anjan Dwaraknath
Stanford University

Primoz Skraba
Queen Mary University of London

Leonidas J. Guibas
Stanford University

Gunnar Carlsson
Stanford University, UnboxAI

Abstract

Topology applied to real world data using persistent homology has started to find applications within machine learning, including deep learning. We present a differentiable topology layer that computes persistent homology based on level set filtrations and edge-based filtrations. We present three novel applications: the topological layer can (i) regularize data reconstruction or the weights of machine learning models, (ii) construct a loss on the output of a deep generative network to incorporate topological priors, and (iii) perform topological adversarial attacks on deep networks trained with persistence features. The code[1] is publicly available and we hope its availability will facilitate the use of persistent homology in deep learning and other gradient based applications.

1 Introduction

Persistent homology, or simply persistence, is a well-established tool in applied and computational topology. In a deep learning setting, persistence has mainly been used as preprocessing to provide topological features for learning (Seng Pun, Xia, and Xian Lee, 2018; Giansiracusa, Giansiracusa, and Moon, 2017; Hofer and al, 2017). There has been work that uses differentiable properties of persistence to incorporate topological information in deep learning (Clough et al., 2019; Liu, Jeng, and Yang, 2016) and regularization (Chen, Ni, et al., 2019); however, such work has focused on specialized applications and specific functions of the persistence diagrams. Another line of work uses applied topology and persistence for deep learning interpretability (Brüel-Gabrielsson and Carlsson, 2018; Gabella et al., 2019), automating the construction of deep learning architectures (Carlsson and Brüel-Gabrielsson, 2018), complexity measures (Guss and Salakhutdinov, 2018; Rieck et al., 2018), and adversarial attacks (Gebhart and Schrater, 2017; Gebhart and Schrater, 2019). Persistence fits naturally in geometric problems and has been applied to a number of geometry processing applications including shape matching (Carlsson, Zomorodian, et al., 2005), optimal pose-matching (Dey et al., 2010), shape segmentation (Skraba, Ovsjanikov, et al., 2010), and surface reconstruction (Brüel-Gabrielsson, Ganapathi-Subramanian, et al., 2019). In (Brüel-Gabrielsson, Ganapathi-Subramanian, et al., 2019) gradient descent was successfully applied to persistence-based optimization. In this work, we consider how gradient descent through persistence may be used more broadly and flexibly than in the specialized applications cited above. As we will see, persistent homology is easily tailored to incorporate topological information into a variety of machine learning problems.

In many deep learning settings there is a natural topological perspective, including images and for 3D data such as point clouds or voxel spaces. In fact, many of the failure cases of generative models are topological in nature (Wu et al., 2016; Goodfellow, Pouget-Abadie, et al., 2014). We show how topological priors can be used to improve such models. It has been speculated (Brüel-Gabrielsson and Carlsson, 2018; Carlsson and Brüel-Gabrielsson, 2018) that models that rely on topological features might have desirable properties besides test accuracy; one such property has been robustness against adversarial attacks (Chakraborty et al., 2018). However, to our knowledge, no such attacks have been conducted. With our layer, such attacks are easy to implement, and we provide illustrative exam-

[1] www.github.com/bruel-gabrielsson/TopologyLayer

Proceedings of the 23$^{\text{rd}}$International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

ples. As a language for describing global properties of data, topology is also useful in exploring properties of generalization. There are some natural measures of topological simplicity (akin to norm regularization) and we show how these can successfully be used to regularize the parameters or weights of machine learning models. Our contributions include: (i) an easy-to-use persistence layer for level set filtrations and edge-based filtrations, (ii) the first regularization using persistence directly on the weights of machine learning models, (iii) the first incorporation of topological priors in deep generative networks in image and 3D data settings, and (iv) the first topological adversarial attacks.

2 Topological Preliminaries

This section contains a review of the relevant topological notions, including persistent homology (persistence), providing an intuitive idea with some additional details provided in the supplementary material. For readers who are unfamiliar with persistence, we refer the reader to a number of excellent introductions and surveys which are available (Edelsbrunner and Harer, 2010) and (Poulenard, Skraba, and Ovsjanikov, 2018; Brüel-Gabrielsson, Ganapathi-Subramanian, et al., 2019) for related work on optimizing over persistence diagrams.

Topological spaces can generally be encoded using cell complexes, which consist of k-dimensional balls, or cells, $(k = 0, 1, 2, \dots)$, and boundary maps from cells in dimension k to cells in dimension $k-1$. Practically, it is convenient to work with simplicial complexes, in which k-dimensional cells are k-dimensional simplices, because boundary maps are determined automatically, but we will continue this section by discussing more general cell complexes. We will assume that the complexes are finite which ensures various technical conditions necessary for persistence (Edelsbrunner and Harer, 2010).

Homology is an *algebraic invariant* of a topological space, associating a vector space H_k to the kth dimension of a cell complex $\mathcal{X}$. Homology is computed by forming a chain complex of $\mathcal{X}$, consisting of vector spaces $C_k(\mathcal{X})$ which are freely generated by the k-cells of $\mathcal{X}$, and boundary maps $\partial_k : C_k(\mathcal{X}) \to C_{k-1}(\mathcal{X})$ which satisfy $\partial_{k-1} \circ \partial_k = 0$. In the case of dimension 0, $\partial_0 = 0$. As a notational convenience, we will use the same symbol for a k-cell $\sigma \in \mathcal{X}$ and the associated basis vector $\sigma \in C_k(\mathcal{X})$. The vector spaces $C_k(\mathcal{X})$ may be over any field, but for the purposes of determining kernels and images of maps exactly finite fields are preferred – in practice we use the finite field with two elements, $\mathbb{Z}/2\mathbb{Z}$. Homology in dimension k is defined as the quotient vector space

$$H_k(\mathcal{X}) = \ker \partial_k / \operatorname{im} \partial_{k+1}$$

An element of $H_k(\mathcal{X})$ is called a homology (equivalence) class, and a choice of representative for a class is called a generator. The dimension of $H_k(\mathcal{X})$ counts the number of k-dimensional features of $\mathcal{X}$. For example, $\dim H_0(\mathcal{X})$ counts the number of connected components, $\dim H_1(\mathcal{X})$ counts the number of holes, and so on. Homology is *homotopy invariant*, meaning that continuous deformations of $\mathcal{X}$ produce the same result.

Persistent homology studies how homology changes over an increasing sequence of complexes $\mathcal{X}_0 \subseteq \mathcal{X}_1 \subseteq \dots \subseteq \mathcal{X}_n = \mathcal{X}$, also called a filtration on $\mathcal{X}$. We consider sublevel set filtrations of a function $f : \mathcal{X} \to \mathbb{R}$. The filtration is defined by increasing the parameter α, with $\mathcal{X}_\alpha = f^{-1}(-\infty, \alpha]$, and the only requirement is that $\mathcal{X}_\alpha$ be a valid cell complex, meaning if a cell is in $\mathcal{X}_\alpha$, its boundary must also be in $\mathcal{X}_\alpha$. We first consider a filtration where cells have a strict ordering, meaning they are added one at a time. The addition of a k-dimensional cell σ at parameter i can have two outcomes. First, if $\partial_k \sigma$ is already in $\operatorname{im} \partial_k$ (meaning $\partial_k \sigma = \partial_k w$ for some $w \in C_k(\mathcal{X}_i \setminus \sigma)$), then $w - \sigma \in \ker \partial_k$. Since the kernel expands by one dimension, the quotient $H_k = \ker \partial_k / \operatorname{im} \partial_{k+1}$ expands by one dimension, and $w - \sigma$ generates the new homology class. The second possibility is that $\partial_k \sigma$ is not already in $\operatorname{im} \partial_k$, which means $\operatorname{im} \partial_k$ expands by one dimension. Because $\partial_{k-1} \circ \partial_k = 0$, $\partial_k \sigma \in \ker \partial_{k-1}$, and previously generated a homology class. Thus, the quotient $H_{k-1} = \ker \partial_{k-1} / \operatorname{im} \partial_k$ will have one fewer dimension, and $\partial_k \sigma$ is a generator for the removed class. In summary, every cell in the filtration either *creates* or *destroys* homology when it appears. The full information about how homology is born and dies over the filtration can be represented as a multi-set of pairs (b, d) where b is the birth parameter of a homology class, and d is the death parameter of that class ($d = \infty$ if it is still present in $\mathcal{X}$). This multiset of pairs for homology in dimension k is known as the k-dimensional persistence diagram of the filtration, $\mathrm{PD}_k(\mathcal{X}_\alpha) = \{(b_i, d_i)\}_{i \in \mathcal{I}_k}$, or the k-dimensional barcode of $\mathcal{X}_\alpha$. As a notational convenience, we will order the indexing of points by decreasing lifetimes i.e. $d_i - b_i \geq d_j - b_j$ for $i < j$.

As persistence diagrams are a collection of points in $\mathbb{R}^2$, there are many notions of distances between diagrams and cost functions on diagrams which depend on the points. We use loss functions that can be expressed in terms of three parameters

$$\mathcal{E}(p, q, i_0; \mathrm{PD}_k) = \sum_{i=i_0}^{|\mathcal{I}_k|} |d_i - b_i|^p \left(\tfrac{d_i + b_i}{2}\right)^q \qquad (1)$$

The parameters p and q define a polynomial function, following those introduced in (Adcock, Carlsson, and Carlsson, 2016). We sum over lifetimes beginning with the i_0 most persistent point in the diagram. Varying

i_0 for PD_k varies the number of k-dimensional features that are not penalized. For example, if $i_0 = 2$, with PD_0 we consider all but the most persistent class, promoting a one connected component. Alternatively, using $i_0 = 2$ with PD_1 will promote one hole. The parameter p can be increased to more strongly penalize the most persistent features, and the parameter q serves to weight features that are prominent later in the filtration. We also use the Wasserstein distance between diagrams – this is defined as the optimal transport distance between the points of the two diagrams. One technicality is that the two diagrams may have different cardinalities, and points may be mapped to the diagonal – see the supplementary material for details.

Differentiation: Given an input filtration $f : \mathcal{X} \to \mathbb{R}$, we can compute the gradient of a functional of a persistence diagram $\mathcal{E}(\mathrm{PD}_k)$. The key is to note that each birth-death pair can be mapped to the cells that respectively created and destroyed the homology class, defining an inverse map

$$\pi_f(k) : \{b_i, d_i\}_{i \in \mathcal{I}_k} \to (\sigma, \tau). \tag{2}$$

In the case where the ordering on cells is strict, as we previously discussed, the map is unique, and we obtain

$$\frac{\partial \mathcal{E}}{\partial \sigma} = \sum_{i \in I_k} \frac{\partial \mathcal{E}}{\partial b_i} \mathbb{I}_{\pi_f(k)(b_i)=\sigma} + \sum_{i \in I_k} \frac{\partial \mathcal{E}}{\partial d_i} \mathbb{I}_{\pi_f(k)(d_i)=\sigma} \tag{3}$$

in which at most one term will have a non-zero indicator. As we will see, many filtrations do not give rise to a strict ordering, because multiple cells can appear at the same parameter value in the filtration. While the persistence diagram is still well-defined, the inverse map 2 may no longer be unique. This can be resolved by extending the total order to a strict order either deterministically or randomly – see (Skraba, Thoppe, and Yogeshwaran, 2017) for a formal proof and description of how this can be done. As a result, $\partial \mathcal{E} / \partial \sigma$ should generally be considered as a subgradient, and a choice of strict ordering selects an element in the subgradient.

Filtrations: While general filtrations could be considered for optimization, we will focus on two different kinds of filtrations that are defined by either points or edges in a complex. For simplicity, we now use simplicial complexes, where each cell is a simplex $(v_0, \ldots, v_k)$, where each (v_j) is a 0-cell (point). We will use the subscript notation σ_i to denote the i-skeleton of a simplex, which consists of the i-dimensional faces. For instance $\sigma_0 = \{(v_j) \mid v_j \in (v_0, \ldots, v_k) = \sigma\}$, and σ_1 consists of all $\binom{k}{2}$ pairs of 0-cells.

First, we consider extensions of filtrations on 0-cells, also known as lower-star filtrations. In particular for sublevel set filtrations, $f((v_0, \ldots, v_k)) = \max_{i=0,\ldots,k} f((v_i))$. This construction is useful for building filtrations on images, where we take $\mathcal{X}$ to be a triangulation of a rectangle with 0-cells defined by the grid of pixels on the image, and $f((v_i))$ is the intensity of a color channel at the pixel (v_i).

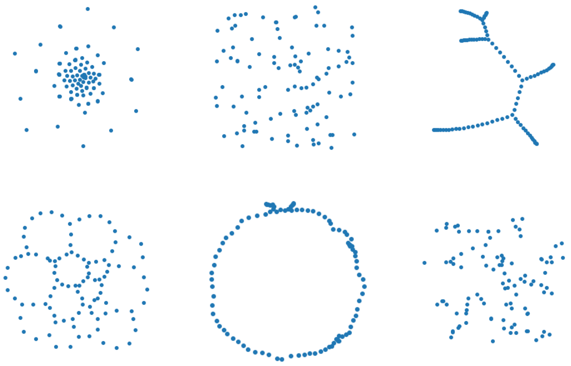

Figure 1: weak Alpha filtrations. Top center: points sampled uniformly from the unit square, Top left: Optimizing to increase $\mathcal{E}(2,0,2;\mathrm{PD}_0)$, Top right: Optimizing to decrease $\mathcal{E}(2,0,2;\mathrm{PD}_0)$. Bottom Left: Optimizing to increase $\mathcal{E}(2,0,1;\mathrm{PD}_1)$, Bottom Right: optimizing to decrease $\mathcal{E}(2,0,1;\mathrm{PD}_1)$, Bottom center: optimizing to increase $\mathcal{E}(2,1,1;\mathrm{PD}_1)$, decrease $\mathcal{E}(2,0,2;\mathrm{PD}_0)$

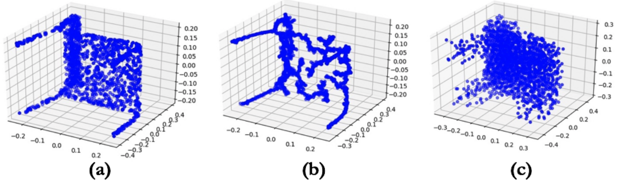

Figure 2: Rips filtrations. (a) points sampled from a chair (b) optimized to decrease $\mathcal{E}(1,0,2;\mathrm{PD}_0)$, (c) increase $\mathcal{E}(1,0,2;\mathrm{PD}_0)$.

The second kind of filtration that we consider extends a filtration on the edges of a complex, also called a flag filtration. For sublevel set filtrations, this has the form $f((v_0, \ldots, v_k)) = \max_{i<j \in 0,\ldots,k} f((v_i, v_j))$ One example of this is based on pairwise distances of points. The Vietoris-Rips, or Rips, filtration, $\mathcal{R}_\alpha$, is the distance-based flag filtration on the clique complex, consisting of all 2^n possible simplices on the vertex set. Even when limiting the space to simplices below a certain dimension, the Rips filtration can become too large to compute with efficiently. A more tractable complex in low dimensional euclidean space uses the Delaunay triangulation of a point cloud as the underlying space. We refer to the distance-based flag filtration on this space as the *weak Alpha filtration.*

Computation: We have implemented a PyTorch (Paszke et al., 2017) extension that performs the described differentiation through persistence diagrams, supporting several standard algorithms written in C++.

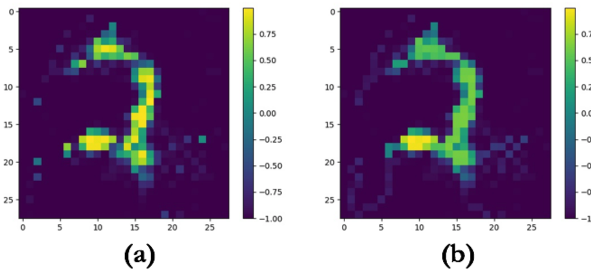

Figure 3: (a) A noisy image of the digit ‘2’. (b) after promoting a single local maximum using $\mathcal{E}(1, 0, 2; \mathrm{PD}_0)$.

Penalty	L1	L2	TV
Def.	$\|\beta\|_1$	$\|\beta\|_2$	$\|\nabla\beta\|_1$
Penalty	TV2	Top1	Top2
Def.	$\|\nabla\beta\|_2$	$\mathcal{E}(1, 0, 2; \mathrm{PD}_0)$	$\mathcal{E}(1, 0, 4; \mathrm{PD}_0)$

Figure 4: Abbreviation for penalties used for regularization of least squares problems.

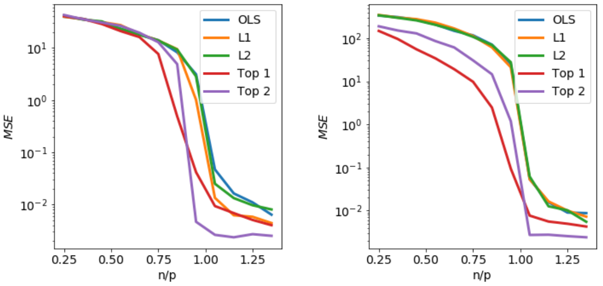

Figure 5: MSE of $\hat{\beta}$ obtained using several regularization schemes as size of training set increases. Left: entries of β_* are drawn i.i.d. from $\{-1, 0, 1\}$. Right: entries of β_* are drawn i.i.d. from $\{1, 2, 3\}$.

The actual method used to compute persistent homology is largely irrelevant for our purposes, as long as we are able to map points in the persistence diagram back to filtration values of individual cells. While our implementation does not rely on external topology libraries, many existing packages could potentially be used or modified to provide the required information. The original persistence algorithm (Zomorodian and Carlsson, 2005) as well as the cohomology algorithm (deSilva, Morozov, and Vejdemo-Johansson, 2011) are based on putting the boundary matrices ∂_k in a form which reveals the birth-death pairs. The worst case complexity is known to be equivalent to matrix multiplication in the number of simplices (Milosavljević, Morozov, and Skraba, 2011), although sparsity of ∂_k typically renders this bound pessimistic. There are many approaches to speeding up calculations in practice (Otter et al., 2017), and if only zero-dimensional homology is of interest, then the union-find algorithm (Cormen et al., 2009) typically performs faster. The dependence of number of simplices on the number of points n depends on the construction. For example, the Alpha complex may have $O(n^{d/2})$ simplices where d is the ambient dimension (Edelsbrunner, 2010), whereas the Rips complex may have as many as $O(n^{k+1})$ simplices where k is the maximal dimension homology we consider. However, in practice, the resulting complexes are approximately linear in n for small d and k. We note that many improvements to the persistence algorithm have been made with the goal of tackling larger spaces. In contrast, we seek to compute persistence using different filtrations on the same small- to medium-sized space rapidly, which may find different optimizations beneficial, although these considerations are beyond the scope of this work.

3 Applications

3.1 Topological Noise Reduction and Regularization

We first demonstrate how functions of persistence diagrams can be effectively used for both optimization of the placement of points, and optimization of functions on a space. We show how one can encourage the formation of lines, clusters, or holes in a set of points using geometric filtrations. We then show how level set filtrations can be used effectively for regularization of parameters in a model by penalizing the number number of local maxima in the parameter topology. While we see there is some benefit to regularization using an appropriate topological penalty, we do not claim superiority to other regularization schemes. Instead we wish to draw attention to the flexibility of topological penalties in both the point cloud and image settings.

In Section 1, we reviewed several applications which use specific topological loss functions. There are many possible losses which may be considered, and here we demonstrate some behaviors that can be promoted using persistence. In Figure 1, we see how a set of 100 random points in the unit square can be moved into different configurations by taking gradients of different functions of weak Alpha persistence diagrams. In Figure 2 we see how points that are sampled from a 3D chair can be moved around using similar functions of Rips persistence diagrams. An analysis of the optimality of one choice over another in any given situation is beyond the scope of this work. We primarily wish to draw attention to the wide variety of behaviors that can be encouraged by varying the choice of function.

Direct optimization on the filtration is not limited to geometric complexes. In Figure 3, we optimize functions on a space. As we will see in Section 3.2, limiting the number of local maxima in an image can improve

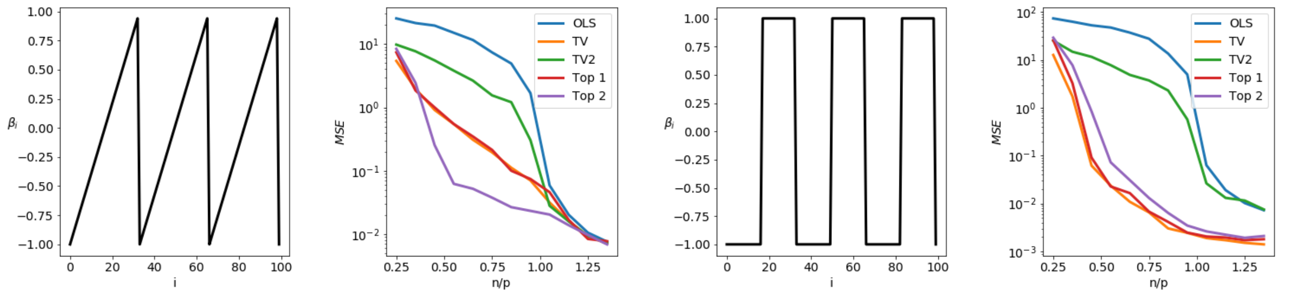

Figure 6: Left to right: Sawtooth β_*. MSE of linear prediction using $\hat{\beta}$ obtained from several regularization schemes as size of training set increases. Boxcar β_*. MSE of linear prediction using $\hat{\beta}$ obtained from same regularization schemes as size of training set increases.

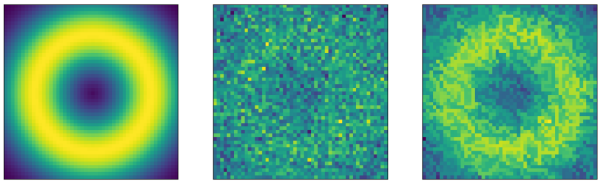

Figure 7: Left: Original image with pixel values in $[0, 1]$. Data is generated with $n/p = 0.5$, with i.i.d. Gaussian noise. Center: ordinary least squares solution. Right: least squares solution with topological penalty $\mathcal{E}(1, 0, 2; \mathrm{PD}_0) + \mathcal{E}(1, 0, 2; \mathrm{PD}_1)$.

the visual quality of generated digits. In this example, we perform optimization directly on the superlevel sets of a noisy image to produce a single global maximum.

While these examples are illustrative, we wish to see how we can use topology directly in a machine learning model for the purposes of regularization, or encoding a prior on some topological structure.

Regularization is used throughout machine learning to prevent over-fitting, or to solve ill-posed problems. In a typical problem, we observe data $\{X_i\}$ and responses $\{y_i\}$, and would like to fit a predictive model with parameters $\hat{\beta}$ that will allow us to make a prediction $\hat{y}_i = f(\hat{\beta}; X_i)$ for each observation. The quality of the model is assessed by a loss function ℓ, such as the mean squared error. However, many models are prone to over-fitting or are ill-posed if there are more unknown parameters than observations, and adding a regularization term $P(\beta)$ can be beneficial. The estimated value of $\hat{\beta}$ for the model becomes

$$\hat{\beta} = \operatorname*{argmin}_{\beta} \sum_{i=1}^{n} \ell\big(y_i, f(\beta; X_i)\big) + \lambda P(\beta) \tag{4}$$

where λ is a free tuning parameter.

Well-known examples of regularization include L_1 regularization $P(\beta) = \|\beta\|_1$ (Lasso) (Tibshirani, 1996), which promotes sparsity, or L_2 regularization $P(\beta) = \|\beta\|_2$ (Ridge regression) (Hoerl and Kennard, 1970) which tends to keep parameters from growing excessively large. Both of these types of regularization can be viewed as making the topological statement that parameter weights should "cluster" around zero, and a similar topological penalty might simply encourage the set of all weights to form clusters by penalizing the sum of lengths of PD_0 from a Rips or weak Alpha filtration on the weights.

Another class of well-known regularization schemes make an assumption about the topology of the set of parameters themselves, and penalize properties of the weights as a function on that space. Examples include penalties on a norm of a finite-difference derivative, such as total variation regularization $P(\beta) = \|\nabla\beta\|_1$ (Rudin, Osher, and Fatemi, 1992), or penalties on the ordering of weights as seen in isotonic regression and its variants (Tibshirani, Hoefling, and Tibshirani, 2011). From the topological point of view, these regularization schemes encourage β to have fewer local maxima and minima, which might be accomplished by penalizing the sum of lengths of PD_0 from a level set filtration.

In Figures 5 and 6, we compare different regularization schemes for several different linear regression problems. Examples are generated according to $y_i = X_i\beta_* + \epsilon_i$, with $X_i \sim N(0, I)$, and $\epsilon_i \sim N(0, 0.05)$. β_* is a feature vector with $p = 100$ features, and an estimate $\hat{\beta}$ is made from n samples by solving Equation 4 with the mean-squared error loss $\ell\big(y_i, f(\beta; X_i)\big) = (y_i - X_i\beta)^2$ using different penalties, and λ is chosen from a logarithmically spaced grid on $[10^{-4}, 10^1]$ via cross-validation for each penalty. We track the mean-squared prediction error for the estimate $\hat{\beta}$ as the number of samples is increased. We also compare to the ordinary least-squares solution, using the smallest 2-norm solution if the problems is under-determined $(n < p)$.

In Figure 5, β_* are chosen uniformly at random from

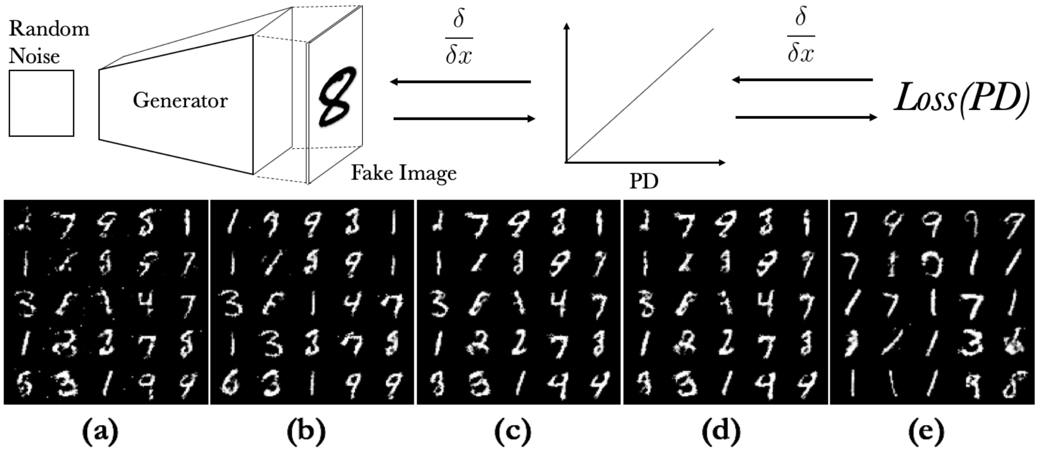

Figure 8: Top: Setup for computing topology loss and backpropogation. (a) Baseline-Generator. (b) Minimize the topology loss with respect to the latent space of Baseline-Generator. (c) Topology-Generator. (d) Train Baseline-Generator with topology-discriminator for 100 batch iterations. (e) Train Baseline-Generator in original GAN-setup for another 60,000 batch iterations.

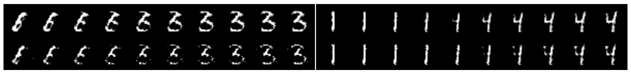

Figure 9: Linear interpolation in latent space. Top row: Topology-Generator. Bottom row: Baseline-Generator.

three different values. On the left, those values are $\{-1, 0, 1\}$, and on the right, $\{1, 2, 3\}$. We consider L_1 and L_2 penalties, as well as two topological penalties using a weak-alpha filtration. The first is $\mathcal{E}(1, 0, 2; \mathrm{PD}_0)$, and the second is $\mathcal{E}(1, 0, 4; \mathrm{PD}_0)$. Both topological penalties are non-negative, and the first penalty is non-zero if β takes more than a single value, and the second penalty is non-zero if β takes more than three distinct values, explicitly encoding that we expect three clusters. In the case where β_* takes values in $\{-1, 0, 1\}$, the L_1 and L_2 penalties slightly outperform ordinary least squares, because while β_* is not truly sparse, some shrinkage seems beneficial. In the case where β_* takes values in $\{1, 2, 3\}$, L_1 and L_2 clearly bias the estimate in an ineffective way and fail to outperform ordinary least squares. In contrast, the two topological penalties clearly do better in both cases.

In Figure 6, the features in β_* are chosen to have three local maxima when the features are given the line topology. On the left, β_* consists of three piecewise-linear sawteeth, and on the right, β_* consists of three piecewise-constant boxcars. The total variation penalty $P(\beta) = \sum_{i=1}^{p} |\beta_{i+1} - \beta_i|$ and a smooth variant $P(\beta) = (\sum_{i=1}^{p} |\beta_{i+1} - \beta_i|^2)^{1/2}$ are considered, as well as two topological penalties. The parameters of the topological penalties are identical to the previous example, but the penalties are now imposed on superlevel set diagrams of β in order to penalize the number of local maxima in β instead of the number of distinct values. In the boxcar problem, total variation regularization does very well, as it encourages piece-wise linear functions, and the two topological penalties perform similarly. In the sawtooth problem, total variation does not do as well because β_* is no longer piece-wise constant, and interestingly the first topological penalty is similarly not as effective, while the second topological penalty performs well in both examples.

Finally, Figure 7 shows a linear regression problem on a 2D image. The topological penalty incorporated information from PD_1 as well as PD_0 to promote a single maximum and a single hole. For visual comparison, we also show the resulting ordinary least squares image.

These examples demonstrate how topological information can be incorporated effectively to add regularization or incorporate prior knowledge into problems. Furthermore, they demonstrate how topological information can be directly encoded, such as penalties on the number of clusters or number of maxima of a function, in a natural way that is difficult to accomplish with more traditional schemes.

3.2 Incorporating Topological Priors in Generative Models

We now use the same topological priors to improve the quality of a deep generative neural network. We start with a Baseline-Generator, pre-trained in a GAN-setup on MNIST, and by training it for a few iterations with a topological loss, we arrive at an improved Topology-Generator. We provide comparisons with other methods applied to the Baseline-Generator.

Generator Evaluation					
Model	MMD-L2	COV-L2	MMD-Wass	COV-Wass	Inception
Baseline-Generator (Images)	28.0±0.1	0.05±0.006	1.56±0.08	0.11±0.01	4.6±0.1
Topology-Generator (Images)	**27.5**±0.1	**0.06**±0.002	**1.52**±0.07	**0.12**±0.01	**5.1**±0.1
Baseline-Generator (3D Voxels)	**33.7**±0.8	0.10±0.01	4.3±1.4	0.65±0.03	N/A
Topology-Generator (3D Voxels)	34.1±0.8	**0.11**±0.02	**2.4**±0.8	0.65±0.03	N/A

Figure 10: Metrics for generator evaluation.

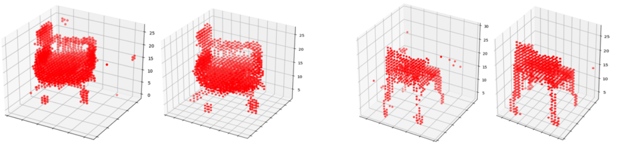

Figure 11: For each pair: (Left) before training with topology loss, (Right) after training with topology loss for 20 batch iterations.

A GAN as in (Goodfellow, Pouget-Abadie, et al., 2014) is trained on MNIST for 32,000 batch iterations with a batch size of 64 (this batch size is used throughout this section). The resulting generator (Baseline-Generator) produces reasonable output but with topological noise, see Figure 8(a). The prior used to improve the Baseline-Generator is identical with that of Figure 3: images should have 1 component in a superlevel set filtration. The loss function (topology loss) is $\mathcal{E}(1,0,2;\mathrm{PD}_0)$. The setup (Figure 8) is used to backpropagate to the latent space of Baseline-Generator, with the generator weights fixed, to minimize the topology loss using SGD; seen Figure 8(b) for results. ALternatively, using the same setup, the Baseline-Generator's weights are updated to minimize the topology loss; we train for 50 batch iterations to arrive at a new generator (Topology-Generator). The output can be seen in Figure 8(c).

For further qualitative comparisons, we train the Baseline-Generator for 100 batch iterations with a discriminator between features of the 0-dim persistence on MNIST images and the generator's output. The features were sums of lengths of the k longest PD_0 features for several choices of k, expressed as $\mathcal{E}(1,0,2;\mathrm{PD}_0) - \mathcal{E}(1,0,3;\mathrm{PD}_0)$, $\mathcal{E}(1,0,2;\mathrm{PD}_0) - \mathcal{E}(1,0,4;\mathrm{PD}_0)$, $\mathcal{E}(1,0,2;\mathrm{PD}_0) - \mathcal{E}(1,0,5;\mathrm{PD}_0)$, and $\mathcal{E}(1,0,2;\mathrm{PD}_0) - \mathcal{E}(1,0,11;\mathrm{PD}_0)$, with the results shown in Figure 8 (d). The output of a generator arrived at by training the Baseline-Generator in the original GAN-setup for another 60,000 batch iterations is shown in Figure 8 (e). Evidently, the topology loss allows the generator to learn in only 50 batch iterations to produce images with a single connected component and the difference is visually significant. These results are similar to using a PD_0-aware discriminator, suggesting that our priors were valid. Updating only the latent space produces cleaner images but they still contain some topological noise. For a closer study, consider the linear interpolation in the latent space of the Baseline-Generator and Topology-Generator in Figure 9. The two different cases behave very differently with respect to the topology. The Baseline-Generator interpolates by letting a disconnected components appear and grow. The Topology-Generator tries to interpolate by deforming the number without creating disconnected components. This might be most obvious in the interpolation from "1" to "4" (Figure 9, right hand side) where the appended structure of the "4" appears as a disconnected component in the baseline but grows out continuously from the "1" in the topology-aware case.

We also quantitatively compare the Baseline-Generator and Topology-Generator to further investigate if any improvements have been made. We use the Minimal Matching Distance (MMD) and Coverage metric as advocated by (Achlioptas et al., 2018) as well as the *Inception score* (Salimans et al., 2016) (a convolutional neural network with 99% test accuracy on MNIST was used instead of the Inception model). MMD-Wass and COV-Wass use the same procedure as MMD-L2 and COV-L2 but instead of the L2 distance between images, the 1-Wasserstein distance between the 0-dim persistence diagrams of the images was used (see Section 2). As seen in Table 10, the Topology-Generator shows improvements for all of these metrics. The results are the average of 5 computations of each metric, with test set sizes of 1,000 for L2 and Inception, and test sets sizes of 100 for Wasserstein distance.

We extend this superlevel set filtration to 3D data in the form of voxel grids. As before, a baseline generator is obtained by training a GAN to generate voxel shapes (chairs only) as in (Wu et al., 2016) and its output after 1,000 epochs (or 333,000 batch iterations) can be seen in Figure 11 as the left hand members in each of the two pairs. The result of training with the topology loss (same as for images) for 20 batch iterations can be seen in Figure 11 as the right hand members in each of the two pairs. We compare some metrics in Table 10; we show the average of 5 computations of each metric, with test set sizes of 100. Note that every voxel chair in the ground truth dataset has identical PD_0, since each chair consists of a connected component of voxels of value 1, among voxels of value 0.

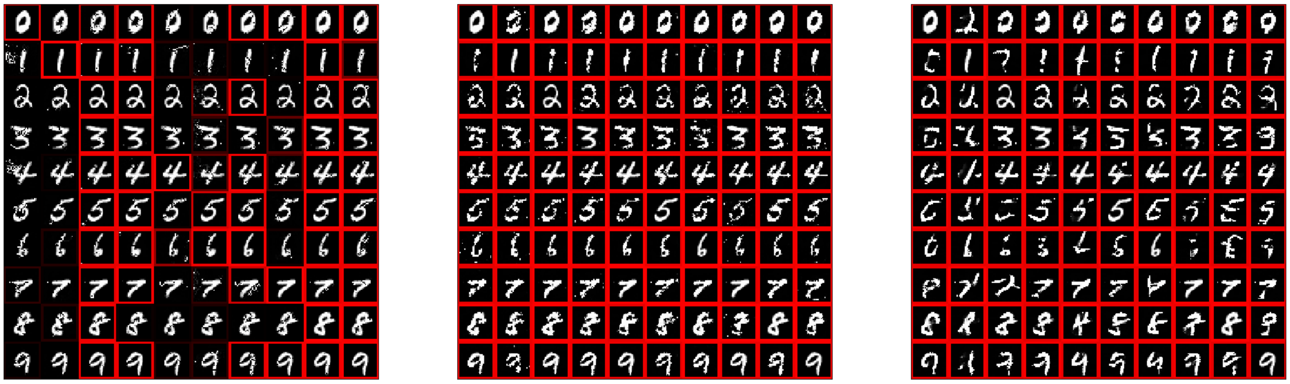

Figure 12: Topological adversarial attack on TopModel, MLPModel and CNNModel. The (i, j)-th cell represents an attack on an image with label i to be classified as label j. Red outlines are successful attacks.

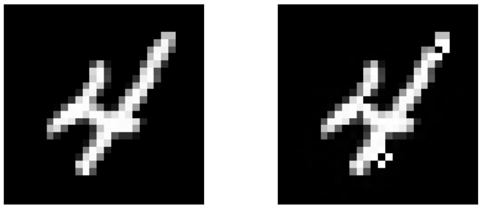

Figure 13: Example of Topological adversarial attack. (Left) original image, (Right) image optimized to be classified as an 8, which introduced two 1 pixel holes.

3.3 Topological Adversarial Attacks

Our topological layer may also be placed at the beginning of a deep network to generate features directly on the data. We can use the fact that our input layer is differentiable to perform adversarial attacks, by backpropagating from the predictions back to the input image. To the best of our knowledge, these are the first adversarial attacks conducted using persistence features. Since standard super-level set persistence is insufficient to classify MNIST digits, we include orientation information by computing the persistence homology during 8 directional sweeps. This is achieved by using the product of the image with fixed functions such as x, y, $\frac{x+y}{2}, \ldots$ etc., where x and y are the image coordinates, as the filtration value, for each of 8 different directions, $\mathcal{E}(p, q, 1; \mathrm{PD}_k)$ for p and q ranging between 0 and 4 resulting in 400 features for training the classification model. The model trained to classify the digits based on these topological features achieved 80-85 % accuracy. Next we performed gradient attack (Goodfellow, Shlens, and Szegedy, 2015) to change the classification of the digit to another target class. We observe that it is harder to train adversarial images compared to CNNs and MLPs. The results are shown in Figure 12. A red outline indicates that the attack was successful. When the attack was conducted on 1,000 images, to retarget to a random class, it had 100% success rate on MLP and CNN models and 25.2% success rate on the TopModel. When the adversarial attacks succeed, the results may offer insight on how the model classifies each digit. For example in Figure 13, the left image is the original image of the digit 4, the right was trained to be classified as an 8; note that two small holes at the top and bottom were sufficient to misclassify the digit. Several examples of the topological attacks provide similar intuition. Attacks on MLP and CNN are qualitatively different, but further work is needed to gauge the extent and utility of such distinctions.

4 Discussion

We present three novel applications using a differentiable topology layer which can be used to promote topological structure in Euclidean data, images, the weights of machine learning models, and to compare adversarial attacks. This only scratches the surface of the possible directions leveraging the differentiable properties of persistence. Without doubt such work will tackle problems beyond those we have presented here, including encouraging topological structure in intermediate activations of deep neural networks or using the layer in the middle of deep networks to extract persistence features where they may be more useful. However, many of the applications we have presented here also deserve further focus. For example, topological regularization, including the penalties we have presented, may have interesting theoretical properties, or closed form solutions. Furthermore, training autoencoders with distances between persistence features may produce stronger results than the functions considered here. Finally, it might prove useful to use topological features to train deep networks that are more robust to adversarial attacks – however, as we show this will require additional work. Topology, in contrast to local geometry, is generally underexploited in machine learning, but changing this could benefit the discipline.

Acknowledgements: RBG and GC were supported by Altor Equity Partners AB through Unbox AI (www.unboxai.org). BN was supported by the US Department of Energy (DE-AC02-76SF00515) while at the SLAC National Accelerator Laboratory. PS was supported by SSHRC Canada (NFRFE-2018-00431) and the Alan Turing Institute - Defense and Security Programme (D015). We are grateful for Panos Achlioptas' insights on the evaluation of our generative models.

References

Achlioptas, Panos et al. (2018). "Learning Representations and Generative Models for 3D Point Clouds". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden, pp. 40–49.

Adcock, Aaron, Erik Carlsson, and Gunnar Carlsson (2016). "The ring of algebraic functions on persistence bar codes". In: *Homology, Homotopy and Applications* 18.1, pp. 381–402. ISSN: 15320073, 15320081. DOI: 10.4310/HHA.2016.v18.n1.a21. URL: http://www.intlpress.com/site/pub/pages/journals/items/hha/content/vols/0018/0001/a021/ (visited on 07/03/2018).

Brüel-Gabrielsson, Rickard and Gunnar Carlsson (2018). "Exposition and Interpretation of the Topology of Neural Networks". In: *arXiv:1810.03234v2*.

Brüel-Gabrielsson, Rickard, Vignesh Ganapathi-Subramanian, et al. (2019). "Topology-Aware Surface Reconstruction for Point Clouds". In: *arXiv preprint arXiv:1811.12543*.

Carlsson, Gunnar and Rickard Brüel-Gabrielsson (2018). "Topological Approaches to Deep Learning". In: *arXiv:1811.01122*.

Carlsson, Gunnar, Afra Zomorodian, et al. (2005). "Persistence barcodes for shapes". In: *International Journal of Shape Modeling* 11.02, pp. 149–187.

Chakraborty, Anirban et al. (2018). "Adversarial Attacks and Defences: A Survey". In: *arXiv preprint arXiv:1810.00069*.

Chen, Chao, Xiuyan Ni, et al. (2019). "A Topological Regularizer for Classifiers via Persistent Homology". In: *Proceedings of Machine Learning Research*. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 2573–2582.

Chen, Xi, Yan Duan, et al. (2016). "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2172–2180.

Clough, James R. et al. (2019). "Explicit topological priors for deep-learning based image segmentation using persistent homology". In: *arXiv preprint arXiv:1901.10244*.

Cormen, Thomas H. et al. (2009). *Introduction to Algorithms*. 3rd. MIT Press.

deSilva, Vin, Dimitriy Morozov, and Mikael Vejdemo-Johansson (2011). "Dualities in persistent (co)homology". In: *Inverse Problems* 27.12.

Dey, Tamal K et al. (2010). "Persistent heat signature for pose-oblivious matching of incomplete models". In: *Computer Graphics Forum*. Vol. 29. Wiley Online Library, pp. 1545–1554.

Edelsbrunner, Herbert (2010). "Alpha Shapes – a Survey". In: *Tessellations in the Sciences; Virtues, Techniques and Applications of Geometric Tilings*. Ed. by R. van de Weygaert et al. Springer, Heidelberg.

Edelsbrunner, Herbert and John Harer (2010). *Computational Topology - an Introduction*. American Mathematical Society, pp. I–XII, 1–241. ISBN: 978-0-8218-4925-5.

Gabella, Maxime et al. (2019). "Topology of Learning in Artificial Neural Networks". In: *arXiv:1902.08160*.

Gebhart, Thomas and Paul Schrater (2017). "Adversary Detection in Neural Networks via Persistent Homology". In: *arXiv preprint arXiv:1711.10056*.

– (2019). "Adversarial Examples Target Topological Holes in Deep Networks". In: *arXiv preprint arXiv:1901.09496*.

Giansiracusa, Noah, Robert Giansiracusa, and Chul Moon (2017). "Persistent homology machine learning for fingerprint classification". In: *arXiv preprint arXiv:1711.09158*.

Goodfellow, Ian J., Jean Pouget-Abadie, et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2672–2680.

Goodfellow, Ian, Jonathon Shlens, and Christian Szegedy (2015). "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations*. URL: http://arxiv.org/abs/1412.6572.

Guss, William H. and Ruslan Salakhutdinov (2018). "On Characterizing the Capacity of Neural Networks using Algebraic Topology". In: *arXiv preprint arXiv:1802.04443*.

Hoerl, Arthur E. and Robert W. Kennard (1970). "Ridge Regression: Biased Estimation for Nonorthogonal Problems". In: *Technometrics* 12.1, pp. 55–67. DOI: 10.1080/00401706.1970.10488634.

Hofer, Christoph and et al (2017). "Deep Learning with Topological Signatures". In: *31st Conference on Neural Information Processing Systems*.

Liu, Jen-Yu, Shyh-Kang Jeng, and Yi-Hsuan Yang (2016). "Applying Topological Persistence in Convolutional Neural Network for Music Audio Signals". In: *arXiv:1608.07373*.

Milosavljević, Nikola, Dmitriy Morozov, and Primoz Skraba (2011). "Zigzag persistent homology in matrix multiplication time". In: *Proceedings of the 27th annual ACM symposium on Computational geometry - SoCG '11*. the 27th annual ACM symposium. Paris, France: ACM Press, p. 216. ISBN: 978-1-4503-0682-9. DOI: 10.1145/1998196.1998229. URL: http://portal.acm.org/citation.cfm?doid=1998196.1998229 (visited on 10/22/2019).

Otter, Nina et al. (2017). "A roadmap for the computation of persistent homology". In: *EPJ Data Science* 6.1, p. 17.

Paszke, Adam et al. (2017). "Automatic Differentiation in PyTorch". In: *NIPS Autodiff Workshop*.

Poulenard, Adrien, Primoz Skraba, and Maks Ovsjanikov (2018). "Topological Function Optimization for Continuous Shape Matching". In: *Computer Graphics Forum*. Vol. 37. Wiley Online Library, pp. 13–25.

Rieck, Bastian et al. (2018). "Neural Persistence: A Complexity Measure for Deep Neural Networks Using Algebraic Topology". In: *arXiv preprint arXiv:1812.09764*.

Rudin, Leonid I., Stanley Osher, and Emad Fatemi (1992). "Nonlinear total variation based noise removal algorithms". In: *Physica D: Nonlinear Phenomena* 60.1, pp. 259–268. ISSN: 01672789. DOI: 10.1016/0167-2789(92)90242-F. URL: https://linkinghub.elsevier.com/retrieve/pii/016727899290242F (visited on 05/21/2019).

Salimans, Tim et al. (2016). "Improved Techniques for Training GANs". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain, pp. 2234–2242.

Seng Pun, Chi, Kelin Xia, and Si Xian Lee (2018). "Persistent-Homology-based Machine Learning and its Applications – A Survey". In: *arXiv preprint arXiv:1811.00252*.

Skraba, Primoz, Maks Ovsjanikov, et al. (2010). "Persistence-based segmentation of deformable shapes". In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, pp. 45–52.

Skraba, Primoz, Gugan Thoppe, and D Yogeshwaran (2017). "Randomly Weighted $d-$ complexes: Minimal Spanning Acycles and Persistence Diagrams". In: *arXiv preprint arXiv:1701.00239*.

Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society, Series B* 58.1, pp. 267–288. URL: http://statweb.stanford.edu/~tibs/lasso/lasso.pdf (visited on 05/22/2019).

Tibshirani, Ryan J., Holger Hoefling, and Robert Tibshirani (2011). "Nearly-Isotonic Regression". In: *Technometrics* 53.1, pp. 54–61. ISSN: 0040-1706, 1537-2723. DOI: 10.1198/TECH.2010.10111. URL: http://www.tandfonline.com/doi/abs/10.1198/TECH.2010.10111 (visited on 05/21/2019).

Wu, Jiajun et al. (2016). "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In: *Advances in neural information processing systems*, pp. 82–90.

Zomorodian, Afra and Gunnar Carlsson (2005). "Computing Persistent Homology". In: *Discrete & Computational Geometry* 33.2, pp. 249–274. ISSN: 1432-0444. DOI: 10.1007/s00454-004-1146-y. URL: https://doi.org/10.1007/s00454-004-1146-y (visited on 06/12/2019).

A Appendix

A.1 Topological Definitions

Here we include the precise definitions of the required notions – this should not be considered as an exhaustive introduction, but rather our goal is to give precise definitions to the concepts used in the paper.

A cell complex can be defined inductively by dimension. Let $\mathcal{X}_0$ be the 0-skeleton of the complex consisting of points, which can be thought of as 0-dimensional balls (cells). The k-skeleton $\mathcal{X}_k$ is obtained by attaching some number of k-dimensional cells by their boundaries to the $(k-1)$-skeleton $\mathcal{X}_{k-1}$ via continuous maps. This process can be repeated indefinitely to form the cell complex $\mathcal{X}$, but in practice we terminate at some finite dimension. While cell complexes can be used fairly flexibly in encoding topological spaces, simplicial complexes are typically preferred in computational settings for their combinatorial description.

Definition A.1. *A k-dimensional simplex $(v_0, \ldots, v_k)$ is the convex combination of $k+1$ vertices, $\{(v_0), \ldots, (v_k)\}$.*

We only consider the situation where the vertices correspond to point in a sufficiently nice ambient space, usually $\mathbb{R}^d$ so that a geometric realization of simplices exists as a subset of the ambient space. Simplices are used to represent a space, but must satisfy some additional constraints so that they form a *simplicial complex.*

Definition A.2. *A simplicial complex $\mathcal{X}$ is a collection of simplices such that*

1. *For every simplex σ in $\mathcal{X}$, every face $\tau \subseteq \sigma$ is also in $\mathcal{X}$.*
2. *For any two simplices σ_1 and σ_2, $\tau = \sigma_1 \cap \sigma_2$ is a face of both σ_1 and σ_2.*

These can be thought of as higher dimensional analogs of graphs or triangular meshes – conversely, a graph is a 1-dimensional simplicial complex.

To define homology, we first construct a *chain group.* In our setting, the k-th chain group $C_k(\mathcal{X})$ is the freely generated group generated by k-dimensional simplices. Importantly, there exists a boundary homomorphism

$$\partial_k : C_k(\mathcal{X}) \to C_{k-1}(\mathcal{X})$$

such that $\partial \circ \partial = 0$. Explicitly,

$$\partial_k(v_0, \ldots, v_k) = \sum_{i=0}^{k} (-1)^i (v_0, \ldots, \hat{v}_i, \ldots, v_k)$$

where $\hat{v}_i$ indicates that the i-th vertex has been removed. Again, we take $\partial_0 = 0$. Homology can then be defined as

$$H_k(\mathcal{X}) = \frac{\ker \partial_k}{\mathrm{im}\ \partial_{k+1}}$$

We consider homology computed over fields, in which case the homology groups are vector spaces. Rather than only consider one simplicial complex $\mathcal{X}$, we can consider an increasing sequence of simplicial complexes, called a *filtration* $\mathcal{X}_0 \subseteq \mathcal{X}_1 \subseteq \ldots \subseteq \mathcal{X}_N$. The requirement is that each $\mathcal{X}_i$ is itself a simplicial complex. We consider the filtration function $f : \mathcal{X} \to \mathbb{R}$ such that each sub/super-level set $f^{-1}(-\infty, \alpha]$, resp. $f^{-1}[\alpha, \infty)$ form a filtration in parameterized by α. Under appropriate finiteness conditions, which are always satisfied for finite simplicial complexes (Edelsbrunner and Harer, 2010).

Theorem A.3. *(Zomorodian and Carlsson, 2005) If homology is computed over a field, then the homology of a filtration admits an* interval decomposition*. The interval decomposition is direct sum of rank 1 elements which exists over an interval $[b, d)$.*

In our setting, whether the intervals are open or closed is not important so we supress this aspect of the notation. This collection of intervals is the *persistence diagram.*

Definition A.4. *A persistence diagram is a collection of points (b_i, d_i), possibly with repetition, along with the diagonal Δ, i.e. all points such that $b = d$.*

The diagonal plays an important part in the definition of distances between diagrams. The main distance we consider is the p-Wasserstein distance between two diagrams

$$W_p(\mathrm{PD}_k(f), \mathrm{PD}_k(g)) = \inf_{\varphi} \left(\sum_{p \in \mathrm{PD}_k(f)} ||p - \varphi(p)||^p \right)^{1/p}$$

where φ is a bijection. This can be viewed as an optimal transport problem, points must either be matched to the other diagram or the diagonal. This can be formulated as a classical linear program if we add to each diagram the projection of the other diagram to the diagonal

$$\mathrm{proj}(d, b) = \left(\frac{d+b}{2}, \frac{d+b}{2} \right)$$

In other words, we can consider the optimal transport problem between $\mathrm{PD}(f) \cup \mathrm{proj}(\mathrm{PD}(g))$ and $\mathrm{PD}(g) \cup \mathrm{proj}(\mathrm{PD}(f))$ where the distance between any two points on the diagonal is 0.

One of the core techniques in this paper is to compute the gradient of an energy function with respect to some

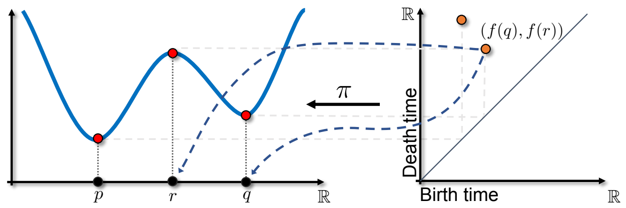

Figure 14: A one dimensional example of a persistence diagram and the inverse map π. The function on the left has critical points at points p, r and q. The local minima create components in the sub-level sets and so represent birth times (x-axis), while the maxima kills one of the components (the younger one) and so is a death time (y-axis). The inverse map for a point in the diagram returns the corresponding critical points/simplicies.

parameters. This is possible through the definition of an inverse map

$$\pi_f(k) : \{b_i, d_i\}_{i \in \mathcal{I}_k} \to (\sigma, \tau)$$

This map is unique and well defined in the case of when a filtration is a strict order, i.e. the simplices are totally ordered in the filtration. This is never the case in our scenarios, where the simplices only form a total order. However this can be resolved by extending the total order to a strict order either deterministically or randomly, see (Skraba, Thoppe, and Yogeshwaran, 2017) for a formal proof and description of how this can be done.

As is the case in our setting, the filtration is not defined directly on the simplices but rather derived from either functions on the vertices or depend on some property of the vertices, e.g. the coordinates of the points that the vertices correspond to. In this case, we require an additional inverse map from simplices to a collection of vertices. This is described in the main paper for two scenarios which we use in the applications. Once both inverse maps are defined, the gradient can be defined in the standard way using the chain rule.

A.2 Comparison

In this paper we use both super-level set filtrations as well as Vietoris-Rips and weak-Alpha filtrations. Figure 15 provides an illustrative comparison of super-level set filtration (on the left) and a Rips filtration (on the right). The digit "9" viewed in terms of the super-level set of its pixel values has one clear connected component and one clear ring, which can be read from the corresponding persistence diagrams. However, the super-level set filtration does not take into consideration distance along the grid and a small (in the 2D sense) hole can be very large from the superlevel set perspective. In the Rips case we lay out all the pixel values in a 3D space and use the euclidean distance. Here there are many more connected components at small to medium filtration values, and the same is true for the number of rings. The Rips does not merely pick up the obvious ring in the "9" but also rings that are formed with respect to the vertical axis.

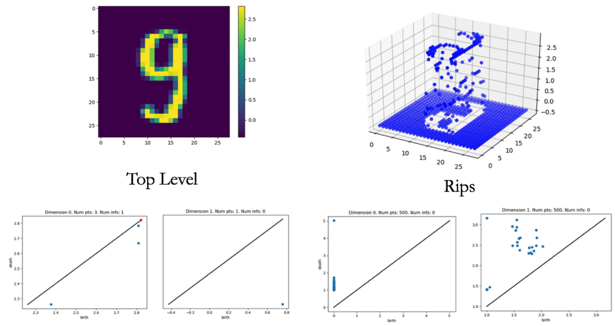

Figure 15: Compare top level set with Rips filtration

A.3 Explore latent space

In this paper we backpropagate to the latent space of generative models (query the latent space) in order to improve on them. However, the ability to topologically query the latent space may not only be used to give us more topologically desirable output but also to explore the nature of the latent space. Consider Figure 16 that consists of images produced by a trained InfoGAN-generator (Chen, Duan, et al., 2016). The type of digit closest to a "2" with a loop on the bottom is arguably a digit "2" without such a loop. However, when we backpropagate to remove the ring we get something that does not look like any MNIST digit. Similarly, if we remove the ring in the digit "6" we get something that does not look very much like any MNIST digit. This gives some indications that the latent space learned by the InfoGAN is relatively topologically flexible.

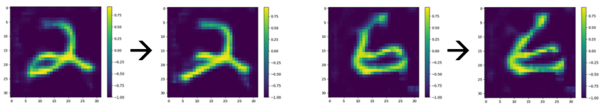

Figure 16: Infogan query

A.4 Topological Mapping

In this paper we train and experiment with networks that use persistence as input. However, with a differentiable layer we can put the persistence computations anywhere in the network. Sometimes it is useful to put certain layers in the middle of a deep network. For example in domain transfer we often want to create a mapping where features will be more useful. The setup might look like in the Figure 17. We do a simpler setup

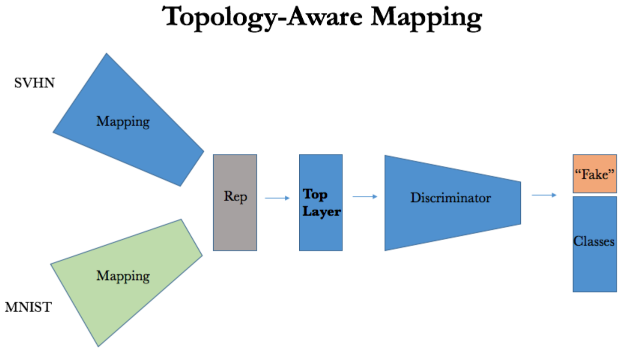

Figure 17: Topological mapping

where we train a network to use superlevel persistence features to classify the naive PD_1-features of MNIST which we define for "1", "2", "3", "4", "5", "7" as zero, for "0", "6", "9" as one, and for "8" as two. We are able to get a 84% accuracy. However, by putting two convolutional layers (the mapping) which maintain the dimensionality of the input before our Topology Layer, we get the results shown in Figure 18 where we ultimate improve our accuracy to 86%. In this case, it seems as if the mapping thickens the digits in the images in order to remove topological noise.

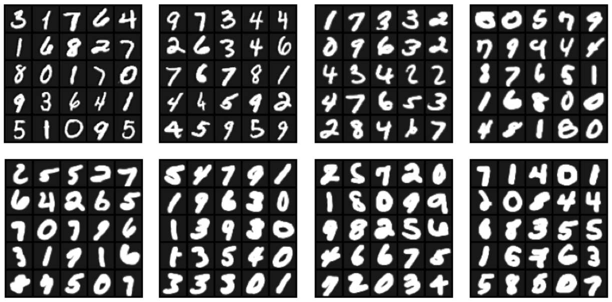

Figure 18: Intermediate Representation in topological mapping

A.5 Topological Adversarial Attacks

A.5.1 Features for classification model

In order to train a classification model using only topological features we must include orientation and directional information into the persistence homology features. This can be achieved by using custom filtration constructed in the following way. First we define 8 directional functions

$$g_\theta(x,y) = \cos(\theta)x + \sin(\theta)y \qquad \theta = 0, \pi/4, \pi/2, \ldots, 7\pi/4$$

These functions are shifted and scaled so that in the domain of the image, they range from 0 to 1. If $I(x,y)$ is the input image, then the filtrations are given by

$$f_{\theta_i}(x,y) = I(x,y) g_{\theta_i}(x,y)$$

Persistence diagrams of dimensions 0 and 1 are computed for each filtration. We then compute 25 features on each persistence diagram given by $\mathcal{E}(p,q,0;\mathrm{PD})$, for p and q ranging between 0 and 4, thus totalling 400 features.

A.6 Regularization Sparsity Visualization

In Figure 19 we see the Rips PD_0 diagram for the weights of a logistic regression model without (left) and with (right) L1 regularization. As the test accuracy improves, we see that the persistence diagram changes as well.

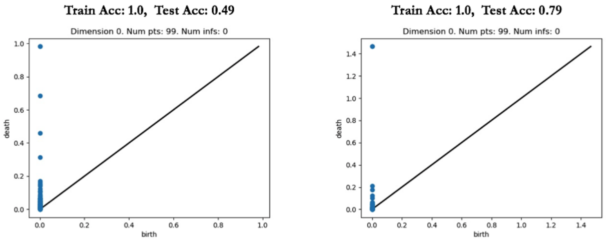

Figure 19: Rips and generalization