Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

doi:10.3934/xx.xxxxxxx

pp. **X–XX**

TOFU: TOPOLOGY FUNCTIONAL UNITS FOR DEEP LEARNING

Christopher Oballe*

University of Notre Dame

Dave Boothe

CCDC Army Research Lab

Piotr J. Franaszczuk

CCDC Army Research Lab and Johns Hopkins University

Vasileios Maroulas*

University of Tennessee, Knoxville

Abstract. We propose ToFU, a new trainable neural network unit with a persistence diagram dissimilarity function as its activation. Since persistence diagrams are topological summaries of structures, this new activation measures and learns the topology of data to leverage it in machine learning tasks. We showcase the utility of ToFU in two experiments: one involving the classification of discrete-time autoregressive signals, and another involving a variational autoencoder. In the former, ToFU yields competitive results with networks that use spectral features while outperforming CNN architectures. In the latter, ToFU produces topologically-interpretable latent space representations of inputs without sacrificing reconstruction fidelity.

1. **Introduction.** Topological data analysis (TDA) encompasses a set of methods that measure the shape of data with tools from algebraic topology (9). Persistent homology (18; 23), the workhorse behind many popular TDA techniques, takes data and converts it to a multiscale topological summary known as a persistence diagram (PD), which can be used for shape-based inference. Since the space of PDs lack a Hilbert space structure, they are not directly amenable to commonly-used statistical learning methods. A large body of work sought to remedy this shortcoming by inventing well-behaved Hilbert space representations of PDs (7; 1; 11; 5; 4; 50). Other works, notably (47) and (37), derive PD representations that serve as sufficient statistics, thereby ensuring that PD summaries retain all statistically-pertinent information for an inference task. Some authors avoid Hilbert space representations altogether, choosing instead to work directly in PD space. This is achieved, for example, by leveraging stability results to push statistical distributions from data space forward to PD space (20), or by adopting tools from point process theory (12; 2; 33; 34).

Date: November 24, 2021.

2020 *Mathematics Subject Classification.* Primary: 55N31, 68T07.

Key words and phrases. Topological data analysis, deep learning, neural networks, machine learning, activation functions, variational autoencoders.

In addition to studying PDs and their properties to create tools for inference, the use of persistent homology to design and understand artificial neural networks (ANNs) is another area of research that lies at the intersection of machine learning and TDA.

The work (3) shows that persistent-homology-derived features can effectively classify ANN dynamics; (26) establishes empirical links between the homology of ANNs and their capacities; similarly, (22) uncovers topological patterns in the weights of trained CNNs, and shows that the topological structure of the weights correlates with the CNN's ability to generalize. Persistent homology has also been used in ANNs to regularize topology in the output at certain layers. Two works that leverage ideas from persistent homology in autoencoders are (45) and (38); the former uses the Wasserstein distance between distributions to introduce a novel regularization term for latent space distributions, while the latter introduces a persistent-homology-loss term that promotes similar topology in the input and latent spaces. A general framework for controlling the topology of layer outputs in ANNs with PD loss functions is introduced in (6).

Our work proposes a new trainable ANN unit that uses a PD dissimilarity function as its activation. Since persistence diagrams are topological summaries of structures, this new activation measures and learns the topology of data to leverage it in machine learning tasks. Unlike previous works, which exploit PD-inverse maps to promote desired topological characteristics in output features, our method learns parameters that live in PD space, which are used to topologically distinguish inputs. We refer to our proposed ANN unit as the **To**pological **F**unctional **U**nit (ToFU) since its activation is a functional on the space of PDs. ToFU is parameterized by a PD and learns pertinent topology in the data itself. In particular, ToFU learns a PD that aids an ANN in its intended task. For example, if ToFU is used in an ANN designed for binary classification, ToFU may learn a PD that is more similar to PDs of one class versus those of the other, thereby distinguishing the two classes by their topologies. Moreover, since ToFU solely considers the persistent homology of data, the parameters it learns are robust to all rigid transformations of input data such as rotations and translations. To summarize, the main contributions of our are work are:

1. a new trainable ANN unit that uses a PD dissimilarity function as its activation,
2. a signal classification example where ToFU learns features that outperform traditional topological vectorizations and remain competitive with those derived from Fourier analysis, and
3. a variational autoencoder architecture that demonstrates how ToFU learns pertinent topology present in the data itself.

The paper is organized as follows. Section 2 covers the necessary background to formulate and understand our method. In particular, Section 2.1 summarizes artificial neural networks and the mathematical formulation we use to describe them. Section 2.2 reviews computational topology, specifically persistent homology with cubical filtrations. In Section 3, we present ToFU along with accompanying examples. Section 3.1 describes a novel encoder architecture that uses ToFU to learn latent space representations. Section 4 contains two experiments that showcase ToFu's utility. Finally, we end with discussions in Section 5.

2. **Preliminaries.**

We begin by briefly reviewing the ideas from artificial neural networks and computational topology pertinent to our work.

2.1. **Deep Learning.**

In this section, we define artificial neural networks (ANNs) and introduce accompanying terminology that we use throughout the paper. ANNs are function approximators that are widely used in machine learning for their expressive capabilities. For thorough expositions on ANNs and their role in machine learning, see (25), (42), and references therein.

Units and layers. The building blocks of ANNs are units and layers. Units are defined by transformations and activations. A transformation $T : \mathbb{R}^d \to \mathcal{X}$ is a function from an input Euclidean space $\mathbb{R}^d$ to a Polish space $\mathcal{X}$, and an activation $a : \mathcal{X} \to \mathbb{R}$ is a real-valued function. A unit $\phi : \mathbb{R}^d \to \mathbb{R}$ is given by the composition $\phi = a \circ T$. In practice, T usually has learnable parameters while a does not, but this may not be required. In fact, our unit ToFU uses a nonlearnable transformation that computes the PD of the input; see Section 2.2 and 3 for details. A common choice for T is the affine transformation, L, given by

$$L(\mathbf{x}) := \mathbf{x}\mathbf{w}^\intercal + b, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is a row vector and $\mathbf{w} \in \mathbb{R}^d$ and $\mathbf{b} \in \mathbb{R}$ are learnable parameters known as the weights and bias, respectively. Two common choices for a are the rectified linear unit and sigmoid activations given, respectively, by $\mathrm{ReLU}(x) = \max(0, x)$ and $\sigma(x) = (1+e^{-x})^{-1}$. A layer $\Phi : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ is a function that is comprised of units, $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_{d_{\text{out}}}(\mathbf{x})]$. We call Φ_{l+1} *dense* if each of its units contains an affine transformation (see Equation (1)) of the form $L(\mathbf{x}) = \Phi_l(\mathbf{x})\mathbf{w}^\intercal + b$.

Artificial neural networks. An artificial neural network (ANN) is a function $A : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ given by $A(\mathbf{x}) = \Phi_L \circ \Phi_{L-1} \circ \cdots \circ \Phi_2 \circ \Phi_1(\mathbf{x})$ where Φ_l are layers such that the input dimension of Φ_{l+1} is the same as the output dimension of Φ_l for all $l \in \{1, 2, \ldots, L-1\}$. Notice that there is no constraint on the output dimension of the final layer ϕ_L. ANNs are explicitly parameterized by the collection of all learnable parameters in each of their layers, $\boldsymbol{\theta}$, which we signify by writing $A_{\boldsymbol{\theta}}$.

Supervised training of an ANN requires a set of labelled examples $\mathcal{T} := \{(\mathbf{x}_n, \mathbf{y}_n)\}_{i=n}^{|\mathcal{T}|}$, where $\mathbf{x}_n$ and $\mathbf{y}_n$, respectively, denote an input and an output, and a pertinent loss function $\ell(\mathbf{y}_n, A_{\boldsymbol{\theta}}(\mathbf{x}_n))$, which measures the discrepancy between the true output and the output of the ANN. Common choices of ℓ include cross-entropy (for classification) and squared error (for regression). With $\mathcal{T}$ and ℓ in hand, an ANN is trained by searching for a solution to

$$\hat{\boldsymbol{\theta}} := \operatorname*{argmin}_{\boldsymbol{\theta}} \frac{1}{|\mathcal{T}|} \sum_{(\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{T}} \ell(\mathbf{y}_n, A_{\boldsymbol{\theta}}(\mathbf{x}_n)), \tag{2}$$

by means of some version of (stochastic) gradient descent. The gradient for each set of parameters $\boldsymbol{\theta}_l$ at layer Φ_l is obtained through backpropagation (49), an algorithm that iteratively applies the chain rule to compute $\frac{\partial L}{\partial \boldsymbol{\theta}_l}$. Hence, to be trainable via backpropagation, any layer in an ANN should be differentiable with respect to its learnable parameters. In most cases, gradient updates of parameters in $A_{\boldsymbol{\theta}}$ are not performed using the entire training set $\mathcal{T}$, but rather by randomly partitioning $\mathcal{T}$ into subsets of equal size, known as minibatches, then performing a separate gradient update with each minibatch substituted in place of $\mathcal{T}$ in Equation (2). Moreover, the gradients used to update parameters are typically multiplied by a tunable scaling factor, which stabilizes parameter values and speeds up learning.

Usually, the minibatching procedure is repeated in order to iterate through the entirety of $\mathcal{T}$ multiple times. In the language of deep learning, the size of minibatches, the number of times one iterates through $\mathcal{T}$, and the scaling factor used for gradient descent are known as the batch size, the number of epochs, and the learning rate, respectively.

Variational autoencoders. Here, we give a high level summary of variational autoencoders (VAEs). For a thorough introduction to VAEs, please see (28) and references therein. The goal of a VAE is to create a probabilistic generative model for data by learning a joint distribution, $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$, over data $\mathbf{x} \in \mathbb{R}^d$ and a latent variable $\mathbf{z} \in \mathbb{R}^h$, where $h < d$. Here, $\boldsymbol{\theta}$ denotes learnable parameters of an ANN. The distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ implies a probabilistic encoding-decoding scheme based on drawing from its posteriors: given data $\mathbf{x}$, create an encoding $\mathbf{z}$ by drawing from $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, then perform decoding by drawing from $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. A naive loss function for training the generative model is

$$L(\Theta) = D_{KL}(p_\Theta(\mathbf{x})||p_{true}(\mathbf{x})) \tag{3}$$

$$= -\mathbb{E}_{p_\Theta(\mathbf{x})}(\log p_{true}(\mathbf{x})) + \mathbb{E}_{p_\Theta(\mathbf{x})}(\log p_\Theta(\mathbf{x})), \tag{4}$$

where $p_{true}(\mathbf{x})$ denotes the true probability density of the data, D_{KL} denotes Kullback-Leibler divergence, and $\mathbb{E}_{p_\Theta(\mathbf{x})}$ denotes expectation with respect to $p_\Theta(\mathbf{x})$. However, Equation (4) is rarely implemented as a loss function because of several practical limitations. First, $p_{true}(\mathbf{x})$ is seldom known and must be estimated. Second, the density $p_\Theta(\mathbf{x}) = \int p_\Theta(\mathbf{x}|\mathbf{z}) p_\Theta(\mathbf{z})\, d\mathbf{z}$ generally has no closed form and hence must also be estimated, for instance by using a Monte Carlo approximation. Employing these estimators introduces a level of noise into training that greatly hinders learning. Moreover, since $p_\Theta(\mathbf{x}|\mathbf{z}) \propto p_\Theta(\mathbf{x}) p_\Theta(\mathbf{z}|\mathbf{x})$, the integral for $p_\Theta(\mathbf{x})$ must be estimated each time one samples the posterior (a similar situation occurs when sampling from the posterior $p_\Theta(\mathbf{z}|\mathbf{x})$), which is computationally expensive and introduces a high level of variance in network predictions. Therefore, most VAE implementations avoid the use of Equation (4) in training, and instead typically use the following design choices. First, one assumes a simple prior, with no dependence on network parameters, over the latent variable , e.g. $p_\Theta(\mathbf{z}) = p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$, the standard normal density in $\mathbb{R}^h$. Next, the posteriors $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ and $q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})$ are implemented as separate neural networks with respective parameters $\boldsymbol{\theta}$ and $\boldsymbol{\vartheta}$. By Bayes' rule, $p(\mathbf{z})$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ imply the posterior distribution, $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) / \int p(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, d\mathbf{z}$. Therefore, a sensible loss function to train the VAE is,

$$L(\boldsymbol{\theta}, \boldsymbol{\vartheta}) = D_{KL}(q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})), \tag{5}$$

however since the posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ involves the same problematic integral in the rightmost term of Equation (4), a loss function based on the evidence lower bound (ELBO), which is related to Equation (5), is used instead:

$$L(\boldsymbol{\theta}, \boldsymbol{\vartheta}) = \mathbb{E}_{q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})}\Big(\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\Big) - D_{KL}\Big(q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\Big). \tag{6}$$

It can be shown that maximizing Equation (6) simultaneously maximizes the evidence of the model, $p_{\boldsymbol{\theta}}(\mathbf{x})$, and minimizes the Kullback-Leibler divergence $D_{KL}(q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})))$; see (28) or Appendix 5.1.

2.2. **Computational Topology.** In this section, we provide a succinct overview of the ideas from computational topology that underpin our method. For a thorough treatment of computational topology, see (17; 51; 9), and references therein. One notable difference in our exposition from that in other computational topology papers is that we focus our discussion on cubical complexes (48) in lieu of simplicial complexes. We make this choice purely for the sake of simplicity, because the data in our experiments (Sections 4.1 and 4.2) are naturally modelled by functions over grids, which are efficiently represented as filtered cubical complexes. Our method works for persistence diagrams computed by any homology theory, including simplicial.

Intuitively, cubical complexes are built by gluing cubes together at faces. Formally, an elementary interval $I \subset \mathbb{R}$ is a subset of the form $\{l\}$ or $[l, l+1]$ for any $l \in \mathbb{Z}$; we refer to the former as a degenerate interval. An elementary cube is a product $Q := I_1 \times I_2 \times \cdots \times I_k \subset \mathbb{R}^k$ of elementary intervals, and its dimension (denoted $\dim Q$) is the number of nondegenerate intervals in the product. If Q' and Q are elementary cubes with $Q' \subset Q$, we say Q' is a face of Q. Moreover, if $\dim Q' = \dim Q - 1$, we call Q' a facet of Q. A cubical complex $\mathcal{K}$ is any subset that can be written as a union of elementary cubes, and any subset of $\mathcal{K}$ that forms a cubical complex is referred to as a subcomplex of $\mathcal{K}$. We denote the collection of k-dimensional cubes in $\mathcal{K}$ by $\mathcal{K}_k$.

Cubical homology. After building a cubical complex from data, our next step is to compute homology to obtain a quantitative descriptor of topology. Homology provides a mathematically rigorous framework to measure connectivity and detect the presence of holes in topological structures. Homology computations are done with tools from abstract linear algebra (see (16)), which studies linear operators between general vector spaces defined over arbitrary fields. In all of our examples, we choose the field as $\mathbb{Z}_2$, which is defined as the set $\{0, 1\}$ with addition and multiplication modulo 2, the vector spaces of interest are the so-called $k-$chains of a cubical complex $\mathcal{K}$, the set of formal sums $C_k := \{\sum_i \alpha_i Q_i : Q_i \in \mathcal{K}_k, \alpha_i \in \mathbb{Z}_2\}$. The linear operators we are concerned with are the $k-$th boundary maps $\partial : \mathcal{K}_k \to \mathcal{K}_{k-1}$, given by

$$\partial_k(Q) = \begin{cases} 0, & k = 0 \\ \sum_{Q' \in \mathcal{F}(Q)} Q', & k > 0 \end{cases} \tag{7}$$

where $\mathcal{F}(Q)$ are the facets of Q. The boundary map extends to $\partial_k : C_k \to C_{k-1}$ through linear extension of Equation (7). Elements of $\ker \partial_k$ and $\mathrm{Im}\partial_k$ are called k-cycles and k-boundaries, respectively. The k-th homology group, H_k, is defined as the quotient space (see Appendix 5.2), $H_k := \ker \partial_k / \mathrm{Im}\partial_{k+1}$. Generators of H_k are referred to as homological features. Informally, these are $k-$dimensional holes in a cubical complex. In Appendix 5.2, we provide an example of a full cubical homology computation over $\mathbb{Z}_2$. Homology computation over arbitrary fields is addressed in (51) and a general form of the boundary operator for cubical complexes is given in (27).

Filtrations and persistent homology. Rather than compute the homology for a fixed cubical complex built atop data, the more common approach in TDA is to compute the homology for a nested family of complexes while tracking the appearances and disappearances of homological features. This method, called persistent homology, is well-suited for quantifying large-scale topological structure in data since it is robust

to minor perturbations in samples; see (18). Persistent homology is defined through filtrations.

Definition 2.1. Let $\mathcal{K}$ be a cubical complex, and suppose $f : Q_{\mathcal{K}} \to \mathbb{R}$, where $Q_{\mathcal{K}}$ denotes the set of elementary cubes in $\mathcal{K}$, satisfies (i) $f(Q') \leq f(Q)$ whenever Q' is a face of Q. Define $\mathcal{K}(a) := f^{-1}(-\infty, a]$ and notice that (i) implies $\mathcal{K}(a)$ is a subcomplex of $\mathcal{K}$ for every $a \in \mathbb{R}$. Taking $a_1 < a_2 < \cdots < a_n$ to be the values of f for every simplex in $\mathcal{K}$ and denoting $\mathcal{K}(a_i) := \mathcal{K}_i$, we obtain an increasing sequence of subcomplexes $\emptyset = \mathcal{K}_0 \subset \mathcal{K}_1 \subset \cdots \subset \mathcal{K}_n = \mathcal{K}$, which we call the filtration of f.

A commonly-used filtration, and the one we adopt in all of our examples, is the lower-star filtration, which is defined by $\mathcal{K}(a) := \{Q \in \mathcal{K} : \max_{v \in Q} f(v) \leq a\}$ where v denotes a 0-cube (i.e., a vertex of the cube Q). An example of a lower star filtration is shown in Figure 1.

Formally, persistent homology is defined in terms of persistence modules (14), but for our purposes, an intuitive description of persistent homology suffices. Therefore, we proceed here with an intuitive description, deferring a rigorous treatment to Appendix 5.2. A filtration can be visualized mentally as building $\mathcal{K}$ by the addition of cells, and with the addition of cubes during a filtration, homological features are created or destroyed. Those whose addition spawns a homological feature are called positive, while their counterparts that kill features are called negative. It has been shown in (19) that, assuming at most one simplex arises during every nonzero filtration index (which can be assured with data by adding jitter when necessary), each homological feature which occurs during a filtration maps to a pair, (Q_b, Q_d), where Q_b and Q_d are the positive and negative cubes that create and destroy the feature, respectively. The collection $\{(f(Q_b), f(Q_d)\}_{(Q_b,Q_d)\in\mathcal{P}} \cup \Delta$, where $\mathcal{P}$ is the set of all positive-to-negative cube pairs for the filtration of f, and $\Delta := \{(b, d \in \mathbb{R}^2 : b = d)\}$, is known as a persistence diagram (PD), which we henceforth denote by $\mathcal{D}$. The inclusion of the diagonal, Δ, in the definition of a PD is to ensure that bijections between distinct PDs can always be defined, which is important for constructing distances between PDs. The first and second coordinates of points in $\mathcal{D}$ are called birth and death, respectively–so named because they provide the appearance and disappearance "times" of homological features during a filtration. A PD serves as a topological summary of data. Each point in a PD corresponds to a homological feature, and the persistence of the point (i.e., its death coordinate minus its birth coordinate) is related to feature's scale.

PD dissimilarity functions and differentiation. Consider two PDs, $\mathcal{D}$ and $\mathcal{D}'$. It has been shown in (36) that the space of PDs equipped with either the p-Wasserstein or bottleneck metric, defined, respectively, by

$$W_p(\mathcal{D}, \mathcal{D}') := \inf_{\eta:\mathcal{D}\to\mathcal{D}'} \left(\sum_{\mathbf{p}\in\mathcal{D}} \|\mathbf{p} - \eta(\mathbf{p})\|_p^p \right)^{1/p} \tag{8}$$

$$B(\mathcal{D}, \mathcal{D}') := \inf_{\eta:\mathcal{D}\to\mathcal{D}'} \max_{\mathbf{p}\in\mathcal{D}} \|\mathbf{p} - \eta(\mathbf{p})\|_\infty \tag{9}$$

is a Polish space, where $\mathbf{p} := (b, d)$ is a point in a PD, $\|\cdot\|_p$ and $\|\cdot\|_\infty$ denote the Minkowski distance and supremum norm, respectively, and η denotes a bijection between two PDs, $\mathcal{D}$ and $\mathcal{D}'$. The inclusion of the diagonal, $\Delta := \{(b, d \in \mathbb{R}^2 : b = d)\}$, in $\mathcal{D}$ and $\mathcal{D}'$ ensures such a bijection exists. Equations (8) and (9) are examples

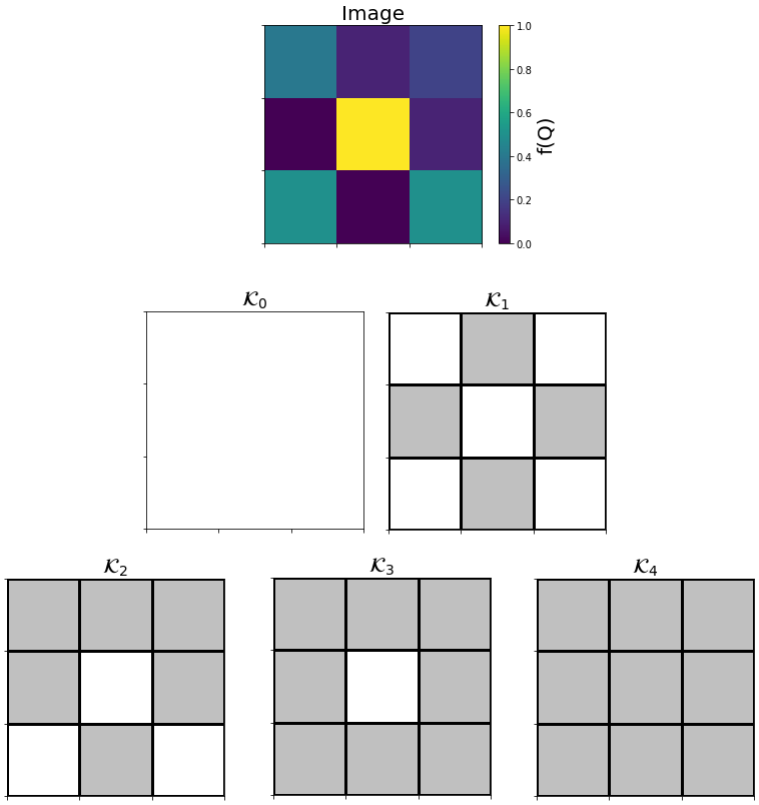


FIGURE 1. Here, we show a lower star filtration of image data. We model the image as a function f on a cubical complex $\mathcal{K}$ whose constituent elementary cubes correspond to pixels. The value $f(Q)$ of a pixel Q is its intensity. At the beginning of the filtration, $\mathcal{K}_0$, there are no cubes present. The four cubes with the lowest intensity appear in $\mathcal{K}_1$. During this time in the filtration, there is also a 1-cycle directly in the center of the image. Since this 1-cycle is not the boundary of any 2-chains, it corresponds to a 1-dimensional homological feature. More cubes are added in $\mathcal{K}_2$ and $\mathcal{K}_3$. Finally, the last cube is added at $\mathcal{K}_4$ and the 1-dimensional feature that appeared at $\mathcal{K}_1$ is annihilated.

of PD dissimilarity functions, which enable one to quantify topological differences between diagrams in a dataset.

For our applications, we omit the diagonal from PDs for computational reasons (see Section 3), and adopt a PD dissimilarity function reminiscent of Equation (8) and other well-studied Wasserstein-like distances (31; 32; 43; 35), namely the minimal-cost matching function (see Figure 2),

$$m(\mathcal{D}';\mathcal{D}) := \min_{\gamma\in\Pi} \sum_{\mathbf{p}\in\mathcal{D}} \|\mathbf{p}_- \gamma(\mathbf{p})\|_2^2, \tag{10}$$

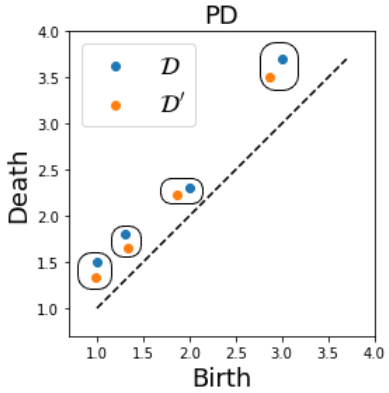


FIGURE 2. The minimal cost matching (Equation (10)) for two PDs, $\mathcal{D}$ and $\mathcal{D}'$.

with $|\mathcal{D}'| \geq |\mathcal{D}|$, where $|\cdot|$ denotes cardinality and Π is the set of injections from $\mathcal{D}$ to $\mathcal{D}'$. The minimizer, γ^* in Equation (10) may be computed via the Hungarian algorithm (29). Fixing $\mathcal{D}$ in Equation (10) yields a functional $m_{\mathcal{D}} : \mathcal{D}' \to \mathbb{R}$ from the space of PDs to the real numbers, and it was shown in (41) that one can define a derivative of $m_{\mathcal{D}}$ with respect to $\mathcal{D}$ that holds almost surely. We summarize the construction of this derivative here as it is important in our framework, but encourage the reader to refer to (41) for a full rigorous treatment. We also include a detailed derivative computation in Appendix 5.2. Let $\mathcal{D} = \{\mathbf{p}_n\}_{n=1}^{|\mathcal{D}|} = \{(b_n, d_n)\}_{n=1}^{|\mathcal{D}|}$, fix $\mathcal{D}'$, and by a slight abuse of notation, denote the first and second components of $\gamma(\mathbf{p})$ by $\gamma(b)$ and $\gamma(d)$, respectively, so in particular $\gamma(\mathbf{p}) = [\gamma(b), \gamma(d)]$. For almost all $\mathcal{D}$, the minimal-cost matching, γ^*, does not change in a neighborhood of $\mathcal{D}$, where a neighborhood is defined by one of the PD metrics, Equations (8) and (9). To be precise, there exists $\epsilon > 0$ such that if we perturb each point in $\mathcal{D}$ to create a new diagram $\tilde{\mathcal{D}}$ satisfying $W_p(\mathcal{D}, \tilde{\mathcal{D}}) < \epsilon$, then $\gamma^*(\mathbf{p}_n) = \gamma^*(\tilde{\mathbf{p}_n})$ for all $\mathbf{p}_n \in \mathcal{D}$ and $\tilde{\mathbf{p}_n} \in \tilde{\mathcal{D}}$. Therefore, $\frac{\partial m}{\partial \mathbf{p}_n}$ is locally well-defined by

$$\frac{\partial m}{\partial \mathbf{p}_n} = \Big[\frac{\partial m}{\partial b_n}, \frac{\partial m}{\partial d_n}\Big] \tag{11}$$

$$= 2\Big[b_n - \gamma^*(b_n),\ d_n - \gamma^*(d_n)\Big] \tag{12}$$

$$= 2\big(\mathbf{p}_n - \gamma^*(\mathbf{p}_n)\big), \tag{13}$$

since $\gamma^*(\mathbf{p}_n) \in \mathcal{D}'$ is constant in Wasserstein/bottleneck neighborhoods of $\mathcal{D}$.

3. **ToFU: Topological Functional Units.** For the remainder of this paper, $\mathbf{x}$ denotes a data point modelled as a function $f : G \to \mathbb{R}$, where $G \subset \mathbb{Z}^d$ is an integer grid. We identify $\mathbf{x}$ with a filtered cubical complex by (i) identifying each element $g \in G$ with an elementary cube of dimension d, say Q_g, (ii) imposing topological structure on the resulting collection of cubes, and (iii) taking the filtration value of Q_g to be $f(g)$. Step (ii) is informed by pre-exisiting topological structure in G on a case-by-case basis. For one-dimensional signals (Section 4.1), $\mathbf{x} = \{x_n\}_{n=0}^N \subset \mathbb{R}$, we define the set of cubes as $\cup_{n=0}^N [n, n+1]$ and take x_n as the filtration value for $[n, n+1]$. For image data (Section 4.2), where $d = 2$, we use a construction as in

Figure 1. $\mathcal{D} = \{\mathbf{p}_n\}_{n=1}^N$ denotes a persistence diagram with an arbitrary ordering of its points, and $[\mathbf{p}_1; \mathbf{p}_2; \ldots; \mathbf{p}_N]$ is the $N \times 2$ matrix defined by stacking elements of $\mathcal{D}$ row-wise. We refer to the latter as a matrix version of $\mathcal{D}$. Moreover, $\mathcal{D}_{\mathbf{x}}$ is the persistence diagram corresponding to a filtration of some data $\mathbf{x}$.

Recall by fixing $\mathcal{D}$ in Equation (10), we obtain a functional $m_{\mathcal{D}} : \mathcal{D}' \to \mathbb{R}$ from the space of PDs to the real numbers. Consider an ANN unit, $\phi_{\boldsymbol{\mathcal{D}}} : \mathbb{R}^d \to \mathbb{R}$, with transformation $T(\mathbf{x}) := \mathcal{D}_{\mathbf{x}}$, and whose activation is given by

$$a_{\boldsymbol{\mathcal{D}}}(\mathcal{D}_{\mathbf{x}}) = \frac{1}{2} m(\mathcal{D}_{\mathbf{x}}; \boldsymbol{\mathcal{D}}), \tag{14}$$

where $\boldsymbol{\mathcal{D}}$ is a persistence diagram that parameterizes $\phi_{\boldsymbol{\mathcal{D}}}$. Using Equation (13), we define the gradient of $\phi_{\boldsymbol{\mathcal{D}}}$ with respect to $\boldsymbol{\mathcal{D}}$, evaluated at $\mathcal{D} = \{\mathbf{p}_n\}_{n=1}^N$, by

$$\frac{\partial}{\partial \boldsymbol{\mathcal{D}}} \phi_{\boldsymbol{\mathcal{D}}} |_{\boldsymbol{\mathcal{D}}=\mathcal{D}} = [\mathbf{p}_1; \mathbf{p}_2; \ldots; \mathbf{p}_N] - [\gamma^*(\mathbf{p}_1); \gamma^*(\mathbf{p}_2); \ldots; \gamma^*(\mathbf{p}_N)], \tag{15}$$

where γ^* is a minimal−cost matching from $\boldsymbol{\mathcal{D}}$ to $\mathcal{D}_{\mathbf{x}}$, in accordance with Equation (10). Since γ^* respects shuffling of indices of the points in $\mathcal{D}$, Equation (15) is well-defined for any ordering of $\mathcal{D}$ up to permutation of rows, and a gradient update of $\boldsymbol{\mathcal{D}}$ may be performed with any ordering of $\mathcal{D}$. Through backpropagation, Equation (15) allows for optimization of loss functions with respect to the topological parameter $\boldsymbol{\mathcal{D}}$. We henceforth refer to the ANN unit $\phi_{\mathcal{D}}$ as **To**pological **F**unctional **U**nit (ToFU). The number of learnable points in ToFU, N in Equation (15) is a hyperparameter. This design decision explains our choice to omit the diagonal from PDs and the cardinality assumption in Equation (10). In particular, excluding the diagonal prevents new points from arising in $\boldsymbol{\mathcal{D}}$ as a result of matches to the diagonal, which fixes the cardinality of $\boldsymbol{\mathcal{D}}$ and thereby limits the number of learnable parameters in the model. Limiting the number of learnable parameters prevents overfitting and expedites training. Moreover, the assumption $|\boldsymbol{\mathcal{D}}| \leq |\mathcal{D}_{\mathbf{x}}|$ allows N to be much smaller than the cardinality of the input diagram $\mathcal{D}_{\mathbf{x}}$, which has the advantage of boosting computational speed at the potential expense of neglecting pertinent topological information.

Example. Here, we investigate ToFU in a classification problem with synthetic data. We generate two classes of PDs; all PDs in both classes have five points. Class 1 PDs are drawn from two distributions. Namely, the two types of PDs in Class 1 are generated by making 5 independent draws from $\mathcal{N}^*((0, 0.3), \sigma I)$ and $\mathcal{N}^*((0.3, 0.6), \sigma I)$, respectively, with $\sigma = 0.1$ and $\mathcal{N}^*$ denoting the bivariate normal distribution truncated to $\{(b, d) \in \mathbb{R}^2 : b < d\}$, chosen so that points in the sampled PDs do not live below the diagonal. Class 2 PDs are sampled by making 5 independent draws from $\mathcal{N}^*((0.6, 0.9), \sigma I)$. Figure 3 shows samples of the Class 1 and Class 2 PDs in blue and orange, respectively. In particular, examples from both types of Class 1 PDs are shown independently as upright and inverted blue triangles. We sample 300 PDs in total; 100 for each type of Class 1 PD, respectively, and 100 Class 2 PDs. Note that this leads to class imbalance, which makes the correct classification of class 2 PDs a more difficult task. The ANN architecture we consider in this problem is summarized in Table 1.

We choose to use only 1 ToFU with 1 learnable point to facilitate visualization of the learned diagram $\boldsymbol{\mathcal{D}}$ over multiple independent training loops. The learnable point is initialized by drawing from a uniform distribution on $\{(b, d) \in [0, 1] : d > b\}$. During training, we use a binary cross-entropy loss function, a learning rate of 0.1, a batch size of 32, and train our ANN for 20 epochs; see Section 2.1. The learned

TABLE 1. The ANN architecture we used for classification of simulated PDs.

Layer	Description
1	ToFU. 1 unit. 1 learnable point.
2	Dense. 32 units. ReLU activations.
3	Dense. 16 units. ReLU activations.
4	Dense. 8 units. ReLU activations.
5	Dense. 1 unit. Sigmoid activation.

1-point diagrams $\boldsymbol{\mathcal{D}}$ for 20 different independent training loops are shown in Figure 3 as x's, and their colors denote test accuracies on an 80/20 training test split.

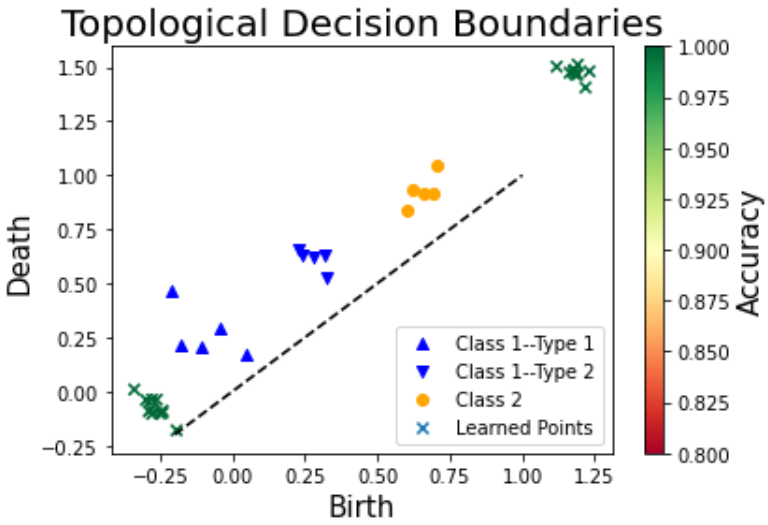


FIGURE 3. One-point PDs learned by ToFU for classification. The learned diagrams are color coded by accuracy. Example PDs from both classes in the classification task are also shown. Class 1 consisted of two types of PDs, depicted as upright and inverted triangles, respectively. Learned PDs corresponding to high classification accuracy fell into two groups– those with birth times earlier and later than points in Class 1 and Class 2, respectively.

All learned PDs $\boldsymbol{\mathcal{D}}$ were able to perfectly separate the test sets. The learned diagrams fell into one of two categories, characterized, respectively, by earlier and later birth times.

Since Equation (14) depends on a minimal cost matching, the initialization method of the learnable points can greatly influence the learned parameter $\boldsymbol{\mathcal{D}}$. We illustrate the role of point initialization with three examples. In each example, we use Equation (15) to minimize

$$L(\boldsymbol{\mathcal{D}}) := \frac{1}{|\mathscr{D}|} \sum_{\mathcal{D} \in \mathscr{D}} m(\mathcal{D}; \boldsymbol{\mathcal{D}}) \tag{16}$$

via gradient descent, where $\mathscr{D}$ denotes a fixed set of PDs. Equation (16) is reminiscent of mean-squared error, and we may loosely interpret our approximate minimum to Equation (16) as a "PD of best fit" to the set $\mathscr{D}$.

Figure 4 shows an example where point initialization determines the solution; we consider a PD with two points (shown respectively as a blue triangle and orange dot) and a learnable diagram $\boldsymbol{\mathcal{D}}$ with one point. Two separate initializations and their corresponding gradient updates are shown in green and blue, respectively).

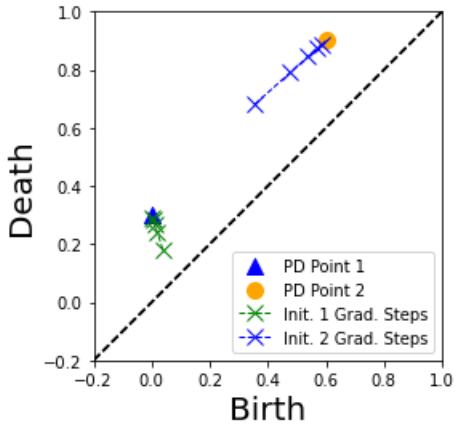


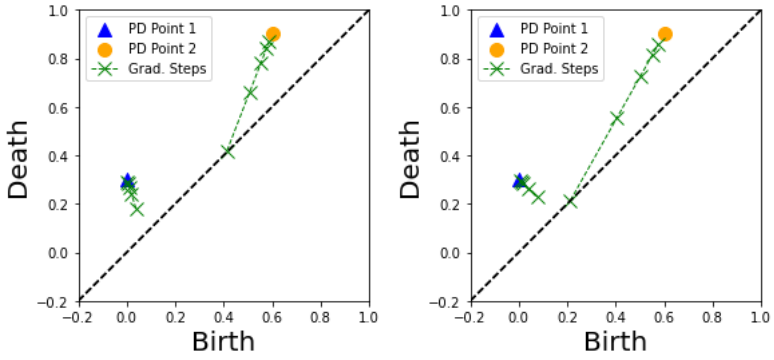
FIGURE 4. Gradient descent with ToFU layer using Equation (15). Because the gradient depends on a minimal−cost matching, initialization of weights determines the solution when there are fewer learnable points than those in data.

Another extreme, one in which a unique solution for $\boldsymbol{\mathcal{D}}$ exists and hence point initialization has no effect, is depicted in Figure (5). Here, we once again consider a PD with two points, but now the learnable diagram $\boldsymbol{\mathcal{D}}$ also has two points (shown in green). Two optimizations with different initialization for $\boldsymbol{\mathcal{D}}$ are shown in Figure 5(a) and (b), respectively. Notice in Figure 5(b) that although both learnable points are initialized closer to the blue point than to the orange point, the minimal-cost matching ensures that only a single learnable point is paired to the blue point during gradient updates. In this case, the learnable points always move toward the PD point to which they are initially matched.

Finally, we consider an example with multiple PDs (shown as upright and inverted triangles, respectively) in Figure (6). Each PD in Figure 6, as well as the learnable PD $\boldsymbol{\mathcal{D}}$, has three points. Notice in Figure 6(a) and (b) that the learnable points in $\boldsymbol{\mathcal{D}}$ move to the region between both PDs. In general, one expects $\boldsymbol{\mathcal{D}}$ to converge to a barycenter of PDs in the training data.

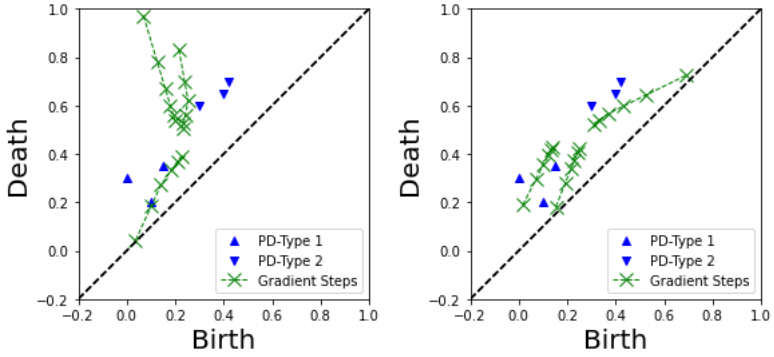
In all three examples, ToFU learns a sensible value for $\boldsymbol{\mathcal{D}}$, but Figures 4 and 6 demonstrate that the value for $\boldsymbol{\mathcal{D}}$ is not necessarily unique. The nonuniqueness does not preclude the use of ToFU in ANNs, which can learn useful relationships in data despite the use of nonconvex loss functions (with respect to network parameters). For a detailed discussion of nonconvex loss functions and ANNs, see (13).

3.1. **Topologically−Based Encodings.** In this section, we present an architecture for a variational autoencoder utilizing ToFU to create latent representations



(A) (B)

FIGURE 5. Gradient descent with ToFU layer that has two learnable points. Different initializations are shown in (a) and (b).



(A) (B)

FIGURE 6. Gradient descent to minimize the average of Equation (10) for two PDs using a ToFU layer that has three learnable points. Different initializations are shown in (a) and (b).

that encode topological information about data. The novel part of our architecture is the encoder, which we now describe in detail.

The first step performed in our encoder sends the input data $\mathbf{x} \in \mathbb{R}^d$ through a layer comprised of ToFUs whose activations are denoted, respectively, by $\phi_{\boldsymbol{\mathcal{D}}_1}, \phi_{\boldsymbol{\mathcal{D}}_2}, \ldots, \phi_{\boldsymbol{\mathcal{D}}_C} : \mathbb{R}^d \to \mathbb{R}$, where C is the user-selected number of units. We define $\phi(\mathbf{x}) := (\phi_{\boldsymbol{\mathcal{D}}_i}(\mathbf{x}))_{i=1}^{C} \in \mathbb{R}^C$, and two pairs of multi-layer ANNs, $\mu_{\mathbf{x}}, \sigma_{\mathbf{x}} : \mathbb{R}^d \to \mathbb{R}^h$ and $\mu_\phi, \sigma_\phi : \mathbb{R}^C \to \mathbb{R}^{h_T}$. The next step in our encoder passes $\mathbf{x}$ to $\mu_{\mathbf{x}}$ and $\sigma_{\mathbf{x}}$, as well as ϕ to μ_ϕ and σ_ϕ, then both pairs are used to parameterize the latent distribution $q(\mathbf{z}|\mathbf{x})$. In particular,

$$q(\mathbf{z}|\mathbf{x}) = \mathcal{N}\Big(\mathbf{z}; \mu, \Sigma\Big) \tag{17}$$

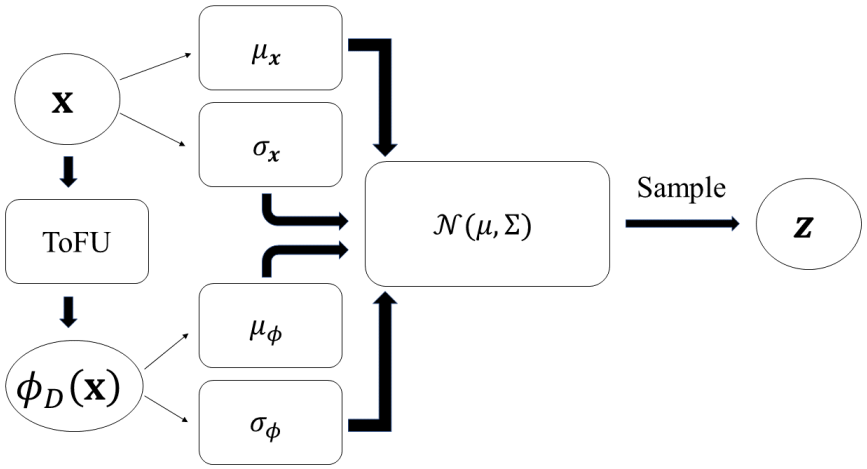


FIGURE 7. A schematic of the encoder in our topological VAE architecture. Here, we choose C = 1 in ϕ for simplicity.

where $\mu := (\mu_{\mathbf{x}}, \mu_\phi) \in \mathbb{R}^{h+h_T}$ is the vector whose first h elements and last h_T elements are $\mu_{\mathbf{x}}$ and μ_ϕ, respectively, $\Sigma := \text{diag}(\sigma_{\mathbf{x}}, \sigma_\phi) \in \mathbb{R}^{(h+h_T)\times(h+h_T)}$ is the diagonal matrix whose first h and last h_T diagonal entires are $\sigma_{\mathbf{x}}$ and σ_ϕ, respectively, and $\mathcal{N}\Big(\mathbf{z}; \mu, \Sigma\Big)$ denotes the $(h + h_T)-$dimensional Gaussian density with mean μ and covariance matrix Σ. Our topological encoder is summarized in Figure 7. The discussion around Equation (17) is summarized in the following proposition.

Proposition 1. Let $\mathbf{x} \in \mathbb{R}^d$, consider a persistence diagram $\mathcal{D}_x$ computed from a filtration constructed with $\mathbf{x}$, and define $\phi(\mathbf{x}) := (\phi_{\boldsymbol{\mathcal{D}}_i}(\mathbf{x}))_{i=1}^C \in \mathbb{R}^C$, where each $\phi_{\boldsymbol{\mathcal{D}}_i} : \mathbb{R}^d \to \mathbb{R}$ is given by Equation (14). Suppose further we have two pairs of multilayer ANNs, $\mu(\mathbf{x}), \sigma(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^h$ and $\mu_T(\phi), \sigma_T(\phi) : \mathbb{R}^C \to \mathbb{R}^{h_T}$, and consider the distribution $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}\Big(\mathbf{z}; (\mu, \mu_T), \text{diag}(\sigma, \sigma_T)\Big)$. Then, the last C dimensions of $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$ only depend on $\mathcal{D}_{\mathbf{x}}$.

Proposition 1 follows by noticing $q(\mathbf{z}|\mathbf{x})$ in Equation (17) is an uncorrelated multivariate Gaussian density then inspecting its marginal distributions. Using our encoder, one can incorporate topological information about the input data into latent space representations in an interpretable manner.

With our encoder in hand, we are free to select any architecture suitable for the data to build a decoder. When $\mathbf{x} \in \mathbb{R}^d$ is grey-scale image data, given by a $d = H \times W$ array of pixels, the architecture we adopt for the decoder is as follows. The output of the encoder, $\mathbf{z}$, is passed through to a multi-layer ANN $\Phi_{\text{dec}} : \mathbb{R}^{h+h_T} \to \mathbb{R}^d$, which parameterizes the data likelihood,

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{B}(\Phi_{\text{dec}}), \tag{18}$$

where $\mathcal{B}(\Phi_{\text{dec}})$ is the d-dimensional Bernoulli distribution with independent marginals.

4. Experiments.

In this section, we investigate ToFU in two experimental settings. First, we consider ToFU in a classifcation problem, then explore its use in a variational autoencoder.

PDs have varying cardinalities so to ensure fixed input sizes to our ANNs, we pad PDs with zeros when necessary. ToFU is implemented in TensorFlow 2.0, and we intend to make the code publically available. All of our demos and experiments were performed on a CPU.

4.1. **Signal Classification Using ToFU.** We consider a classification problem with discrete time autoregressive (AR) signals, which have been widely studied in signal processing (40). It was shown in (21) the power spectral densities (PSDs) of AR signals can be explictly computed from AR simulation parameters, and we use this fact to create distinct collections of signals $\mathcal{A}_j$ for $j = 1, 2, 3, 4$, each characterized by nonzero peaks in their PSDs at 6 Hz, 10 Hz, 14 Hz, and 21 Hz, respectively. These specific frequency choices are motivated by their interest amongst neuroscientists and neurologists who study human brain activity using electroencephalography (EEG) signals (8), and the capabilities of AR models to simulate EEG signals (24). For each j, $|\mathcal{A}_j| = 800$, and signals in $\mathcal{A}_j$ have the same non-zero peak in their PSDs, with varying damping factors sampled uniformly from $\{4, 5, \ldots, 32\}$. Hence, the $\mathcal{A}_j$ are comprised of signals that oscillate at the same frequency, but with different strengths. Every signal also has a fixed-width PSD peak at zero to simulate the monotonically decreasing PSDs that occur in a wide variety of natural signals (this phenomenon is known as $1/f$ behavior in literature).

We use our method to distinguish the different $\mathcal{A}_j$ based on the sublevel set topologies of their constituent signals. PDs are computed for the signals in each $\mathcal{A}_j$, which we denote by $\mathscr{D}_j$. With $\mathscr{D}_j$ in hand, we built a 4-layer ANN (3 hidden, 1 output) to classify each PD. The first layer of our ANN consists of a ToFU layer. To measure the accuracy of our classifier, we used an $80 - 20$ split with each $\mathscr{D}_j$ to build training and test sets. For the sake of comparison, we consider three other ANNs whose architecture has the same last three layers, but the ToFU layer replaced. The first of these has an untrainable layer that computes PSDs with Welch's method (44), which approximates PSDs by averaging periodograms from overlapping windows of the signals. The second network, which we refer to as Conv1, has a single convolutional layer with 8 channels and filter sizes of 3, followed by average pooling with the same filter size, which downsamples an image through windowed averaging. The third network, which we call Conv2, has 64 channels with filter sizes of 128, followed by average pooling; the rationale behind this network was that it could theoretically learn to compute periodograms in a single layer. The difference at the beginning of each of these networks largely has to do with the type of data it accepts as input. The Welch, Conv1, and Conv2 networks all take 1-dimensional signals as input, and hence 1-dimensional convolutions were chosen to learn feature maps in the initial layer since 1-dimensional convolutions (i) require 1-dimensional signals as input and (ii) have been well-studied as a method to compute machine learning features of signals. Each of these networks also used a non-learnable average pooling layer to downsample, which is a standard practice, as downsampling is well-known to improve performance in convolutional networks. Conversely, the ToFU network takes persistence diagrams as input, so we used a ToFU layer, as opposed to convolution, to learn a vectorization. In each of these networks, the final 3 layers after the initial vectorization layer were fixed. Indeed, we used this high level design precisely to control for the structure of each network.

To compare our method to other topological approaches, we consider two more networks that are trained on persistence diagram vectorizations. The first of these networks employs a convolutional architecture with persistence images (PIs) (1).

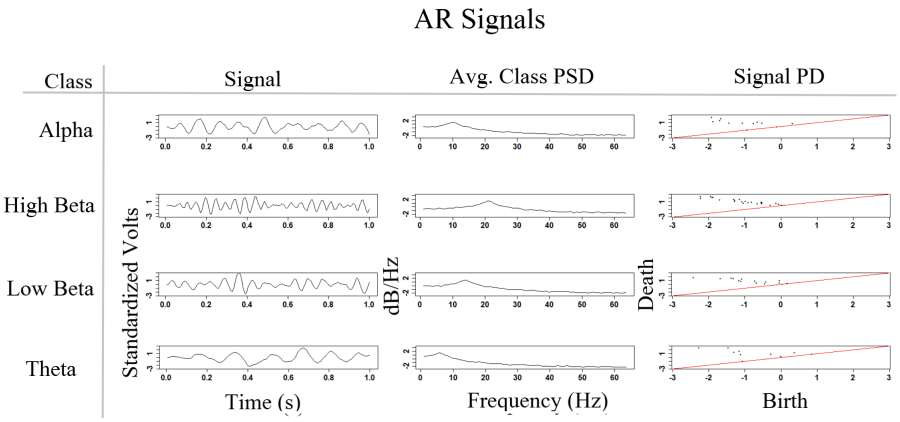


FIGURE 8. Examples from the AR signals dataset. Here, we show signals with damping factor β=4. The average log PSD for each class, estimated by averaging periodograms, is shown in the second column.

This network requires 2-dimensional signal inputs, i.e. pixelated images, which precludes the use of one-dimensional convolutions. Consequently, we employ a 2-dimensional convolutional architecture based on the ubiquitous LeNet-5 CNN (30) using PIs with a resolution of 20×20 as input. This architecture is specified by sequential layers:

1. Convolutional layer with 6 channels, a filter size of 5×5 and a stride of 1.
2. Average pooling.
3. Convolutional layer with 16 channels, a filter size of 5×5 and a stride of 1.
4. Average pooling.
5. Dense layer with 120 units and ReLU activations.
6. Dense layer with 84 units and ReLU activations.
7. Dense layer with 4 units and softmax activation.

The second topological vectorization network is trained on persistence landscapes (PLs) (7). Since PLs can be represented as one-dimensional signals, we once again use the Conv1 architecture when training on PLs. Both PIs and PLs have hyperparameters that require tuning. For PIs, we train with three different bandwidths over different scales, 0.1, 1, and 10. Similarly, for PLs, we train with three different landscape numbers, 1, 5 and 10. For both vectorizations, we use the hyperparameters with the highest training accuracy. Each network is trained for 100 epochs with a learning rate of 0.001, and the highest test accuracies achieved by each during training are reported in Table 2.

We observe that the ToFU network is competitive with the one that used hand-designed features, and that it outperforms both convolutional and PD vectorization networks. We acknowledge that deeper CNNs would likely have competitive performance with the ToFU network, but at the expense of more complicated network architectures.

4.2. **Variational Autoencoder.** In this section, we present an example demonstrating how ToFU can learn latent space encodings in variational autoencoders that

TABLE 2. Test accuracies for each ANN trained for AR signal classification. Conv1 and Conv2 refer to the 8 and 64 channel networks, respectively, while PLs and PIs refer to the networks trained on persistence landscapes and persistence images.

Model	Test Accuracy
Welch	98.91
ToFU	98.12
PLs	96.41
PIs	95.94
Conv1	92.66
Conv2	88.12

richly incorporate topological information about the input data. Please see Section 2.1 for overview of variational autoencoders, and (28) for a thorough introduction.

For this experiment, we create a synthetic categorical dataset of images whose classes are topologically distinguishable; see Figure 9. Our dataset mimics observations of die rolls on a flat surface. The Die dataset consists of 6 classes $\mathcal{C}_k$, each characterized by a fixed number k of nonzero pixels. Elements of $\mathcal{C}_k$ are generated by the following procedure:

1. Make k independent draws from $\mathcal{U}[0.85, 2]$.
2. Arrange the k samples in a fixed pattern at the center of a 21-by-21 grid.
3. Apply random vertical and horizontal translations to the grid.

Each nonzero pixel in elements of $\mathcal{C}_k$ corresponds to a homological feature of degree 1 in the cubical filtration. Hence, 1-dimensional homology readily distinguishes the classes. Moreover, since the 1-dimensional PDs are invariant under translations of the grid, they serve as natural descriptors for $\mathcal{C}_k$.

We generate 1000 examples for each class and compute their 1-dimensional PDs, $\mathcal{C}_k = \{\mathcal{C}_k^i\}_{i=1}^{1000}$ and $\mathcal{D}_k = \{\mathcal{D}_k^i\}_{i=1}^{1000}$. The $\mathcal{C}_k$ and $\mathcal{D}_k$ are used to train a variational autoencoder (VAE) that uses ToFU to learn a latent space representation; see Section 3.1. We henceforth refer to this VAE as the ToFU variational autoencoder (ToFU-VAE). The encoder for ToFU-VAE follows the architecture laid out in Section 3.1 with: i) $C = 1$ in the ToFU layer, where the number of learnable points equals 6, ii) $\mu_{\mathbf{x}}, \sigma_{\mathbf{x}}, \mu_\phi$, and σ_ϕ specified by two-layer ANNs, with dense layers comprised of 64 units that have ReLU activations, and 3) the dimensions that define the latent space given by $h = h_T = 1$. We select a 2-dimensional latent space for ease of visualization. The decoder for ToFU-VAE also follows the architecture described in Section 3.1, with Φ_{dec} specified by an ANN with two dense layers, the first of which has 64 units with ReLU activations, and the second has $21 \times 21 = 441$ units with sigmoid activations.

For the sake of comparison, we also train a typical VAE without ToFU. The encoder-decoder architecture for the typical VAE is the same as that of ToFU-VAE except that: i) the ToFU layer, μ_ϕ, and σ_ϕ are omitted, ii) $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu_{\mathbf{x}}, \Sigma_{\mathbf{x}})$; in particular, the latent space distribution only depends on $\mathbf{x}$ through $\mu_{\mathbf{x}} : \mathbb{R}^d \to \mathbb{R}^h$ as well as another two-layer ANN with the same architecture as μ_x that outputs a diagonal covariance matrix, $\Sigma_{\mathbf{x}} : \mathbb{R}^d \to \mathbb{R}^{h \times h}$, and iii) the output dimension of μ_x is

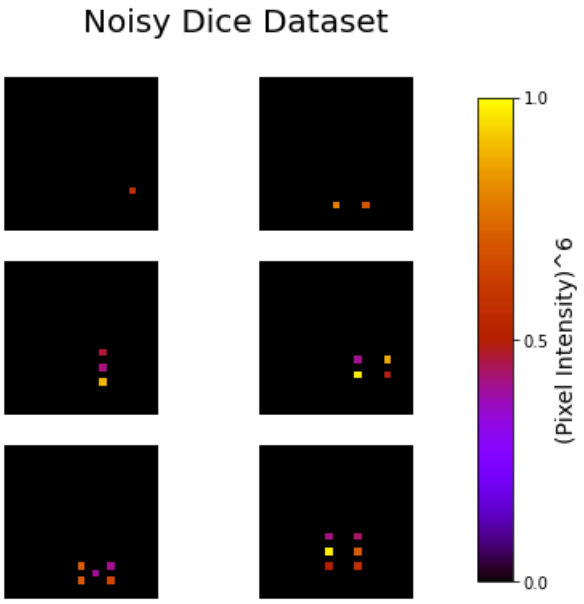


FIGURE 9. Shown above is a single example from each of the six classes in our synthetic dataset. We apply a nonlinear transformation to pixel values for visual clarity.

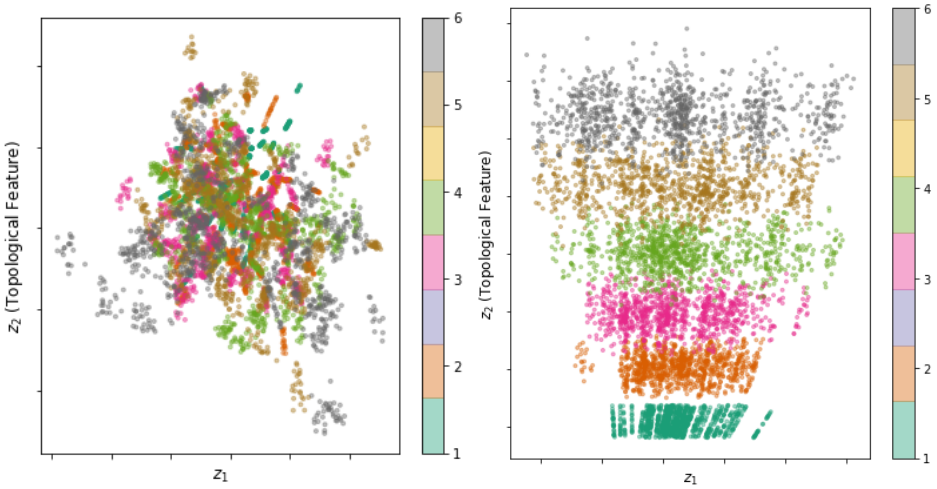
increased to 2 so that the latent space dimension of the typical VAE matches that of ToFU-VAE.

Both ToFU-VAE and the typical VAE are trained for 2 epochs with a batch size of 32 and a learning rate of 0.001. The resulting latent space representations are shown in Figure 10, where we have explicitly labeled the topological dimension in (b), which depicts the latent space of ToFU-VAE.

Figure 10 shows a marked difference in the latent space representation of the typical VAE and ToFU-VAE. As expected, the topological dimension of ToFU-VAE (z_2 in Figure 10(b)) completely encodes the topology of each class. This leads to a more interpretable latent space representation than that produced by the typical VAE. Namely, ToFU-VAE produced a joint distribution whose marginals describe input data with similar global topology. Moreover, Table 3 shows that this increased interpretability incurs no cost in reconstruction error.

TABLE 3. Test reconstruction errors for both VAEs.

ANN	Test Recon. Err.
Typical VAE	0.0847
ToFU-VAE	0.0806



(A) (B)

FIGURE 10. Latent space representations of the (a) typical VAE and (b) ToFU-VAE. The ToFU-VAE latent space shows clear separation based on the topology of each class.

5. **Discussion and Future Directions.** We have introduced a new ANN unit, ToFU, that is parameterized by a learnable PD and employs a PD dissimilarity function as its activation. Our examples demonstrated that ToFU learns pertinent topology in data, which may be leveraged for data science applications that bear topology in mind.

Our classification example in Section 3 depicted how ToFU differentiates topologically-distinct classes in the space of PDs. In particular, ToFU learns a PD whose topological distance to PDs in the data differs across classes. Section 4.1 considered a signal classification problem inspired by neuroscience. In this experiment, ToFU achieved competitive performance with spectral features without relying on deep networks, a feat that was unmatched by CNNs. The signal classification problem exemplifies that ToFU learns high-level descriptors of data useful by their inherent nature. Finally, we used ToFU to create a novel variational autoencoder (VAE) whose latent space marginal distrbutions are solely dictated by the topology of data. We showed how our new VAE produces interpretable latent space representations without sacrificing reconstruction accuracy on a synthetic dataset.

The features learned by ToFU are invariant to a set of transformations that preserve the large-scale topological structure in inputs, for example rotations and translations. Such transformations are commonly encountered in data science problems, wherein something like an image may be rotated or translated, but its class label remains unaffected (we considered examples of this nature in Sections 4.1 and 4.2, where, respectively, phase shifts of signals and image translations did not alter class labels). To help ANNs recognize these transformations with limited training data at their disposal, practitioners often rely on data augmentation to generate synthetic training examples. ToFU reduces the need for data augmentation since roto-translational invariant features are learned by design. As a future direction,

we will investigate more sophisticated methods for using ToFU in data augmentation, for example by using ToFU-VAE as a generative model to create training data with desired topological characteristics. Additionally, it will be beneficial to compare ToFU to a larger collection of TDA methods, for example sliced Wasserstein kernels (10) or persistence codebooks (50), on a larger repository of datasets.

In isolation, gradient-based optimization of Equation (14) is closely-related to finding the Fréchet mean of PDs with respect to the Wasserstein metric; this is a well-studied problem (36), and a Fréchet mean for a finite collection of PDs is known to exist, although not necessarily a unique one. A later work (46) introduced a gradient-descent algorithm to find Fréchet means, and established its convergence to a local minimum under mild conditions. The issue of nonuniqueness is considered in (39) wherein the authors propose the Probabilistic Fréchet Mean (PFM). The PFM is interpreted as a probabilistic mixture of PDs and, unlike the typical Fréchet mean, is unique. While our work does not use the gradient-descent algorithm from (46) or PFMs from (39), incorporating them into our framework constitute an interesting areas for further research. From a computational standpoint, (15) introduces an algorithm for the fast computation of Wasserstein barycenters that is amenable to GPU computations, and leveraging this work in the implementation of ToFU can reap computational benefits potentially executable within a quantum framework. While the examples considered in this paper employ ToFU directly on the input data, there is nothing in principle that prevents ToFU's use in deeper layers of an ANN. In the future, we will investigate the use ToFU to discover informative topological structure in hidden representations of ANNs.

Deep learning benefits from sound inductive biases. Therefore, the use of topological descriptors in ANNs can augment performance whenever topology is a defining characteristic in data. In this fashion, ToFU is a step toward harmonizing the expressive capabilities of deep learning with high-level mathematical intuitions of data.

Acknowledgments. All authors would like to thank an anonymous reviewer for their comments, which substangtially improved this manuscript. CO's research was sponsored by the Army Research Laboratory (ARL) and was accomplished under Cooperative Agreement Number W911NF-19-2-0302. VM's work was partially supported by the ARO W911NF-21-1-0094, NSF DMS-1821241, DMS-2012609, and ARL and was accomplished under Cooperative Agreement Number W911NF-19-2-0328. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES.

[1] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta and L. Ziegelmeier, Persistence images: A stable vector representation of persistent homology, *The Journal of Machine Learning Research*, **18** (2017), 218–252.

[2] R. J. Adler and S. Agami, Modelling persistence diagrams with planar point processes, and revealing topology with bagplots, *Journal of Applied and Computational Topology*, **3** (2019), 139–183.

[3] J.-B. Bardin, G. Spreemann and K. Hess, Topological exploration of artificial neuronal network dynamics, *Network Neuroscience*, **3** (2019), 725–743.
[4] E. Berry, Y.-C. Chen, J. Cisewski-Kehe and B. T. Fasy, Functional summaries of persistence diagrams., *J. Appl. Comput. Topol.*, **4** (2020), 211–262.
[5] C. A. Biscio and J. Møller, The accumulated persistence function, a new useful functional summary statistic for topological data analysis, with a view to brain artery trees and spatial point process applications, *Journal of Computational and Graphical Statistics*, **28** (2019), 671–681.
[6] R. Brüel-Gabrielsson, B. J. Nelson, A. Dwaraknath, P. Skraba, L. J. Guibas and G. Carlsson, A topology layer for machine learning, *arXiv preprint arXiv:1905.12200*.
[7] P. Bubenik, Statistical topological data analysis using persistence landscapes, *The Journal of Machine Learning Research*, **16** (2015), 77–102.
[8] G. Buzsaki, *Rhythms of the Brain*, Oxford University Press, 2006.
[9] G. Carlsson, Topology and data, *Bulletin of the American Mathematical Society*, **46** (2009), 255–308.
[10] M. Carriere, M. Cuturi and S. Oudot, Sliced wasserstein kernel for persistence diagrams, in *International Conference on Machine Learning*, PMLR, 2017, 664–673.
[11] F. Chazal, B. T. Fasy, F. Lecci, A. Rinaldo and L. Wasserman, Stochastic convergence of persistence landscapes and silhouettes, in *Proceedings of the thirtieth annual symposium on Computational geometry*, 2014, 474–483.
[12] F. Chazal and V. Divol, The density of expected persistence diagrams and its kernel based estimation, 2019.
[13] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous and Y. LeCun, The loss surfaces of multilayer networks, in *Artificial intelligence and statistics*, 2015, 192–204.
[14] W. Crawley-Boevey, Decomposition of pointwise finite-dimensional persistence modules, *Journal of Algebra and its Applications*, **14** (2015), 1550066.
[15] M. Cuturi and A. Doucet, Fast computation of wasserstein barycenters, vol. 32 of Proceedings of Machine Learning Research, PMLR, Bejing, China, 2014, 685–693, URL `http://proceedings.mlr.press/v32/cuturi14.html`.
[16] D. S. Dummit and R. M. Foote, *Abstract algebra*, vol. 3, Wiley Hoboken, 2004.
[17] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction*, American Mathematical Society, Providence, RI, 2010.
[18] H. Edelsbrunner, D. Letscher and A. Zomorodian, Topological persistence and simplification, in *Proceedings 41st annual symposium on foundations of computer science*, IEEE, 2000, 454–463.
[19] H. Edelsbrunner, D. Letscher and A. Zomorodian, Topological persistence and simplification, in *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, IEEE, 2000, 454–463.
[20] B. T. Fasy, F. Lecci, A. Rinaldo, L. Wasserman, S. Balakrishnan, A. Singh et al., Confidence sets for persistence diagrams, *The Annals of Statistics*, **42** (2014), 2301–2339.
[21] P. J. Franaszczuk and K. J. Blinowska, Linear model of brain electrical activity—eeg as a superposition of damped oscillatory modes, *Biological cybernetics*, **53** (1985), 19–25.
[22] R. B. Gabrielsson and G. Carlsson, Exposition and interpretation of the topology of neural networks, in *2019 18th IEEE International Conference On*

Machine Learning And Applications (ICMLA), IEEE, 2019, 1069–1076.

[23] R. Ghrist, Barcodes: the persistent topology of data, *Bulletin of the American Mathematical Society*, **45** (2008), 61–75.

[24] S. Gordon, P. Franaszczuk, W. Hairston, M. Vindiola and K. McDowell, Comparing parametric and nonparametric methods for detecting phase synchronization in eeg, *Journal of Neuroscience Methods*, **212** (2013), 247–258.

[25] K. Gurney, *An introduction to neural networks*, CRC press, 1997.

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

[27] T. Kaczynski, K. Mischaikow and M. Mrozek, *Computational homology*, vol. 157, Springer Science & Business Media, 2006.

[28] D. P. Kingma and M. Welling, An introduction to variational autoencoders, *arXiv preprint arXiv:1906.02691*.

[29] H. W. Kuhn, The hungarian method for the assignment problem, *Naval research logistics quarterly*, **2** (1955), 83–97.

[30] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, **86** (1998), 2278–2324.

[31] A. Marchese and V. Maroulas, Signal classification with a point process distance on the space of persistence diagrams, *Advances in Data Analysis and Classification*, **12** (2018), 657–682.

[32] V. Maroulas, C. P. Micucci and A. Spannaus, A stable cardinality distance for topological classification, *Advances in Data Analysis and Classification*, **14** (2020), 611–628.

[33] V. Maroulas, J. L. Mike and C. Oballe, Nonparametric estimation of probability density functions of random persistence diagrams., *Journal of Machine Learning Research*, **20** (2019), 1–49.

[34] V. Maroulas, F. Nasrin and C. Oballe, A bayesian framework for persistent homology, *SIAM Journal on Mathematics of Data Science*, **2** (2020), 48–74.

[35] F. Mémoli, The gromov–wasserstein distance: A brief overview, *Axioms*, **3** (2014), 335–341.

[36] Y. Mileyko, S. Mukherjee and J. Harer, Probability measures on the space of persistence diagrams, *Inverse Problems*, **27** (2011), 124007.

[37] A. Monod, S. Kalisnik, J. A. Patino-Galindo and L. Crawford, Tropical sufficient statistics for persistent homology, *SIAM Journal on Applied Algebra and Geometry*, **3** (2019), 337–371, URL `http://dx.doi.org/10.1137/17M1148037`.

[38] M. Moor, M. Horn, B. Rieck and K. Borgwardt, Topological autoencoders, *arXiv preprint arXiv:1906.00722*.

[39] E. Munch, K. Turner, P. Bendich, S. Mukherjee, J. Mattingly and J. Harer, Probabilistic fréchet means for time varying persistence diagrams, *Electronic Journal of Statistics*, **9** (2015), 1173–1204.

[40] A. V. Oppenheim, J. R. Buck and R. W. Schafer, *Discrete-time signal processing. Vol. 2*, Upper Saddle River, NJ: Prentice Hall, 2001.

[41] A. Poulenard, P. Skraba and M. Ovsjanikov, Topological function optimization for continuous shape matching, in *Computer Graphics Forum*, vol. 37, Wiley Online Library, 2018, 13–25.

[42] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, Cambridge university press, 2014.

[43] P. Skraba and K. Turner, Wasserstein stability for persistence diagrams, *arXiv preprint arXiv:2006.16824*.
[44] O. Solomon Jr, Psd computations using welch's method, *STIN*, **92** (1991), 23584.
[45] I. Tolstikhin, O. Bousquet, S. Gelly and B. Schoelkopf, Wasserstein auto-encoders, *arXiv preprint arXiv:1711.01558*.
[46] K. Turner, Y. Mileyko, S. Mukherjee and J. Harer, Fréchet means for distributions of persistence diagrams, *Discrete & Computational Geometry*, **52** (2014), 44–70.
[47] K. Turner, S. Mukherjee and D. M. Boyer, Persistent homology transform for modeling shapes and surfaces, *Information and Inference: A Journal of the IMA*, **3** (2014), 310–344.
[48] H. Wagner, C. Chen and E. Vuçini, Efficient computation of persistent homology for cubical data, in *Topological methods in data analysis and visualization II*, Springer, 2012, 91–106.
[49] P. J. Werbos, *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, vol. 1, John Wiley & Sons, 1994.
[50] B. Zieliński, M. Lipiński, M. Juda, M. Zeppelzauer and P. Dłotko, Persistence codebooks for topological data analysis, *Artificial Intelligence Review*, **54** (2021), 1969–2009.
[51] A. Zomorodian and G. Carlsson, Computing persistent homology, *Discrete & Computational Geometry*, **33** (2005), 249–274.

Received xxxx 20xx; revised xxxx 20xx.

Appendix.

5.1. **Variational autoencoders.** In this section, we derive the relationship between the Kullback-Leibler divergence and evidence lower bound (ELBO) given in Equations (5) and Equation (6), respectively. To this end, notice

$$
\begin{aligned}
D_{KL}(q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) &= -\mathbb{E}_{q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})}\Big(\log \frac{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})}\Big) && (19)\\
&= -\mathbb{E}_{q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})}\Big(\log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) - \log q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})\Big) && (20)\\
&= -\mathbb{E}_{q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})}\Big(\log p(\mathbf{z}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) && (21)\\
&\qquad\qquad - \log p_{\boldsymbol{\theta}}(\mathbf{x}) - \log q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})\Big)\\
&= \log p_{\boldsymbol{\theta}}(\mathbf{x}) \underbrace{-\mathbb{E}_{q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})}\Big(\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\Big) + D_{KL}(q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{-ELBO(\boldsymbol{\theta},\boldsymbol{\vartheta})}, && (22)
\end{aligned}
$$

where Equation (21) follows from Bayes' rule and Equation (22) follows by linearity of the expectation and the fact that $p_{\boldsymbol{\theta}}(\mathbf{x})$ does not depend on the density $q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})$. From Equation (22), we deduce

$$
\begin{aligned}
ELBO(\boldsymbol{\theta},\boldsymbol{\vartheta}) &= \log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{KL}(q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) && (23)\\
&\leq \log p_{\boldsymbol{\theta}}(\mathbf{x}). && (24)
\end{aligned}
$$

The inequality in (24) follows since Kullback-Leibler divergence is always non-negative, and the quantity $p_{\boldsymbol{\theta}}(\mathbf{x})$ is known as the evidence of the model $p_{\boldsymbol{\theta}}$ given

data $\mathbf{x}$. Equations (23) and (24) establish the ELBO as a lower bound for the log likelihood of the model $p_\theta(\mathbf{x})$ given data $\mathbf{x}$, and show that the bound becomes tight whenever $D_{KL}(q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) = 0$. Consequently, maximizing the ELBO increases the evidence of the model while reducing Kullback-Leibler divergence, $D_{KL}(q_{\boldsymbol{\vartheta}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))$.

5.2. **Computational topology.**

5.2.1. *Quotient spaces.* Let V be a vector space over a field F and suppose N is a subspace of V. Given $v \in V$, we define an equivalence class, $[v]$, by $[v] := \{v + n : n \in N\}$. The quotient space V/N is defined as the collection of equivalence classes $\{[v] : v \in V\}$ along with the addition and scalar multiplication operations:

$$[v] + [w] := [v + w], \quad v, w \in V \tag{25}$$

$$\lambda[v] := [\lambda v], \quad \lambda \in F. \tag{26}$$

It can be shown that Equations (25) and (26) are well-defined (16), and moreover that V/N is a vector space. Informally, V/N represents the vector space that one obtains from V by "zeroing out" the elements in N. As a simple example, we can consider $V = \mathbb{R}^2$ and $N = \{(x, 0) : x \in \mathbb{R}\}$ (the x-axis) over $\mathbb{R}$ with standard addition and multiplication. Then, for $(x, y) \in \mathbb{R}^2$, we have $[(x, y)] := \{(x + r, y) : r \in \mathbb{R}\} = \{(x, y) \in \mathbb{R}^2 : x \in \mathbb{R}\}$. We observe that our equivalence classes are the lines parallel to the x-axis, which are entirely parameterized by their y-coordinate. In a sense, we have "zeroed out" the x-coordinate to obtain a new one-dimensional vector space that has the same structure as $\mathbb{R}$. Formally, we say $\mathbb{R}^2/\mathbb{R}$ is isomorphic to $\mathbb{R}$ and write $\mathbb{R}^2/\mathbb{R} \simeq \mathbb{R}$.

5.2.2. *Homology example.* We explicitly compute the homology group $H_1 := \ker \partial_1 / \mathrm{Im} \partial_2$ over $\mathbb{Z}_2$ for a simple cubical complex to illustrate that homological computations boil down to abstract linear algebra.

Consider a cubical complex, $\mathcal{K}$, comprised of 0-,1-, and 2-dimensional cubes that we denote by $\{v_1, v_2, v_3, v_4, v_5, v_6\}, \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, and $\{f_1\}$, respectively; see Figure 11. To compute H_1, we need to construct its associated boundary operators, ∂_1 and ∂_2, which can be represented as matrices since the boundary operator is linear. In particular,

$$\partial_1 = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}, \text{ and } \partial_2 = \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \end{array} \begin{array}{c} \begin{array}{c} f_1 \end{array} \\ \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{array}, \tag{27}$$

by computing each boundary operator using $\{v_1, v_2, v_3, v_4, v_5, v_6\}, \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, and $\{f_1\}$ as respective bases for the 0-, 1-, and 2-chains. Next, we compute $\ker \partial_1$ by constructing a basis for the null space of ∂_1, which, bearing in mind that $1+1 = 0$ in $\mathbb{Z}_2$, can be done by the standard method of augmented matrix row reduction. The basis we obtain from this procedure is $\{[1, 1, 1, 1, 0, 0, 0]^\intercal, [0, 1, 0, 0, 1, 1, 1]^\intercal\}$, which corresponds to the set of 1-chains $\{e_1 + e_2 + e_3 + e_4, e_2 + e_5 + e_6 + e_7\}$. Since $\mathcal{C}_2$ is spanned by f_1, the image of ∂_2 is spanned by $[1, 1, 1, 1, 0, 0, 0]^\intercal$, which corresponds

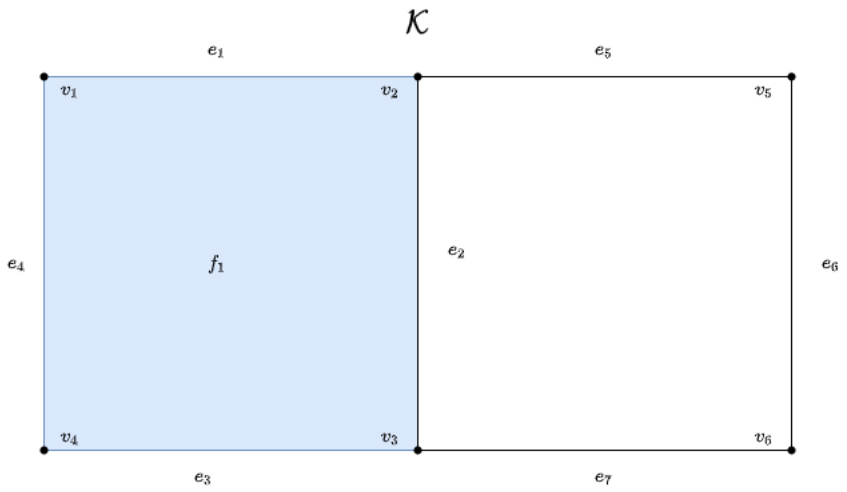


FIGURE 11. A cubical complex, $\mathcal{K}$ that we use to demonstrate homological computations. The 0-,1-, and 2-dimensional cubes in $\mathcal{K}$ are labelled as $\{v_1, v_2, v_3, v_4, v_5, v_6\}$, $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, and $\{f_1\}$, respectively.

to the 1-chain $e_1 + e_2 + e_3 + e_4$. Finally, $H_1 := \ker \partial_1 / \mathrm{Im}\partial_2 \simeq \{\lambda(e_2 + e_5 + e_6 + e_7) : \lambda \in \mathbb{Z}_2\}$ (the vector space spanned by $e_2 + e_5 + e_6 + e_7$) is obtained by "zeroing out" the elements in $\ker \partial_1$ that appear in $\mathrm{Im}\partial_2$. We conclude that H_1 is generated by one element. Intuitively, this element represents the hole enclosed by the edges e_2, e_5, e_6, and e_7.

5.2.3. *Persistence modules.* A filtration, Definition (2.1), implies an inclusion of chain complexes

$$C_k(\mathcal{K}_0) \subset C_k(\mathcal{K}_1) \subset \cdots \subset C_k(\mathcal{K}_n),$$

where $C_k(\mathcal{K})$ denotes the k-chains of $\mathcal{K}$, which in turn induce linear maps $i^*_{j,j+1}$ between homology groups, $H_k(\mathcal{K}_j)$ and $H_k(\mathcal{K}_{j+1})$ of $\mathcal{K}_j$ and $\mathcal{K}_{j+1}$, respectively,

$$H_k(\mathcal{K}_0) \xrightarrow{i^*_{0,1}} H_k(\mathcal{K}_1) \xrightarrow{i^*_{1,2}} \ldots \xrightarrow{i^*_{n-1,n}} H_k(\mathcal{K}_n), \tag{28}$$

by tracking where elements of $\ker \partial_k$ and $\mathrm{Im}\partial_{k+1}$ are sent under inclusions. The family of homology groups along with the sequence of maps in Equation (28) is known as a (finite) persistence module, which we denote by $\mathcal{M}$. Given two persistence modules, $\mathcal{M}_1 = \{H_k(\mathcal{K}_j), i^*_{j,j+1}\}_{j=0}^{n-1}$ and $M_2 = \{H_k(\mathcal{K}'_j), \iota^*_{j,j+1}\}_{j=0}^{n-1}$, we define their direct sum, $\mathcal{M}_1 \oplus \mathcal{M}_2$, by $\mathcal{M}_1 \oplus \mathcal{M}_2 = \{H_k(\mathcal{K}_j) \oplus H_k(\mathcal{K}'_j), i^*_{j,j+1} \oplus \iota^*_{j,j+1}\}_{j=0}^{n-1}$. A natural question to ask is if persistence modules can be decomposed into a direct sum of persistent modules. By the Structure Theorem for Persistence Modules (14), the answer to this question is in the affirmative. In particular, if we define the interval module $I(b, d)$ by

$$I(b,d) := \underbrace{\mathbf{0} \xrightarrow{0} \mathbf{0} \xrightarrow{0} \ldots \xrightarrow{0} \mathbf{0}}_{b-1 \text{ times}} \underbrace{\xrightarrow{0} \mathbf{v} \xrightarrow{id} \mathbf{v} \xrightarrow{id} \ldots \mathbf{v}}_{d-b \text{ times}} \underbrace{\xrightarrow{0} \mathbf{0} \xrightarrow{0} \mathbf{0} \xrightarrow{0} \ldots \xrightarrow{0} \mathbf{0}}_{n-(d-1) \text{ times}}, \tag{29}$$

where $\mathbf{0}, \mathbf{v}, 0$, and id denote the zero vector space, an arbitrary one-dimensional vector space, the zero map, and the identity map, respectively, then any finite persistence module decomposes into a direct sum of B interval modules,

$$\mathcal{M} = \bigoplus_{m=1}^{B} I(b_m, d_m). \tag{30}$$

Under mild conditions, the coordinates (b_m, d_m) of $\mathcal{M}$'s interval decomposition map uniquely to a pair of cubes, (Q_{b_m}, Q_{d_m}), where Q_{b_m} and Q_{d_m} are positive and negative cubes that create and destroy a homological feature, respectively. As was discussed in Section 2.2, the collection $\{(f(Q_{b_m}), f(Q_{d_m})\}_{m=1}^{B}$, along with the diagonal $\{(x, y) \in \mathbb{R}^2 : x = y\}$ defines a persistence diagram.

5.2.4. *Differentiability with respect to persistence diagrams.* To provide more detail for Equations (11) - (13), we compute the derivative of Equation (10) with respect to points in the diagram $\mathcal{D}$ using the limit definition. To this end, fix $\mathcal{D}'$ and suppose $\mathcal{D} = \{\mathbf{p}_n\}_{n=1}^{N} = \{(b_n, d_n)\}_{n=1}^{N}$. By definition of the derivative,

$$\frac{\partial m}{\partial \mathbf{p}_n} = \Big[\frac{\partial m}{\partial b_n}, \ \frac{\partial m}{\partial d_n}\Big], \tag{31}$$

and hence to compute the left hand side of Equation (31), it suffices to compute $\frac{\partial m}{\partial b_n}$ and $\frac{\partial m}{\partial d_n}$. We only compute $\frac{\partial m}{\partial b_n}$ as the computation of $\frac{\partial m}{\partial d_n}$ is analogous. To this end, notice

$$\begin{aligned}\frac{\partial m}{\partial b_n} = \lim_{\epsilon \to 0} \frac{1}{\epsilon}\Big[\Big(\min_{\gamma \in \Pi} \sum_{\mathbf{p} \in \mathcal{D} \setminus \mathbf{p}_n} \|\mathbf{p} - \gamma(\mathbf{p})\|_2^2 + (b_n + \epsilon - \gamma(b_n + \epsilon))^2 + (d_n - \gamma(d_n))^2\Big) \\ - \Big(\min_{\gamma \in \Pi} \sum_{\mathbf{p} \in \mathcal{D} \setminus \mathbf{p}_n} \|\mathbf{p} - \gamma(\mathbf{p})\|_2^2 + (b_n - \gamma(b_n))^2 + (d_n - \gamma(d_n))^2\Big)\Big],\end{aligned} \tag{32}$$

where Π is the set of injections from $\mathcal{D}$ to $\mathcal{D}'$, and by a slight abuse of notation, we denote the first and second components of the mapping $\gamma(\mathbf{p})$ by $\gamma(b)$ and $\gamma(d)$, respectively, so in particular $\gamma(\mathbf{p}) = [\gamma(b), \gamma(d)]$. Since $\mathcal{D}'$ is fixed, it has been shown that the minimal cost matching in Equation (32), γ^*, does not change in a neighborhood of $\mathcal{D}$, where a neighborhood is defined by one of the PD metrics, Equations (8) and (9). To be precise, there exists $\delta > 0$ such that if we perturb each point in $\mathcal{D}$ to create a new diagram $\tilde{\mathcal{D}}$ satisfying $W_p(\mathcal{D}, \tilde{\mathcal{D}}) < \delta$, then $\gamma^*(\mathbf{p}_n) = \gamma^*(\tilde{\mathbf{p}_n})$ for all $\mathbf{p}_n \in \mathcal{D}$ and $\tilde{\mathbf{p}_n} \in \tilde{\mathcal{D}}$. Thus, for ϵ sufficiently small, we can drop the minimums in Equation (32) and replace γ with the minimizer γ^* to obtain

$$\begin{aligned}\frac{\partial m}{\partial b_n} &= \lim_{\epsilon \to 0} \frac{(b_n + \epsilon - \gamma^*(b_n))^2 - (b_n - \gamma^*(b_n))^2}{\epsilon} \\ &= 2(b_n - \gamma^*(b_n)).\end{aligned}$$

E-mail address: coballe@nd.edu
E-mail address: david.l.boothe7.civ@mail.mil
E-mail address: piotr.j.franaszczuk.civ@mail.mil
E-mail address: vmaroula@utk.edu