Pull-back Geometry of Persistent Homology Encodings

Anonymous authors
Paper under double-blind review

Abstract

Persistent homology (PH) is a method that allows to create topology-inspired representations of data. In contrast to data-driven representation methods, it is an expert-based method with explicit computations designed by hand. Nonetheless, owing to the complexity of these computations, its properties and interpretation have remained elusive. We investigate the spectrum of the Jacobian of PH data encodings and the pull-back geometry that they induce on the data manifold. Then, by measuring different perturbations and features on the data manifold with respect to this geometry, we can identify which of them are recognized or ignored by the PH encodings. This also allows us to compare different encodings in terms of their induced geometry. Importantly, the approach does not require training and testing on a particular task and permits a direct exploration of PH. We experimentally demonstrate that the pull-back norm can be used as a predictor of performance on downstream tasks and to select suitable PH encodings accordingly.

Keywords: Topological data analysis, data representation, Jacobian spectrum, pull-back geometry, sensitivity analysis

1 Introduction

Persistent homology (PH) is a well-established technique in applied and computational topology (Carlsson, 2009; Oudot, 2015). At its core, PH seeks to create *representations* of data that highlight topological aspects. Recently, they have been found to also capture purely geometric aspects of the data, such as curvature (Collins et al., 2004; Bubenik et al., 2020) and convexity (Turkeš et al., 2022). PH representations have been used particularly in applications where multiscale homological features can be expected to capture relevant information, such as in the prediction of biomolecular properties (Cang & Wei, 2017; Cang et al., 2018; Wang et al., 2020), quantification of similarity in materials (Lee et al., 2017), medical imaging (Singh et al., 2023), or the analysis of tree-like brain artery structures (Bendich et al., 2016). PH also increasingly interfaces with machine learning (see, e.g., Carriere et al., 2020; Hensel et al., 2021). Beside serving as a data representation technique, PH has been used, for instance, to investigate the decision boundaries of neural networks (Ramamurthy et al., 2019) or the transformations of data sets across the layers of a deep neural network (Naitzat et al., 2020). Other work has explored training neural networks to approximate PH features (Hofer et al., 2019; Montúfar et al., 2020; de Surrel et al., 2022), which can facilitate faster computations or serve as a basis for fine-tuning PH representations. Along this thread, a topology-encoding neural network based on PH was proposed by Haft-Javaherian et al. (2020).

Despite the success and interest in PH representations, and although PH computations are defined in a specific and deliberate way, due to their complexity, the properties and interpretation of such representations have remained elusive. Particularly, it is unclear which aspects of the data are highlighted or which ones are suppressed in the PH representations, or which choices of the PH representation parameters are most useful for particular downstream tasks. Recently, there has been growing interest in studying the interpretation and usefulness of PH, as we detail in "Related work" below. However, the majority of these studies evaluate the behavior of PH through the lens of the performance of some chosen model trained on the data representations.

In our work we seek to evaluate the properties of PH encodings irrespective of particular models used to solve specific downstream tasks. Here, we use PH *encoding* to denote the mapping from data to vectorized representations of the associated PH. We use the pull-back geometry induced by the PH encoding map

to investigate its sensitivity to a particular data variation. We understand data variation as an umbrella term, which includes both data perturbations (e.g., translation or dilation of a point cloud), as well as data features (e.g., a label indicating the presence of anomaly or disease). By optimizing the average pull-back norm of the perturbation or gradient vector field (for data perturbations and features, respectively), this approach can guide the choice of suitable PH encodings. We will focus on a common PH encoding called the persistence image (PI) (Adams et al., 2017), which can be thought of as an image-like representation of the input data in terms of multiscale homological features. We note that our methods can be extended to other PH representations, provided they are differentiable and that the representation space can be endowed with a Riemannian manifold structure (for an example, see Anirudh et al., 2016).

Main contributions

- We present an approach that can be used to investigate persistence images and their induced geometry on the manifold of input data sets in terms of the rank, spectrum, and singular vectors of the Jacobian, as well as the pull-back norm of tangent vectors on the data manifold (Section 3)[1]. Based on our approach, we can compare different PH encodings independently of their specific performance on selected tasks, or also guide the selection of suitable encodings for particular tasks.
- We show how the above approach can be used experimentally to identify which data perturbations (such as rotation, translation, or noise) are captured by the encodings and which are ignored (Section 4).
- We show experimentally how the above approach can be used to investigate to what extent a PH encoding can recognize a data feature of interest (*sex* feature in a brain artery trees data set) (Section 5). Finally, we also show that the pull-back norm is related to the performance on a downstream task.

Related work Our discussion falls within the general subject of interpreting a complex nonlinear map by investigating the effect that local input perturbations have on the output. This is conceptually related to topics such as sensitivity analysis (Saltelli, 2002), interpretable machine learning (Samek et al., 2021), sensitivity of outputs to input perturbations (Molnar et al., 2020), activation maximization (Simonyan et al., 2013), relevance propagation (Montavon et al., 2018), adversarial robustness (Engstrom et al., 2019), interpretable controls in implicit generative models (Härkönen et al., 2020), or function parametrizations in artificial neural networks (Du et al., 2019; Kornblith et al., 2019).

Hauser & Ray (2017) investigated the Riemannian geometry on the data manifold that is induced by representations learned using artificial neural networks and show, in particular, that the metric tensor can be found by backpropagating the coordinate representations learned by the network. This shares similarities with our approach, as the induced geometry is essentially the pull-back geometry from representation space by the Jacobian map. Meller & Berkouk (2023) proposed a graph representation for neural networks using a singular value decomposition of the weight matrices. While they tackle the nonlinearity by studying linear maps contained in the nonlinear map consecutively, our emphasis lies in the local linear approximation of the nonlinear map.

There exist a few studies that investigate the sensitivity of PH representations to perturbations and their ability to recognize specific features, as we do in this work. For example, Turkeš et al. (2021) study the sensitivity of a number of PH representations to different types of transformations (such as rotation, translation, change of image brightness or contrast, Gaussian, salt and pepper noise). However, the main method to assess this sensitivity is the performance of a support vector machine trained on the original data representations and tested on the representations of data under transformations, and thus requires training and depends on the choice of a particular classifier. Bubenik et al. (2020) showed in theory and experiments that persistence landscapes can be used to detect curvature of an underlying set based on a sampled point cloud. Turkeš et al. (2022) conducted investigations towards identifying fundamental types of tasks for which PH representations might be most useful. They showed that beside curvature and number of holes, PH representations can be used for detecting convexity. However they focus on three specific tasks (detecting number of holes, convexity, and curvature), whereas we study the alignment of the representations with *any* given feature defined on the

[1]Data and code developed in this research are available at `https://anonymous.4open.science/r/persistent-homology-0915`

space of input data sets. Moreover, in that work the performance of PH is experimentally evaluated via the accuracy of an SVM classifier, whereas our approach does not require training of any model.

More generally, in the context of inverse problems in persistence theory, there are several lines of work that study conditions under which persistence diagram maps are surjective or injective, see the survey of Oudot & Solomon (2020), and references therein. Within this context, our work can be seen as providing a framework related to the study of the injectivity of specific PH encodings. The work of Xenopoulos et al. (2022) deals with local explainability using topological representations. In contrast, we deal with the explainability of the representations. McGuire et al. (2023) measured the dissimilarity between representations learned by neural networks trained on PH encodings and networks trained on raw data. They experimentally demonstrated that networks learn considerably different representations when processing PH encodings instead of raw data. Rieck (2023) compared the expressivity of PH against the Weisfeiler-Lehman hierarchy of graph isomorphism tests, and explored the potential of PH to capture certain graph structures and characteristic properties.

2 Preliminaries on Persistent Homology

The key idea behind PH is to construct a filtration, i.e., a sequence of topological spaces by gradually adding simplices (e.g., vertices, edges, triangles) to the data, and to study the evolution of topological features (components, holes, voids, and higher-dimensional voids) across the filtration. In this section, we give a brief overview of the different filtrations that we consider in this work (Section 2.1), and the persistence image that we will use to represent PH information (Section 2.2).

2.1 Filtration

Let $X = \{x_i \in \mathbb{R}^D \mid i = 1, \ldots, N\}$ denote a point cloud consisting of N points in $\mathbb{R}^D$. To build a filtration on X, we commonly construct a sequence of simplicial complexes on X.

A simplicial complex can be thought of as a space obtained by taking a union of vertices, edges, triangles, tetrahedra and higher-dimensional simplices. Formally, a collection $K \subset 2^X$ of subsets of X is called a *simplicial complex* if $\sigma \in K$ and $\tau \subset \sigma$ imply $\tau \in K$. An element σ in a simplicial complex is called a $(|\sigma| - 1)$-simplex, where $|\sigma|$ is the cardinality of σ. Specifically, one can think of 0-simplices as vertices (i.e., elements of X); 1-simplices as edges; 2-simplices as triangles. We note that according to this definition of a simplicial complex, not every element of X is necessarily a 0-simplex; this is important for the types of filtrations that we consider in our work, such as the DTM filtration.

In the following we focus on simplicial complexes obtained as the clique complex of an R-neighborhood graph. The *clique complex of the R-neighborhood graph* on a point cloud X consists of all subsets σ of X such that the distance between any pair of points in σ is at most R:

$$\mathrm{Cl}(X, R) = \{\sigma \in 2^X \mid B_E(x_i, R) \cap B_E(x_j, R) \neq \varnothing, \forall x_i, x_j \in \sigma\},$$

where $B_E(x, R) = \left\{y \in \mathbb{R}^D \mid d_E(x, y) \leq R\right\}$ denotes the Euclidean R-ball centered at x and d_E denotes the Euclidean distance.

A *filtration* with respect to the clique complex is an indexed collection $\{K_r\}_{r \in \mathbb{R}^{\geq 0}}$ of subsets $K_r \subset \mathrm{Cl}(X, R)$ satisfying the condition that $K_{r_1} \subset K_{r_2}$ if $r_1 \leq r_2$. The construction of a filtration is equivalent to assigning a *filtration value* $\phi(\sigma)$ to each simplex σ in $\mathrm{Cl}(X, R)$ in the following sense. Given $\{K_r\}_{r \in \mathbb{R}^{\geq 0}}$, one can define the filtration value for any simplex σ as $\phi(\sigma) = \inf\{r : \sigma \in K_r\}$. Conversely, given a filtration value for every simplex, one can define the collection of subsets as $K_r = \{\sigma \in \mathrm{Cl}(X, R) : \phi(\sigma) \leq r\}$.

We will focus on some common filtrations built upon $\mathrm{Cl}(X, R)$:

1. The Vietoris-Rips filtration (Vietoris, 1927). This defines the filtration value for each simplex as its diameter: $\phi(\sigma) = \mathrm{Diam}(\sigma) = \max_{x,y \in \sigma} d_E(x, y)$.

2. The distance-to-measure (DTM) filtration (Anai et al., 2020). This defines the filtration value for each vertex as $\phi(\{x_i\}) = \frac{1}{K} \sum_{x_j \in \mathrm{KNN}(x_i)} d_E(x_i, x_j)$, where $\mathrm{KNN}(x_i)$ denotes the set of k nearest neighbors of

x_i in X, so that the outliers have a large filtration function value and appear late in the filtration. Then it defines the filtration value for edges as $\phi(\{x_i, x_j\}) = \phi(\{x_i\}) + \phi(\{x_j\}) + d_E(x_i, x_j)/2$, and for simplices with degree greater than one as $\phi(\sigma) = \max_{x_i, x_j \in \sigma}\{\phi(\{x_i, x_j\})\}$.

3. The height filtration with respect to a vector hyperplane. Let $v \in \mathbb{R}^n$ be the unit normal vector of the vector hyperplane. The corresponding height filtration defines the filtration value of vertices as $\phi(\{x_i\}) = \langle x_i, v\rangle$, and the filtration value of any other simplices as $\phi(\sigma) = \max_{x\in\sigma}\{\phi(\{x\})\}$. We note that this means that, in order to capture features of interest, one needs to set the maximum length for an edge to be present in the height filtration (for details, see Appendix F.3).

2.2 Persistence image

The set K_r includes more and more simplices as the parameter r increases. In persistent homology, we are interested in how the number of components, holes, voids and higher-dimensional holes of K_r changes as we vary the parameter r. More precisely, we consider the rank of the homology vector spaces of K_r as a function of r. By the matrix reduction algorithm (Edelsbrunner et al., 2002), one can identify a birth-death pair $(b, d) \in [0, R]^2$ for every non-trivial homology class that appears in the filtration. Roughly, the homology class first "appears" in K_b and "persists" until K_d, degenerating to the trivial class afterwards. The persistence diagram (PD) is a summary of such information.

Formally, the k-dimensional PD is the multiset of birth-death pairs for all k-dimensional homology classes that appear in the filtration. Although the space of PDs can be endowed with a metric structure, PDs do not lend themselves to processing with techniques that require a Hilbert space structure, including support vector machine (SVM) and principal component analysis (PCA) (Reininghaus et al., 2015). Hence, one often considers vector representations of PDs. The most commonly used ones include persistence images (Adams et al., 2017) and persistence landscapes (Bubenik, 2015).

In our work, we will focus on persistence images (PIs). Let PD be a k-dimensional persistence diagram in birth-death coordinates. One converts this to a multiset $\eta(\mathrm{PD})$ in birth-lifespan coordinates by applying the linear map $\eta(b, d) = (b, l)$ with $l = d - b$ to each birth-death pair (b, d). Given a kernel function $g_{(b,l)}(x, y)$ on $\mathbb{R}^2$ and a weighting function $\alpha(b, l)$, the persistent surface is the function $\psi\colon \mathbb{R}^2 \to \mathbb{R}$ defined by

$$\psi(x, y) = \sum_{(b,l)\in\eta(\mathrm{PD})} \alpha(b, l) g_{(b,l)}(x, y).$$

A persistence image (PI) is a finite-dimensional representation of ψ obtained as follows. One splits a subdomain of ψ by a $P \times P$ grid of regions. Then the PI of resolution P is the matrix whose (i, j)-th entry or pixel is the integration value of ψ over the (i, j)-th region.

For constructing PIs, one needs to choose 1) the resolution P, 2) the kernel function $g_{(b,l)}(x, y)$ and its associated parameters and 3) the weighting function $\alpha(b, l)$. One of the main difficulties in working with PIs is that there is no canonical way to choose these hyperparameters (Adams et al., 2017). Adams et al. (2017) studied the effects of PI parameters on the performance of K-medoids classifiers that take PIs as inputs. However, we note this approach heavily depend on the choice of downstream model. This motivates us to investigate what kind of information of the input data is intrinsically captured by the PI under different choices of the hyperparameters.

A motivation for considering PIs is that they provide differentiable PH representations (see Leygonie et al., 2022), and that, with an appropriate choice of metric, the space of PIs has a Euclidean structure, which simplifies computations (see Section 3.1). We will later consider derivatives of the PI encoding with respect to the input data. These can be obtained using existing automatic differentiation packages and libraries, such as topologylayer (Brüel-Gabrielsson et al., 2019) and Gudhi (The GUDHI Project, 2020). We provide further details about this in Appendix C.

3 Methods: Sensitivity of PH Encoding to Data Variations

We consider an encoding map $f : \mathcal{M} \to \mathcal{N}$, where $\mathcal{M}$ is a space of point clouds and $\mathcal{N}$ is the space of persistence images (Section 3.1). We conceptualize input data variations (Section 3.2) and the resulting changes of the encoding output (Section 3.3).

3.1 Input space and output space

We let $\mathcal{M}$ be the space of point clouds in $\mathbb{R}^D$ that contain exactly N points,

$$\mathcal{M} = \{X \subset \mathbb{R}^D \colon |X| = N\}.$$

A point cloud X can be regarded as a probability distribution on $\mathbb{R}^D$. Hence we can equip $\mathcal{M}$ with the 2-Wasserstein distance (see, e.g. Peyré et al., 2017):

$$d_W(X, Y) = \min_{\omega \in \Omega(X,Y)} \left(\sum_{x \in X} d_E^2(x, \omega(x)) \right)^{\frac{1}{2}}.$$

Here $\Omega(X, Y)$ denotes the set of bijections $\omega \colon X \to Y$ between the sets X, Y, and d_E denotes the Euclidean distance on $\mathbb{R}^D$. Based on this, $\mathcal{M}$ can be endowed with a manifold structure with $\dim(\mathcal{M}) \triangleq m = D \times N$. For simplicity of presentation, in the following we treat $\mathcal{M}$ as a Euclidean space. Nonetheless, our discussion is consistent with the manifold structure and also applies to other types of data with an appropriate manifold structure (see Appendix B for details).

We let $\mathcal{N}$ be the space of persistence images of fixed resolution P. Thus we can interpret $\mathcal{N}$ as a submanifold embedded in $\mathbb{R}^{P \times P}$ endowed with the canonical Euclidean distance, with $\dim(\mathcal{N}) \triangleq n = P^2$. Here again, other choices of the metric are possible, such as a Wasserstein distance between persistence images.[2]

3.2 Data variations

To characterize local variations of the input data, we consider tangent vectors on the data manifold. We conceptualize the intuitive concepts of perturbations and feature variations of the input data in terms of corresponding vector fields on the data manifold.

The *tangent space* at $X \in \mathcal{M}$, denoted $T_X\mathcal{M}$, is the vector space of all vectors emanating from X and tangential to the data manifold $\mathcal{M}$. The dimension of $T_X\mathcal{M}$ is equal to the dimension of the data manifold, $\dim(T_X\mathcal{M}) = \dim(\mathcal{M})$. Each *tangent vector* $v \in T_X\mathcal{M}$ characterizes a local variation of a single point X. A *vector field* specifies one type of variation for the population of $\mathcal{M}$. More specifically, a vector field V on $\mathcal{M}$ is a smooth map $V \colon \mathcal{M} \to \sqcup_X T_X\mathcal{M}$, assigning to each X on $\mathcal{M}$ a tangent vector $V(X) \in T_X\mathcal{M}$.

A *perturbation* is a modification of a point in the data manifold, e.g., by rotation or shearing. This can be described by a map $\pi \colon \mathcal{M} \to \mathcal{M}$ taking a data point $X \in \mathcal{M}$ to a perturbed data point $\pi(X) \in \mathcal{M}$. The perturbation vector field V_π associates to each $X \in \mathcal{M}$ a tangent vector $V_\pi(X)$ capturing the difference between $\pi(X)$ and X (see Figure 1, left).

Definition 1 (Perturbation vector field)**.** Let $\mathcal{M}$ be a manifold, $T_X\mathcal{M}$ the tangent space at $X \in \mathcal{M}$, and $\pi \colon \mathcal{M} \to \mathcal{M}$ a perturbation map. The perturbation vector field induced by π is defined as

$$V_\pi : \mathcal{M} \to \sqcup_X T_X\mathcal{M}; \quad X \mapsto V_\pi(X) = \pi(X) - X.$$

A *feature* ρ is a real-valued smooth function defined on the data manifold, $\rho \colon \mathcal{M} \to \mathbb{R}$, assigning a feature value to each data point. Discrete-valued (categorical) features can be converted to continuous ones by

[2]This can be of interest since persistence images have been found to be unstable with respect to the Euclidean distance (Reininghaus et al., 2015, Theorem 3; Adams et al., 2017, Remark 6).

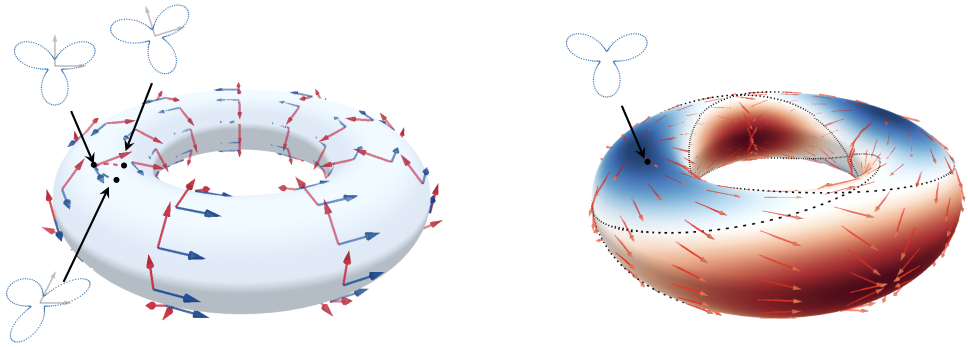

Figure 1: Left: vector fields induced by "rotation" (red arrows) and "shearing" (blue arrows) on the data manifold. Right: gradient vector field induced by a binary feature, where the dashed line is the class boundary, and the continuous feature value represents the probability of the data point belonging to the "red" class.

considering probability distributions or logits of the feature values. For instance, the "cat-or-dog" feature can be converted to a continuous feature $\rho(X) = \text{Prob}(X \text{ is cat}) \in [0, 1]$.

The gradient of a feature introduces a vector field on the data manifold. The gradient vectors point in the direction of steepest increase of the feature, with magnitude indicating the rate (see Figure 1, right).

Definition 2 (Gradient vector field)**.** Let $\mathcal{M}$ be a manifold, $T_X\mathcal{M}$ the tangent space at $X \in \mathcal{M}$, and $\rho\colon \mathcal{M} \to \mathbb{R}$ a one-dimensional feature. The gradient vector field of ρ is the vector field on $\mathcal{M}$ defined as

$$\nabla\rho : \mathcal{M} \to \sqcup_X T_X\mathcal{M}; \quad X \mapsto \nabla\rho(X),$$

such that $\frac{\nabla\rho(X)}{\|\nabla\rho(X)\|} = \text{argmax}_{v\in T_X\mathcal{M}:\|v\|=1} |\frac{\partial}{\partial v}\rho(X)|$ and $\|\nabla\rho(X)\| = \max_{v\in T_X\mathcal{M}:\|v\|=1} |\frac{\partial}{\partial v}\rho(X)|$. Here $\frac{\partial}{\partial v}$ is the directional derivative along v.

Definition 1 and Definition 2 are for the case that $\mathcal{M}$ is a Euclidean space. We provide definitions of perturbation vector field and gradient vector field in the case of general Riemannian manifolds in Appendix B.3. Further, we provide numerical details for estimating such vector fields using finite data sets in Appendix F.3 and F.4.

3.3 Encoding variations

Having characterized data variations in terms of vector fields, the next step is to describe the behavior of the encoding map f in response to these variations. Specifically, we are going to introduce the average pull-back norm of vector fields to quantify the sensitivity of the encoding map to the corresponding data variations. At the outset of this subsection, we emphasize that whether or not it is desirable to have an encoding that is sensitive to a particular data variation depends on the specific practice scenario and whether this variation is perceived as valuable information or as noise that one would like to filter out in the encoding.

Jacobian The *Jacobian* of an encoding map f, denoted by J^f_X, is a linear transformation between tangent spaces that characterizes the local behavior of f. While a tangent vector $v \in T_X\mathcal{M}$ describes one type of data variation at X, the image tangent vector $J^f_X(v)$ describes the resulting variation of the encoding $f(X)$,

$$J^f_X : T_X\mathcal{M} \to T_{f(X)}\mathcal{N}; \quad v \mapsto J^f_X(v).$$

We may write this linear transformation in terms of a Jacobian matrix $J^f_X \in \mathbb{R}^{n\times m}$ with respect to a basis. If there is no risk of confusion, we will omit the super-/subscripts f and X.

The rank of the Jacobian is the dimension of the image of $T_X\mathcal{M}$ under the Jacobian map, $\mathrm{rank}(J_X^f) = \dim(J_X^f(T_X\mathcal{M}))$. It corresponds to the number of degrees of freedom of the data manifold that are captured by the encoding. For instance, $\mathrm{rank}(J_X^f) = \dim(T_X\mathcal{M})$ indicates that f is sensitive to all local data variations, whereas $\mathrm{rank}(J_X^f) = 0$ means that f is invariant under all local variations and thus is constant near X.

Pull-back norm To measure the encoding's effectiveness in capturing a data variation, we introduce the average pull-back norm of a vector field. The *pull-back norm* of a tangent vector $V(X)$ at X is defined as[3]

$$\|V(X)\|_f = \|J_X^f(V(X))\| = \sqrt{V(X)^T \cdot G_X^f \cdot V(X)},$$

where $G_X^f = (J_X^f)^T J_X^f$ is the Gram matrix of the encoding f at X. This measures the sensitivity of f to the variation $V(X)$ at X. To measure the sensitivity across different inputs, we take the average with respect to a distribution on $\mathcal{M}$. In practice, we use the empirical distribution of a given data set $\mathcal{D} = \{X_i\}_{i=1,\ldots,|\mathcal{D}|}$.

Definition 3 (Average pull-back norm)**.** The average pull-back norm of a vector field V with respect to an encoding map f and a data set $\mathcal{D} = \{X_i\}_{i=1,\ldots,|\mathcal{D}|}$ of cardinality $|\mathcal{D}|$ is

$$\|V\|_f = \frac{1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} \|V(X)\|_f.$$

We say that an encoding can detect a data variation characterized by a vector field V if the encoding is sensitive to V, which alludes to the average pull-back norm of V.

Singular value decomposition To gain a more fine-grained insight into the properties of an encoding, we consider the singular value decomposition (SVD) of the Jacobian matrix,

$$J = \tilde{Q}\Lambda Q^T.$$

Here $\tilde{Q} \in \mathbb{R}^{n\times n}, Q \in \mathbb{R}^{m\times m}$ are orthogonal matrices, and $\Lambda \in \mathbb{R}^{n\times m}$ is a diagonal matrix containing in its diagonal the singular values in decreasing order $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{\min(m,n)}$. This sequence of ordered singular value is the *spectrum* of the Jacobian J. Accordingly, the Gram matrix has eigendecomposition $G = J^T J = Q\Lambda^2 Q^T$. We denote the right singular vectors, i.e., the columns of Q, by $q_1, \ldots, q_m$. We will refer to these q_i's as the eigenvectors of the encoding. Any tangent vector $v \in T_X\mathcal{M}$ can be written as $v = \sum \langle v, q_i\rangle q_i$, and its pull-back norm as $\|v\|_f = \sqrt{\sum \lambda_i^2 \langle v, q_i\rangle^2}$. With this, the pull-back norm is decomposed as two parts: the spectrum of Jacobian and the alignment between v and eigenvectors, which is described by the inner product. In particular, the pull-back norm is large if v is aligned with q_i's that have large singular values. This also implies that the eigenvectors with top largest eigenvalues can be regarded as the data variations that the encoding considers most "important". We offer a visualization in Appendix D.

Comparison between encodings Later we will compare different encodings by examining their sensitivity to specific data variations. To place different encodings on the same scaling level, we consider the normalized average pull-back norm $[\sum \|V(X)\|_f/\lambda_1^f]/|\mathcal{D}|$. We provide details about this normalization technique in Appendix F.2. Another way of comparing different encodings is via the Bures-Wasserstein distance (Bhatia et al., 2019) between their Gram matrices. The Bures-Wasserstein distance quantifies the alignment between the eigendecompositions of the Gram matrices. For two positive definite matrices A and B, the Bures-Wasserstein distance is computed as:

$$d_{\mathrm{BW}}(A, B) = \left[\mathrm{Tr}A + \mathrm{Tr}B - 2\mathrm{Tr}(A^{1/2}BA^{1/2})^{1/2}\right]^{1/2}.$$

For matrices that are not strictly positive definite we use the same definition after adding a small multiple of the identity matrix.

[3]Strictly speaking this is a semi-norm, as it may vanish for non-zero tangent vectors.

4 Identifying What Is Recognized

In this section, we seek to identify, for fixed PH encodings, which data variations are recognized and which are ignored. Specifically, we investigate the total amount of data variations that are captured by PH encodings (Section 4.1), and among those captured variations we interpret the most significant ones (Section 4.2). Then, we quantify the "importance" for any data variation (Section 4.3), and measure the dissimilarity of the captured information across different PH encodings (Section 4.4). It is important to notice that, even though our approaches eliminate the potential inference bias induced by downstream models, our results still significantly depend on the choice of the data set. Therefore, we emphasize that this section is dedicated to investigating "what is recognized" by PH encodings *within specific datasets*, rather than obtaining universal knowledge about a PH encoding.

Synthetic data Throughout this section we consider a synthetic data set of point clouds in $\mathbb{R}^2$ sampled from curves in the Radial Frequency Pattern (RFP) family. The curve $\mathrm{RFP}_{(a,w)}$ is parametrized by $\rho(\theta) = 1 + a\cos(w\theta), \theta \in (0, 2\pi]$. Loosely speaking it represents the shape of a flower with w petals of size characterized by a. We take w in $\{3, 4, \ldots, 8\}$ and 10 values of a evenly distributed on the interval $[0.5, 0.9]$. For each curve $\mathrm{RFP}_{(a,w)}$, we evenly sample $N = 150$ points to obtain a point cloud, which is then scaled to the unit square $[0, 1]^2$ (see examples in Appendix F.3). Notably, each curve in the RFP family has the same topology. This allows us to validate the ability of PH to capture information beyond topology. The RFP data set has also been used in studying the importance of specific shape features in shape recognition and object perception (Schmidtmann et al., 2015).

PH encodings We investigate 3 PH encodings: Vietoris-Rips (Rips), DTM, and Height (with respect to the vector hyperplane with normal vector $[1, 0]^T$). For each encoding method we build the corresponding filtration for point cloud data, extract the 1-dimensional PDs, and convert them to PIs with the same PI parameters. For reference we also include a PointNet encoding. PointNet (Qi et al., 2017) is a deep neural network architecture designed for processing point clouds directly. We train the network to predict the number of petals w, achieving a test accuracy of 100%. We take the output of the second-to-last layer of the trained PointNet as the output of the PointNet encoding. More details concerning the filtration and PI parameters, and the PointNet encoding are provided in Appendix F.3.

4.1 Spectrum of the Jacobian

As explained in Section 3.3, the rank of the Jacobian indicates the maximum amount of information, in a dimension sense, that can be captured by the encoding. The spectrum, i.e., the sequence of ordered singular values $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$, provides a fuller picture, indicating to what extent different eigenvectors are highlighted.

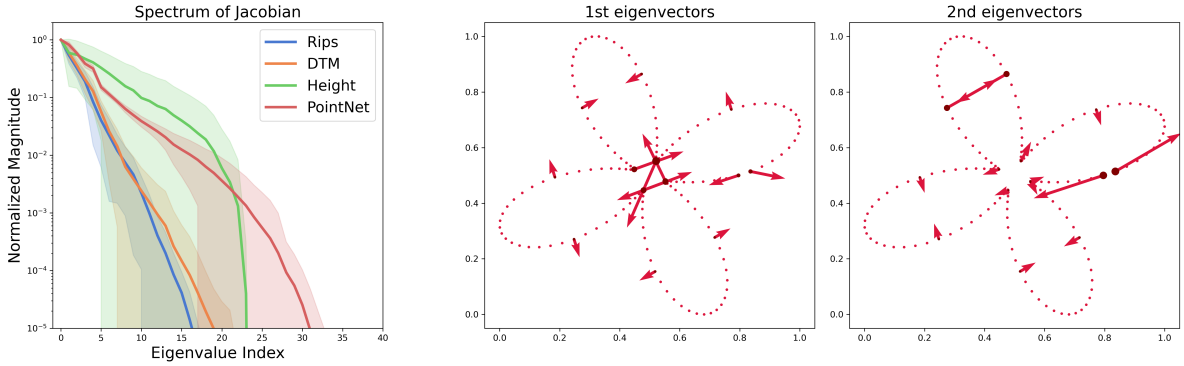

Figure 2: Left: The normalized spectrum of the Jacobian for different encodings. Shown is the mean and standard error of the ordered normalized singular values over different input point clouds. Right: The top two eigenvectors of the Jacobian for the Rips encoding at a particular input point cloud.

The left plot in Figure 2 reports the normalized spectrum of the Jacobian for different encodings, which is the sequence of ordered normalized singular values

$$1 \geq \frac{\lambda_2}{\lambda_1} \geq \cdots \geq \frac{\lambda_{\mathrm{rank}(J)}}{\lambda_1} > 0.$$

The first observation is that the rank of the Jacobian is much smaller than the dimension of the point cloud space and the PI space. Indeed, note that the dimension of $\mathcal{M}$ is $m = D \times N = 2 \times 150 = 300$, and the dimension of $\mathcal{N}$ is $n = P \times P = 20 \times 20 = 400$. On the other hand, the normalized singular values decay to 10^{-5} before index 40. We conclude that the four encodings under consideration capture only a small set of variations in the input data and discard many others. Secondly, while Rips, DTM, and Height all have a similar number of singular values larger than 10^{-5}, Rips and DTM exhibit a sharper initial decay than Height. This implies that Height has a larger effective rank[4], while Rips and DTM concentrate their attention more specifically on a few variations. This difference in decay rates may stem from the fact that the size of holes, which is captured by Rips and DTM encodings, is influenced by fewer variations compared to the position of holes, which is information retained by the Height encoding.

The middle and right plots in Figure 2, show the top two eigenvectors for the Rips encoding at an example point cloud X. An eigenvector corresponds to a list of vectors attached to individual points in the point cloud. These eigenvectors provide insight into the nature of the "important" data variations. More technically, these variations correspond to the most effective way to change the birth/death parameters of certain homology classes.[5] For instance, in the middle plot of Figure 2, the vectors on the petals depict variations that narrow/broaden the petals, which in turn change the death parameter of the corresponding homology classes. The eigenvectors can also be used to obtain point saliency maps, which we discuss in Appendix E. However, we observe that the eigenvectors do not necessarily have an obvious intuitive description. Hence, interpretations are needed to bridge the gap between abstract eigenvectors and human-understandable concepts.

4.2 Alignment between eigenvectors and perturbation tangent vectors

To interpret the eigenvectors of the encoding, we consider their alignment with different perturbation vector fields.

Perturbations on the data manifold We consider eight types of perturbations applied to the input data, illustrated in Figure 3. The first two, *rotation* and *translation*, are Euclidean motions, i.e., transformations that preserve the Euclidean distance. They are used to test the fact that pointwise distance-based encodings, namely Rips and DTM, should remain invariant under such variations. The *dilation*, *strecth_x*, and *shearing* variations serve to test the sensitivity of the encoding to invertible linear transformations of the data. The next two variations are used to test the robustness of PH encoding against noise: the *noising* variation adds coordinate-wise Gaussian noise at each point in the point cloud; the *wiggly* variation adds a sine-type noise at every point in the point cloud along the normal direction. Lastly, the *convex* variation transforms the point cloud towards the boundary of its convex hull through a linear interpolation. We present a visualization of the effects of *shearing* and *convex* on the PH associated with Rips filtration and Height filtration in Appendix F.3.

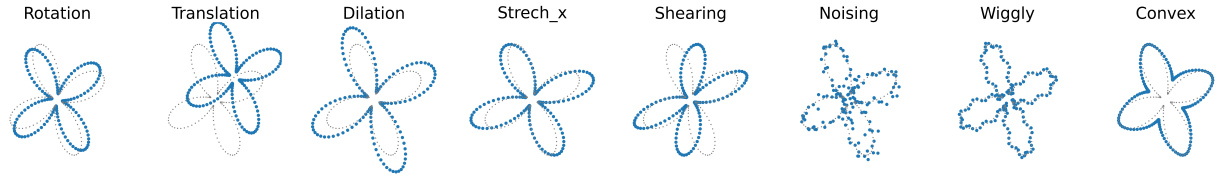

Figure 3: Eight types of perturbations on a RFP pointcloud.

[4]The effective rank is the number of singular values that have a similar order of magnitude as the top singular value.
[5]This bears some resemblance to adversarial perturbations considered in neural networks.

We examine the angles between the top eigenvectors of the different encodings and different perturbation tangent vectors. In Figure 4, we record the average inner product between the perturbation tangent vectors and the top four eigenvectors of each encoding method:

$$\frac{1}{\mid \mathcal{D}_{\mathrm{RFP}} \mid} \sum_{X \in \mathcal{D}_{\mathrm{RFP}}} \left| \left\langle \frac{V_\pi(X)}{\|V_\pi(X)\|}, q_i^f \right\rangle \right|, \quad i = 1, 2, 3, 4.$$

The eigenvectors q_i^f depend on X. Since each encoding map f induces a different orthonormal basis $\{q_1^f, q_2^f, \ldots, q_m^f\}$ on the tangent space of the data manifold, a fixed tangent vector will have different coordinates with respect to the different encodings. Figure 4 serves as a sort of dictionary, showing how each perturbation (rotation, translation, dilation, etc.) is expressed in the language of each encoding.

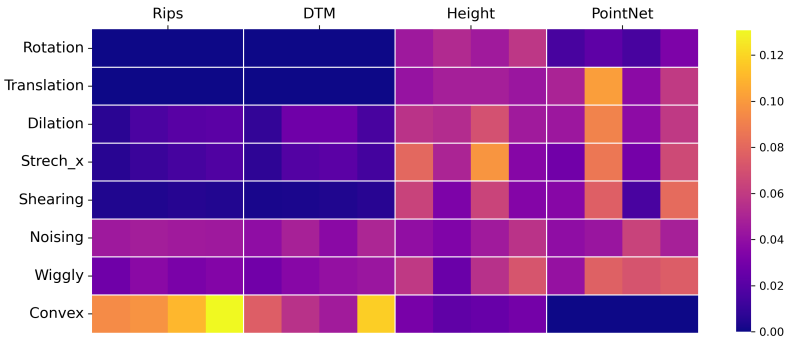

Figure 4: Absolute inner product between perturbation vectors and top four eigenvectors of different encodings. A higher value implies greater alignment, i.e., greater sensitivity to a perturbation.

For Rips and DTM encodings, we find that the top eigenvectors exhibit a relatively strong alignment with *convex*. The corresponding average inner product is around 0.1. This indicates that the most "important" data variation for Rips and DTM are closely related to convexity. This is consistent with the fact that these encodings capture geometric properties, such as birth values of holes in the filtration (that increase under the *convex* perturbation, see Figure 3). At the same time, the top eigenvectors of Rips and DTM are orthogonal to *rotation* and *translation*. This indicates that Euclidean motion is not as relevant in the Rips and DTM encodings, as is to be expected from the definitions of these encodings.

For the Height encoding, the top eigenvectors have a significant alignment with Euclidean motions and *stretch_x*. This makes sense, since the Height encoding is designed to collect information on the position of holes. On the other hand, we do not observe a strong alignment between top eigenvectors of Height encoding and *convex*. This might seem to be in contradiction with Turkeš et al. (2022), who demonstrated that PH on height filtration can be used for detecting convexity. Note, however, that they use 0-dimensional PH on cubical complexes (or analogously, on Rips complexes on geodesic distances), which recognizes concave shapes by their multiple connected components (0-dimensional cycles) for at least some height filtration directions. 0-dimensional PH with respect to such a filtration would see two connected components (two petals) for a while, that would then merge into one at some point. This would happen earlier under *convex* perturbation (i.e., one of the connected components would die sooner), so that the alignment can be expected to be more significant in that case.

For the PointNet encoding, the top eigenvector has a relatively strong alignment with *translation*. This is consistent with a previous observation by Turkeš et al. (2022) where the PointNet did not perform well in classification tasks when the test data was corrupted by translations. Note akin to the approach adopted by Turkeš et al. (2022), we do not use data augmentation techniques during the training for PointNet. This might lead to the sensitivity of the trained PointNet to *translation*. On the other side, PointNet is robust under *convex* perturbations. This can be attributed to the nature of RPF data set. Recall the RPF data set comprises point clouds defined by two independent parameters, a and w, which characterize the size and

the number of the petals, respectively. While the PointNet is trained to identify the number of petals, it can easily learn from the data set to ignore the size of petals. Notice increasing the petal size bears strong resemblance to *convex* perturbation (see examples of RPF point clouds in Appendix F.3). Therefore, one can loosely infer that the RPF data set is "inherently" augmented by *convex* perturbation, and consequently the trained PointNet might learn from the data to ignore *convex* information.

4.3 Pull-back norm of perturbation vector fields

In some scenarios, one is interested in the sensitivity of an encoding to certain types of perturbations (Ren et al., 2020). In Figure 5 (left) we evaluate the average pull-back norm of different perturbation vector fields with respect to different encodings. The pull-back norm takes into account not only the alignment with the encoding eigenvectors but also the magnitude of the corresponding singular values.

For DTM and Rips, we find that noising, wiggly, and convex, have a significantly larger pull-back norm than the other data variations. This is consistent with the Jacobian spectrum in Figure 2 (which indicates DTM and Rips has faster-decaying spectrum and therefore captures only few data variations), and the alignment information in Figure 4 (which indicates alignment of the top eigenvectors with these particular variations). The Height encoding has a relatively large pull-back norm for many of the considered data variations, including dilation, stretch, and shearing. Also the pull-back norm associated with *convex* perturbations with respect to Height encoding exhibits a moderate average value and a large variance. This implies Height encoding is sensitive to *convex* perturbations in certain point clouds, while being less sensitive in others. This aligns with the alignment information in Figure 4, which indicates the most "important" data variations seen by Height encoding, on average, are not closely related with *convex*. Similar to Height encoding, PointNet also leads to relatively large pull-back norms, but with a different profile and with exception of convex, which has a small pull-back norm under PointNet. Rips and DTM encodings have a faster-decaying Jacobian spectrum than Height and PointNet, indicating that they are sensitive to fewer data variations.

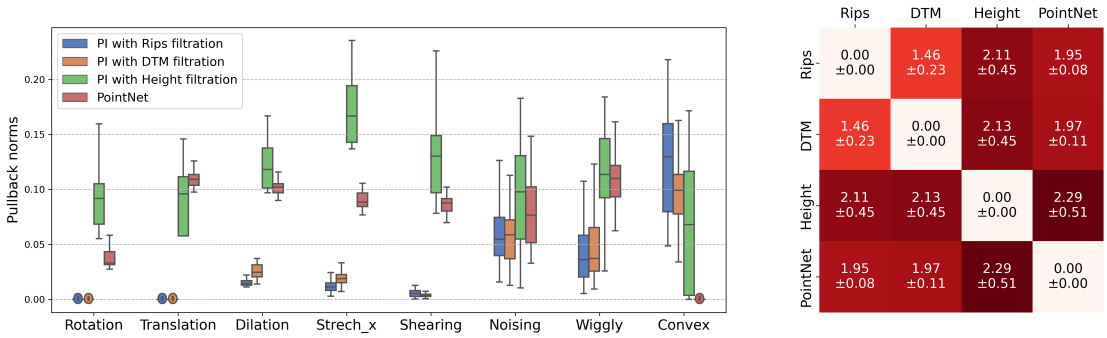

Figure 5: Left: Average pull-back norm of different perturbation vector fields with respect different encodings. Right: Bures-Wasserstein distance between Gram matrices $J^T J$ of different encodings.

4.4 Distance between Gram matrices

Along this thread, we can also investigate the relationship between encodings. The right panel of Figure 5 shows the average Bures-Wasserstein distance between the Gram matrices of different encodings.

The average distance matrix shown in the right part of Figure 5 indicates that all encodings are different, whereby some are more similar and some are more dissimilar (see also alignment pattern in Figure 4). Rips and DTM are closest each other, while PointNet and Height both differ significantly from Rips and DTM. This indicates that Rips and DTM capture similar information which is different from the information that is captured by Height and PointNet. This makes sense, since the DTM filtration function is the average distance to neighbors, which approximates the distance function that underlies the Rips filtration; moreover, the data we consider does not contain outliers. We also find that although Height and PointNet give relatively similar pull-back norms for rotation, translation, and dilation, overall these two encodings are very different.

5 Selecting Hyperparameters

In this section, we shift our focus to the problem: how do we select the hyperparameters of the encoding in order to detect a data feature of interest. We investigate the impact of hyperparameters on the rank and spectrum of Jacobian (Section 5.1) and on the pull-back norm of gradient vector fields of the interested data feature (Section 5.2). Finally, we demonstrate there is a strong correlation between the pull-back norms of gradient vector fields and the downstream task performance, where the task objective is to predict that data feature (Section 5.3).

Real-world data In this section we utilize the brain artery tree data (Bendich et al., 2016). This data set comprises 96 artery trees in $\mathbb{R}^3$ (see Figure 6, left). These artery trees are obtained by applying a tube-tracking algorithm to Magnetic Resonance Angiography (MRA) images from 96 human subjects. We randomly subsample three point clouds from the vertices of each artery tree, with each point cloud containing $N = 500$ points. Then we normalize the sampled point clouds to the unit cube $[0, 1]^3$.

PH encoding We focus on the Vietoris-Rips encoding and 1-dimensional PDs. We investigate two hyperparameters involved in the construction of PIs: 1) the resolution P, and 2) the variance γ^2 of the Gaussian kernel (see Figure 6, right). We set the baseline PI parameters as $P = 20$, $\gamma^2 = 10^{-4}$, and consider a linear weighting function $\alpha(b, l) = \frac{l}{\max\{l\}}$ (Adams et al., 2017).

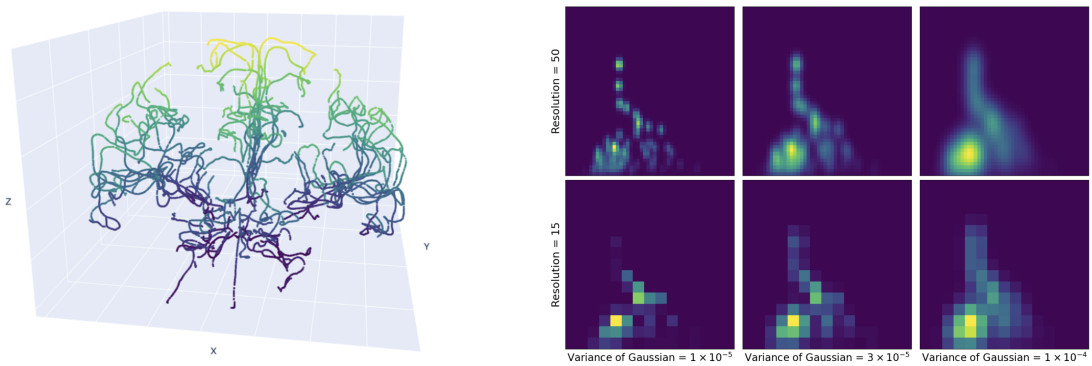

Figure 6: Left: point cloud sampled from a brain artery tree, where the color represents the z-coordinate. Right: the corresponding PI representation with different parameter settings.

5.1 Spectrum of the Jacobian

We begin by analyzing the effects of the resolution P and variance γ^2 of the Gaussian kernel on the spectrum of the Jacobian. In each plot of Figure 7, we varied one parameter while keeping the other fixed at the baseline setting, and present the normalized spectrum. We again observe a low-rank phenomenon, since the average rank is always below 160, while in the baseline setting the dimension of the point cloud and PI spaces are respectively $m = N \times D = 500 \times 3 = 1500$ and $n = P \times P = 20 \times 20 = 400$.

In the left part of Figure 7, we see that the spectrum decays slower as the resolution increases, indicating an increase in rank. This implies that, as one would expect, higher resolution allows the PI to capture more information.

Interestingly, we observe that, as the variance γ^2 increases, the rank of the Jacobian initially increases and then decreases. For fixed resolution, very small variance results in sparse PIs (see the first column in the right panel of Figure 6), where multiple PD points (b, l) may fall into one pixel and can only highlight that pixel; very large variance leads to blurred PIs (see the third column in the right panel of Figure 6), where it also becomes difficult to distinguish between PD points.

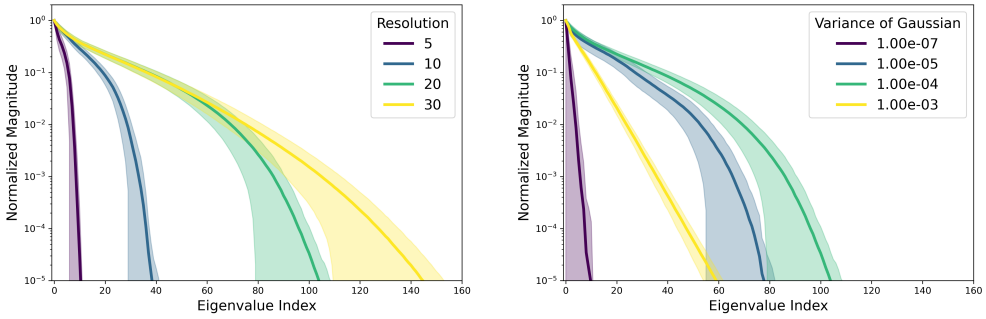

Figure 7: Spectrum of the Jacobian matrix depending on the PI parameters. Left: The effect of changing the resolution P of PI. Right: The effect of changing the variance γ^2 of Gaussian kernel in PI.

5.2 Pull-back norm of feature gradient vector fields

We now explore the effects of the resolution and variance on the pull-back norm of gradient vector fields of the following data feature.

Feature Each point cloud is labeled with a binary *sex* feature, based on the corresponding human subjects' medical information.

Our goal is to locate the optimal parameters of the PI encoding to effectively detect the *sex* feature in the brain artery point clouds. We consider the domain $(P, \gamma^2) \in [15, 30] \times [10^{-5}, 10^{-4}]$. Here the ranges for P and γ^2 are selected based on the values where the spectra in Figure 7 exhibit the slowest decay.

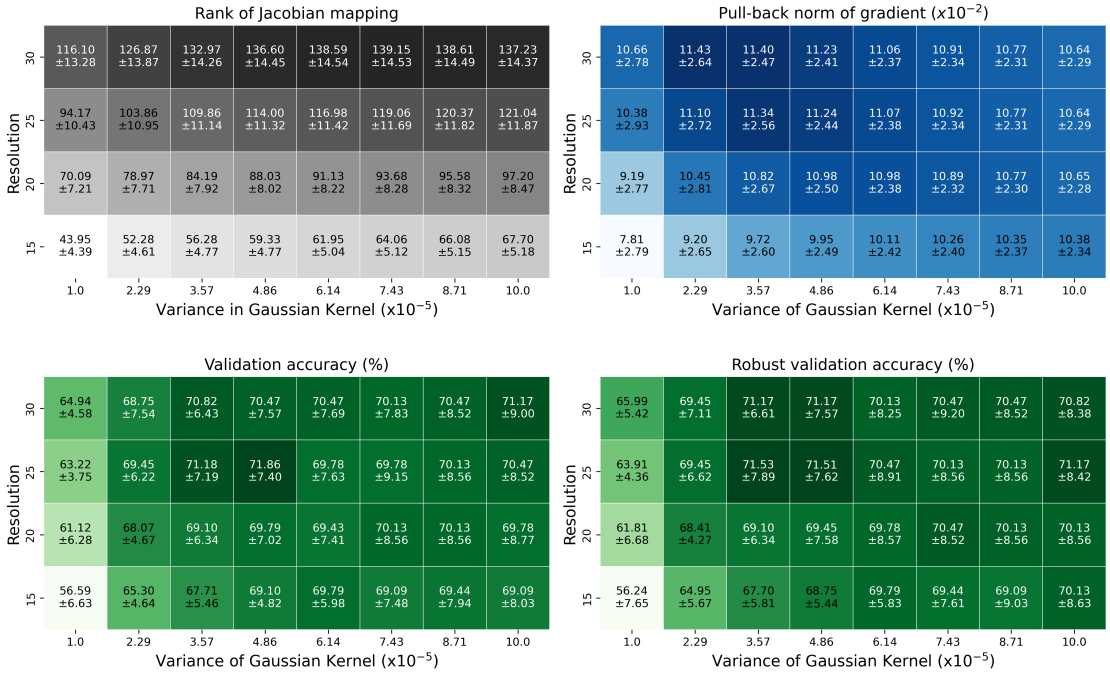

Figure 8: For the brain artery data set, shown is the effect of the resolution (vertical axis) and variance of the Gaussian kernel (horizontal axis) of the PI on the average rank of the Jacobian (upper left), average pull-back norm of the gradient vector field of the *sex* feature (upper right), as well as the test accuracy (lower left) and robust test accuracy (lower right) of the logistic regression model predicting *sex* based on the PI.

In the upper right plot of Figure 8, we present the average pull-back norm of the gradient field of the *sex* feature under different PI parameter choices. The gradient fields are estimated via numerical methods detailed in Appendix F.4. We observe that the pull-back norm generally increases as the resolution P increases, whereas the pull-back norm is not monotonic with γ^2. Moreover, the optimal value for γ^2 varies depending on the choice of resolution. Also, comparison with the upper left part of Figure 8 reveals that the maximum pull-back norm is not necessarily attained for parameters where the rank of the Jacobian is maximal, i.e., when PIs capture the most information about the point cloud.

5.3 Correlation with downstream task performance

We investigate the hypothesis that a high pull-back norm correlates with the performance of a predictor trained on the encoding. To this end we feed PIs generated with different choices of the parameters into logistic regression models and train these to predict the *sex* feature. Here we use logistic regression as the downstream model because of its simplicity, with the intention to minimize the impact of model complexity and training techniques on the task performance. We provide results for convolutional neural networks (CNN) in Appendix F.4.

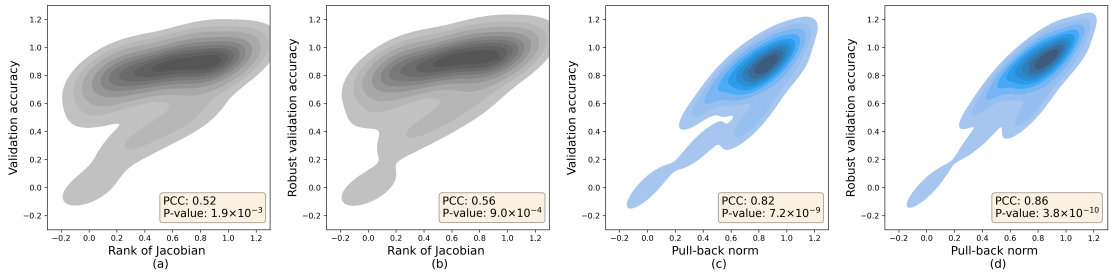

Figure 9: Gaussian kernel density estimation of the joint distribution of four pairs of variables: (a) Jacobian rank vs. validation accuracy; (b) Jacobian rank vs. robust validation accuracy; (c) pull-back norms vs. validation accuracy; and (d) pull-back norms vs. robust validation accuracy, where the downstream models are chosen as logistic regression models.

We evaluate the performance in terms of validation accuracy and robust validation accuracy[6] using cross-validation, which are presented in the lower left and lower right plots in Figure 8. The validation accuracy and robust validation accuracy exhibit a similar pattern to the pull-back norm. Notably, all three quantities reach their maximum at around $P = 30$ and $\gamma^2 = 3.57 \times 10^{-5}$, and their minimum at the lower-left corner.

For a more quantitative comparison, Figure 9 shows a kernel density estimate of the joint distribution of four pairs of variables: Jacobian rank vs. validation accuracy, Jacobian rank vs. robust validation accuracy, pull-back norm vs. validation accuracy, and pull-back norm vs. robust validation accuracy. The plots clearly indicate a strong correlation between the pull-back norm and the performance on the downstream task. The Pearson's correlation coefficient (PCC) between the four pairs of variables, along with the p-value for a two-sided test, are presented in the lower right corner of each plot in Figure 9.

We conclude that for the considered task, there is a significant correlation between the pull-back norm and the downstream task performance. It is also interesting to interpret these results together with the rank of the Jacobian (upper left in Figure 8). The results demonstrate that the improvement in downstream performance is only somewhat correlated with including more information, but it is strongly correlated with including the most relevant information, which is precisely quantified by the pull-back norm. Therefore, we suggest that the proposed framework can be used to select appropriate PH encodings in practice. Note that the procedure is independent of the downstream model architectures and training techniques.

[6]Robust validation accuracy evaluates the accuracy on a test data set subject to additive Gaussian noise on the inputs.

6 Conclusions

The methods and observations presented in this work contribute to addressing some of the main bottlenecks in the practical application of PH, namely how to interpret PH data representations, how to identify which aspects of the data are captured by these representations, and how to effectively select the parameters of PH encodings in order to obtain data representations that are suitable for solving a particular task.

We presented ways to analyze the most relevant features on the data manifold that are captured by persistence images with different choices of the filtration and compared the results with neural-network-based encodings. For example, we found that while a pretrained PointNet had a relatively high alignment with translation and dilation, the 1-dimensional persistence image encoding with Height filtration had a high alignment with stretch, and the 1-dimensional persistence image encoding with Rips filtration had a high alignment with a data variation that makes the point clouds more convex. At the same time we observe that the response of the encodings to these interpretable input data variations is less than 10% as strong as for other more abstract data variations captured by the singular vectors of the Jacobian; for instance, the maximal value taken by inner products between unit tangent vectors representing interpretable data variations and unit singular vectors is less than 0.1.

We demonstrated on a real-world data set that feature alignment as measured by the Jacobian permits PH parameter tuning without the need to train a classifier on top of the data representation in order to select the parameters based on the test accuracy. Rather, one can select the parameters based on the pull-back norm of the features of interest, and perform training using the data representation with the highest pull-back norm.

Limitations and future work Our analysis is based on the structure of the Jacobian of the data encoding, which by nature focuses only on local variations of the input data. In future it will be interesting to further advance these methods in regard to non-local data variations, where synthetic notions of derivatives such as our empirical evaluation of the vector fields, and ideas such as the application of the iterated closest point method could serve as a point of departure. The analysis of non-linear transformations via Gram matrices has seen a number of recent advances in the context of artificial neural networks. It will be interesting to explore possible synergies between those investigations and PH data encodings. In our work we focus on studying persistence images, however we note that the methods can be applied to any PH representations that are differentiable, provided that the space of such representations can be embedded into a Riemannian manifold.

References

Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, 18(1):218–252, 2017.

Hirokazu Anai, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hiroya Inakoshi, Raphaël Tinarrage, and Yuhei Umeda. DTM-based filtrations. In *Topological Data Analysis*, pp. 33–66. Springer, 2020.

Rushil Anirudh, Vinay Venkataraman, Karthikeyan Natesan Ramamurthy, and Pavan Turaga. A Riemannian framework for statistical analysis of topological persistence diagrams. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 68–76, 2016.

Paul Bendich, James S Marron, Ezra Miller, Alex Pieloch, and Sean Skwerer. Persistent homology analysis of brain artery trees. *Annals of Applied Statistics*, 10(1):198, 2016.

Paul J Besl and Neil D McKay. A method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pp. 586–606. Spie, 1992.

R. Bhatia, T. Jain, and Y. Lim. On the Bures–Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37(2):165–191, 2019.

Andrew J Blumberg and Michael Lesnick. Stability of 2-parameter persistent homology. *Foundations of Computational Mathematics*, pp. 1–43, 2022.

Rickard Brüel-Gabrielsson, Bradley J Nelson, Anjan Dwaraknath, Primoz Skraba, Leonidas J Guibas, and Gunnar Carlsson. A topology layer for machine learning. *arXiv preprint arXiv:1905.12200*, 2019.

Peter Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015.

Peter Bubenik, Michael Hull, Dhruv Patel, and Benjamin Whittle. Persistent homology detects curvature. *Inverse Problems*, 36(2):025008, 2020.

Zixuan Cang and Guo-Wei Wei. TopologyNet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions. *PLOS Computational Biology*, 13(7):e1005690, 2017.

Zixuan Cang, Lin Mu, and Guo-Wei Wei. Representability of algebraic topology for biomolecules in machine learning based scoring and virtual screening. *PLOS Computational Biology*, 14(1):e1005929, 2018.

Gunnar E. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46:255–308, 2009. URL `https://api.semanticscholar.org/CorpusID:1472609`.

Mathieu Carriere, Frederic Chazal, Yuichi Ike, Theo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2786–2796. PMLR, 2020. URL `https://proceedings.mlr.press/v108/carriere20a.html`.

Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

Anne Collins, Afra Zomorodian, Gunnar Carlsson, and Leonidas J Guibas. A barcode shape descriptor for curve point cloud data. *Computers & Graphics*, 28(6):881–894, 2004.

Thibault de Surrel, Felix Hensel, Mathieu Carrière, Théo Lacombe, Yuichi Ike, Hiroaki Kurihara, Marc Glisse, and Frédéric Chazal. Ripsnet: A general architecture for fast and robust estimation of the persistent homology of point clouds. In *Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022*, volume 196 of *Proceedings of Machine Learning Research*, pp. 96–106. PMLR, 2022. URL `https://proceedings.mlr.press/v196/surrel22a.html`.

Manfredo Perdigao Do Carmo and J Flaherty Francis. *Riemannian geometry*, volume 6. Springer, 1992.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1675–1685. PMLR, 2019. URL `https://proceedings.mlr.press/v97/du19c.html`.

Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & computational geometry*, 28:511–533, 2002.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1802–1811. PMLR, 2019. URL `https://proceedings.mlr.press/v97/engstrom19a.html`.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL `http://jmlr.org/papers/v22/20-451.html`.

Mohammad Haft-Javaherian, Martin Villiger, Chris B. Schaffer, Nozomi Nishimura, Polina Golland, and Brett E. Bouma. A topological encoding convolutional neural network for segmentation of 3d multiphoton images of brain vasculature using persistent homology. In *CVPR Workshops*, pp. 4262–4271. Computer Vision Foundation / IEEE, 2020.

Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9841–9850. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6fe43269967adbb64ec6149852b5cc3e-Paper.pdf.

Michael Hauser and Asok Ray. Principles of riemannian geometry in neural networks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/0ebcc77dc72360d0eb8e9504c78d38bd-Paper.pdf.

Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, 2021. URL https://www.frontiersin.org/articles/10.3389/frai.2021.681108.

Christoph D. Hofer, Roland Kwitt, and Marc Niethammer. Learning representations of persistence barcodes. *Journal of Machine Learning Research*, 20(126):1–45, 2019. URL http://jmlr.org/papers/v20/18-358.html.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

Yongjin Lee, Senja D Barthel, Paweł Dłotko, S Mohamad Moosavi, Kathryn Hess, and Berend Smit. Quantifying similarity of pore-geometry in nanoporous materials. *Nature Communications*, 8(1):1–8, 2017.

J. Leygonie, S. Oudot, and U. Tillmann. A framework for differential calculus on persistence barcodes. *Foundations of Computational Mathematics*, 22:1069–1131, 2022. doi: https://doi.org/10.1007/s10208-021-09522-y.

Sarah McGuire, Shane Jackson, Tegan Emerson, and Henry Kvinge. Do neural networks trained with topological features learn different internal representations? In *NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, pp. 122–136. PMLR, 2023.

Dan Meller and Nicolas Berkouk. Singular value representation: A new graph perspective on neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 3353–3369. PMLR, 2023.

Christoph Molnar, Giuseppe Casalicchio, and B. Bischl. Interpretable machine learning - a brief history, state-of-the-art and challenges. In *PKDD/ECML Workshops*, 2020.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018. URL https://www.sciencedirect.com/science/article/pii/S1051200417302385.

Guido Montúfar, Nina Otter, and Yu Guang Wang. Can neural networks learn persistent homology features? In *TDA & Beyond*, 2020. URL https://openreview.net/forum?id=pqpXM1Wjsxe.

Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. Topology of deep neural networks. *The Journal of Machine Learning Research*, 21(184):1–40, 2020.

Steve Oudot and Elchanan Solomon. Inverse problems in topological persistence. In Nils A. Baas, Gunnar E. Carlsson, Gereon Quick, Markus Szymik, and Marius Thaule (eds.), *Topological Data Analysis*, pp. 405–433, Cham, 2020. Springer International Publishing.

S.Y. Oudot. *Persistence Theory: From Quiver Representations to Data Analysis*. Mathematical Surveys and Monographs. American Mathematical Society, 2015. ISBN 9781470425456. URL https://books.google.fr/books?id=if8dCwAAQBAJ.

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*, 12:2825–2830, 2011.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Center for Research in Economics and Statistics Working Papers*, (2017-86), 2017.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.

Karthikeyan Natesan Ramamurthy, Kush Varshney, and Krishnan Mody. Topological data analysis of decision boundaries with application to model selection. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5351–5360. PMLR, 2019. URL `https://proceedings.mlr.press/v97/ramamurthy19a.html`.

Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4741–4748, 2015.

Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.

Bastian Rieck. On the expressivity of persistent homology in graph learning. *arXiv preprint arXiv:2302.09826*, 2023.

Andrea Saltelli. Sensitivity analysis for importance assessment. *Risk Analysis*, 22(3):579–590, 2002. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/0272-4332.00040`.

Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J. Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109:247–278, 2021.

Gunnar Schmidtmann, Ben J Jennings, and Frederick AA Kingdom. Shape recognition: convexities, concavities and things in between. *Scientific Reports*, 5(1):1–11, 2015.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. URL `http://dblp.uni-trier.de/db/journals/corr/corr1312.html#SimonyanVZ13`.

Yashbir Singh, Colleen M. Farrelly, Quincy A. Hathaway, Tim Leiner, Jaidip Jagtap, Gunnar E. Carlsson, and Bradley J. Erickson. Topological data analysis in medical imaging: current state of the art. *Insights into Imaging*, 14(1):58, 2023. URL `https://doi.org/10.1186/s13244-023-01413-w`.

The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.3.0 edition, 2020. URL `https://gudhi.inria.fr/doc/3.3.0/`.

Loring W Tu. Manifolds. In *An Introduction to Manifolds*, pp. 47–83. Springer, 2011.

Renata Turkeš, Jannes Nys, Tim Verdonck, and Steven Latré. Noise robustness of persistent homology on greyscale images, across filtrations and signatures. *PLOS ONE*, 16(9):e0257215, 2021.

Renata Turkeš, Guido Montúfar, and Nina Otter. On the effectiveness of persistent homology. In *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=DRjUkfExCix`.

Leopold Vietoris. Über den höheren zusammenhang kompakter räume und eine klasse von zusammenhangstreuen abbildungen. *Mathematische Annalen*, 97(1):454–472, 1927.

Menglun Wang, Zixuan Cang, and Guo-Wei Wei. A topology-based network tree for the prediction of protein–protein binding affinity changes following mutation. *Nature Machine Intelligence*, 2(2):116–123, 2020.

Peter Xenopoulos, Gromit Chan, Harish Doraiswamy, Luis Gustavo Nonato, Brian Barr, and Claudio Silva. Topological representations of local explanations. *arXiv preprint arXiv:2201.02155*, 2022.

Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. Pointcloud saliency maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1598–1606, 2019.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

Appendix

The appendix is organized into the following sections.

- Appendix A: Notation
- Appendix B: Riemannian manifold structure of the space of point clouds
- Appendix C: Differentiability of the PI encoding
- Appendix D: Visualizing the Jacobian of the encoding over the data manifold
- Apeendix E: Point saliency maps for PH encodings
- Appendix F: Details on the experiments

A Notation

Table 1 provides a summary of the notation.

B Riemannian manifold structure of the space of point clouds

Let $\mathcal{M}$ denote the collection of all point clouds in $\mathbb{R}^D$ that contain exactly N points,

$$\mathcal{M} = \{X \subset \mathbb{R}^D \colon |X| = N\}.$$

Recall that the 2-Wasserstein distance d_W between two point clouds of the same size in $\mathbb{R}^D$ is defined as

$$d_W(X,Y) = \min_{\omega \in \Omega(X,Y)} \left(\sum_{x \in X} d_E^2(x, \omega(x)) \right)^{\frac{1}{2}}$$

where ω is an bijection between X and Y, $\Omega(X,Y)$ contains all bijections, and d_E denotes the Euclidean metric on $\mathbb{R}^D$. The 2-Wasserstein distance defined above induces a metric topology on $\mathcal{M}$.

Compared to other distances in the space of point clouds, for instance the Gromov-Hausdorff distance which is commonly used in the study of stability theory of persistent homology (see, e.g., Blumberg & Lesnick, 2022), the 2-Wasserstein distance endows the space of point clouds with a favorable manifold structure. This manifold structure ensures that every small neighborhood is isometric to an Euclidean open set.

We discuss the topological manifold structure (Appendix B.1) and Riemannian manifold structure (Appendix B.2) on the space of point clouds. Then we introduce the Riemannian style definition for perturbation vector fields and gradient vector fields (Appendix B.3).

B.1 Manifold structure

We proceed to establish a manifold structure on $\mathcal{M}$.

Proposition 4. *Let $\mathcal{M}$ be the set containing all point clouds in $\mathbb{R}^D$ with N distinct points, and d_W be the 2-Wasserstein distance on $\mathcal{M}$. For any point cloud $X \in \mathcal{M}$, there exists a Wasserstein ball $B_W(X, \varepsilon_X)$ and an injective mapping $\xi_X : B_W(X, \varepsilon_X) \to \mathbb{R}^{D \times N}$ such that*

$$d_W(Y,Z) = d_E(\xi_X(Y), \xi_X(Z)), \quad \forall Y, Z \in B_W(X, \varepsilon_X).$$

Proof. Consider a point cloud $X = \{x_i\}_{i=1}^N$ in $\mathcal{M}$. For arbitrary $\varepsilon > 0$, we construct an injective mapping ξ_X from $B_W(X, \varepsilon)$ to $\mathbb{R}^{D \times N}$, then choose a radius ξ_X such that the above equation holds. Denote $[N] = \{1, 2, \ldots, N\}$. The map $\xi_X(X)$ can be characterized by a total order in X, $\tau \colon [N] \to X$, where

$$\xi_X(X) = [\tau(1), \tau(2), \ldots, \tau(N)] \in \mathbb{R}^{D \times N}.$$

Table 1: Notations and definitions

Notation	Definition
$\mathcal{M}$	The manifold of point clouds
m	Dimension of manifold $\mathcal{M}$
D	Dimension of the space where point clouds are located
N	Number of points in point clouds
X	Point cloud
x	Point in a point cloud
d_W	Wasserstein distance between point clouds
$[N]$	The set $\{1, 2, \ldots, N\}$
$\Omega(X, Y)$	The set of bijections between point clouds X and Y
$\omega : X \to Y$	Bijection between point clouds X and Y
d_E	Euclidean distance
K, K_r	Simplicial complex
σ	Simplex
$\phi : K \to \mathbb{R}$	Filtration function
b	Birth parameter of homology classes
d	Death parameter of homology classes
l	Lifespan parameter of homology classes
$T_X\mathcal{M}$	Tangent space at X on $\mathcal{M}$
v	Tangent vector
$V \colon \mathcal{M} \to \sqcup_X T_X\mathcal{M}$	Vector field
$V(X)$	Tangent vector at X assigned by V
$\pi \colon \mathcal{M} \to \mathcal{M}$	Perturbation mapping on the space of point clouds
V_π	Vector field induced by perturbation π
$\rho \colon \mathcal{M} \to \mathbb{R}$	One-dimensional feature function on the space of point clouds
$\nabla\rho$	Gradient vector field of function ρ
$\mathcal{N}$	The manifold of persistence images
n	Dimension of manifold $\mathcal{N}$
η	Transformation on PD points from birth-death to birth-lifespan coordinate
$g_{b,l}$	Gaussian kernel located at (b, l)
P	Resolution of persistence images
γ^2	Variance of Gaussian kernel in persistence images
$\alpha(b, l)$	Weighting function
ψ	Persistence surface
$f : \mathcal{M} \to \mathcal{N}$	Encoding map from $\mathcal{M}$ to $\mathcal{N}$
J_X^f, J^f, J_X, J	Jacobian mapping (Jacobian matrix) of map f at X
G_X^f, G^f, G_X, G	Gram matrix of map f at X
λ_i^f, λ_i	The i-th largest singular value of the Jacobian mapping
q_i^f, q_i	The i-th eigenvector of the PH encoding mapping
$\lVert \cdot \rVert_f$	Pull-back norm induced by mapping f
$\mathcal{D}$	Finite data set of point clouds
d_{BW}	Bures-Wasserstein distance between positive-definite matrices

τ reorders X by assigning $[N] = \{1, 2, \ldots, N\}$ to $\{x_i\}_{i=1}^N$. For any other point cloud $Y \in B_W(X, \varepsilon)$, there exists an optimal transport plan between X and Y, denoted by $\omega_{XY} : X \to Y$, satisfying

$$\omega_{XY} = \underset{\omega \in \Omega(X,Y)}{\arg\min} \left(\sum_{x \in X} d_E^2(x, \omega(x)) \right)^{\frac{1}{2}}.$$

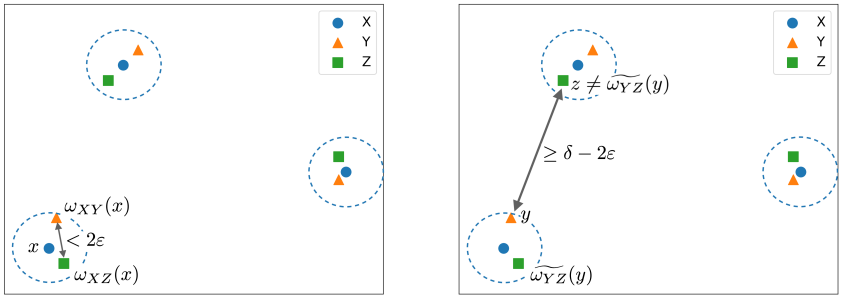

Figure 10: Shown are point clouds Y (orange triangles) and Z (green squares) in a Wasserstein neighborhood of point cloud X (blue circles). Right: both $\omega_{XY}(x) \in Y$ and $\omega_{XZ}(x) \in Z$ are close to $x \in X$, and hence they are close to each other. Left: the distance between y and $z \neq \widetilde{\omega_{YZ}}(y) \in Z$ is lower bounded.

This maps assigns each element in X to a distinct element in Y. We define an embedding ξ_X from Y to $\mathbb{R}^{D\times N}$ as follows:

$$\xi_X(Y) = [\omega_{XY} \circ \tau(1), \omega_{XY} \circ \tau(2), \ldots, \omega_{XY} \circ \tau(N)] \in \mathbb{R}^{D\times N}.$$

We proceed to show that ξ_X is an injective embedding. For any $Y, Z \in B_W(X, \varepsilon)$ with $\xi_X(Y) = \xi_X(Z)$, consider optimal transport plans ω_{XY} between X and Y, and ω_{XZ} between X and Z. Since $\xi_X(Y) = \xi_X(Z)$, we have $\omega_{XY}(x) = \omega_{XZ}(x), \forall x \in X$. Hence, for any $y \in Y$,

$$y = \omega_{XY} \circ (\omega_{XY})^{-1}(y) = \omega_{XZ} \circ (\omega_{XY})^{-1}(y) \in Z.$$

Notice $\omega_{XZ} \circ (\omega_{XY})^{-1}$ is a bijection between Y and Z. Therefore, $Y = Z$ and ξ_X is injective.

Next we calculate the radius ξ_X that preserves the distance between any two point clouds. The goal is to find a radius ε such that for any $Y, Z \in B_W(X, \varepsilon)$, the Wasserstein distance between the point clouds Y and Z,

$$d_W(Y, Z) = \left(\sum_{y \in Y} d_E^2(y, \omega_{YZ}(y)) \right)^{\frac{1}{2}},$$

is equal to the Euclidean distance between the embedding $\xi_X(Y)$ and $\xi_X(Z)$,

$$\begin{aligned} d_E(\xi_X(Y), \xi_X(Z)) &= \left(\sum_{i=1}^{N} d_E^2(\omega_{XY} \circ \tau(i), \omega_{XZ} \circ \tau(i)) \right)^{\frac{1}{2}} \\ &= \left(\sum_{x \in X} d_E^2(\omega_{XY}(x), \omega_{XZ}(x) \right)^{\frac{1}{2}} \\ &= \left(\sum_{y \in Y} d_E^2(y, \omega_{XZ} \circ (\omega_{XY})^{-1}(y) \right)^{\frac{1}{2}}. \end{aligned}$$

Notice it suffices to find a radius ε such that for any $Y, Z \in B_W(X, \varepsilon)$, $\omega_{YZ} = \omega_{XZ} \circ (\omega_{XY})^{-1}$. Equivalently, the optimal bijection between Y and Z is given by the composition $\omega_{XZ} \circ (\omega_{XY})^{-1} : Y \to X \to Z$. The key idea is that if Y and Z are both sufficiently close to X in the sense of the Wasserstein distance, then the distance between $y \in Y$ and $(\omega_{XY})^{-1}(y)$ and the distance between $z \in Z$ and $(\omega_{XZ})^{-1}(z)$ will be small. Hence, each point $y \in Y$ will be close to $\omega_{XZ} \circ (\omega_{XY})^{-1}(y)$ and thus we will have $\omega_{YZ} = \omega_{XZ} \circ (\omega_{XY})^{-1}$. The situation is illustrated in Figure 10.

Denote $\omega_{XZ} \circ (\omega_{XY})^{-1}$ as $\widetilde{\omega_{YZ}}$. For any point cloud $Y \in B_W(X, \varepsilon)$, we have

$$d_W(X, Y) = \left(\sum_{x \in X} d_E^2(x, \omega_{XY}(x)) \right)^{\frac{1}{2}} < \varepsilon.$$

Hence,

$$\max_{x\in X} d_E(x, \omega_{XY}(x)) < \varepsilon.$$

This suggests for any two point clouds $Y, Z \in B_W(X, \varepsilon)$,

$$\max_{x\in X} d_E(\omega_{XY}(x), \omega_{XZ}(x)) < 2\varepsilon.$$

Equivalently,

$$\max_{y\in Y} d_E(y, \widetilde{\omega_{YZ}}(y)) < 2\varepsilon.$$

Let δ denote the minimal pairwise distance of points in X:

$$\delta = \min_{x_1,x_2\in X} d_E(x_1, x_2).$$

Note δ is strictly greater than zero since points in X are mutually different. For a fixed point $y \in Y$, any point z in Z other than $\widetilde{\omega_{YZ}}(\mathrm{y})$ has a lower-bounded distance from y:

$$\min_{z\in Z\setminus\{\widetilde{\omega_{YZ}}(y)\}} d_E(y, z) \geq \delta - 2\varepsilon, \quad \forall y \in Y.$$

Now consider $\varepsilon_X = \frac{\delta}{8}$. We have

$$d_E(y, \widetilde{\omega_{YZ}}(y)) < 2\varepsilon_X = \frac{1}{4}\delta < \frac{3}{4}\delta = \delta - 2\varepsilon_X \leq \min_{z\in Z\setminus\{\widetilde{\omega_{YZ}}(y)\}} d_E(y, z), \quad \forall y \in Y.$$

Equivalently,

$$\widetilde{\omega_{YZ}}(y) = \arg\min_{z\in Z} d_E(y, z), \quad \forall y \in Y.$$

This means $\omega_{YZ} = \widetilde{\omega_{YZ}}$, which completes the proof. □

Corollary 5. *Let $\mathcal{M}$ be the set containing all point clouds in $\mathbb{R}^D$ with N distinct points. $\mathcal{M}$, together with the metric topology induced by 2-Wasserstein distance, forms a manifold of dimension $D \times N$.*

Proof. By Proposition 4, for every $X \in \mathcal{M}$ one can find a neighborhood $B_W(X, \varepsilon_X)$ and an injective mapping $\xi_X : B_W(X, \varepsilon_X) \to \mathbb{R}^{D\times N}$ satisfying

$$d_W(Y, Z) = d_E(\xi_X(Y), \xi_X(Z)), \quad \forall Y, Z \in B_W(X, \varepsilon_X).$$

Notice ξ_X is a bijective isometry between $(B_W(X, \varepsilon_X), d_W)$ and $(\xi_X(B_W(X, \varepsilon_X)), d_E)$. Hence, ξ_X is open and continuous. Consequently, ξ_X is a homeomorphism, and $\{\xi_X : B_W(X, \varepsilon_X) \to \mathbb{R}^{D\times N}\}_{X\in\mathcal{M}}$, serving as an atlas, endows $\mathcal{M}$ with the manifold structure. □

B.2 Riemannian metric structure

Next we introduce the Riemannian metric structure for the manifold of point clouds, and show the distance induced by the Riemannian metric coincides with the Wasserstein distance. To this end, consider an alternative definition for the space of point clouds:

$$\mathcal{M}' = \{X \colon [N] \to \mathbb{R}^D \mid X \text{ is injective}\}/\sim_{S_N} .$$

The equivalence relation is defined by:

$$X_1 \sim_{S_N} X_2 \quad \Leftrightarrow \quad \exists \nu \in S_N \colon X_1 \circ \nu = X_2 \quad \Leftrightarrow \quad \mathrm{Im}(X_1) = \mathrm{Im}(X_2).$$

Here S_N denotes the N-symmetric group and $\mathrm{Im}(X)$ denotes the image of mapping X. Two mappings are deemed equivalent when their images are identical. Note the image of each mapping $X : [N] \to \mathbb{R}^D$ is a point cloud in $\mathbb{R}^D$ as we defined earlier, i.e. $\mathrm{Im}(X) \in \mathcal{M}$. In fact, the mapping $\mathcal{M}' \to \mathcal{M} \colon [X] \mapsto \mathrm{Im}(X)$ gives the

identification between the original and new definitions for the space of point clouds. For simplicity, we use one representative X to denote the equivalence class $[X]$ and also refer to X's as point clouds.

This definition allows defining smooth curves in the point clouds space. Specifically, a smooth curves in $\mathcal{M}'$ is an element of the following set:

$$\mathcal{C} = \{\gamma\colon [N] \times I \to \mathbb{R}^D \mid \gamma(\cdot, t) \text{ is injective}, \forall t \in I; \gamma(k, \cdot) \in C^\infty(I), \forall k \in [N]\}/\sim_{S_N} .$$

Here I denote the closed unit interval $[0, 1]$ and the equivalence relation is defined by

$$\gamma_1 \sim_{S_N} \gamma_2 \quad \Leftrightarrow \quad \exists \nu \in S_N\colon \gamma_1(\nu(\cdot), \cdot)) = \gamma_2(\cdot, \cdot)) \quad \Leftrightarrow \quad \mathrm{Im}(\gamma_1(\cdot, t)) = \mathrm{Im}(\gamma_2(\cdot, t)), \forall t.$$

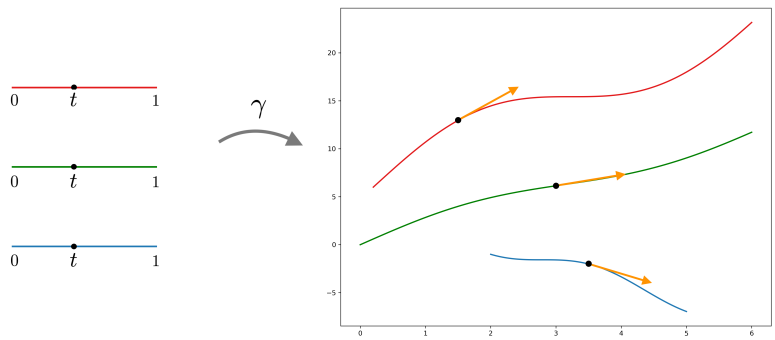

Figure 11: Shown is a curve in the point cloud space $[\gamma] \in \mathcal{C}$, which maps the set $\{1, 2, 3\} \times [0, 1]$ (left) to the plane $\mathbb{R}^2$ (right). In the right panel, the collection of black dots represent the image of γ at a specific time $t \in [0, 1]$, which forms a point cloud in $\mathbb{R}^2$; the collection of orange arrows represents the velocity tangent vector at γ_t along the curve γ.

A curve in point cloud space $[\gamma] \in \mathcal{C}$ is essentially a collection of N curves in $\mathbb{R}^D$ such that at each time t the N points on those curves form a point cloud (see Figure 11). For simplicity we use γ to denote the equivalence class $[\gamma]$.

The *tangent space* at point cloud X is defined as

$$T_X\mathcal{M}' = \{V\mid_{\mathrm{Im}(X)}\colon \mathrm{Im}(X) \to \mathbb{R}^D \mid V \text{ a vector field on } \mathbb{R}^D\}.$$

Define a *Riemannian metric* on tangent space $T_X\mathcal{M}'$:

$$g_X(V, W) = \sum_{x \in \mathrm{Im}(X)} \langle V(x), W(x)\rangle.$$

Endowed with this Riemannian metric, $\mathcal{M}'$ is a Riemannian manifold, denoted by $(\mathcal{M}', g)$. We now discuss the distance induced by this Riemannian metric. For a curve in point cloud space $\gamma \in \mathcal{C}$, the *velocity tangent vector* at γ_t along curve γ is defined as (see an example in the right panel in Figure 11):

$$\dot{\gamma}_t\colon \mathrm{Im}(\gamma_t) \to \mathbb{R}^D; \quad x \to \partial_t\gamma(k_x, t).$$

Here $k_x \in [N]$ is the preimage of point x under γ_t, i.e., $\gamma_t(k_x) = x$. The *length of curve* is given by the length functional L:

$$L\colon \mathcal{C} \to \mathbb{R}; \quad \gamma \mapsto \int_I \sqrt{g_{\gamma_t}(\dot{\gamma}_t, \dot{\gamma}_t)}dt.$$

The distance induced by the Riemannian metric is the minimal length of curves between two points on the manifold.

Definition 6 (Riemannian distance)**.** Let $(\mathcal{M}, g)$ be a Riemannian manifold. The Riemannian distance between two points $X, Y \in \mathcal{M}$ is defined as

$$d_g(X, Y) = \inf\{L(\gamma) : \gamma \text{ a smooth curve in } \mathcal{M} \text{ connecting } X \text{ and } Y\}.$$

We point out the Riemannian distance coincides with the Wasserstein distance. To see this, we need the following result by Do Carmo & Flaherty Francis (Lemma 2.3, Chap. 9, 1992).

Lemma 7 (Do Carmo & Flaherty Francis, 1992)**.** Let X_0, X_1 be two points in Riemannian manifold $\mathcal{M}$, and γ a curve joining X_0 to X_1. Then γ minimizes the length functional if and only if γ minimizes the energy functional defined as follows:

$$E\colon \mathcal{C} \to \mathbb{R}; \quad \gamma \mapsto \int_I g_{\gamma_t}(\dot{\gamma}_t, \dot{\gamma}_t)dt.$$

Moreover, when γ is the minimizer, $L(\gamma) = \sqrt{E(\gamma)}$.

Now we proceed to introduce the main statement of this subsection.

Proposition 8. *Let $(\mathcal{M}', g)$ be the Riemannian manifold of point clouds. Then the Riemannian distance is equivalent to Wasserstein distance, $d_g = d_W$.*

Proof. Consider two point clouds $X_0, X_1 \in \mathcal{M}'$. Let L be the length functional, and E be the energy functional of all curves connecting X_0 and X_1. For any curve γ joining X_0 to X_1, we have

$$\begin{aligned}
E(\gamma) &= \int_I g_{\gamma_t}\left(\dot{\gamma}_t, \dot{\gamma}_t\right) dt \\
&= \int_I \sum_{x \in \mathrm{Im}(\gamma_t)} \langle \dot{\gamma}_t(x), \dot{\gamma}_t(x) \rangle dt \\
&= \int_I \sum_{x \in \mathrm{Im}(\gamma_t)} \|\partial_t \gamma(k_x, t)\|^2 dt \\
&= \int_I \sum_{k=1}^{N} \|\partial_t \gamma(k, t)\|^2 dt \\
&= \sum_{k=1}^{N} \int_I \|\partial_t \gamma(k, t)\|^2 dt \\
&= \sum_{k=1}^{N} E[\gamma(k, \cdot)].
\end{aligned}$$

The above equations along with Lemma 7 indicate the minimizer of the length functional L coincides with the minimizer of $\sum_{k=1}^{N} E(\gamma(k, \cdot))$. Recall for each k, $\gamma(k, \cdot)$ is a curve in $\mathbb{R}^D$ joining $X_0(k)$ to $X_1(k)$. In Euclidean spaces, it is known that straight lines minimize the length functional, which implies, by Lemma 7, they also minimize $E(\gamma(k, \cdot))$. Specifically for fixed end points $X_0(k)$ and $X_1(k)$, $E(\gamma(k, \cdot))$ has minimal value $d_E^2(X_0(k), X_1(k))$. Therefore, finding the minimizer of $\sum_{k=1}^{N} E(\gamma(k, \cdot))$ is equivalent to finding the bijection between $\mathrm{Im}(X_0)$ and $\mathrm{Im}(X_1)$ that produces the minimal value of the sum of squared distances between points paired by the bijection. This exactly coincides with the optimal transport problem. In conclusion, we have

$$\begin{aligned}
d_g(X_0, X_1) &= \min_{\gamma} L(\gamma) \\
&= \min_{\gamma} \sqrt{E(\gamma)} \\
&= \min \left(\sum_{k=1}^{N} E[\gamma(k, \cdot)] \right)^{\frac{1}{2}} \\
&= \min_{\omega \in \Omega(\mathrm{Im}(X_0), \mathrm{Im}(X_1))} \left(\sum_{x \in \mathrm{Im}(X_0)} d^2(x, \omega(x)) \right)^{\frac{1}{2}} \\
&= d_W(\mathrm{Im}(X_0), \mathrm{Im}(X_1))
\end{aligned}$$

This is what was claimed. □

B.3 Vector fields

In this section, we provide definitions for perturbation vector fields and gradient vector fields for the manifold of point clouds.

Assume $\pi : \mathcal{M} \to \mathcal{M}$ is a perturbation mapping. As introduced in Section 3.2, the perturbation vector fields V_π in the Euclidean space is defined as:

$$V_\pi(X) = \pi(X) - X, \quad \forall X \in \mathcal{M}.$$

The main idea behind this definition is that each tangent vector $V_\pi(X)$ specifies the direction of the straight line connecting X and $\pi(X)$. For general Riemannian manifolds, the notion of straight lines is generalized by minimizing geodesics. Formally, the curve between two points on manifold that minimizes the length functional is a *minimizing geodesic*. Then the question arises whether there exists a minimizing geodesic between any two points on the manifold. To address this, we introduce the concepts of geodesically completeness and the Hopf-Rinow theorem.

As introduced in Appendix B.2, the Riemannian metric induces a distance d_g on the manifold $\mathcal{M}$. A sequence $\{X_i\}_{i\in\mathbb{Z}^+}$ of points on $(\mathcal{M}, d_g)$ is a d_g-*Cauchy sequence* if for any positive number ε there exists a positive integer N such that $d_g(X_i, X_j) < \varepsilon, \forall i, j > N$.

Definition 9 (Geodesically complete manifold)**.** The Riemannian manifold $(\mathcal{M}, g)$ is geodesically complete if any d_g-Cauchy sequence $\{X_i\}_{i\in\mathbb{Z}^+}$ converges in $\mathcal{M}$: $\exists Y \in \mathcal{M}$ such that $\lim_{i\to\infty} d_g(X_i, Y) = 0$.

Since the Euclidean space $\mathbb{R}^D$ is a complete metric space, automatically the Riemannian manifold of point clouds is geodesically complete. Also notice this manifold is connected, since there exists a path connecting any two point clouds. The Hopf-Rinow theorem ensures the existence of minimizing geodesics between any two point clouds.

Theorem 10 (Hopf-Rinow theorem)**.** *Let $(\mathcal{M}, g)$ be a connected Riemannian manifold. If $(\mathcal{M}, d_g)$ is geodesically complete, there exists a minimizing geodesic between any two points on $\mathcal{M}$.*

Now we formally define the perturbation vector fields.

Definition 11 (Perturbation vector field)**.** Let $(\mathcal{M}, g)$ be a geodesically complete manifold, and $\pi\colon \mathcal{M} \to \mathcal{M}$ be a perturbation mapping. The perturbation vector field V_π is defined as

$$V_\pi\colon \mathcal{M} \to \sqcup_X T_X\mathcal{M}; \quad X \mapsto V_\pi(X) = \dot{\gamma}_{X,\pi(X)}(0),$$

where $\gamma_{X,\pi(X)}$ is a minimizing geodesic $\gamma_{X,\pi(X)}\colon I \to \mathcal{M}$ with $\gamma_{X,\pi(X)}(0) = X$ and $\gamma_{X,\pi(X)}(1) = \pi(X)$.

We proceed to define the Riemannian gradient vector field. Assume ρ is a real-valued smooth function on $\mathcal{M}$. In the cases of Euclidean spaces, the gradient vector can be characeterized by the following property:

$$\langle \nabla\rho(X), v\rangle = \frac{\partial}{\partial v}\rho, \quad \forall X \in \mathbb{R}^m, v \in T_X\mathbb{R}^m = \mathbb{R}^m.$$

For general Riemannian manifolds, the notion of directional derivatives is generalized by derivations.

Definition 12 (Derivative)**.** Let $\mathcal{M}$ be a manifold, and $C^\infty(\mathcal{M})$ be the space of smooth functions on $\mathcal{M}$. A derivative at $X \in \mathcal{M}$ is a linear map $\partial\colon C^\infty(\mathcal{M}) \to \mathbb{R}$ satisfying the Leibniz identity:

$$\partial(fg) = \partial(f) \cdot g(X) + \partial(g) \cdot f(X).$$

For a fixed point $X \in \mathcal{M}$, it turns out that each tangent vector $v \in T_X\mathcal{M}$ can be uniquely associated with a derivative, denoted by ∂_v, in the sense that $\partial_v(\rho)$ measures the rate of change of the function value $\rho(X)$, moving through X with the velocity specified by v. Detailed discussion regarding the equivalence between tangent vectors and derivations can be found in Tu (Chap. 8, 2011). Now we provide the definition of the Riemannian gradient.

Definition 13 (Gradient vector field)**.** Let $(\mathcal{M}, g)$ be a Riemannian manifold, and $\rho\colon \mathcal{M} \to \mathbb{R}$ a smooth function. The gradient vector field of ρ, denoted by $\nabla\rho$, is defined as the vector field

$$\nabla\rho\colon \mathcal{M} \to \sqcup_X T_X\mathcal{M}; \quad X \mapsto \nabla\rho(X)$$

satisfying the property:

$$g_X(\nabla\rho(X), v) = \partial_v(\rho), \quad \forall X \in \mathcal{M}, v \in T_X\mathcal{M}.$$

Note that Definition 11 and Definition 13 are applicable to other types of data, provided that the data space can be equipped with a geodesically complete Riemannian manifold structure.

C Differentiability of the PI encoding

We provide details for the differentiability of PI encoding and computation the Jacobian. We start with a brief review of the PI computation pipeline. Let X be a point cloud and $\mathrm{Cl}(X, R)$ be the clique complex of the R-neighborhood graph. The pipeline for computing k-dimensional PH and constructing PI involves the following steps (Section 2):

- Determine a filtration map: $\phi\colon \mathrm{Cl}(X, R) \to \mathbb{R}$. This induces a filtration.
- Pair simplices such that every non-trivial k-dimensional homology class in the filtration is associated to a pair of simplices (σ_b, σ_d). Roughly, one can think of σ_b as the k-simplex that creates the homology class and σ_b as the $(k+1)$-simplex that trivializes the homology class.
- Obtain the persistence diagram PD, as a multiset of birth-death values $(\phi(\sigma_b), \phi(\sigma_d))$ and transform PD into a multiset $\eta(\mathrm{PD})$ of birth-lifespan pairs $(\phi(\sigma_b), \phi(\sigma_d) - \phi(\sigma_b))$.
- Choose a smooth kernel $g_{(b,l)}$, e.g., Gaussian kernel, and a smooth weighting function $\alpha(b, l)$, e.g., linear function of l, and construct the persistence surface ψ:

$$\psi(x, y) = \sum_{(b,l)\in\eta(\mathrm{PD})} \alpha(b, l) g_{(b,l)}(x, y).$$

- Obtain the persistence image PI in the form of a $P \times P$ matrix whose (i, j)-th entry is:

$$\mathrm{PI}_{ij} = \int_{\mathrm{pixel}_{ij}} \psi(x, y).$$

  Here $\cup_{ij}\mathrm{pixel}_{ij}$ forms a grid subdivision of a subdomain of ψ. In our experiments, we consider a rectangle subdomain $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ and evenly-spaced rectangle pixels. Specifically, consider grid points $x_k = x_{\min} + k\Delta x$ for $k = 0, 1, \cdots, P$, where $\Delta x = \frac{x_{\max} - x_{\min}}{P}$, and $y_l = y_{\min} + l\Delta y$ for $l = 0, 1, \cdots, P$, where $\Delta y = \frac{y_{\max} - y_{\min}}{P}$. Set $\mathrm{pixel}_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ for $i = 0, 1, \cdots, P-1$ and $j = 0, 1, \cdots, P-1$. We then estimate the integral value by

$$\mathrm{PI}_{ij} = s \cdot \psi(x_i, y_i),$$

  where s is the area of pixel_{ij}, i.e., $s = \Delta x \Delta y$.

Notice ψ, as a sum of Gaussian kernels, is differentiable with respect to the coordinates of the kernel centers (b, l). Meanwhile, the filtration mapping is differentiable with respect to the coordinate of every point in X. For example, the filtration value in Vietoris-Rips filtration, $\phi(\sigma) = \mathrm{Diam}(\sigma) = \max_{x,y\in\sigma} d_E(x, y)$, is differentiable with respect to the coordinate of any point in X. Hence, the PI encoding is differentiable.

More technically, as discussed in Appendix B.1, for every point cloud $X \in \mathcal{M}$, there exists a neighborhood U of X and a coordinate map $\xi\colon U \to \mathbb{R}^{N\times D}$. We claim that the PI encoding map $f\colon \mathcal{M} \to \mathbb{R}^{P\times P}$ is smooth in the sense that the following composition map is smooth:

$$f \circ \xi^{-1}\colon \xi(U) \to \mathbb{R}^{P\times P}.$$

Here we omit the manifold notation for the output space $\mathcal{N}$, given that $\mathcal{N}$ is a submanifold in $\mathbb{R}^{P\times P}$. Specifically, assume $\tau\colon [N] \to X$ is the total order that characterizes $\xi(X)$, i.e., $\xi(X) = [\tau(1), \tau(2), \ldots, \tau(N)]$. For any $(i,j) \in [P] \times [P]$, $k \in [N]$, the partial derivative $\frac{\partial \mathrm{PI}_{ij}}{\partial \tau(k)}$ can be formulated as follows:

$$\begin{aligned}\frac{\partial \mathrm{PI}_{ij}}{\partial \tau(k)} &= \sum_{\sigma} \frac{\partial \mathrm{PI}_{ij}}{\partial \phi(\sigma)} \frac{\partial \phi(\sigma)}{\partial \tau(k)} \\ &= \sum_{\sigma} \frac{\partial s \cdot \psi(x_{ij}, y_{ij})}{\partial \phi(\sigma)} \frac{\partial \phi(\sigma)}{\partial \tau(k)}.\end{aligned}$$

For computing the Jacobian in our experiments, we use the Gudhi library (The GUDHI Project, 2020) version 3.8.0 and Tensorflow version 2.12.0. Specifically, we use Gudhi library to collect PDs, especially `Gudhi.tensorflow.RipsLayer` class to collect PDs with respect to the Rips filtration. We then manually compute PIs as described above using Tensorflow. Finally, we collect the Jacobian for the whole pipeline with `tensorflow.GradientTape.jacobian` function.

D Visualizing the Jacobian of the encoding over the data manifold

In this section we visualize the singular value decomposition (SVD) of Jacobian on a toy data set, aiming at providing further intuition for the eigenvectors of Jacobian mappings. We consider point clouds that are uniformly sampled from axis-aligned ellipses of width w and height h. These are illustrated in the left panel of Figure 12. To visualize the Jacobian at an input point cloud X, we plot the pull-back unit ball around X in the data manifold,

$$B^*(X, 1) = \{v \in T_X\mathcal{M} : \|v\|_f = 1\}.$$

This corresponds to the preimage of a unit ball in $T_{f(X)}\mathcal{N}$. Notably, the equation $1 = \|v\|_f = \sum \lambda_i \langle v, q_i\rangle^2$ indicates that the pull-back unit ball forms an m-dimensional ellipsoid with semi-axes $q_1, q_2, \ldots, q_m$, and the lengths of these semi-axes are given by $\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}, \ldots, \frac{1}{\sqrt{\lambda_m}}$.

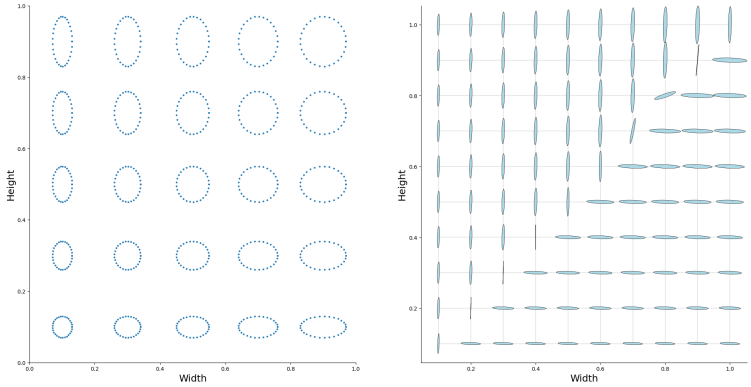

Figure 12: Visualization of the pull-back metric on a toy data set. Left: The space of point clouds sampled from ellipses of width w and height h. Right: The pull-back unit ball at different locations on the space of ellipses. Shorter semi-axes of the pull-back unit ball correspond to larger eigenvalues of the Jacobian.

In the right panel of Figure 12 we plot the pull-back unit balls for the encoding f given by the 1-dimensional PI with respect to the Vietoris-Rips filtration. The plot reveals that the eigenvectors of the encoding associated

with the larger eigenvalue consistently align with the direction of increasing $\min\{w, h\}$. This alignment is in accordance with what we would expect the Vietoris-Rips filtration to capture on this specific data set, since the death value depends on the radius of the inner circumcircle of a hole. Consequently, variations in the length of the major axis have minimal impact on PI. Conversely, altering the length of the minor axis directly affects the death parameter and changes the PI.

E Point saliency maps for PH encodings

In the context of interpretability, point saliency maps are commonly used tools to explain the decisions made by trained models (Montavon et al., 2018; Zheng et al., 2019). These maps assign importance scores to each point in an input point cloud (or to each pixel in an input image), indicating their significance in relation to the model's prediction. In our study, we employ a similar strategy to visualize the importance of each point in an input point cloud with respect to the PH encoding.

Given a point cloud X and an encoding map f, we define the encoding saliency score for each point $x_i \in X$ as

$$s_f(x_i) = \left\| \frac{\partial f}{\partial x_i} \right\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm, which is the generalized Euclidean norm for matrices. Here, $\frac{\partial f}{\partial x}$ represents the Jacobian matrix of the encoding, which is a matrix of format $P^2 \times D$ obtained by flattening the PI into a vector in $\mathbb{R}^{P^2}$ and taking the partial derivative of each output pixel with respect to each coordinate of each point in the input point cloud. Similarly, $\frac{\partial f}{\partial x_i}$ is the vector of partial derivatives of the encoding with respect to the coordinates of the i-th point x_i in the point cloud. This score quantifies the sensitivity of the representation to variations on the coordinates of x.

In Figure 13, we plot the point saliency score with respect to the Rips encoding used in Section 5 for point clouds sampled from eight synthetic curves in $\mathbb{R}^2$. For each point cloud we highlight individual points according to their saliency scores. We observe that endpoints and points related to the inner circumcycle are often highlighted. These points correspond to the simplices that create/destroy certain homology classes in the filtration. For instance, in the curved rhombus in the first-row, second-column panel, the endpoints (yellow) on the upper right side are highlighted. Note that the edge between these two endpoints will connect the gap and create a homology class. Therefore, variations on these two points can significantly change the birth parameter of the corresponding homology class. Meanwhile, the two points (green) on the upper left and lower right sides are highlighted. They are related to the inner circumcycle of the rhombus, and are vertices of the triangle that destroys the homology class. Therefore variations on these two points can change the death parameter of the homology class significantly.

Before concluding this section, it's important to note that the saliency scores shown in Figure 13 are heavily influenced by the sampling process. For example, many of the highlighted endpoints shown in Figure 13

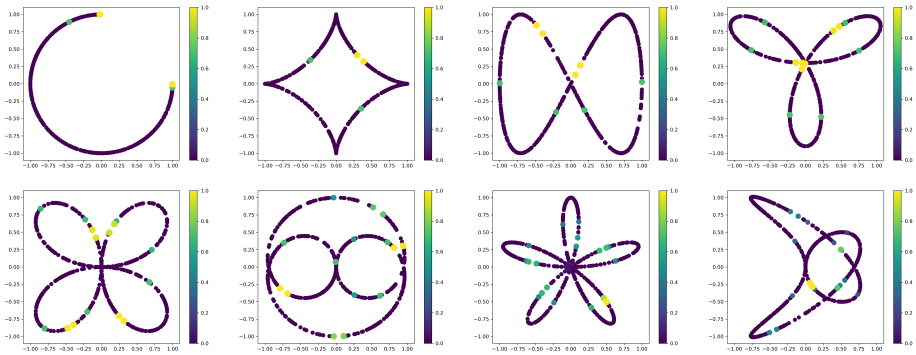

Figure 13: Point saliency score with respect to 1-dimensional PIs on Vietoris-Rips filtration for synthetic 2-dimensional point clouds.

are a result of sparse sampling. Consequently, the resulting saliency scores might be more indicative of the sampling rather than the underlying shape itself. To better comprehend the shape, one potential approach is to compute the average saliency score across multiple different samplings.

F Details on the experiments

F.1 Reproducibility

The data and code developed for this research are made available at `https://anonymous.4open.science/r/persistent-homology-0915`.

F.2 Jacobian normalization

Throughout our work, we use pull-back norm to quantify the sensitivity of the encoding method to data variations. One needs to be careful when comparing the pull-back norms induced by different encodings, since encodings may have different scaling levels. For example, consider Euclidean data space $\mathcal{M} = \mathbb{R}^2$, and two encoding mappings $f_1 = x + 0.1y$, $f_2 = x + 10y$. The pull-back norms of a tangent vector $v = [1, 0]^T$ with respect to f_1 and f_2 are

$$\|v\|_{f_1} = \|J^{f_1} v\| = \left\| \begin{pmatrix} 1 & 0 \\ 0 & 0.1 \end{pmatrix} \cdot [1, 0]^T \right\| = 1,$$
$$\|v\|_{f_2} = \|J^{f_2} v\| = \left\| \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix} \cdot [1, 0]^T \right\| = 1.$$

In terms of "absolute sensitivity", f_1 and f_2 has the same level of sensitivity to variation v. More specifically, when variation v is applied to a data point X, both $f_1(X)$ and $f_2(X)$ would change with distance approximately 1 in the representation space. However, in terms of "relative sensitivity", f_1 is more sensitive to v. The reason is that for f_1, the vector v is the eigenvector of the Jacobian with the largest eigenvalue; while for f_2, v is the eigenvector with the smallest eigenvalue. Equivalently, for f_1, v has the largest pull-back norm among all tangent vectors with the same norm as v, whereas for f_2, v has the smallest pull-back norm.

In Section 4.3, our goal is to study and compare the *focus* of different encodings. In Section 5.2 we search for the encoding whose primary *focus* is on the data variations of interest. Hence, in both sections we remove the scaling factor by considering the normalized Jacobian. Specifically, we divide the Jacobian matrix by its largest singular value, $\tilde{J}^f = J^f / \lambda_1^f$. For vector fields V we consider the normalized average pull-back norm:

$$\frac{1}{|\mathcal{D}|} \sum_{X \in \mathcal{D}} \|\tilde{J}^f \cdot V(X)\|.$$

Returning to the previous example, the normalized pull-back norm for vector v with respect to f_1 and f_2 are $\frac{\|J^{f_1} v\|}{\lambda_1^{f_1}} = \frac{1}{1} = 1$ and $\frac{\|J^{f_2} v\|}{\lambda_1^{f_2}} = \frac{1}{10} = 0.1$, respectively.

F.3 Identifying what is recognized

We provide details for the experiments in Section 4.

Radial Frequency Pattern data set Figure 14 shows some examples in the Radial Frequency Patterns (RFP) data set $\mathcal{D}_{\text{RFP}}$.

PH parameters For the Vietoris-Rips encoding, we set *maximal_edge_length* as 1. For the DTM filtration, we set *maximal_edge_length* as 0.5 and parameter m as 0.02. For the Height filtration, we set *maximal_edge_length* as 0.1. We note that we need to set the maximum edge length to a small value to be able to capture topological features of interest (for instance, one wants to avoid connecting different outer

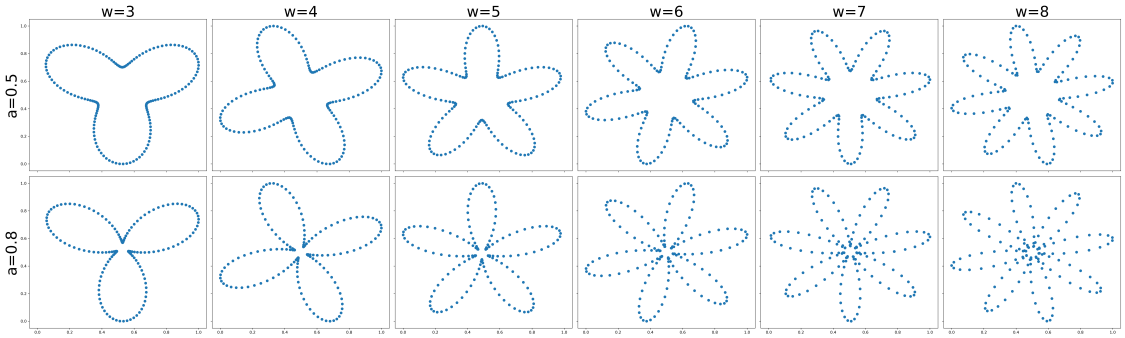

Figure 14: Visualization of the Radial Frequency Patter (RFP) data.

regions of petals); alternatively, one could use geodesic distances or cubical complexes, see also (Turkeš et al., 2022, Fig.20) for a detailed discussion. Notice there's no need to set a small maximum edge length for Rips and DTM filtrations since the filtration value of an edge in Rips and DTM filtrations takes the distance between its two vertices into account. In contrast, in Height filtration the filtration value of an edge is defined by the maximal filtration value of its vertices, which implies any two points will be immediately connected by an edge between them, if exists, after they appear in the filtration. For the construction of PI, we set the resolution P as 20, variance γ^2 of the Gaussian kernel as 10^{-4}, and the range of the image as $[0,1] \times [0,1]$. The weighting function is set as $\alpha(b,l) = l$. The implementation utilized Tensorflow version 2.12.0 and Gudhi (The GUDHI Project, 2020) version 3.8.0.

Visualization of the effects of perturbation on PH We visualizes the effects of *shearing* perturbation and *convex* perturbation on the persistent homology with respect to Rips filtration (see the upper panel in Figure 15) and Height filtration (see the lower panel in Figure 15). We omit plots associated with DTM filtration as they closely resemble those associated with Rips filtration. Please note that while these visualizations provide insights into the effects of perturbations on PH, they are highly dependent on the specific point clouds under consideration. In fact, the effects of perturbations on PH can vary significantly across different point clouds.

As shown in the upper penal in Figure 15, the *shearing* changes the sparsity of points in the point clouds and hence changes the birth values (x coordinates) of certain points in PD with respect to Rips filtration. Meanwhile, *shearing* changes the size of petals and hence changes the death values (y coordinates) of certain PD points with respect to Rips filtration. On the other hand, *convex* has the effect of "opening up" the central region of the point clouds. Consequently, some loops appear later in the filtration, notably the five loops that already exist in the second column of the first row but do not show up in the second column of the third row. Consequently, *convex* also induces changes in the birth values of PD points associated with these loops. We note that, in comparison to *shearing*, the *convex* perturbation has a more significant impact on PI. This is consistent with the findings illustrated in Figure 5, where Rips encoding is more sensitive to *convex* compared to *shearing*.

In the lower panel of Figure 15, we can observe distinct effects of these two perturbations on the PH associated with the Height filtration. The *shearing* perturbation directly changes the x coordinates of the points in the original points clouds, which correspond to their filtration values, and consequently changes the PI noticeably. On the other hand, the *convex* perturbation changes the PH in a more significant way. Specifically, certain small loops that initially appeared in the filtration (as seen in the first row of the lower panel in Figure 15) no longer appear in the *entire* filtration after perturbation (as seen in the third row of the lower panel in Figure 15). This observation arises from the small value we assigned to the *maximal_edge_length* parameter for Height filtration. In fact, for some point clouds, the central points originally have a distance smaller than *maximal_edge_length*. Then *convex* perturbation can pull these central points apart, causing their distance to exceed the *maximal_edge_length* threshold. Consequently, edges connecting these central points vanish from the filtration, resulting in the disappearance of certain small loops. In such cases, as shown in Figure 15, the *convex* can significantly change the PI associated with Height filtration. However, for other point clouds

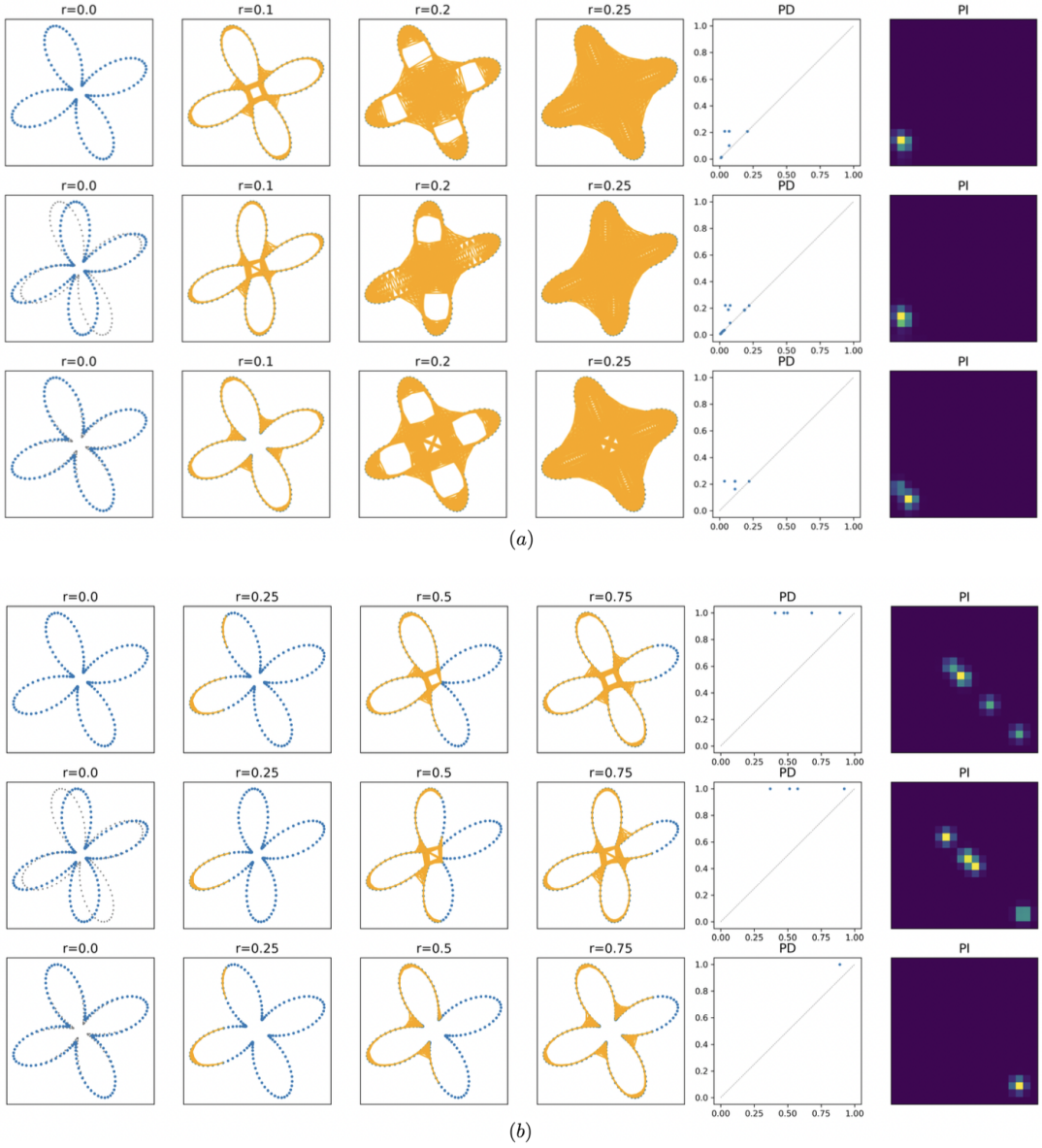

Figure 15: Figure (a) visualizes the effects of different perturbations on persistent homology with respect to Rips filtration. The first four columns illustrate the simplicial complex K_r in the Rips filtration with parameter r ranging from 0.0 to 0.25. The fifth and sixth columns show the corresponding PDs and PIs, respectively. The first row is associated with the original point cloud; while the second and third rows are associated with the point cloud perturbed by *shearing* and *convex*, respectively. Figure (b) visualizes the effects of perturbations on persistent homology with respect to Height filtration, following the same row-column arrangement as in Figure (a).

where the central points initially have a distance greater than *maximal_edge_length*, the corresponding PI may remain relatively unchanged after the *convex* perturbation. This is again consistent with the results shown in Figure 5, where the pull-back norms of the *convex* perturbation associated with Height filtration exhibit a significant range of variation.

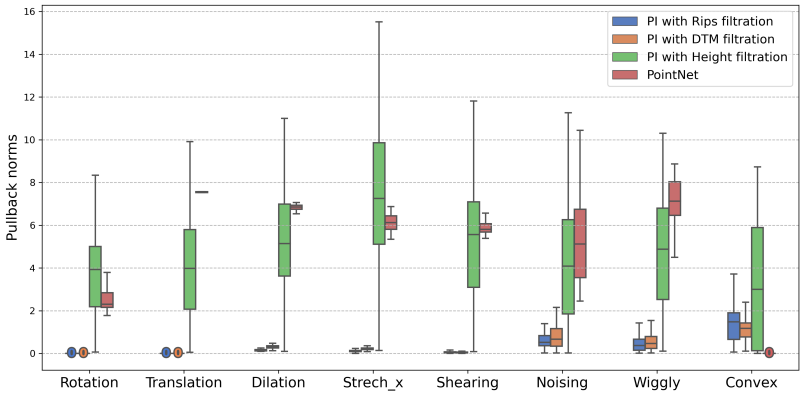

Figure 16: Unnormalized average pull-back norm of different perturbation vector fields with respect different encodings.

PointNet architecture and training details In our experiments, we labeled each point cloud data $X \in \mathcal{D}_{\text{RFP}}$ with the number of petals, i.e. the parameter w of the curve $\text{RFP}_{a,w}$ from which X is sampled. This produces 8 classes in total and we train the PointNet model to classify each point cloud in the data set. The PointNet model consists of a 1-dimensional convolutional layer with 64 filters and kernel size 1, followed by batch normalization and rectified linear unit (ReLU) activation. Then global max pooling is applied to obtain a permutation-invariant representation. This is followed by two fully connected layers with 128 and 64 hidden units, respectively, with ReLU activation. The final output layer uses softmax activation to produce theprobability distribution over the output classes. We trained the model with a batch size of 32 for 100 epochs, using a learning rate of 0.001. The optimization algorithm used was Adam, and the model was trained using the cross-entropy loss function. We note data augmentation techniques are used for training PointNet in literatures (see, e.g., Qi et al., 2017). However, we do not augment the data during training, in order to ensure a fair comparison with other encodings. The implementation utilized Tensorflow version 2.12.0.

Perturbation vector field estimation Let $\mathcal{D} = \{X_i\}_{i \in \mathcal{I}}$ be a finite data set of point clouds, and $\pi : \mathcal{M} \to \mathcal{M}$ a perturbation mapping defined on the data manifold. In the case where the data lies in Euclidean space, i.e. $\mathcal{M} = \mathbb{R}^m$, one can compute the perturbation vectors as following:

$$V_\pi(X) = \pi(X) - X, \quad \forall X \in \mathcal{D}.$$

However, in the case of general Riemannian data manifold, the subtraction between any two points on the manifold may not be well-defined. To address this, we control the perturbation mapping such that for every $X \in \mathcal{D}$ the perturbed point cloud $\pi(X)$ lies in a small neighborhood of X and calculate the perturbation vector via the local coordinate system. Specifically, as shown in Appendix B.1, for each point cloud $X \in \mathcal{D}$, one can find a neighborhood $X \in U_X \subset \mathcal{M}$ and an injective isometry $\xi_X : U_X \to \mathbb{R}^m$. We control the perturbation mapping sends every point X to U_X, i.e.

$$\pi(X) \in U_X, \quad \forall X \in \mathcal{D}.$$

Then the perturbation vectors can computed with the subtraction on the Euclidean domain $\xi_X(U_X)$ as follows:

$$V_\pi(X) = \xi_X(\pi(X)) - \xi_X(X), \quad \forall X \in \mathcal{D}.$$

Unnormalized pull-back norms As discussed in Appendix F.2, we consider the normalized average pull-back norms for perturbation tangent vector fields in Section 4.3. We present the results of unnormalized average pull-back norms in Figure 16. Note that one could conclude from Figure 16 that, on average, Height is more sensitive to *convex* than Rips in terms of "absolute sensitivity". However, we reach the opposite conclusion from Figure 5 in terms of "relative sensitivity".

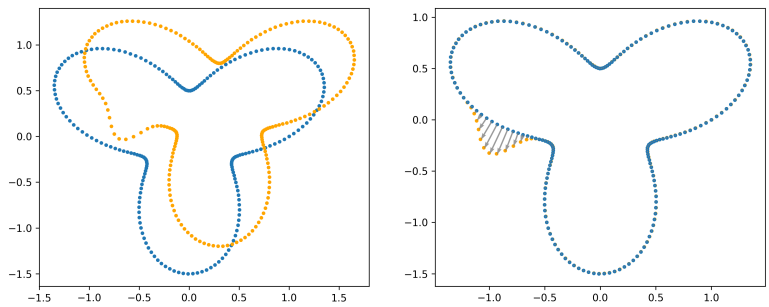

Figure 17: Left: shown are two point clouds in the plane. Right: the orange point cloud is transformed by iterated closest point algorithm and the grey vectors represent the difference vector between blue point clouds and the transformed orange point cloud.

F.4 Selecting hyperparameters

We provide further details on Section 5.

PH parameters In this section, we focus on Vietoris-Rips encodings. We set the parameter *maximal_edge_legnth* as 0.25, and the range of PI as $[0.0, 0.25] \times [0.0, 0.25]$. The implementation utilized Gudhi (The GUDHI Project, 2020) version 3.8.0 and Tensorflow version 2.12.0.

Gradient vector field estimation Let $\mathcal{D} = \{X_i\}_{i \in \mathcal{I}}$ be a finite data set of point clouds, and $\rho(X_i)$ be the corresponding feature values of point clouds in $\mathcal{D}$. In the case when the data lies in Euclidean space, we can estimate the gradient vectors with the finite difference method (FDM) as follows:

$$\overline{\nabla \rho(X)} = X' - X, \quad X' = \operatorname*{argmax}_{Y \in \mathcal{D}} \frac{\mid \rho(Y) - \rho(X) \mid}{d_E(X, X')}, \quad \forall X \in \mathcal{D}.$$

For binary categorical feature, i.e. $\rho(X) \in \{0, 1\}$, the formula can be modified as

$$\overline{\nabla \rho(X)} = X' - X, \quad X' = \operatorname*{argmin}_{Y \in \{Z \in \mathcal{D} : \rho(Z) \neq \rho(X)\}} d_E(X, Y), \quad \forall X \in \mathcal{D}.$$

However, in the case of general Riemannian data manifold, the subtraction between any two points on $\mathcal{M}$ may not be well-defined. To address this, when estimating gradient vector located at X, we send other point clouds in the data set to a neighborhood of X via transformation that preserves the feature value. In our experiment, we use Euclidean transformation since the *sex* feature is irrelevant to the position or orientation of the brain artery trees. Then FDM can be applied to the transformed data set, via the coordinate system on that neighborhood.

Specifically, we utilize the iterated closest point (ICP) method (Chen & Medioni, 1992; Besl & McKay, 1992). Let $E(\mathbb{R}^D)$ be the collection of all Euclidean transformations in $\mathbb{R}^D$

$$E(\mathbb{R}^D) = \{\iota : \mathbb{R}^D \to \mathbb{R}^D : d_E(x, y) = d_E(\iota(x), \iota(x)), \forall x, y \in \mathbb{R}^D\}.$$

Note each ι can be naturally extended to a transformation on point clouds: $\iota(X) \triangleq \{\iota(x) : x \in X\}$. Let ζ be a error function in the sense that $\zeta(X, Y)$ measures the "difference" between X and Y. Given two point clouds X and Y, the ICP algorithm searches the Euclidean transformation that gives the minimal error value:

$$\iota_{X,Y} = \operatorname*{argmin}_{\iota \in E(\mathbb{R}^D)} \zeta(X, \iota(Y)).$$

Here $\iota_{X,Y}$ is Euclidean transformation found by ICP algorithm for point clouds X and Y. Define the ICP discrepancy between point clouds X and Y (not necessarily a distance) as

$$d_{\mathrm{ICP}}(X, Y) = d_W(X, \iota_{X,Y}(Y)),$$

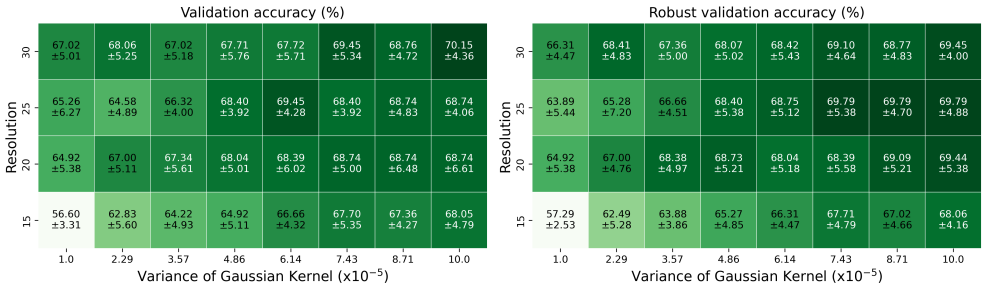

Figure 18: Right: test accuracy of the convolutional neural network (CNN) trained with PIs produced by different parameter settings. Left: robust test accuracy of the convolutional neural network (CNN) trained with PIs produced by different parameter settings.

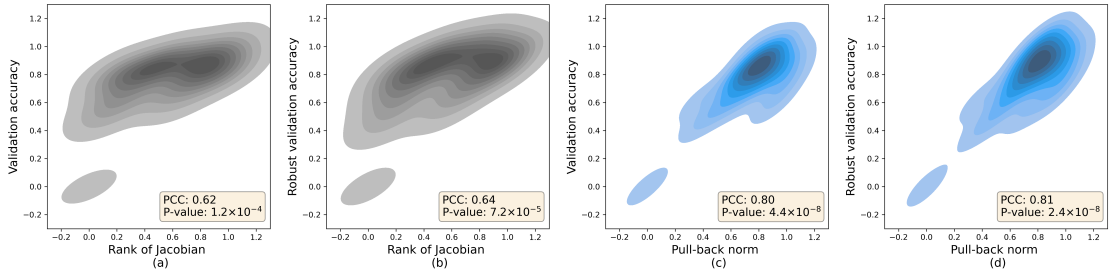

Figure 19: Gaussian kernel density estimation of the joint distribution of four pairs of variables: (a) Jacobian rank vs. validation accuracy; (b) Jacobian rank vs. robust validation accuracy; (c) pull-back norms vs. validation accuracy; and (d) pull-back norms vs. robust validation accuracy, where the downstream models are chosen as convolutional neural networks.

where d_W is Wasserstein distance between point clouds. Let U_X be a neighborhood of X and ξ_X a coordinate map $\xi_X : U_X \to \mathbb{R}^m$. Estimate the gradient vector field for binary categorical feature ρ as follows (see an illustration in Figure 17):

$$\overline{\nabla \rho(X)} = \xi_X(\iota_{X,X'}(X')) - \xi_X(X), \quad X' = \underset{Y \in \{Z \in \mathcal{D} : \rho(Z) \neq \rho(X)\}}{\operatorname{argmin}} d_{\mathrm{ICP}}(X, Y), \quad \forall X \in \mathcal{D},$$

The implementation utilized python library Open3D (Zhou et al., 2018) version 0.17.0 and POT (Flamary et al., 2021) version 0.9.0.

Downstream tasks and performance In Section 5, we fed PIs produced by different parameter settings to logistic regression models to predict the *sex* feature. Specifically, we normalize the PIs such that the pixel values range within $[0, 1]$. We implemented logistic regression models using Scikit-learn (Pedregosa et al., 2011) version 1.2.2 with default hyperparameters. When evaluating the model, we use a 7-fold cross validation. And for robust evaluation, we add identically and independent distributed Gaussian noise with variance 10^{-2} to each coordinate of each point in input point clouds.

Here, we also include the results for a convolutional neural network (CNN) as the downstream model. The CNN model takes PIs as inputs, then begins with a convolutional layer with 32 filters and a 3×3 kernel, followed by a ReLU activation function. Then max pooling with a pool size of 2×2 is applied. Subsequently, another convolutional layer with 64 filters and a 3×3 kernel is added, also followed by a ReLU activation and max pooling. The resulting outputs are then flattened and passed through a fully connected layer with 64 neurons and ReLU activation. Finally, a single neuron with a sigmoid activation function is used for binary classification. To train the model, we employ Adam optimizer with the cross-entropy loss function. The implementation utilized Tensorflow version 2.12.0.

We represent the downstream performance of the CNN model in Figure 18. Shown in Figure 19 is the kernel density estimation of the joint distribution of four pairs of variables: Jacobian rank vs. validation accuracy, Jacobian rank vs. robust validation accuracy, pull-back norm vs. validation accuracy, and pull-back norm vs. robust validation accuracy. Additionally, the Pearson correlation coefficient and p-value of a two-sided test are presented at the lower right corner of each point in Figure 19. We observe that the correlation between pull-back norms and downstream performances remains significant.

We conjecture that complex models, such as CNNs, are able to obtain good downstream performance even if the average pull-back norm is low, so long as it is not zero. Intuitively, when the feature information is indeed contained in the encoded representation but is not significantly pronounced, a simple model may not be able to extract this information, but a complex model, along with appropriate training techniques, can still learn to extract and utilize this information.