

A Survey of Vectorization Methods in Topological Data Analysis

Dashti Ali, Aras Asaad, Maria-Jose Jimenez, Vidit Nanda, Eduardo Paluzo-Hidalgo, Manuel Soriano-Trigueros

Abstract—Attempts to incorporate topological information in supervised learning tasks have resulted in the creation of several techniques for vectorizing persistent homology barcodes. In this paper, we study thirteen such methods. Besides describing an organizational framework for these methods, we comprehensively benchmark them against three well-known classification tasks. Surprisingly, we discover that the best-performing method is a simple vectorization, which consists only of a few elementary summary statistics. Finally, we provide a convenient web application which has been designed to facilitate exploration and experimentation with various vectorization methods.

Index Terms—Topological data analysis, persistent homology, barcodes, vectorization methods

1 INTRODUCTION

PROPELLED by deep theoretical foundations and a host of computational breakthroughs, topological data analysis emerged roughly three decades ago as a promising method for extracting insights from unstructured data [1], [2], [3], [4]. The principal instrument of the enterprise is persistent homology; this consists of three basic steps, each relying on a different branch of mathematics.

- 1) *Metric geometry*: construct an increasing family $\{X_t\}$ of cell complexes around the input dataset X , where the indexing t is a scale parameter in $\mathbb{R}_{\geq 0}$.
- 2) *Algebraic topology*: compute the d -th homology vector spaces $H_d(X_t)$ for scales t in $\mathbb{R}_{\geq 0}$ and dimensions d in $\mathbb{Z}_{\geq 0}$.
- 3) *Representation theory*: decompose each family of vector spaces $\{H_d(X_t) \mid t \geq 0\}$ into irreducible summands, thus producing a **barcode**.

The resulting barcodes are finite multisets of real intervals $[p, q] \subset \mathbb{R}$, which admit concrete geometric interpretations in low dimensions — see Figure 1. The ultimate goal is to infer the coarse geometry of X across various scales by examining the longer intervals in its barcodes. Crucially, once the method for constructing $\{X_\epsilon\}$ from X has been fixed, the entire persistent homology pipeline is unsupervised: one

- *Dashti Ali is with Koya University, Iraq. Email: dashti.ali@koyauniversity.org*
- *Aras Asaad is with School of computing, University of Buckingham, UK and with Koya University, Iraq. Email: aras.asaad@buckingham.ac.uk:*
- *Maria-Jose Jimenez and Vidit Nanda are with Mathematical Institute, University of Oxford, United Kingdom. E-mail: {nanda, jimenez}@maths.ox.ac.uk*
- *Maria-Jose Jimenez, Eduardo Paluzo-Hidalgo and Manuel Soriano-Trigueros are with Departamento de Matematica Aplicada I, Universidad de Sevilla, Spain. E-mail: {majiro, epaluzo, msoriano4}@us.es*
- *Manuel Soriano-Trigueros is with Institute of Science and Technology Austria, Austria. E-mail: msoriano@ist.ac.at*

Manuscript received ; revised .

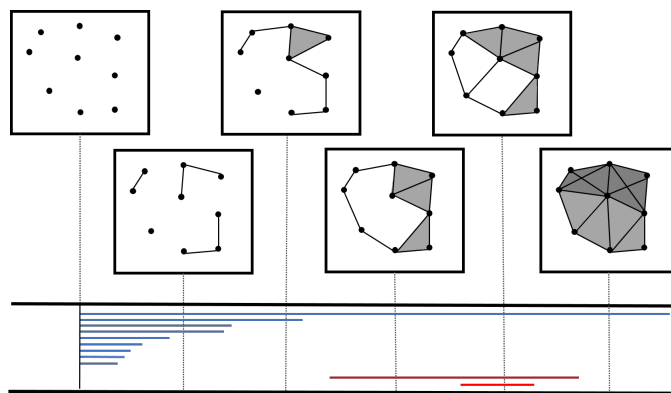


Fig. 1. An increasing family of cell complexes built around a point cloud dataset; the associated barcode in dimensions 0 (blue) and 1 (red) catalogues the connected components and cycles respectively.

requires neither labelled data nor hyperparameter tuning to produce barcodes from X .

At the other end of the data analysis spectrum lies supervised machine learning using contemporary neural networks, which are replete with billions of tunable parameters and gargantuan training datasets [5]. The practical aspects of deep neural networks appear to be light years ahead of the underlying theory. It nevertheless remains the case that machine learning has driven astonishing progress in the systematic automation of several important classification tasks. One direct consequence of these success stories is the irresistible urge to combine topological methods with machine learning. The most common avenue for doing so is to turn barcodes into vectors (lying in a convenient Euclidean space) which then become input for suitably-trained neural networks.

The good news, at least from an engineering perspective, is that barcodes are inherently combinatorial objects, and as such, they are remarkably easy to vectorize. Several dozen vectorization methods have been proposed across the last decade, and new ones continue to appear with alarming

frequency and increasing complexity — the reader will encounter thirteen of them here. The bad news, on the other hand, comes in the form of three serious challenges which must be confronted by those who build or use such vectorizations:

- 1) Given the large number of options, even established practitioners are not aware of all the vectorization techniques; similarly, knowledge of which vectorizations are suitable for which types of data is difficult – if not impossible – to glean from the published literature.
- 2) There is a natural metric between barcodes called the *bottleneck distance*; when it is endowed with this metric, the space of barcodes becomes infinite-dimensional and highly nonlinear. As such, it does not admit any faithful embeddings into finite-dimensional vector spaces.
- 3) Even the *stable* vectorizations, which preserve distances by mapping barcodes into infinite-dimensional vector spaces, may suffer from a lack of discriminative power in practice: by design, they are poor at distinguishing between datasets whose coarse structures are similar and whose differences reside in finer scales.

In This Paper

Here we seek to comprehensively describe, catalogue and benchmark vectorization methods for persistent homology barcodes. The first contribution of this paper is the following taxonomy of the known methods, which we hope will serve as a convenient organizational framework for beginners and experts alike —

- 1) *Statistical* vectorizations: these summaries consist of basic statistical quantities;
- 2) *Algebraic* vectorizations: these are generated from polynomials;
- 3) *Curve* vectorizations: these come from maps $\mathbb{R} \rightarrow H$, where H is a vector space;
- 4) *Functional* vectorizations: these are maps of the form $X \rightarrow H$ for $X \neq \mathbb{R}$;
- 5) *Ensemble* vectorizations: these are generated from collections of training barcodes.

There are unavoidable overlaps between these five categories. When such an overlap occurs, we have placed the given vectorization technique in the earliest relevant category among those in the list above; thus, an algebraic vectorization given by polynomial functions of basic statistical quantities will be placed in category (1) rather than category (2). The reader might claim, quite reasonably, that category (3) should be subsumed into category (4). However, the sheer number of curve-based vectorizations compelled us to set them apart.

The second contribution of this paper is a comprehensive benchmarking of thirteen vectorization techniques across these five categories on three well-known image classification datasets. These datasets were selected to simultaneously (a) provide an increasing level of difficulty for topological methods, and (b) to be instantly recognizable for the broader machine learning community. These are: the **Outex**

texture database [6], the **SHREC14** shape retrieval dataset [7], and the **Fashion-MNIST** database [8]. Surprisingly, the best-performing vectorization *in all three cases* is a rather naïve one obtained by collecting basic statistical quantities associated to (the multiset of) intervals in a given barcode.

Our third contribution is a companion web application which computes and visualizes all thirteen vectorization techniques which have been investigated in this paper. In addition to running online¹, this web app can also be downloaded² and run locally on more challenging datasets.

Not In This Paper

Vectorization methods form but a small part of the ever expanding interface between topological data analysis and machine learning. As such, there are several related techniques which are not benchmarked here. The precise inclusion criteria for our study in this paper are as follows.

- 1) We restrict our attention to those methods which produce genuine vectors from barcodes. In particular, kernel methods [9], [10] are beyond the scope of this paper.
- 2) We only consider those vectorizations that are either straightforward for us to implement, or have an easily accessible and trusted implementation. For instance, path signature based vectorizations [11], [12] are excluded.
- 3) We do not compare machine learning architectures designed for the explicit purpose of inferring (persistent) homology [13], [14], [15].
- 4) We do not touch upon various attempts to design or study neural networks using tools from topological data analysis [16], [17].
- 5) Finally, even among methods which satisfy the first four criteria, we have discarded techniques which regularly obtained a classification accuracy below fifty percent.

Similar Efforts

The authors of [18] have summarised – but not compared – several vectorization and kernel methods for barcodes. Another summary (sans comparison) may be found in [19], with emphasis on metric aspects of the chosen vectorizations. The work of [20] describes a common overarching framework for what we have called curve vectorizations here. More recently, [21] and [22] have described and compared five and four vectorization methods respectively.

Outline

Notation and preliminaries involving barcodes are established in Section 2. In Sections 3 and 4 we introduce the thirteen vectorizations (suitably organised into our taxonomy) and the three datasets. Section 5 contains the results of our experiments whose finer details have been relegated to Appendices A and B. We provide a description of the web app in Section 6 and some brief concluding remarks in Section 7.

1. <https://persistent-homology.streamlit.app>
2. <https://github.com/dashtiali/vectorisation-app>

2 PERSISTENCE BARCODES FROM DATA

At its core, persistent homology studies sequences of finite-dimensional vector spaces $V = \{V_i \mid 0 \leq i \leq n\}$ and linear maps $a = \{a_i : V_{i-1} \rightarrow V_i \mid 1 \leq i \leq n\}$:

$$V_0 \xrightarrow{a_1} V_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} V_n.$$

Such sequences (V, a) are called *persistence modules*. Among the simplest examples are *interval modules* — for each pair of integers $p \leq q$ with $[p, q] \subset [0, n]$, the corresponding interval module $(I^{[p,q]}, c^{[p,q]})$ has

$$\dim I_i^{[p,q]} = \begin{cases} 1 & \text{if } p \leq i \leq q \\ 0 & \text{otherwise;} \end{cases}$$

similarly, the map $c_i^{[p,q]}$ is the identity whenever $p+1 \leq i \leq q$ and zero otherwise.

2.1 Structure and Stability

Every persistence module decomposes into a direct sum of interval modules. In particular, we have the following structure theorem [23], [24].

Theorem 2.1. For every persistence module (V, a) , there exists a unique set $\mathbf{Bar}(V, a)$ of subintervals of $[0, n]$ along with a unique function $\mathbf{Bar}(V, a) \rightarrow \mathbb{Z}_{>0}$ denoted $[p, q] \mapsto \mu_{p,q}$ for which we have an isomorphism

$$(V, a) \simeq \bigoplus_{[p,q] \in \mathbf{Bar}(V,a)} \left(I^{[p,q]}, c^{[p,q]} \right)^{\mu_{p,q}}.$$

Thus, the algebraic object (V, a) may be fully recovered (up to isomorphism) from purely combinatorial data consisting of the set of intervals $\mathbf{Bar}(V, a)$ and the multiplicity function μ . Alternately, one may view $\mathbf{Bar}(V, a)$ itself as a multiset with $\mu_{p,q}$ copies of each interval $[p, q]$. This multiset is called the **barcode** of (V, a) . It is often useful in applications to let the vector spaces V_i be indexed by real numbers rather than integers. With this modification in place, $\mathbf{Bar}(V)$ becomes a collection of real intervals $[p, q] \subset \mathbb{R}$.

The most important property of persistence modules, beyond the structure theorem, is their **stability** [24]. There is a natural metric on the set of persistence modules called the *interleaving distance* and a metric on the set of barcodes called the *bottleneck distance*.

Theorem 2.2. The assignment $(V, a) \mapsto \mathbf{Bar}(V, a)$ is an isometry from the space of persistence modules (with interleaving distance) to the space of barcodes (with bottleneck distance).

The advantage of this theorem is that barcodes remain robust to (certain types of) perturbations of the original dataset, thus conferring upon the topological data analysis pipeline a degree of noise-tolerance. The significant difficulty from a statistical perspective, however, is that the metric space of persistence barcodes with bottleneck distance is nonlinear — even averages can not be defined for arbitrary collections of barcodes [25], [26], [27].

2.2 Barcodes from Data

Persistence modules arise naturally from a wide class of datasets. The first step in topological data analysis involves imposing the structure of a filtered cell complex — either simplicial [28, Chapter 8] or cubical [29] — from the data [1], [2], [3]. The two most prominent examples of filtered cell complex structures arising from data are as follows.

- 1) Given a finite point cloud $X \subset \mathbb{R}^n$, one constructs a family of increasing simplicial complexes $\{S_\epsilon \mid \epsilon \geq 0\}$ defined as follows. A collection $\{x_0, \dots, x_k\}$ forms a k -simplex in S_ϵ if and only if the (Euclidean) distance between x_i and x_j is no larger than ϵ for all i, j in $\{0, \dots, k\}$. Since there are only finitely many ϵ values at which new simplices are introduced, the filtration is indexed by a subset of the natural numbers. The collection S_ϵ is called the **Vietoris-Rips** filtration of X . These filtrations can be similarly defined for any metric space.
- 2) Consider a grayscale image I , given in terms of $m \times n$ pixels with intensity values in the set $\{0, 1, \dots, 255\}$. This naturally forms a two-dimensional cubical complex, which can be endowed with the **upper-star** filtration by intensity values. In particular, each elementary cube of dimension < 2 appears at the smallest intensity encountered among the 2-dimensional cubes in its immediate neighbourhood. Higher-dimensional cubical filtrations may be similarly generated from higher-dimensional pixel grids.

There are several other filtration types which may be used to model point and image datasets. Once the data has been suitably modeled by a filtered simplicial or cubical complex, persistence modules are obtained by computing **homology** groups with coefficients in a field. In general, these homology groups are not invariant to the choice of filtration. The reader who is interested in the definition and computation of homology is urged to either consult standard algebraic topology references such as [30, Ch 2] or see the more recent [3], [4], [31], [32].

A substantial difficulty in topological data analysis is that although persistent homology barcodes can be readily associated with a large class of datasets, the space of all such barcodes is notoriously unpleasant to encounter from a statistical perspective. Fortunately, barcodes are combinatorial objects which can be mapped to Hilbert spaces in a plethora of reasonable ways. Indeed, across the last decade, such **vectorization methods** have been proposed by various authors, and our main purpose in this work is to benchmark many of these methods against standard classification tasks.

3 VECTORIZATION METHODS FOR BARCODES

Throughout this section, we assume knowledge of the barcode $B := \mathbf{Bar}(V, a)$ of an \mathbb{R} -indexed persistence module along with its multiplicity function $\mu : B \rightarrow \mathbb{Z}_{>0}$. We note that for each interval $[p, q]$ in B the numbers p and q are called its *birth* and *death* respectively, and the length $q - p$ is called its *lifespan*.

3.1 Statistical Vectorizations

The first and simplest category of vectorizations considered in this paper are generated from basic statistical quantities associated to the given barcode. Variants of the following vectorization have been defined and used on several occasions — see for instance [33, sec 2.3], [20, Sec 6.2.1] and [18, Sec 4.1.1].

Definition 3.1. The **persistence statistics** vector of $\mu : B \rightarrow \mathbb{Z}_{>0}$ consists of:

- 1) the mean, the standard deviation, the median, the interquartile range, the full range, the 10th, 25th, 75th and 90th percentiles of the births p , the deaths q , the midpoints $\frac{p+q}{2}$ and the lifespans $q - p$ for all intervals $[p, q]$ in B counted with multiplicity;
- 2) the total number of bars (again counted with multiplicity), and
- 3) the *entropy* of μ , defined as the real number

$$E_\mu := - \sum_{[p,q] \in B} \mu_{p,q} \cdot \left(\frac{q-p}{L_\mu} \right) \cdot \log \left(\frac{q-p}{L_\mu} \right),$$

where L_μ is the weighted sum

$$L_\mu := \sum_{[p,q] \in B} \mu_{p,q} \cdot (q-p). \quad (1)$$

The entropy from Definition 3.1(3) was introduced in [34], [35]. Our second statistical vectorization is from [36], where entropy has been upgraded to a real-valued piecewise constant function rather than a single number.

Definition 3.2. The **entropy summary** function of $\mu : B \rightarrow \mathbb{Z}_{>0}$ is the map $S_\mu : \mathbb{R} \rightarrow \mathbb{R}$ given by

$$S_\mu(t) = - \sum_{[p,q] \in B} \mathbb{1}_{p \leq t < q} \cdot \mu_{p,q} \cdot \left(\frac{q-p}{L_\mu} \right) \cdot \log \left(\frac{q-p}{L_\mu} \right).$$

Here $\mathbb{1}_\bullet$ is the indicator function — it equals 1 when the conditional \bullet is true and it equals 0 otherwise. The number L_μ appearing in the expression above is defined in (1).

The entropy summary function has also been called the *life entropy curve*, e.g., in [20].

3.2 Algebraic Vectorizations

The vectorizations in this category are generated using polynomial maps constructed from the barcode $\mu : B \rightarrow \mathbb{Z}_{>0}$.

The first example considered here is from [37]. It becomes convenient, for the purpose of defining it, to arbitrarily order the intervals in B as $\{[p_i, q_i] \mid 1 \leq i \leq n\}$ with the understanding that each $[p, q]$ occurs $\mu_{p,q}$ times in this ordered list.

Definition 3.3. The ring of **algebraic functions** on $\mu : B \rightarrow \mathbb{Z}_{>0}$ consists of all those \mathbb{R} -polynomials f in variables $\{x_1, y_1, \dots, x_n, y_n\}$ for which the following property holds: there exist polynomials $\{g_i \mid 1 \leq i \leq n\}$ satisfying

$$\frac{\partial f}{\partial x_i} + \frac{\partial f}{\partial y_i} = (x_i - y_i) \cdot g_i.$$

(Here $\partial f / \partial x_i$ indicates the partial derivative of f with respect to x_i , and so forth).

The desired vectorization is obtained by selecting finitely many algebraic functions from this ring and evaluating them at $x_i = p_i$ and $y_i = q_i$ for all i . The feature maps generated by making such choices are sometimes called *Adcock-Carlsson coordinates* — see for instance [38]. Letting q_{\max} be the maximum death-value encountered among the intervals in B , four of the most widely-used algebraic functions are:

$$\begin{aligned} f_1 &= \sum_i p_i(q_i - p_i) & f_2 &= \sum_i (q_{\max} - q_i)(q_i - p_i) \\ f_3 &= \sum_i p_i^2(q_i - p_i)^4 & f_4 &= \sum_i (q_{\max} - q_i)^2(q_i - p_i)^4 \end{aligned}$$

Small changes in the barcode (in terms of bottleneck distance) are liable to create large fluctuations in the associated algebraic functions. The methods of tropical geometry were used in [39] to address the bottleneck instability of algebraic functions. In this setting, the standard polynomial operations $(+, \times)$ are systematically replaced by $(\max, +)$. To define the resulting vectorization, we once again use an ordering $\{[p_i, q_i] \mid 1 \leq i \leq n\}$ of the intervals in B .

Definition 3.4. A **tropical coordinate function** for $\mu : B \rightarrow \mathbb{Z}_{>0}$ is a function F of variables $\{x_1, y_1, \dots, x_n, y_n\}$ which is both tropical and symmetric as described below.

- 1) *Tropical*: there is an expression for F which uses only the operations \max , \min , $+$ and $-$ on the variables $\{x_i\}$ and $\{y_i\}$.
- 2) *Symmetric*: any permutation of $\{1, \dots, n\}$, when applied to both $\{x_i\}$ and $\{y_i\}$, leaves F unchanged.

Let λ_i be the lifespan $q_i - p_i$ of the i -th interval in B . To generate feature maps from the tropical coordinate functions described above, one simply evaluates them at $x_i = \lambda_i$ and y_i equal to either $\max(r\lambda_i, p_i)$ or $\min(r\lambda_i, p_i)$ for a positive integer parameter r . Examples of such tropical coordinate features include:

$$\begin{aligned} F_1 &= \max_i \lambda_i \\ F_2 &= \max_{i < j} (\lambda_i + \lambda_j) \\ F_3 &= \max_{i < j < k} (\lambda_i + \lambda_j + \lambda_k) \\ F_4 &= \max_{i < j < k < l} (\lambda_i + \lambda_j + \lambda_k + \lambda_l) \\ F_5 &= \sum_i \lambda_i \\ F_6 &= \sum_i \min(r\lambda_i, p_i), \end{aligned}$$

and the somewhat more complicated

$$F_7 = \sum_j \left[\max_i (\min(r\lambda_i, p_i) + \lambda_i) - (\min(r\lambda_j, p_j) + \lambda_j) \right].$$

These seven tropical coordinates were used in [39] for performing classification on the MNIST database, with $r = 28$.

The third and final algebraic vectorization considered here is generated by extracting complex polynomials from barcodes [40], [41]. In what follows, the symbol i should be interpreted as $\sqrt{-1}$ (and not as an index for the intervals in

B). Consider the three continuous maps $R, S, T : \mathbb{R}^2 \rightarrow \mathbb{C}$ defined as follows:

$$\begin{aligned} R(x, y) &= x + iy \\ S(x, y) &= \begin{cases} \frac{y-x}{\alpha\sqrt{2}} \cdot (x + iy) & \text{if } (x, y) \neq (0, 0) \\ 0 & \text{otherwise} \end{cases} \\ T(x, y) &= \frac{y-x}{2} \cdot [(\cos \alpha - \sin \alpha) + i(\cos \alpha + \sin \alpha)], \end{aligned}$$

where α is the norm $\sqrt{x^2 + y^2}$.

Definition 3.5. Given a barcode $\mu : B \rightarrow \mathbb{Z}_{>0}$, let $X : \mathbb{R}^2 \rightarrow \mathbb{C}$ be any one of the three functions R, S, T defined above. The **complex polynomial** vectorization of μ of type X is the sequence of coefficients of the complex polynomial in one variable z given by

$$C_X(z) := \prod_{[p,q] \in B} [z - X(p, q)]^{\mu_{p,q}}.$$

In practice, it is customary to either take only the first few highest degree coefficients of $C_X(z)$ or to multiply it by a suitable power of z . This is done to guarantee that the feature vectors assigned to a collection of barcodes all have the same dimension.

Other Algebraic Vectorizations: In the subsequent section, we describe how to extract vectorizations by using barcode data to build curves which take values in a vector space. Once such a curve has been extracted, one can compute its *path signature* via iterated integrals [42]. The path signature resides in the tensor algebra of the target vector space; elements of the tensor algebra are equivalent to coefficients of non-commuting polynomials, and hence constitute algebraic vectorizations of barcodes — see [11], [12] for examples of this approach.

3.3 Curve Vectorizations

There are several interesting ways of turning barcodes into one or more curves, which for our purposes here mean (piecewise) continuous maps from \mathbb{R} to a convenient vector space. Feature vectors can then be constructed by sampling the given curve at finite subsets of \mathbb{R} . Perhaps the simplest and most widely used curve-based vectorization is the following.

Definition 3.6. The **Betti curve** of $\mu : B \rightarrow \mathbb{Z}_{>0}$ is the curve $\beta_\mu : \mathbb{R} \rightarrow \mathbb{R}$ given by

$$\beta_\mu(t) = \sum_{[p,q] \in B} \mathbb{1}_{p \leq t < q} \cdot \mu_{p,q}.$$

Here $\mathbb{1}_\bullet$ is the indicator function as described in Definition 3.2, so this function counts the number of intervals (with multiplicity) in B which contain t . Very similar in spirit (and formula) to the Betti curve is the following vectorization from [20].

Definition 3.7. The **lifespan curve** of $\mu : B \rightarrow \mathbb{Z}_{>0}$ is the map $L_\mu : \mathbb{R} \rightarrow \mathbb{R}$ given by

$$L_\mu(t) = \sum_{[p,q] \in B} \mathbb{1}_{p \leq t < q} \cdot \mu_{p,q} \cdot (q - p).$$

It is not difficult to create very different-looking Betti and lifespan curves from two barcodes which have arbitrarily

small bottleneck distance — we can always add lots of very small intervals to a given barcode without changing its bottleneck distance by a significant amount. One way to rectify the bottleneck instability of Betti and lifespan curves is to test the containment not only of t in each interval $[p, q] \in B$, but rather of the largest subinterval of the form $[t - s, t + s]$. This modification leads to one of the oldest and best-known stable curve vectorizations [43], [44], as defined below.

Definition 3.8. The **persistence landscape** of the barcode $\mu : B \rightarrow \mathbb{Z}_{>0}$ is a sequence of curves $\{\Lambda_i^\mu : \mathbb{R} \rightarrow [-\infty, \infty] \mid i \in \mathbb{Z}_{>0}\}$ defined as follows. The image $\Lambda_i^\mu(t)$ of each t in \mathbb{R} equals

$$\sup \left\{ s \geq 0 \mid \left(\sum_{[p,q] \in B} \mathbb{1}_{[t-s, t+s] \subset [p,q]} \cdot \mu_{p,q} \right) \geq i \right\}.$$

By convention, the supremum over the empty set is zero. Moreover, since our barcode B is assumed to be finite, the landscape functions Λ_i^μ become identically zero for sufficiently large i . An alternate approach to defining persistence landscapes comes from the function $\Delta : B \times \mathbb{R} \rightarrow \mathbb{R}$, given by

$$\Delta([p, q], t) := \max(\min(t - p, q - t), 0). \quad (2)$$

For each $i \in \mathbb{Z}_{>0}$, the curve Λ_i^μ from Definition 3.8 equals the i -th largest number in the multiset that contains $\mu_{p,q}$ copies of $\Delta([p, q], t)$ for each interval $[p, q]$ in B . The fourth and final curve vectorization that we consider here was introduced in [45], and it is also defined in terms of the functions Δ from (2).

Definition 3.9. Let $w : B \rightarrow \mathbb{R}_{>0}$ be any function, which we will denote $[p, q] \mapsto w_{p,q}$. The w -weighted **persistence silhouette** of $\mu : B \rightarrow \mathbb{Z}_{>0}$ is the map $\phi_\mu^w : \mathbb{R} \rightarrow \mathbb{R}$ defined as the weighted average

$$\phi_\mu^w(t) := \frac{\sum w_{p,q} \cdot \mu_{p,q} \cdot \Delta([p, q], t)}{\sum w_{p,q} \cdot \mu_{p,q}}.$$

Here both sums on the right are indexed over all $[p, q] \in B$, and Δ is defined in (2).

Reasonable choices of weight functions are provided by setting $w_{p,q} = (q - p)^\alpha$ for a real-valued scale parameter $\alpha \geq 0$. For small α , the shorter intervals dominate the value of the silhouette curve, whereas for large α it is the longer intervals which play a more substantial role — see [45, Sec 4] for details.

Other Curve Vectorizations: See the *envelope embedding* from [11], the *accumulated persistence function* in [46], and the *persistent Betti function* of [47]. In [48], the persistent Betti function is decomposed along the Haar basis to produce a vectorization. More recently, [20] provides a general framework for constructing several different curve vectorizations.

3.4 Functional Vectorizations

Here we catalogue those barcode vectorizations which are given by maps from spaces other than \mathbb{R} . The first, and perhaps most prominent member of this category is the following vectorization from [49]. Its definition below makes

use of two auxiliary components besides the given barcode $\mu : B \rightarrow \mathbb{Z}_{>0}$. The first is a continuous, piecewise-differentiable function $f : \mathbb{R}^2 \rightarrow \mathbb{R}_{\geq 0}$ satisfying $f(x, 0) = 0$ for all $x \in \mathbb{R}$. And the second is a collection of smooth probability distributions $\Psi := \{\psi_{p,q} \mid [p, q] \in B\}$ where $\psi_{p,q}$ has mean $(p, q - p)$.

Definition 3.10. The persistence surface of $\mu : B \rightarrow \mathbb{Z}_{>0}$ with respect to f and Ψ (as described above) is the function $\mathbb{R}^2 \rightarrow \mathbb{R}$ given by

$$\rho_{f,\Psi}^\mu(x, y) = \sum_{[p,q] \in B} \mu_{p,q} \cdot f(p, q - p) \cdot \psi_{p,q}(x, y).$$

The **persistence image** $\mathbf{I}_{f,\Psi}^\mu$ of μ with respect to (f, Ψ) assigns a real number to every subset $Z \subset \mathbb{R}^2$; this number is given by integrating the persistence surface over Z :

$$\mathbf{I}_{f,\Psi}^\mu(Z) = \iint_Z \rho_{f,\Psi}^\mu(x, y) dx dy.$$

In order to obtain a vector from the persistence image, one lets Z range over grid pixels in a rectangular subset of \mathbb{R}^2 and renormalizes the resulting array of numbers, thus producing a grayscale image. Standard choices of f and $\Psi = \{\psi_{p,q}\}$ are:

$$f(x, y) = \begin{cases} 0 & t \leq 0 \\ t/\lambda_{\max} & 0 < t < \lambda_{\max} \\ 1 & t > \lambda_{\max} \end{cases}$$

$$\psi_{p,q}(x, y) = \frac{1}{2\pi\sigma^2} \cdot \exp\left(-\frac{(x-p)^2 + (y-(q-p))^2}{2\sigma^2}\right).$$

Here λ_{\max} is the largest lifespan $\max_{[p,q] \in B} (q - p)$ encountered among the intervals in B , and σ is a user-defined parameter which forms the common standard deviation of every $\psi_{p,q}$ in sight.

The second and final functional vectorization which we will examine was introduced in the paper [38]. Set $\mathbb{W} := \{(x, y) \in \mathbb{R}^2 \mid y > 0\}$, and note that points $(x, y) \in \mathbb{W}$ parameterize intervals $[x, x + y] \subset \mathbb{R}$ of strictly positive length that could possibly lie in a given barcode. Let $C_c(\mathbb{W})$ be the set of all continuous functions $f : \mathbb{W} \rightarrow \mathbb{R}$ with compact support³. The given barcode $\mu : B \rightarrow \mathbb{Z}_{>0}$ induces a function $V_\mu : C_c(\mathbb{W}) \rightarrow \mathbb{R}$ via

$$V_\mu(f) = \sum_{[p,q] \in B} \mu_{p,q} \cdot f(p, q - p). \quad (3)$$

A subset T of $C_c(\mathbb{W})$ is called a *template system* if for any distinct pair $\mu_1 : B_1 \rightarrow \mathbb{Z}_{>0}$ and $\mu_2 : B_2 \rightarrow \mathbb{Z}_{>0}$ of barcodes, there exists at least one $f \in T$ so that $V_{\mu_1}(f) \neq V_{\mu_2}(f)$.

Definition 3.11. Fix an integer $n > 0$ and let $\text{Sub}_n(T)$ be the collection of all size n subsets of a template system T as described above. The **template function vectorization** of $\mu : B \rightarrow \mathbb{Z}_{>0}$ with respect to T is the map $\tau : \text{Sub}_n(T) \rightarrow \mathbb{R}^n$ defined as follows. Given $f = \{f_1, \dots, f_n\}$ in $\text{Sub}_n(T)$, the associated vector in \mathbb{R}^n is

$$\tau^\mu(f) := (V_\mu(f_1), \dots, V_\mu(f_n)),$$

3. In other words, $C_c(\mathbb{W})$ contains those continuous real-valued functions on \mathbb{W} which evaluate to 0 outside the intersection of a sufficiently large rectangle with \mathbb{W} in \mathbb{R}^2 .

where $V_\mu(f_i)$ is as defined in (3).

Two convenient choices of T , called *tent functions* and *interpolating polynomials*, have been highlighted in [38]. Tent functions are indexed by points $(u, v) \in \mathbb{R}^2$ and require an additional parameter $\delta > 0$; they have the form

$$g_{u,v}^\delta(x, y) = \max\left(1 - \frac{1}{\delta} \cdot \max(|x - u|, |y - v|), 0\right) \quad (4)$$

By construction, each such function is supported on the square of side length 2δ around the point (u, v) in the birth-lifespan plane. The normal pipeline for selecting finitely many template functions requires covering a sufficiently large bounded subset of \mathbb{W} with a square grid and then selecting the appropriate tent functions supported on grid cells. We direct interested readers to [38, Sections 6 and 7] for details on interpolating polynomials and for suggestions on how one might select suitable n and $f \in \text{Sub}_n(T)$ for a given classification task.

Other Functional Vectorizations: See the *generalised persistence landscape* in [50] and the *crocker stacks* of [51].

3.5 Ensemble Vectorizations

Our last category contains two methods which require access to a sufficiently large collection of *training* barcodes $\mu_i : B_i \rightarrow \mathbb{Z}_{>0}$ in order to generate a vectorization. The first of these methods, introduced in [52], [53], is a modification of the template system vectorization from Definition 3.11. We recall that $\mathbb{W} \subset \mathbb{R}^2$ is defined as $\{(x, y) \mid 0 \leq x < y\}$ and that every barcode B is identified with a subset $P(B) \subset \mathbb{W}$ via the map that sends intervals $[p, q]$ of positive length to points (p, q) .

Definition 3.12. The **adaptive template system** induced by a collection of barcodes $\{\mu_i : B_i \rightarrow \mathbb{Z}_{>0}\}$ is obtained via the following two steps. Letting $P \subset \mathbb{W}$ be the union $\bigcup_i P(B_i)$, one

- 1) identifies finitely many ellipses $E_j \subset \mathbb{W}$ which tightly contain P , and then
- 2) constructs suitable functions g_j supported on E_j , as described in (5) below.

The desired vectorization of a new barcode $\mu : B \rightarrow \mathbb{Z}_{>0}$ is now obtained by using these g_j , rather than tent functions, as template functions in Definition 3.11. Three different methods for finding the E_j can be found in [52, Sec 3]. Let v^* denote the transpose of a given vector v in \mathbb{R}^2 . Now each ellipse E with centre $x = (x_1, x_2)^*$ corresponds to a symmetric 2×2 matrix A satisfying

$$E = \{z \in \mathbb{R}^2 \mid (z - x)^* A (z - x) = 1\}.$$

Setting $h(z) := (z - x)^* A (z - x)$, the adaptive template function g supported on E is

$$g(z) = \begin{cases} 1 - h(z) & h(z) < 1 \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

The second instance of an ensemble vectorization framework which we benchmark in this paper is from [54]. Let $\mu_i : B_i \rightarrow \mathbb{Z}_{>0}$ be a collection of training barcodes as before, and fix a dimension parameter $b \in \mathbb{Z}_{>0}$. Much like the adaptive template systems of Definition 3.12, the *automatic*

topology-oriented learning (ATOL) vectorization is a two-step process for mapping each B_i to a vector space, which in this instance is always \mathbb{R}^b .

Definition 3.13. The **ATOL contrast functions** corresponding to the collection of barcodes $\{\mu_i : B_i \rightarrow \mathbb{Z}_{>0}\}$ and parameter $b \in \mathbb{Z}_{>0}$ are obtained as follows:

- 1) Treating the point clouds

$$P_i := \{(p, q) \in \mathbb{R}^2 \mid [p, q] \in B_i \text{ and } q > p\}$$

as discrete measures on \mathbb{R}^2 , one estimates their average measure E .

- 2) Let $\mathbf{z} := (z_1, z_2, \dots, z_b)$ be a point sample in \mathbb{R}^2 drawn (in independent, identically distributed function) along E . Define the real numbers $\sigma_i(\mathbf{z})$ for $1 \leq i \leq b$ by

$$\sigma_i(\mathbf{z}) := \frac{1}{2} \max_{j \neq i} \|z_j - z_i\|_2,$$

where $\|\bullet\|_2$ denotes the usual Euclidean norm on \mathbb{R}^2 .

The contrast functions $\{\Omega_i : \mathbb{R}^2 \rightarrow \mathbb{R} \mid 1 \leq i \leq b\}$ are now given by

$$\Omega_i(x) = \exp\left(-\frac{\|x - z_i\|}{\sigma_i(\mathbf{z})}\right).$$

The reader is directed to [54, Algorithm 1] for further details. Once the contrast functions have been produced in the manner described above, the corresponding **ATOL vectorization** of a given barcode $\mu : B \rightarrow \mathbb{Z}_{>0}$ equals $(\Omega_1^\mu, \dots, \Omega_b^\mu)$, where

$$\Omega_i^\mu := \sum_{[p,q] \in B} \mu_{p,q} \cdot \Omega_i(p, q).$$

Other Ensemble Vectorizations: The *persistence codebooks* approach from [55] proposes three different types of barcode vectorizations; these are based on bag-of-word embeddings, VLAD (vector of locally aggregated descriptors), and Fisher Vectors respectively.

4 DATASETS

The vectorization methods described in the preceding section have been benchmarked against three standard datasets; these are described below and arranged in increasing order of difficulty for topological methods. All three of them have been used in the past for comparing vectorizations (or kernels) for persistence barcodes [9], [10], [11], [38], [52], [56].

4.1 Outex

Outex is a database of images developed for the assessment of texture classification algorithms [6] — see Fig. 2, right-bottom, for some samples of textures from the 68 categories. Each texture class contains 20 images of size 128×128 pixels, which results in 1,360 images in total. We designed a reduced version of the experiment by randomly selecting 10 of the total 68 classes in the dataset, which we refer to as **Outex10** below. The full classification is referred to as **Outex68**. In both cases, a train/test split of 70/30 has been applied.

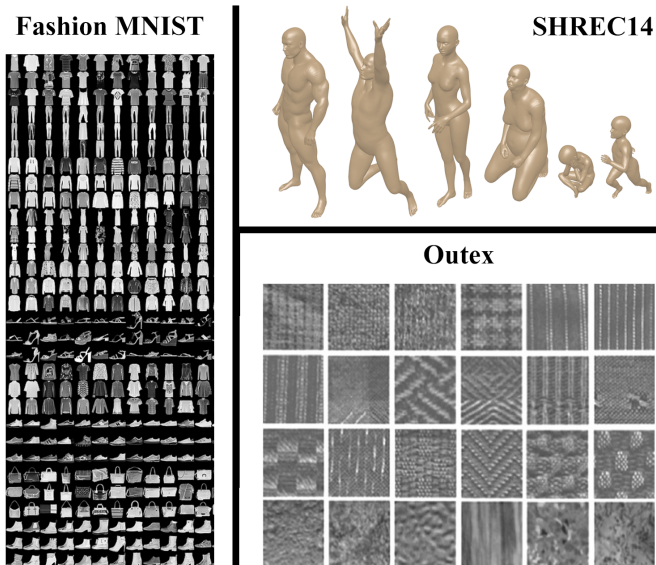


Fig. 2. Samples from datasets used in our experiments

We treat each image as a cubical complex; the filtration is induced by considering the pixel intensity on the 2-dimensional cells, which is inherited by other cells via the lower-star and upper-star filtrations. Persistent homology barcodes are computed in dimensions 0 and 1 using the GUDHI library [57]. No pre-processing has been applied to the images.

4.2 SHREC14

The Shape Retrieval of a non-rigid 3D Human Models dataset, usually abbreviated SHREC14 [7], is designed to test shape classification and retrieval algorithms. It contains real and synthetic human shapes and poses stored as 3D meshes (which are already simplicial complexes). We use the synthetic part of the dataset; this constitutes a classification task with 15 classes (5 men, 5 women and 5 children), each one with 20 different poses — see the upper-right corner of Fig. 2.

We apply the Heat Kernel Signature (HKS) to obtain filtrations [9], [58]. For a fixed real parameter $t > 0$, this filtration assigns to each mesh point x the value

$$\text{HKS}_t(x) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \cdot \phi_i(x)^2 \quad (6)$$

Here λ_i and ϕ_i are eigenvalues and corresponding eigenfunctions of (a discrete approximation to) the Laplace-Beltrami operator of the given mesh. Every simplex of dimension > 0 is assigned the largest value of HKS_t encountered among its vertices. We used the pre-computed barcodes (for such filtrations across a range of t -values) which have been provided in the repository⁴ accompanying [21]. Of the 300 samples, 70% were used for training and the other 30% for testing.

4. https://github.com/barnesd8/machine_learning_for_persistence

Vectorization Method	Score	Parameters	Estimator
Persistence Statistics	0.992		SVM, rbf kernel, C_1, γ_1
Entropy Summary	0.975	100	SVM, rbf kernel, C_1, γ_1
Algebraic Functions	0.992		SVM, lin. kernel, C_3
Tropical Coordinates	0.975	250	SVM, lin. kernel, C_4
Complex Polynomial	0.950	5, R	SVM, rbf kernel, C_1, γ_1
Betti Curve	0.908	200	SVM, rbf kernel, C_1, γ_1
Lifespan Curve	0.975	100	SVM, rbf kernel, C_1, γ_1
Persistence Landscape	0.975	50, 20	SVM, rbf kernel, C_2, γ_2
Persistence Silhouette	0.983	100, 0	SVM, rbf kernel, C_1, γ_1
Persistence Image	0.938	1, 25	RF, n=500
Template Function	0.958	35, 20	SVM, rbf kernel, C_1, γ_1
Adaptive Template S.	0.975	GMM, 40	SVM, rbf kernel, C_1, γ_1
ATOL	0.967	32	SVM, lin. kernel, C_4

TABLE 1

Outex10 results. The relevant parameter values are $C_1 = 936.5391, \gamma_1 = 0.0187, C_2 = 914.9620, \gamma_2 = 0.0061, C_3 = 86.0442,$ and $C_4 = 998.1848.$

Vectorization Method	Score	Param.	Estimator
Persistence Statistics	0.934		SVM, rbf kernel, C_1, γ_1
Entropy Summary	0.859	100	SVM, poly kernel, $C_2, \gamma_2, \text{deg}=2$
Algebraic Functions	0.875		SVM, lin. kernel, C_4
Tropical Coordinates	0.887	50	SVM, lin. kernel, C_5
Complex Polynomial	0.846	10, R	SVM, lin. kernel, C_4
Betti Curve	0.804	200	SVM, rbf kernel, C_1, γ_1
Lifespan Curve	0.842	100	SVM, rbf kernel, C_1, γ_1
Persistence Landscape	0.822	50, 20	SVM, rbf kernel, C_3, γ_3
Persistence Silhouette	0.844	100, 1	SVM, lin. kernel, C_4
Persistence Image	0.762	1, 150	RF, n=500
Template Function	0.831	35, 20	RF, n=200
Adaptive Template	0.819	GMM, 50	SVM, lin. kernel, C_6
ATOL	0.854	16	SVM, lin. kernel, C_7

TABLE 2

Outex68 results. The optimal parameter values are $C_1 = 936.5391, \gamma_1 = 0.0187, C_2 = 957.5357, \gamma_2 = 0.0120, C_3 = 914.9620, \gamma_3 = 0.0061,$ $C_4 = 998.1848, C_5 = 884.1255, C_6 = 143.1201$ and $C_7 = 494.0596.$

4.3 FMNIST

The Fashion-MNIST database contains 28×28 grayscale images (7,000 images per class, with 10 classes) — see the left side of Fig. 2 for some sample images. We split this dataset into 60,000 training and 10,000 testing images.

The filtration used for generating barcodes is as follows: we performed padding, median filter, and shallow thresholding before computing *canny edges* [59]. Then each pixel is given a filtration value equalling its distance from the edge-pixels. Finally, all other cells inherit filtration values from the top pixels via the lower star filtration rule.

5 RESULTS

Here we report the classification accuracy of the thirteen vectorization methods from Section 3 on each of the three datasets from Section 4. Implementation details and parameter choices are provided in Appendix A. The source code is available at the following GitHub repository: <https://github.com/Cimagroup/vectorization-maps>.

5.1 Outex

Table 1 displays the classification accuracy for the smaller (and easier) experiment on 10 classes. As one might expect,

all techniques perform rather well, with **Persistence Statistics** and **Algebraic Functions** sharing the best performance with 99.2% accuracy each, followed closely by Persistent Silhouettes with 98.3% each.

Results from the full experiment with 68 classes are contained in Table 2; as one might expect, the performance of every single vectorization degrades in the passage from Outex10 to Outex68. Here **Persistence Statistics** is the clear winner by a significant margin, earning 93.4% accuracy. Tropical Coordinates ranks second with 88.7%. Setting aside the outstanding performance of Persistence Statistics, it appears clear from these results that the algebraic vectorizations perform far better on Outex68 than the vectorizations from the other categories.

We note that the authors of [20] have also used Outex to compare the performance of various curve vectorizations, with Persistence Statistics being used as a baseline. They also obtained their best results with Persistence Statistics.

5.2 SHREC14

We used 10 different t -values $t_1 < t_2 < \dots < t_{10}$, as in [9], [38], [52], for generating filtrations via the heat kernel from (6). At t_{10} we found several sparse or empty barcodes, which led us to discard that classification problem. Table 3 collects the best performance for each method across the

Vectorization Method	Score	Param.	Estimator
Persistence Statistics	0.947	t_5	RF, n=100
Entropy Summary	0.723	$t_6, 200$	RF, n=300
Algebraic Functions	0.909	t_5	RF, n=500
Tropical Coordinates	0.844	$t_6, 50$	SVM, lin. kernel, C_5
Complex Polynomial	0.889	$t_6, 20, S$	SVM, lin. kernel, C_6
Betti Curve	0.728	$t_5, 200$	RF, n=100
Lifespan Curve	0.878	$t_7, 200$	SVM, lin. kernel, C_7
Persistence Landscape	0.889	$t_6, 50, 10$	SVM, rbf kernel, C_1, γ_1
Persistence Silhouette	0.867	$t_6, 200, 2$	SVM, rbf kernel, C_2, γ_2
Persistence Image	0.916	$t_5, 1, 10$	RF, n=100
Template Function	0.944	$t_5, 14, 0.7$	SVM, rbf kernel, C_3, γ_3
Adaptive Template Sys.	0.889	$t_5, \text{GMM}, 15$	SVM, lin. kernel, C_8
ATOL	0.933	$t_8, 16$	SVM, rbf kernel, C_4, γ_4

TABLE 3

Best performance of each method on SHREC14. The parameters are $C_1 = 835.6257, \gamma_1 = 0.0002, C_2 = 212.6281, \gamma_2 = 0.0031, C_3 = 879.1425, \gamma_3 = 0.0010, C_4 = 936.5391, \gamma_4 = 0.0187, C_5 = 141.3869, C_6 = 625.0300, C_7 = 998.1848, C_8 = 274.500$.

first 9 values of t ; it also contains values of the optimal parameters (see Appendix A) and the optimal values of t .

Persistence Statistics yielded the best classification accuracy of 94.7%, followed closely by Template Functions at 94.4%. One remarkable feature of these results is that the dataset does not appear to favour any one category of vectorizations over the other — it is possible to achieve over 88% accuracy by using a suitable statistical, algebraic, curve, functional or ensemble vectorization. In fact, only the curve-based vectorizations failed to achieve over 90% accuracy on this dataset. The variation of classification accuracy with the heat kernel parameter t is discussed in Appendix B.

5.3 FMNIST

The results of our experiments on FMNIST are recorded in Table 4. We note that these experiments only used information contained in the 0-dimensional barcodes and that the SVM classifier was not used. The classification accuracy of all the methods is much lower than the corresponding figures for the two preceding datasets. Once more, the **Persistence Statistics** vectorization takes the top spot with 74.9% and Template Functions are slightly behind at 74.7%

Vectorization Method	Accuracy	Parameters
Persistence Statistics	0.749	
Entropy Summary	0.696	30
Algebraic Functions	0.710	
Tropical Coordinates	0.696	10
Complex Polynomial	0.661	10, R
Betti Curve	0.618	50
Lifespan Curve	0.692	30
Persistence Landscape	0.694	30, 5
Persistence Silhouette	0.670	30, 0
Persistence Image	0.698	1, 12
Template Functions	0.747	10, 2
Adaptive Template System	0.602	GMM, 5
ATOL	0.730	16

TABLE 4

FMNIST results. All the scores have been achieved for Random Forest classifier with 100 trees.

One rather surprising aspect of these results is the fact that Adaptive Template Systems performed far worse than

ordinary Template Functions despite having recourse to 60,000 training barcodes. We do not have a clear explanation for this phenomenon, particularly in light of a fairly competitive performance from ATOL (which was also exposed to the same training data).

6 WEB APPLICATION

In order to illustrate and visualize the vectorization methods described here, we have built an interactive web application **Brava** that runs on any modern browser; it is available at

<https://persistent-homology.streamlit.app/>

The app has been built in Python using the Streamlit library along with several existing Python libraries. The sidebar contains options for selecting different types of input data and displays several options for data visualization. One sample image/point-cloud from each of the three datasets used in this paper has been pre-loaded, but the user is free to upload their own data. Specifications, formatting guidelines, and downloading instructions are available in our GitHub repository:

<https://github.com/dashtiali/vectorisation-app>

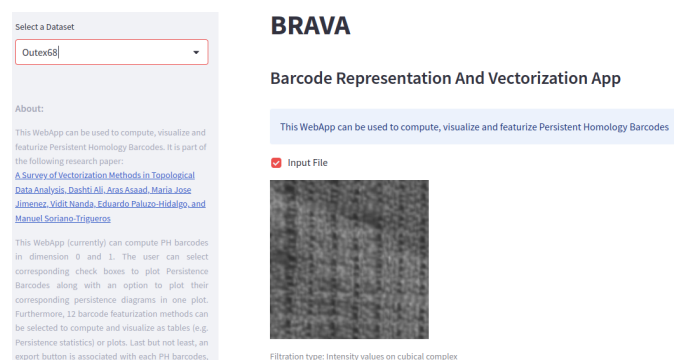


Fig. 3. A screenshot of the web app

All of the barcode vectorization methods considered in this paper can be computed and visualized in different formats (tables, bar graphs, scatter plots), depending on the

Persistence Barcodes

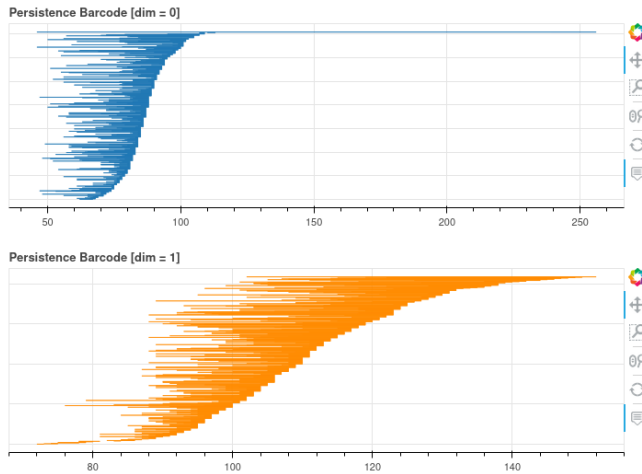


Fig. 4. Intervals in barcodes of dimensions 0 and 1 as displayed by the web app.

type of vectorization being invoked. Barcodes are computed by default in dimensions 0 and 1, depicted as in Figure 4.

The vectorizations are depicted in Brava as follows.

- 1) The Persistence Statistics vectorization is numerical, so we show its values in a table, as in Figure 5.

Persistence Statistics

Persistence Statistics [dim = 0]

	Mean	STD	Median	IQR	Range	P10	P25	P75	P90
Births	73.8873	11.5833	76.0000	15.5000	64.0000	58.0000	66.0000	81.5000	87.0000
Deaths	87.5718	12.2165	87.0000	10.0000	189.0000	77.0000	82.0000	92.0000	99.6000
Midpoints	80.7296	9.1496	80.0000	11.0000	91.0000	63.0000	74.0000	88.0000	94.0000
Lifespans	13.6845	15.2305	10.0000	15.0000	209.0000	2.0000	4.0000	19.0000	29.0000

	Count	Entropy
Pd0	355.0000	5.4686

Fig. 5. The Persistence Statistics vectorization as shown in the web app.

- 2) Algebraic vectorizations are illustrated as bar graphs. In Figure 6, for instance, one finds bars whose heights correspond to values attained by the 7 chosen tropical coordinate polynomials on the input barcodes.
- 3) Curve vectorizations, such as persistence landscapes, are depicted via piecewise-linear graphs (see Figure 7). Sliders have been provided to set the resolution parameter.
- 4) Persistence images are displayed as heat maps — see Figure 8.
- 5) Template Functions, their adaptive version, and ATOL are all displayed as bar graphs with heights of bars indicating the values of the selected functions. Figure 9, for instance, depicts Template Functions.

It is our hope that users will benefit from the ability to generate these visualizations without having to write any code of their own. In order to facilitate downstream

Tropical Coordinates

Log values are displayed for visualization

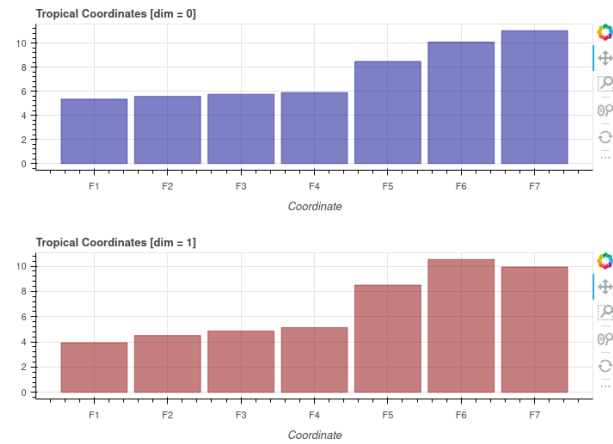


Fig. 6. A visualization of the Tropical Coordinates vectorization from the web app.

Persistence Landscapes

Resolution = 100

Number of landscapes

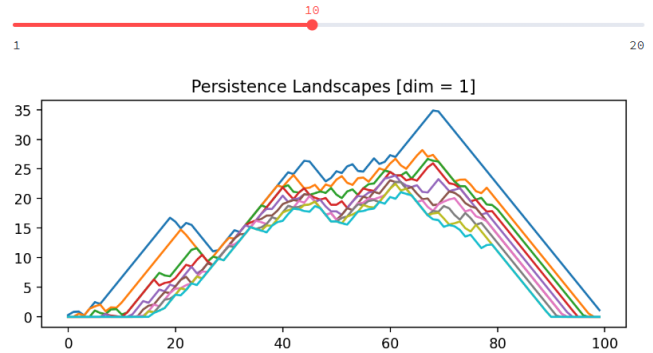


Fig. 7. Persistence landscapes in the web app

Persistence Image

Bandwidth of the Gaussian kernel = 1, Weight function = Constant (i.e lambda x1)

Resolution

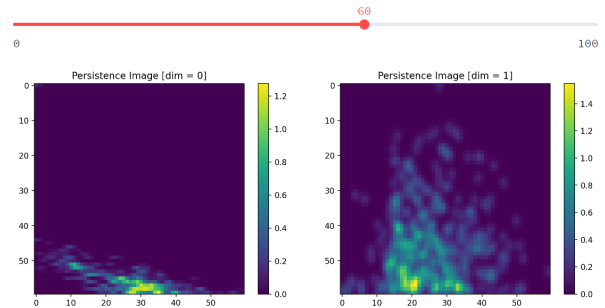


Fig. 8. Persistence images as shown in the web app

analysis, the web app also provides the ability to download the vectors generated by each vectorization method.

Template Function

Number of bins in each axis = 5, Padding = 2

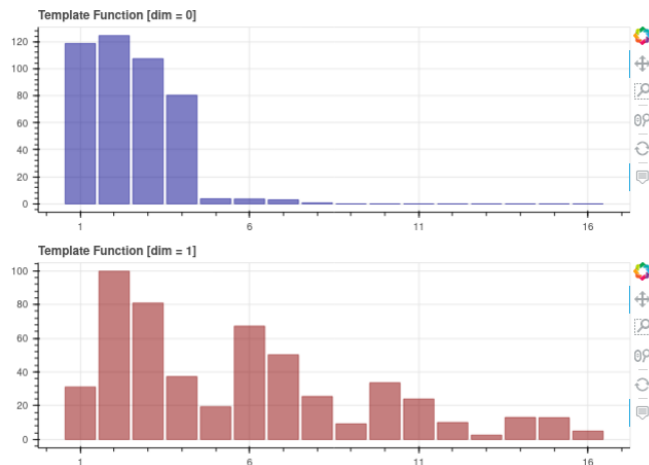


Fig. 9. The web app visualization of template functions

7 CONCLUSION

At the time of writing, it remains difficult to accurately pinpoint those attributes which might make a given vectorization method a good choice for a particular classification problem. There are no powerful theorems or immutable doctrines available to guide scientists who wish to incorporate topological information into machine learning pipelines. In the absence of such theoretical foundations, the best that one can expect are principled heuristics supported by reproducible empirical evidence. This paper is an outcome of our efforts to provide such evidence. En route, we have organized thirteen available vectorization methods into five categories in Section 3 and provided a web application which will allow others to conduct their own experiments involving these methods.

One possible conclusion that may be drawn from the results of Section 5 is that we can dispense with sophisticated vectorization techniques and only use (some variant of) Persistence Statistics. We do not necessarily suggest such a course of action. While it is certainly true that Persistence Statistics earned top honors in all of our experiments and is much faster to compute than the alternatives, there are other factors to consider. In particular, no comparative study such as ours can be truly exhaustive. There is always the chance that making different choices – for instance, using another dataset for classification, or adding some new polynomials to one of the algebraic vectorizations – could dramatically update our priors about which methods perform best.

APPENDIX A

IMPLEMENTATION AND PARAMETER DETAILS

We have made use of several existing software packages, such as GUDHI [60], Teaspoon⁵ or Scikit-learn [61], as well as our own implementations in some cases. Salient information regarding each method has been provided in the list below. Full details can be found in the GitHub repository accompanying this paper⁶.

5. <https://lizliz.github.io/teaspoon/index.html>

6. <https://github.com/Cimagroup/vectorization-maps>

A.1 Persistence Statistics

The persistence statistics vectorization from Definition 3.1 requires no additional parameters. We have implemented this method ourselves.

A.2 Entropy Summary Function

We have used the GUDHI implementation of the entropy summary function from Definition 3.2. There is a single resolution parameter which selects the grid points on which the entropy summary function is sampled.

A.3 Algebraic Functions

The algebraic functions of Definition 3.3 are implemented in the Teaspoon package. For reasons which remain unclear to us, this implementation includes a fifth tropical polynomial $f_5 = \max_i \{(q_i - p_i)\}$ beyond the four ordinary polynomials f_1, \dots, f_4 which were described after Definition 3.3. We do not expect that removing this function will improve the results described below.

A.4 Tropical Coordinates

We have implemented the tropical polynomials F_1, \dots, F_7 described after Definition 3.4. The parameter r has been optimized over the set $\{10, 50, 250, 500, 800\}$ for Outex and SHREC14, and over $\{10, 50, 250\}$ for FMNIST.

A.5 Complex Polynomials

We have used the GUDHI implementation of complex polynomials, which have been described in Definition 3.5. We generated the polynomials with respect to all three of the transformations $R, S, T : \mathbb{R}^2 \rightarrow \mathbb{C}$. The number of coefficients used was chosen from $\{5, 10, 20\}$ for Outex and SHREC14 and $\{3, 5, 10\}$ for FMNIST.

A.6 Betti Curve

The Betti curve vectorization from Definition 3.6 has been implemented in GUDHI, and it only requires a resolution parameter. This parameter was chosen from $\{50, 100, 200\}$ for Outex and SHREC14 and $\{15, 30, 50\}$ for FMNIST.

A.7 Lifespan Curve

We implemented the lifespan curve ourselves, with a resolution parameter optimised across the set $\{50, 100, 200\}$ for Outex and SHREC14 and across the set $\{15, 30, 50\}$ for FMNIST.

A.8 Persistence Landscapes

We have used the GUDHI implementation of persistence landscapes (see Definition 3.8). There are two parameters to consider: the resolution (to identify the grid points where each landscape is sampled) and the total number of landscapes used. The resolution was optimized over $\{50, 100, 200\}$ for Outex and SHREC14, and over $\{15, 30, 50\}$ for FMNIST; the number of landscapes ranged over $\{2, 5, 10, 20\}$ for Outex and SHREC14 and over $\{1, 2, 3, 5\}$ for FMNIST.

Method	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Pers Stat	0.729	0.785	0.662	0.704	0.947	0.910	0.915	0.915	0.908
Ent Sum	0.378	0.333	0.522	0.536	0.656	0.723	0.633	0.656	0.530
Alg Fun	0.467	0.456	0.556	0.567	0.909	0.878	0.863	0.833	0.711
Trop Coord	0.505	0.556	0.522	0.612	0.822	0.844	0.833	0.767	0.800
Com Poly	0.322	0.456	0.400	0.467	0.856	0.889	0.844	0.850	0.790
Bet Cur	0.511	0.467	0.628	0.660	0.728	0.633	0.611	0.644	0.536
Lif Cur	0.456	0.411	0.593	0.639	0.789	0.833	0.878	0.833	0.798
Pers Land	0.700	0.511	0.789	0.778	0.878	0.889	0.857	0.833	0.789
Pers Sil	0.400	0.378	0.556	0.589	0.811	0.867	0.856	0.856	0.656
Pers Img	0.644	0.691	0.795	0.856	0.916	0.794	0.871	0.811	0.718
Temp Func	0.778	0.735	0.933	0.789	0.944	0.919	0.908	0.932	0.922
Ad Temp Sys	0.802	0.872	0.833	0.727	0.889	0.856	0.889	0.844	0.633
ATOL	0.828	0.786	0.911	0.833	0.906	0.867	0.900	0.933	0.867

TABLE 5
Best Results for SHREC14 for various vectorization methods across nine t -values.

A.9 Persistence Silhouette

We have used the GUDHI implementation of persistence silhouettes (see Definition 3.9). The resolution parameter was optimized over $\{50, 100, 200\}$ for Outex and SHREC14 and over $\{15, 30, 50\}$ for FMNIST; the weight w ranged over $\{0, 1, 2, 5, 10, 20\}$ for Outex and SHREC14 and $\{0, 1, 2, 5\}$ for FMNIST.

A.10 Persistence Images

Persistence images (from Definition 3.10) have been implemented in GUDHI. The resolution parameter r , which results in images of size $r \times r$, ranged over $\{25, 75, 150\}$ for Outex, over $\{10, 20, 40\}$ for SHREC14, and over $\{3, 6, 12, 20\}$ for FMNIST. Bandwidth values of the Gaussian kernel (σ in Definition 3.10) were chosen from $\{0.05, 1\}$ for Outex and from $\{0.05, 0.5, 1\}$ for both, SHREC14 and FMNIST.

A.11 Template Functions

We have used code from the repository⁷ provided with the paper [38] for computing template functions (see Definition 3.11). We use tent functions as described in (2), which require two parameters: a grid resolution δ and a padding parameter π (for enlarging the area covered by the square grid). We optimized over

- δ in $\{35, 50, 65\}$ and π in $\{20, 25, 30\}$ for Outex;
- δ in $\{3, 4, \dots, 14, 15\}$ and π in $\{0.5, 0.6, \dots, 1.1, 1.2\}$ for SHREC14;
- δ in $\{2, 3, 5, 10\}$ and π in $\{0.5, 1, 2\}$ for FMNIST.

A.12 Adaptive Template Systems

The implementation of adaptive template systems (Definition 3.12) has also been sourced from the same repository as template functions. We have used the Gaussian mixture model for generating ellipsoidal domains, and require only one parameter: the number of clusters. This has been optimized over

- $\{10, 20, 30, 40, 50\}$ for Outex,
- $\{5, 10, 15, 20, 25, 30, 35, 40, 45\}$ for SHREC14, and
- $\{3, 4, 5, 10, 15\}$ for FMNIST.

7. https://github.com/lucho8908/adaptive_template_systems

A.13 ATOL

The ATOL vectorization from Definition 3.13 has been implemented in GUDHI, and it also requires the number of functions b as a parameter. We have optimized this over $\{2, 4, 8, 16, 32, 64\}$ for Outex and over $\{2, 4, 8, 16\}$ for both SHREC14 and FMNIST.

A.14 Dimensions, Classifiers and Hyperparameters

In the case of Outex, we have concatenated vectors arising from barcodes of dimensions 0 and 1; for SHREC14, the vectors computed from only dimension 1 barcodes performed better, so the results are only reported for them. Finally, only dimension 0 barcodes were taken to build vectors for FMNIST. We considered both Support Vector Machine (SVM) [62] and Random Forest (RF) [63] classifiers. Due to convergence issues, only RF has been performed for FMNIST.

For each parameter of each vectorization method, we accomplished a hyperparameter optimization process based on random search (when optimizing SVM and RF jointly) or grid search (for optimizing RF), with 5-fold cross-validation on the training data, to find the best (hyper)parameters for both the machine learning models and the vectorization methods; then, we assigned to each method the parameters with the best average score among all the 5-fold cross-validation scheme; finally the vectorization methods were evaluated on the test dataset 100 times, to report the average accuracy.

APPENDIX B

HEAT KERNEL PARAMETER DEPENDENCE

As mentioned in Section 4, the filtration for SHREC14 is generated using the Heat Kernel Signature (6) which depends on a single parameter t . In Table 5 we depict the best classification accuracy of each vectorization method across all 9 values of t which were used in our experiments.

We note that for small values of t , the ensemble vectorizations perform best, whereas for intermediate and larger values both ensemble and functional vectorizations achieve good performance. The algebraic and curve based vectorizations perform quite poorly for low t -values, but tend to become more competitive between t_5 and t_8 .

ACKNOWLEDGMENTS

M.J. Jimenez, E. Paluzo-Hidalgo and M. Soriano-Trigueros are funded by the Spanish grant Ministerio de Ciencia e Innovacion TED2021-129438B-I00, grant PID2019-107339GB-I00, and by the REXASI-PRO project of the European Union's Horizon 2020 research and innovation program under grant agreement No 101070028-2.

M.J. Jimenez is also funded by a grant of Convocatoria de la Universidad de Sevilla para la recualificacion del sistema universitario español, 2021-23, funded by the European Union, NextGenerationEU.

V. Nanda is supported by EPSRC grant EP/R018472/1 and by US AFOSR grant FA9550-22-1-0462.

We are grateful to the team of GUDHI and TEASPOON developers, for their work and their support. We are also grateful to Streamlit for providing extra resources to deploy the web app online on Streamlit community cloud.

REFERENCES

- [1] R. Ghrist, "Barcodes: the persistent topology of data," *Bull. Amer. Math. Soc. (N.S.)*, vol. 45, no. 1, pp. 61–75, 2008. [Online]. Available: <http://dx.doi.org.proxy.libraries.rutgers.edu/10.1090/S0273-0979-07-01191-3>
- [2] G. Carlsson, "Topology and data," *Bull. Amer. Math. Soc. (N.S.)*, vol. 46, no. 2, pp. 255–308, 2009. [Online]. Available: <http://dx.doi.org.proxy.libraries.rutgers.edu/10.1090/S0273-0979-09-01249-X>
- [3] V. Nanda and R. Sazdanovic, "Simplicial models and topological inference in biological systems," in *Discrete and Topological Models in Molecular Biology*. Springer, 2014, pp. 109–141.
- [4] S. Y. Oudot, *Persistence theory: from quiver representations to data analysis*. American Mathematical Soc., 2017, vol. 209.
- [5] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer, 2018.
- [6] T. Ojala, T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen, and S. Huovinen, "Outex-new framework for empirical evaluation of texture analysis algorithms," in *2002 International Conference on Pattern Recognition*, vol. 1. IEEE, 2002, pp. 701–706.
- [7] D. Pickup, X. Sun, P. L. Rosin, R. R. Martin, Z. Cheng, Z. Lian, M. Aono, A. Ben Hamza, A. Bronstein, M. Bronstein, S. Bu, U. Castellani, S. Cheng, V. Garro, A. Giachetti, A. Godil, J. Han, H. Johan, L. Lai, B. Li, C. Li, H. Li, R. Litman, X. Liu, Z. Liu, Y. Lu, A. Tatsuma, and J. Ye, "SHREC'14 track: Shape retrieval of non-rigid 3d human models," in *Proceedings of the 7th Eurographics workshop on 3D Object Retrieval*, ser. EG 3DOR'14. Eurographics Association, 2014.
- [8] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
- [9] J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt, "A stable multi-scale kernel for topological machine learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4741–4748.
- [10] M. Carriere, M. Cuturi, and S. Oudot, "Sliced wasserstein kernel for persistence diagrams," in *International conference on machine learning*. PMLR, 2017, pp. 664–673.
- [11] I. Chevyrev, V. Nanda, and H. Oberhauser, "Persistence paths and signature features in topological data analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 1, pp. 192–202, 2020.
- [12] C. Giusti and D. Lee, "Signatures, lipschitz-free spaces, and paths of persistence diagrams," *arXiv preprint arXiv:2108.02727*, 2021.
- [13] M. Carrière, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda, "Perslay: A neural network layer for persistence diagrams and new graph topological signatures," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2786–2796.
- [14] C. D. Hofer, R. Kwitt, and M. Niethammer, "Learning representations of persistence barcodes." *J. Mach. Learn. Res.*, vol. 20, no. 126, pp. 1–45, 2019.
- [15] A. Keros, V. Nanda, and K. Subr, "Dist2Cycle: a simplicial neural network for homology localization," in *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, 2022.
- [16] M. Moor, M. Horn, B. Rieck, and K. Borgwardt, "Topological autoencoders," in *International conference on machine learning*. PMLR, 2020, pp. 7045–7054.
- [17] G. Carlsson and R. B. Gabrielsson, "Topological approaches to deep learning," in *Topological data analysis*. Springer, 2020, pp. 119–146.
- [18] C. S. Pun, K. Xia, and S. X. Lee, "Persistent-homology-based machine learning and its applications—a survey," *arXiv preprint arXiv:1811.00252*, 2018.
- [19] A. Som, K. N. Ramamurthy, and P. Turaga, "Geometric metrics for topological representations," in *Handbook of Variational Methods for Nonlinear Geometric Data*. Springer, 2020, pp. 415–441.
- [20] Y. Chung and A. Lawson, "Persistence curves: A canonical framework for summarizing persistence diagrams," *Adv. Comput. Math.*, vol. 48, no. 1, p. 6, 2022. [Online]. Available: <https://doi.org/10.1007/s10444-021-09893-4>
- [21] D. Barnes, L. Polanco, and J. A. Perea, "A comparative study of machine learning methods for persistence diagrams," *Frontiers in Artificial Intelligence*, vol. 4, 2021. [Online]. Available: <https://www.frontiersin.org/article/10.3389/frai.2021.681174>
- [22] F. Conti, D. Moroni, and M. A. Pascali, "A topological machine learning pipeline for classification," *Mathematics*, vol. 10, no. 17, p. 3086, 2022.
- [23] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete & Computational Geometry*, vol. 33, no. 2, pp. 249–274, 2005.
- [24] F. Chazal, V. de Silva, M. Glisse, and S. Oudot, *The Structure and Stability of Persistence Modules*. Springer, 2016.
- [25] K. Turner, Y. Mileyko, S. Mukherjee, and J. Harer, "Fréchet means for distributions of persistence diagrams," *Discrete & Computational Geometry*, vol. 52, pp. 44–70, 2014.
- [26] V. de Silva and V. Nanda, "Geometry in the space of persistence modules," in *Proceedings of the 29th Annual Symposium on Computational Geometry*, ACM, 2013, pp. 397–404.
- [27] P. Bubenik, V. de Silva, and V. Nanda, "Higher interpolation and extension for persistence modules," *SIAM Journal on Applied Algebra and Geometry*, vol. 1, no. 1, pp. 272–284, 2017.
- [28] M. A. Armstrong, *Basic Topology*. Springer, 1983.
- [29] T. Kaczynski, K. M. Mischaikow, M. Mrozek, and K. Mischaikow, *Computational homology*, ser. Applied mathematical sciences (Springer-Verlag New York Inc.); v. 157. New York: Springer, 2004.
- [30] A. Hatcher, *Algebraic topology*. Cambridge University Press, 2002.
- [31] H. Edelsbrunner and J. Harer, *Computational Topology - an Introduction*. American Mathematical Society, 2010.
- [32] T. K. Dey and Y. Wang, *Computational Topology for Data Analysis*. Cambridge University Press, 2022.
- [33] A. Asaad, D. Ali, T. Majeed, and R. Rashid, "Persistent homology for breast tumor classification using mammogram scans," *Mathematics*, vol. 10, no. 21, 2022. [Online]. Available: <https://www.mdpi.com/2227-7390/10/21/4039>
- [34] H. Chintakunta, T. Gentimis, R. Gonzalez-Diaz, M. Jimenez, and H. Krim, "An entropy-based persistence barcode," *Pattern Recognit.*, vol. 48, no. 2, pp. 391–401, Feb. 2015. [Online]. Available: <https://doi.org/10.1016/j.patcog.2014.06.023>
- [35] M. Rucco, F. Castiglione, E. Merelli, and M. Pettini, "Characterisation of the idiosyncratic immune network through persistent entropy," in *Proceedings of ECCS 2014*. Springer International Publishing, 2016, pp. 117–128. [Online]. Available: https://doi.org/10.1007/978-3-319-29228-1_11
- [36] N. Atienza, R. Gonzalez-Diaz, and M. Soriano-Trigueros, "On the stability of persistent entropy and new summary functions for topological data analysis," *Pattern Recognition*, vol. 107, p. 107509, 2020.
- [37] A. Adcock, E. Carlsson, and G. Carlsson, "The ring of algebraic functions on persistence bar codes," *Homology, Homotopy Appl.*, vol. 18, p. 381–402, 2016.
- [38] J. A. Perea, E. Munch, and F. A. Khasawneh, "Approximating continuous functions on persistence diagrams using template functions," *Foundations of Computational Mathematics*, 2022. [Online]. Available: <https://doi.org/10.1007/s10208-022-09567-7>
- [39] S. Kališnik, "Tropical coordinates on the space of persistence barcodes," *Foundations of Computational Mathematics*, vol. 19, no. 1, p. 101–129, 2019.
- [40] M. Ferri and C. Landi, "Representing size functions by complex polynomials," *Proc. Math. Met. in Pattern Recognition*, vol. 9, pp. 16–19, 1999.

[41] B. Di Fabio and M. Ferri, "Comparing persistence diagrams through complex vectors," in *Image Analysis and Processing — ICIAP 2015*, V. Murino and E. Puppo, Eds. Cham: Springer International Publishing, 2015, pp. 294–305.

[42] I. Chevrete and A. Kormilitzin, "A primer on the signature method in machine learning," *arXiv:1603.03788 [stat.ML]*, 2016.

[43] P. Bubenik, "Statistical topological data analysis using persistence landscapes," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 77–102, Jan. 2015. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2789272.2789275>

[44] P. Bubenik and P. Dłotko, "A persistence landscapes toolbox for topological statistics," *Journal of Symbolic Computation*, vol. 78, pp. 91–114, 2017.

[45] 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*, ser. SOCG'14. New York, NY, USA: Association for Computing Machinery, 2014, p. 474–483. [Online]. Available: <https://doi.org/10.1145/2582112.2582128>

[46] 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*, vol. 28, no. 3, pp. 671–681, 2019.

[47] K. Xia, "Persistent similarity for biomolecular structure comparison," *Communications in Information and Systems*, vol. 18, no. 4, pp. 269–298, 2018.

[48] Z. Dong, C. Hu, C. Zhou, and H. Lin, "Vectorization of persistence barcode with applications in pattern classification of porous structures," *Computers & Graphics*, vol. 90, pp. 182–192, 2020.

[49] 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," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 218–252, 2017.

[50] E. Berry, Y.-C. Chen, J. Cisewski-Kehe, and B. T. Fasy, "Functional summaries of persistence diagrams," *Journal of Applied and Computational Topology*, vol. 4, no. 2, pp. 211–262, 2020.

[51] L. Xian, H. Adams, C. M. Topaz, and L. Ziegelmeier, "Capturing dynamics of time-varying data via topology," *Foundations of Data Science*, vol. 4, no. 1, 2022.

[52] L. Polanco and J. A. Perea, "Adaptive template systems: Data-driven feature selection for learning with persistence diagrams," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 1115–1121.

[53] S. Tymochko, E. Munch, and F. A. Khasawneh, "Adaptive partitioning for template functions on persistence diagrams," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 1227–1234.

[54] M. Royer, F. Chazal, C. Levrard, Y. Umeda, and Y. Ike, "Atol: Measure vectorization for automatic topologically-oriented learning," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130. PMLR, 4 2021, pp. 1000–1008. [Online]. Available: <https://proceedings.mlr.press/v130/royer21a.html>

[55] B. Zieliński, M. Lipiński, M. Juda, M. Zeppelzauer, and P. Dłotko, "Persistence codebooks for topological data analysis," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1969–2009, 2021.

[56] W. Guo, K. Manohar, S. L. Brunton, and A. G. Banerjee, "Sparse-tda: Sparse realization of topological data analysis for multi-way classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1403–1408, 2018.

[57] P. Dłotko, "Cubical complex," in *GUDHI User and Reference Manual*, 3.6.0 ed. GUDHI Editorial Board, 2022. [Online]. Available: https://gudhi.inria.fr/doc/3.6.0/group__cubical__complex.html

[58] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Computer graphics forum*, vol. 28. Wiley Online Library, 2009, pp. 1383–1392.

[59] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[60] The GUDHI Project, *GUDHI User and Reference Manual*, 3.6.0 ed. GUDHI Editorial Board, 2022. [Online]. Available: <https://gudhi.inria.fr/doc/3.6.0/>

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg,

J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[62] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[63] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.



Dashti Ali is a machine learning and software developer. He obtained his MSc in scientific computation from the university of Nottingham in 2013. He does research in machine learning and applications of applied and computational topology with particular interest in digital image analysis. As a software developer based in Canada, he develops cross platform applications. In addition, he is a research fellow at Koya University.



Aras Asaad is a mathematician and machine learning researcher. His research interests are mainly in applications of applied topology and geometry, with particular interest in computer vision. He obtained his PhD in mathematics and computing from The University of Buckingham in 2020. He is an honorary junior research fellow in school of computing at The University of Buckingham and research fellow at Koya University.



Maria-Jose Jimenez does her research on applications of computational topology, with special interest on biological data. She obtained her PhD in mathematics in 2003 from Universidad de Sevilla, where she is an Associate professor. She is currently doing a one-year research stay in the Mathematical Institute, University of Oxford.



Vidit Nanda is an applied and computational algebraic topologist. He obtained his PhD in mathematics from Rutgers University in 2012, and has held postdoctoral positions at the University of Pennsylvania, the Alan Turing Institute and the Institute for Advanced Study. In 2018 he joined the Mathematical Institute at the University of Oxford as an Associate Professor.



Eduardo Paluzo-Hidalgo has research interests primarily in machine learning and computational topology. He obtained his PhD in mathematics in 2021 from Universidad de Sevilla. He is currently a researcher in the Department of Applied Math I at Universidad de Sevilla, Spain, and a member of the Combinatorial Image Analysis group.



Manuel Soriano-Trigueros specializes in topological persistence and topological data analysis. He obtained his PhD in mathematics at Universidad de Sevilla in 2023. In 2017, he obtained an MSc in Mathematics from the same university. He also studied one year as an exchange student at Warwick University. Currently, he is a postdoctoral researcher at the Institute of Science and Technology Austria.