RipsNet: a general architecture for fast and robust estimation of the persistent homology of point clouds

Felix Hensel, Marc Glisse, Frédéric Chazal, Thibault de Surrel, Mathieu Carriere, Théo Lacombe, Hiroaki Kurihara, Yuichi Ike

▶ **To cite this version:**

Felix Hensel, Marc Glisse, Frédéric Chazal, Thibault de Surrel, Mathieu Carriere, et al.. RipsNet: a general architecture for fast and robust estimation of the persistent homology of point clouds. Proceedings of Machine Learning Research, 2022, Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022, 196, pp.96-106. hal-03867083

HAL Id: hal-03867083
https://inria.hal.science/hal-03867083

Submitted on 24 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

RIPSNET: A GENERAL ARCHITECTURE FOR FAST AND ROBUST ESTIMATION OF THE PERSISTENT HOMOLOGY OF POINT CLOUDS

Felix Hensel*, **Marc Glisse & Frédéric Chazal**
Université Paris-Saclay, CNRS, Inria,
Laboratoire de Mathématiques d'Orsay, France

Thibault de Surrel* **& Mathieu Carrière***
Université Côte d'Azur, Inria, France

Théo Lacombe
LIGM, Laboratoire Informatique Gaspard Monge
Univ Gustave Eiffel, CNRS, LIGM,
F-77454 Marne-la-Vallée, France

Hiroaki Kurihara
Fujitsu Ltd., Kanagawa, Japan

Yuichi Ike
Graduate School of Information Science and Technology
The University of Tokyo, Japan

ABSTRACT

The use of topological descriptors in modern machine learning applications, such as persistence diagrams (PDs) arising from Topological Data Analysis (TDA), has shown great potential in various domains. However, their practical use in applications is often hindered by two major limitations: the computational complexity required to compute such descriptors exactly, and their sensitivity to even low-level proportions of outliers. In this work, we propose to bypass these two burdens in a data-driven setting by entrusting the estimation of (vectorization of) PDs built on top of point clouds to a neural network architecture that we call *RipsNet*. Once trained on a given data set, RipsNet can estimate topological descriptors on test data very efficiently with generalization capacity. Furthermore, we prove that RipsNet is robust to input perturbations in terms of the 1-Wasserstein distance, a major improvement over the standard computation of PDs that only enjoys Hausdorff stability, yielding RipsNet to substantially outperform exactly-computed PDs in noisy settings. We showcase the use of RipsNet on both synthetic and real-world data. Our implementation will be made freely and publicly available as part of the open-source library `Gudhi`[1].

1 INTRODUCTION

The knowledge of topological features (such as connected components, loops, and higher dimensional cycles) that are present in data sets provides a better understanding of their structural properties at multiple scales, and can be leveraged to improve statistical inference and prediction. Topological Data Analysis (TDA) is the branch of data science that aims to detect and encode such topological features, in the form of *persistence diagrams* (PD). A PD is a (multi-)set of points D in $\mathbb{R}^2$, in which each point $p \in D$ corresponds to a topological feature of the data whose size is encoded by its coordinates. PDs are descriptors of a general nature and allow flexibility in their computation. As such, they have been successfully applied to many different areas of data science, including, e.g., material science (Buchet et al., 2018), genomic data (Cámara, 2017), and 3D-shapes (Li et al., 2014). In the present work, we focus on PDs stemming from point cloud data, referred to as *Rips PDs*, which find natural use in shape analysis (Chazal et al., 2009; Gamble & Heo, 2010) but also in other domains such as time series analysis (Perea & Harer, 2015; Pereira & de Mello, 2015; Umeda, 2017),

*These authors contributed equally to the work.

[1] `https://gudhi.inria.fr/`

or in the study of the behavior of deep neural networks (Guss & Salakhutdinov, 2018; Naitzat et al., 2020; Birdal et al., 2021).

A drawback of Rips PDs computed on large point clouds is their computational cost. Furthermore, even though these topological descriptors enjoy stability properties with respect to the input point cloud in the Hausdorff metric (Chazal et al., 2014), they are fairly sensitive to perturbations: moving a single point in an arbitrarily large point cloud can alter the resulting Rips PD substantially.

In addition, the lack of linear structure (such as addition and scalar multiplication) of the space of PDs hinder the use of PDs in standard machine learning pipelines, which are typically developed to handle inputs belonging to a finite dimensional vector space. This burden motivated the development of *vectorization methods*, which allow to map PDs into a vector space while preserving their structure and interpretability. Vectorization methods can be divided into two classes, *finite-dimensional embeddings* (Bubenik, 2015; Adams et al., 2017; Carrière et al., 2015; Chazal et al., 2015; Kališnik, 2018), turning PDs into elements of Euclidean space $\mathbb{R}^d$, and *kernels* (Carrière et al., 2017; Kusano et al., 2016; Le & Yamada, 2018; Reininghaus et al., 2015), that implicitly map PDs to elements of infinite-dimensional Hilbert spaces.

In this work, we propose to overcome the previous limitations of Rips PDs, by learning their finite-dimensional embeddings directly from the input point cloud data sets with neural network architectures. This approach allows not only for a much faster computation time, but also for increased robustness of the topological descriptors.

Contributions. More specifically, our contributions in this work are summarized as follows.

- We introduce RipsNet, a DeepSets-like architecture capable of learning finite-dimensional embeddings of Rips PDs built on top of point clouds.
- We study the robustness properties of RipsNet. In particular, we prove that for a given point cloud X, perturbing a proportion $\lambda \in (0, 1)$ of its points can only change the output of RipsNet by $\mathcal{O}(\lambda)$, while the exact persistence diagram may change by a fixed positive quantity even in the regime $\lambda \to 0$.
- We experimentally showcase how RipsNet can be trained to produce fast, accurate, and useful estimations of topological descriptors. In particular, we observe that using RipsNet outputs instead of exact PDs yields better performances for classification tasks based on topological properties.

Related work. Our RipsNet architecture is directly based on *DeepSets* (Zaheer et al., 2017), a particular case of equivariant neural network (Cohen, 2021) designed to handle point clouds as inputs. Namely, DeepSets essentially consists of processing a point cloud $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ via

$$X \mapsto \phi_2 \left(\mathbf{op}(\{\phi_1(x_i)\}_{i=1}^n) \right), \tag{1}$$

where $\mathbf{op}$ is a *permutation invariant operator* on sets (such as sum, mean, maximum, etc.) and $\phi_1 : \mathbb{R}^d \to \mathbb{R}^{d'}$ and $\phi_2 : \mathbb{R}^{d'} \to \mathbb{R}^{d''}$ are parametrized maps (typically encoded by neural networks) optimized in the training phase. Eq. (1) makes the output of DeepSets architectures invariant to permutations, a property of Rips PDs that we want to reproduce in RipsNet. Note also that DeepSets have also been used recently in TDA for defining the PersLay layer (Carrière et al., 2020) for PDs whose is to learn how to compute the best PD embedding w.r.t. a given data science task. Even though PersLay and RipsNet have similar architectures, their goals are different, as RipsNet does not aim at learning the best PD embedding, but at approximating the PD computation itself.

There exist a few works attempting to compute or estimate (vectorizations of) PDs through the use of neural networks. In (Som et al., 2020), the authors propose a convolutional neural network (CNN) architecture to estimate persistence images (see below) computed on 2D-images. Similarly, in (Montúfar et al., 2020), the authors provide an experimental overview of specific PD features (such as, e.g., their tropical coordinates (Kališnik, 2019)) that can be learned using a CNN, when PDs are computed on top of 2D-images. On the other hand, RipsNet is designed to handle the (arguably harder) situation where input data are point clouds of arbitrary cardinality instead of 2D-images (i.e., vectors). Very recently, (Yan et al., 2022) addressed a similar question in the context of graphs. Fairly similar to our approach, the recent work (Zhou et al., 2021) also aims at learning to compute topological descriptors on top of point clouds via a neural network. However, note that

our methodology is quite different: while our approach based on DeepSets architectures allows to process point clouds directly, the approach proposed in (Zhou et al., 2021) requires the user to equip the point clouds with graph structures (that depend on hyper-parameters mimicking Rips filtrations) and is suited to 3D point clouds only. Furthermore, a key difference between our approach and the aforementioned works is that we provide a theoretical study of our model that provides insights on its behavior, particularly in terms of robustness to noise (which shows in our experimental results, see Section 4.2 and Table 1), while the other works are mostly experimental.

Supplementary material. Proofs of theoretical results, as well as additional experiments on UCR-timeseries data and additional persistence diagram vectorization, can be found in (de Surrel et al., 2022). Code will be incorporated in future versions of `Gudhi`, a version to reproduce the experiments is provided at `https://github.com/hensel-f/ripsnet`.

2 BACKGROUND

In this section, we recall the basics of persistence theory. We refer the interested reader to (Cohen-Steiner et al., 2009; Edelsbrunner & Harer, 2010; Oudot, 2015) for a thorough treatment.

Persistence diagrams. Let $\mathcal{X}$ be a topological space, and $f\colon \mathcal{X} \to \mathbb{R}$ a real-valued continuous function. The α-*sublevel set* of $(\mathcal{X}, f)$ is defined as $\mathcal{X}_\alpha = \{x \in \mathcal{X} \,:\, f(x) \leq \alpha\}$. Increasing α from $-\infty$ to $+\infty$ yields an increasing nested sequence of sublevel sets, called the *filtration* induced by f. It starts with the empty set and ends with the entire space $\mathcal{X}$. Ordinary persistence keeps track of the times of appearance and disappearance of topological features (connected components, loops, cavities, etc.) in this sequence. For instance, one can store the value α_b, called the *birth time*, for which a new connected component appears in X_{α_b}. This connected component eventually merges with another one for some value $\alpha_d \geq \alpha_b$, which is stored as well and called the *death time*. One says that the component *persists* on the corresponding interval $[\alpha_b, \alpha_d]$. Similarly, we save the $[\alpha_b, \alpha_d]$ values of each loop, cavity, etc. that appears in a specific sublevel set X_{α_b} and disappears (gets "filled") in X_{α_d}. This family of intervals is called the barcode, or *persistence diagram* (PD), of $(\mathcal{X}, f)$, and can be represented as a multiset (i.e., elements are counted with multiplicity) of points supported on the open half-plane $\{(\alpha_b, \alpha_d) \in \mathbb{R}^2 \,:\, \alpha_b < \alpha_d\} \subset \mathbb{R}^2$. The information of connected components, loops, and cavities is represented in PDs of dimension 0, 1, and 2, respectively.

Filtrations for point clouds. Let $X = \{x_1, \ldots, x_N\}$ be a finite point cloud in $\mathcal{X} = \mathbb{R}^d$, and $\alpha \geq 0$. Let $f_X\colon \mathcal{X} \to \mathbb{R}, v \mapsto \min_{x \in X} \|v - x\|$ denote the distance of $v \in \mathbb{R}^d$ to X. In this case, the sublevel set $\mathcal{X}_\alpha$ is given by the union of d-dimensional closed balls of radius α centered at x_i $(i = 1, \ldots, N)$. From this filtration, different types of PDs can be built, called Čech, Rips, and alpha filtrations, which can be considered as being equivalent for the purpose of this work (in particular, RipsNet can be used seamlessly with any of these choices, see (de Surrel et al., 2022)) Due to its computational efficiency in low-dimensional settings, we use the alpha filtration in our numerical experiments.

Metrics between persistence diagrams. The space of PDs can be equipped with a parametrized metric d_s, $1 \leqslant s \leqslant \infty$ which is rooted in algebraic considerations and whose proper definition is not required in this work. In the particular case $s = \infty$, this metric will be referred to as the *bottleneck* distance between PDs. Of importance is, that the space of PDs $\mathcal{D}$ equipped with such metrics lacks linear (Hilbert; Euclidean) structure (Carriere & Bauer, 2018; Bubenik & Wagner, 2020).

The lack of linear structure of the metric space $(\mathcal{D}, d_s)$ prevents a faithful use of PDs in standard machine learning pipelines, for which one typically requires inputs to belong to a finite-dimensional vector space. A natural workaround is thus to seek for a *vectorization* of PDs (PV), that is a map $\phi : (\mathcal{D}, d_s) \to (\mathbb{R}^d, \|\cdot\|)$ for some dimension d. Provided the map ϕ satisfies suitable properties (e.g., being Lipschitz, injective, etc.), one can turn a sample of diagrams $\{D_1, \ldots, D_n\} \subset \mathcal{D}$ into vectors $\{\phi(D_1), \ldots, \phi(D_n)\} \subset \mathbb{R}^d$ which can subsequently be used in machine learning.

Various vectorization techniques, with success in applications, have been proposed (Carrière et al., 2015; Chazal et al., 2015; Kališnik, 2018). In this work, we focus on the *persistence image* (Adams et al., 2017) —though the approach developed in this work adapts faithfully to any other vectorization as persistence landscapes (Bubenik, 2015), as detailed in (de Surrel et al., 2022).

Persistence images (PI). Given a PD D, computing its persistence image essentially boils down to putting a Gaussian $g_u(z) \coloneqq \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|z-u\|^2}{2\sigma^2}\right)$, with fixed variance σ^2, on each of its points u and weighing it by a piecewise differentiable function $w : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ (typically a function of the distance of u to the diagonal in $\mathbb{R}^2$) and then discretizing the resulting surface on a fixed grid to obtain an image. Formally, one starts by rotating the diagram D via the map $T\colon \mathbb{R}^2 \to \mathbb{R}^2, (b, d) \mapsto (b, d-b)$. The *persistence surface* of D is defined as $\rho_D(z) \coloneqq \sum_{u \in T(D)} w(u) g_u(z)$, where w satisfies $w(x, 0) = 0$. Now, given a compact subset $A \subset \mathbb{R}^2$ partitioned into domains $A = \bigsqcup_{i=1}^{k} P_i$—in practice a rectangular grid regularly partitioned in $(n \times n)$ pixels—we set $I(\rho_D)_P \coloneqq \int_P \rho_D dz$. The vector $(I(\rho_D)_{P_i})_{i=1}^n$ is the persistence image of D. The transformation $\mathrm{PI}\colon X \mapsto \mathrm{Dgm}(X) \mapsto I(\rho_{\mathrm{Dgm}(X)})$ defines a finite-dimensional vectorization as shown in Figure 1.

3 RIPSNET

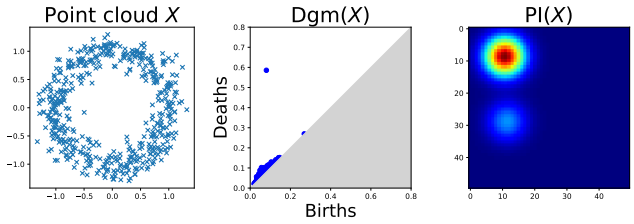

Figure 1: Pipeline to extract PIs from point clouds. The main bump in $\mathrm{PI}(X)$ corresponds to the circle underlying the point cloud X. The smaller bump accounts for smaller loops inferred when filtering X.

Motivation and definition. In the pipeline illustrated in Figure 1, the computation of the PD $\mathrm{Dgm}(X)$ from an input point cloud X is the most complex operation involved: it is both computationally expensive and introduces non-differentiability in the pipeline. Moreover, as detailed in (de Surrel et al., 2022) for the specific case of persistence images (PI), the output vectorization can be highly sensitive to perturbations in the input point cloud X: moving a single point $p_i \in X$ can arbitrarily change $\mathrm{PI}(X)$ even when n is large. This instability can be a major limitation when incorporating persistence vectorizations (PVs) of diagrams in practical applications.

To overcome these difficulties, we propose a way to bypass this computation by designing a neural network architecture that we call *RipsNet* ($\mathbf{RN}$). The goal is to learn a function, denoted by $\mathbf{RN}$ as well, able to reproduce persistence vectorizations for a given distribution of input point clouds $X \sim \mathbf{P}$ after training on a sample $\{X_i\}_{i=1}^n$ with labels being the corresponding vectorizations $\{\mathrm{PV}(X_i)\}_{i=1}^n$.

As $\mathbf{RN}$ takes point clouds $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ of potentially varying sizes as input, it is natural to expect it to be *permutation invariant*. An efficient way to enforce this property is to rely on a DeepSets architecture (Zaheer et al., 2017). Namely, it consists of decomposing the network into a succession of two maps $\phi_1\colon \mathbb{R}^d \to \mathbb{R}^{d'}$ and $\phi_2\colon \mathbb{R}^{d'} \to \mathbb{R}^{d''}$ and a permutation invariant operator $\mathbf{op}$—typically the sum, the mean, or the maximum— $\mathbf{RN}\colon X \mapsto \phi_2\left(\mathbf{op}(\{\phi_1(x)\}_{x \in X})\right)$. For each $x \in X$, the map ϕ_1 provides a representation $\phi_1(x)$; these point-wise representations are gathered into a single one via the permutation invariant operator $\mathbf{op}$, and the map ϕ_2 is finally applied on it.

In practice, ϕ_1 and ϕ_2 are themselves parameterized by neural networks; in this work, we will consider simple feed-forward fully-connected networks (see Section 4 for the architecture hyper-parameters), though more general architectures could be considered. The parameters characterizing ϕ_1 and ϕ_2 are tuned during the training phase, where we minimize the $L2$-loss $\sum_{i=1}^n \|\mathbf{RN}(X_i) - \mathrm{PV}(X_i)\|^2$, over a set of training point clouds $\{X_i\}_i$ with corresponding pre-computed vectorizations $\{\mathrm{PV}(X_i)\}_i$.

Once properly trained (assuming good generalization properties), when extracting topological information of a point cloud, an important advantage of using $\mathbf{RN}$ instead of PV lies in the computational efficiency: while the exact computation of PDs and vectorizations has cubic complexity in the number of simplices in the filtration (which can itself be exponential in the number of points in X), running the forward pass of a trained network is significantly faster, as showcased in Section 4. Moreover, $\mathbf{RN}$ also satisfies some strong robustness properties, as detailed in the following. This yields a substantial advantage over exact PVs when the data contain some perturbations such as noise, outliers, or adversarial attacks.

Wasserstein stability of RipsNet. We now show that RipsNet satisfies robustness properties . A convenient formalism to demonstrate these properties is to represent a point cloud $X = \{x_1, \dots, x_N\}$ by a probability measure $m_X := \frac{1}{N}\sum_{i=1}^N \delta_{x_i}$, where δ_{x_i} denotes the Dirac mass located at $x_i \in \mathbb{R}^d$. Let $\mu(f)$ denote $\int f \mathrm{d}\mu$ for a map f and a probability measure μ. Such measures can be compared by Wasserstein distances $W_p,\ p \geq 1$, which are defined for any two probability measures μ, ν supported on a compact subset $\Omega \subset \mathbb{R}^d$ as $W_p(\mu,\nu) := \left(\inf_\pi \iint \|x-y\|^p \mathrm{d}\pi(x,y)\right)^{\frac{1}{p}}$, where the infimum is taken over measures π, supported on $\mathbb{R}^d \times \mathbb{R}^d$, with marginals μ and ν.

In this section, we set $\mathbf{op}$ as the mean operator: $\mathbf{op}(\{y_1, \dots, y_N\}) = \frac{1}{N}\sum_{i=1}^N y_i$, and let $\mathbf{RN} = \phi_2 \circ \mathbf{op} \circ \phi_1$, where ϕ_1, ϕ_2 are Lipschitz-continuous maps with Lipschitz constants C_1, C_2 respectively.

Let us first investigate the pointwise stability of $\mathbf{RN}$. If $X = \{x_1, \dots, x_{N-1}, x_N\} \subset \Omega$ and $X' = \{x_1, \dots, x_{N-1}, x'_N\} \subset \Omega$, then $W_1(m_X, m_{X'}) \leq \frac{1}{N}\|x_N - x'_N\|$. Therefore, moving a single point x_N of X to another location x'_N changes the W_1 distance between the two measures by at most $\mathcal{O}(1/N)$. More generally, moving a fraction $\lambda \in (0,1)$ of the points in X affects the Wasserstein distance in $\mathcal{O}(\lambda)$. From this, we can derive a first stability result for RipsNet.

Proposition 3.1. *For any two point clouds X, Y, and any $p \geq 1$, one has*

$$\|\mathbf{RN}(X) - \mathbf{RN}(Y)\| \leq C_1 C_2 \cdot W_1(m_X, m_Y) \leq C_1 C_2 \cdot W_p(m_X, m_Y).$$

In particular, this result implies that moving a small proportion of points λ in Ω in a point cloud X does not affect the output of $\mathbf{RN}$ by much. We refer to it as a "pointwise stability" result in the sense that it describes how $\mathbf{RN}$ is affected by perturbations of a fixed point cloud X. Note that in contrast, Rips PDs, as well as their vectorizations, are not robust to such perturbations: moving a single point of X, even in the regime $\lambda \to 0$, may change the resulting PD by a fixed positive amount, preventing a similar result to hold for PVs.

The pointwise stability result of Proposition 3.1 can now be used to obtain a good theoretical understanding on how RipsNet behaves in practical learning settings. For this, we consider the following model: let P be a law on some compact set $\Omega \subset \mathbb{R}^d$, fix $N \in \mathbb{N}$, and let $\mathbf{P}$ denote $P^{\otimes N}$, that is, $X \sim \mathbf{P}$ is a random point cloud $X = \{x_1, \dots, x_N\}$ where the x_i's are i.i.d. $\sim P$.

In practice, given a training sample $X_1, \dots, X_n \sim \mathbf{P}$, RipsNet is trained to minimize the empirical risk $\widehat{\mathcal{R}}_n$ which, hopefully, yields a small theoretical risk $\mathcal{R}$, where

$$\widehat{\mathcal{R}}_n := \frac{1}{n}\sum_{i=1}^n \|\mathbf{RN}(X_i) - \mathrm{PV}(X_i)\| \quad \text{and} \quad \mathcal{R} := \int \|\mathbf{RN}(X) - \mathrm{PV}(X)\| \mathrm{d}\mathbf{P}(X).$$

Remark 3.2. The question to know whether "$\widehat{\mathcal{R}}_n$ small $\Rightarrow$ $\mathcal{R}$ small" is related to the capacity of RipsNet to generalize properly. Providing a theoretical setting where such an implication should hold is out of the scope of this work, but can be checked empirically by looking at the performances of RipsNet on validation sets.

We now consider a standard contamination model (Huber, 1964): given a point cloud $X \sim \mathbf{P}$, we randomly replace a fraction $\lambda = \frac{N-K}{N} \in (0,1)$ of its points[2] by corrupted observations distributed with respect to some law Q. Let $Y \sim Q^{\otimes N-K} =: \mathbf{Q}$ and $F(X,Y)$ denote this corrupted point cloud.

Lemma 3.3. *Let $C(P,Q) := \mathbb{E}_{P\otimes Q}[\|x-y\|]$. Then, $\mathbb{E}_{\mathbf{P}\otimes\mathbf{Q}}[W_1(F(X,Y),X)] \leq \lambda C(P,Q)$. In particular, if P, Q are supported on a compact set $\Omega \subset \mathbb{R}^d$ with diameter $\leq L$, the bound is λL.*

This result is similar to the well-known robustness properties of the Wasserstein distance for contamination models, as detailed in, e.g., (Staerman et al., 2021). We can now state our main result.

Proposition 3.4. *One has $\int \|\mathbf{RN}(F(X,Y)) - \mathrm{PV}(X)\| \mathrm{d}\mathbf{P}(X)\mathrm{d}\mathbf{Q}(Y) \leq \lambda C_1 C_2 \cdot C(P,Q) + \mathcal{R}$. In particular, if P, Q are supported on a compact subset of $\mathbb{R}^d$ with diameter $\leq L$, one has*

$$\int \|\mathbf{RN}(F(X,Y)) - \mathrm{PV}(X)\| \mathrm{d}\mathbf{P}(X)\mathrm{d}\mathbf{Q}(Y) \leq \mathcal{O}(\lambda + \mathcal{R}).$$

Therefore, if RipsNet achieves a low theoretical test risk ($\mathcal{R}$ small) and only a small proportion λ of points is corrupted, RipsNet will produce outputs similar, in expectation, to the persistence vectorizations $\mathrm{PV}(X)$ of the clean point cloud.

[2] As the x_i's are i.i.d., we may assume without loss of generality that the last $N-K$ points are replaced.

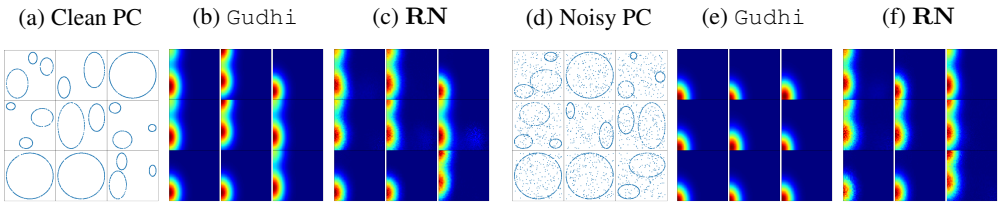

Figure 3: Point clouds, Gudhi, and $\mathbf{RN}_{\text{synth}}$ outputs on clean (a,b,c) and noisy input data (d,e,f).

4 Numerical Experiments

In this section, we illustrate the properties of our general architecture **RN** presented in the previous sections. The approach we use is the following: we first train a **RN** architecture on a training data set Tr_1, comprised of point clouds PC_1 with their corresponding labels L_1 and persistence vectorizations PV_1 computed via alpha filtrations. Note that this training step does not require the labels L_1 of the point clouds since the targets are the persistence vectorizations PV_1. Then, we use both **RN** and Gudhi to compute the persistence vectorizations of three data sets: a second training data set $\mathrm{Tr}_2 = (\mathrm{PC}_2, \mathrm{L}_2)$, a test data set $\mathrm{Te} = (\mathrm{PC}, \mathrm{L})$, and a noisy test data set $\widetilde{\mathrm{Te}} = (\widetilde{\mathrm{PC}}, \widetilde{\mathrm{L}})$ (as per our noise model explained in Section 3). All three data sets are comprised of labeled point clouds only. At this stage, we also measure the computation time of **RN** and Gudhi for generating outputs.

To obtain quantitative scores, we finally train two machine learning classifiers: one on the labeled **RN** PDs from Tr_2, and the other on the labeled Gudhi PDs from Tr_2, which we call $\mathrm{Cl}_{\mathbf{RN}}$ and $\mathrm{Cl}_{\texttt{Gudhi}}$, respectively. The classifiers $\mathrm{Cl}_{\mathbf{RN}}$ and $\mathrm{Cl}_{\texttt{Gudhi}}$ are then evaluated on the test PDs computed with **RN** and Gudhi, respectively, on both Te and $\widetilde{\mathrm{Te}}$. See Figure 2 for a schematic overview.

We also generate scores using alphaDTM-based filtrations (Anai et al., 2019) computed with the Python package Velour with parameters $m = 5\%$ (respectively $m = 0.75\%$ for the 3D-shape experiments), $p = 2$, in the exact same way as we did for Gudhi. We let $\texttt{Gudhi}^{\mathrm{DTM}}$ and $\mathrm{Cl}_{\texttt{Gudhi}^{\mathrm{DTM}}}$ denote the corresponding model and classifier, respectively. Note that alphaDTM-based filtrations (which are modifications of the alpha filtrations in the presence of noise) usually require manual tuning, which, contrary to **RN** parameters, cannot be optimized during training. Hence, in our experiments, we manually tuned those parameters. Finally, note that we also added some (non-topological) baselines in each of our experiments to provide a sense of what other methods are capable of in terms of accuracy scores. However, our main purpose is to show that **RN** can provide a much faster and more noise-robust alternative to Gudhi: the most important comparison is between **RN** and both Gudhi and $\texttt{Gudhi}^{\mathrm{DTM}}$.

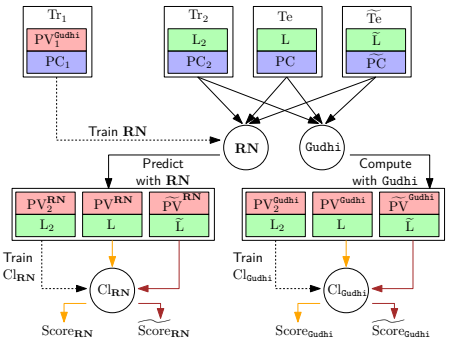

Figure 2: Scheme of our experimental setup.

In the following, we apply our experimental setup to two types of data. First, we focus on synthetic data generated by a simple generative model. Next, we consider 3D-shape data from the ModelNet10 data set (Wu et al., 2015). The computations were run on a computing cluster on 4 Xeon SP Gold 2.6GHz CPU cores with 8GB of RAM per core. Accuracy scores (of all methods) are provided in Table 1, and running times (of topological methods) are provided in Table 2.

4.1 Synthetic Data

Data set. Our synthetic data set consists of samplings of unions of circles in the Euclidean plane $\mathbb{R}^2$. These unions are made of either one, two, or three circles, and we use the number of circles as the labels of the point clouds. Each point cloud has $N = 600$ points, and $N - K = 200$ corrupted points, i.e., $\lambda = 1/3$, when noise is added.

We train a RipsNet architecture $\mathbf{RN}_{\text{synth}}$ on a data set Tr_1 of 3300 point clouds, using 3000 point clouds for training and 300 as a validation set. The persistence diagrams PD_1 were computed with

alpha filtrations in dimension 1 with Gudhi, and then vectorized into normalized persistence images of resolution 50×50, leading to 2500-dimensional vectors. The image bandwidth was estimated as the 0.2-quantile of all pairwise distances between the birth-persistence transforms of the PD points, and the image weight was defined as $10 \cdot \tanh(y - x)$.

Our architecture $\mathbf{RN}_{\text{synth}}$ is structured as follows. The permutation invariant operator is $\mathbf{op} = \text{sum}$, ϕ_1 is made up of three fully connected layers of 30, 20, and 10 neurons with ReLU activations, ϕ_2 consists of three fully connected layers of 50, 100, and 200 neurons with ReLU activations, and a last layer with sigmoid activation. We used the mean squared error (MSE) loss with Adamax optimizer with $\varepsilon = 5 \cdot 10^{-4}$, and early stopping after 200 epochs with less than 10^{-5} improvement.

Synth. Data	$\text{Cl}^{\text{XGB}}_{\texttt{Gudhi}}$	$\text{Cl}^{\text{XGB}}_{\texttt{Gudhi}^{\text{DTM}}}$	$\text{Cl}^{\text{XGB}}_{\mathbf{RN}}$	$\mathbf{DS}_1$	$\mathbf{DS}_2$
PI	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.1}$	81.6 ± 5.3	66.4 ± 2.3	66.0 ± 2.4
$\widetilde{\text{PI}}$	33.3 ± 0.0	65.0 ± 1.3	$\mathbf{77.4 \pm 4.4}$	66.8 ± 1.0	66.6 ± 2.3
λ (%)	$\text{Cl}^{\text{NN}}_{\texttt{Gudhi}}$	$\text{Cl}^{\text{NN}}_{\texttt{Gudhi}^{\text{DTM}}}$	$\text{Cl}^{\text{NN}}_{\mathbf{RN}}$	$\text{Cl}^{\text{NN}}_{\text{TopologyNet}}$	pointnet
0	30.4 ± 4.0	30.9 ± 2.0	53.9 ± 2.4	67.7 ± 3.4	$\mathbf{81.6 \pm 1.1}$
2	30.3 ± 3.2	31.0 ± 2.7	53.2 ± 2.5	64.7 ± 3.7	$\mathbf{74.5 \pm 1.6}$
5	29.9 ± 4.0	31.0 ± 2.7	55.1 ± 3.3	60.8 ± 3.9	$\mathbf{63.4 \pm 1.6}$
10	25.2 ± 3.2	29.5 ± 3.1	51.0 ± 2.1	$\mathbf{52.4 \pm 2.5}$	50.6 ± 1.5
15	22.9 ± 4.6	25.7 ± 3.1	46.9 ± 3.0	$\mathbf{47.0 \pm 3.6}$	44.9 ± 1.7
25	14.4 ± 4.0	18.1 ± 2.6	$\mathbf{42.6 \pm 2.5}$	36.9 ± 3.6	11.0 ± 0.2
50	14.0 ± 3.4	13.1 ± 1.9	$\mathbf{31.6 \pm 3.3}$	26.9 ± 2.7	10.9 ± 0.0

Table 1: Accuracy scores of classifiers trained on Gudhi, $\texttt{Gudhi}^{\text{DTM}}$ and $\mathbf{RN}$ PVs generated from several data sets. The highest accuracy of the topology-based classifiers (middle) is highlighted in red, and the highest accuracy over all models in bold font.

Finally, we evaluate $\mathbf{RN}_{\text{synth}}$ and Gudhi using default XGBoost classifiers $\text{Cl}^{\text{XGB}}_{\mathbf{RN}_{\text{synth}}}$ and $\text{Cl}^{\text{XGB}}_{\texttt{Gudhi}}$ from Scikit-Learn, trained on a data set Tr_2 of 3000 point clouds and tested on a clean test set Te and a noisy test set $\widetilde{\text{Te}}$ of 300 point clouds each. We compare it against two DeepSets baselines trained directly on the point clouds: $\mathbf{DS}_1$ resp. $\mathbf{DS}_1$ with fully connected layers of sizes (50, 30, 10, 3) resp. (50, 3). Both architectures have ReLU activations, except for the last layer, $\mathbf{op} = \text{sum}$, default Adam optimizer, cross entropy loss, and early stopping after 200 epochs with less than 10^{-4} improvement.

Results. We show a few point clouds of Te and $\widetilde{\text{Te}}$, as well as their corresponding vectorized PDs and estimated vectorizations with $\mathbf{RN}_{\text{synth}}$, in Figure 3. Accuracies and running times (averaged over 10 runs) are given in the first four rows of Table 1 and the first two rows of Table 2.

As one can see from Figure 3 and the the first four rows of Table 1, $\mathbf{RN}_{\text{synth}}$ manages to learn features that are visually similar to the PD vectorizations generated by Gudhi, with comparable accuracies on clean data. However, features generated by $\mathbf{RN}_{\text{synth}}$ are much more robust and thus contain complementary information to the purely topological features; even though $\text{Cl}^{\text{XGB}}_{\texttt{Gudhi}}$ and $\text{Cl}^{\text{XGB}}_{\texttt{Gudhi}^{\text{DTM}}}$ see their accuracies largely decrease when noise is added, $\text{Cl}^{\text{XGB}}_{\mathbf{RN}_{\text{synth}}}$ accuracy only decreases slightly. Note that the decrease of accuracy is more moderate for $\text{Cl}^{\text{XGB}}_{\texttt{Gudhi}^{\text{DTM}}}$ since $\texttt{Gudhi}^{\text{DTM}}$ is designed to be more robust to outliers. Also, one can see from Table 2 that running times are much more favorable for $\mathbf{RN}_{\text{synth}}$, with an improvement of 2 (resp. 3) orders of magnitude over Gudhi (resp. $\texttt{Gudhi}^{\text{DTM}}$).

Data	Gudhi (s)	$\texttt{Gudhi}^{\text{DTM}}$ (s)	$\mathbf{RN}$ (s)
PI	69.5 ± 3.1	173.7 ± 13.3	$\mathbf{0.4 \pm 0.0}$
$\lambda = 2\%$	118.4 ± 4.7	178.5 ± 8.1	$\mathbf{0.2 \pm 0.0}$
$\lambda = 5\%$	117.8 ± 4.5	180.0 ± 9.2	$\mathbf{0.2 \pm 0.0}$

Table 2: Running times of topological methods.

4.2 3D-SHAPE DATA AND RIPSNET GENERALIZATION CAPACITY

Data set. We also run experiments on Princeton's ModelNet10 data set, comprised of 10 classes of 3D meshes. In order to obtain point clouds in $\mathbb{R}^3$, we sample 1024 points on the surfaces of the 3D meshes, which are then centered and normalized to be contained in the unit sphere. We have $2393/598$ and $406/229$ training and test samples at our disposal for the training of RipsNet, and for the training of neural net classifiers, respectively. The architecture of these classifiers (NN) is very simple, consisting of only two consecutive fully connected layers of 100 and 50 neurons.

For the sake of simplicity, we focus on persistence images of resolution 25×25 with weight function $(y - x)^2$ only, and consider the combination of PDs of dimension 0 and 1. The vectorization parameters were estimated as in Section 4.1 . The final RipsNet architecture, using $\mathbf{op} = \text{mean}$, was found via a 3-fold cross-validation over several models, and again optimized with Adam optimizer.

To showcase the robustness of $\mathbf{RN}$, we introduce noise fractions λ in $\{0.02, 0.05, 0.1, 0.25, 0.5\}$. In addition, we compare $\mathbf{RN}$ to TopologyNet, a recently introduced model (Zhou et al., 2021) which predicts topological descriptors on 3D point clouds, but relies on building a k-NN graph on top of each point cloud (we used $k = 16$ in this work) and hence is more task-specific and less general than RipsNet in architectural design. TopologyNet architecture and training parameters are detailed in (Zhou et al., 2021). As an independent non-topological baseline, we also compared RipsNet to `pointnet` (Qi et al., 2017), a standard neural network for 3D point cloud classification.

Results. In addition to the neural net classifiers, we also train XGBoost classifiers, the results of which, as well as additional results of the neural net classifiers and running times, are reported in the last parts of Table 1 and Table 2. Due to class imbalances, the accuracy of the best possible *constant* classifier is 22.2%. As the sampling of the point clouds, as well as the addition of noise, are random, we repeat this process 10 times in total. Subsequently, we train the classifiers on each of these data sets, without retraining $\mathbf{RN}$, and report the mean and standard deviation. In our experiments, the average running time of TopologyNet (for approximating PDs and their vectorizations) is $28.0 \pm 0.2s$. Hence, the running times of both `Gudhi` and TopologyNet are outperformed by $\mathbf{RN}$ by three and two orders of magnitude respectively, see Table 2. The accuracy of $\mathrm{Cl}^{\mathrm{NN}}_{\mathbf{RN}}$ substantially surpasses those of $\mathrm{Cl}^{\mathrm{NN}}_{\texttt{Gudhi}}$ and $\mathrm{Cl}^{\mathrm{NN}}_{\texttt{Gudhi}^{\mathrm{DTM}}}$ for all values of λ and remains much more robust for high levels of noise. RipsNet and TopologyNet exhibit similar behaviour: TopologyNet performs slightly better in low noise settings while RipsNet is more robust to noise. However, contrary to RipsNet, TopologyNet does not enjoy any theoretical robustness result and indeed deteriorates significantly for higher noise levels. TopologyNet is designed to handle 3D point clouds and relies on building a k-NN graph on input point clouds, and thus leverages more information, yielding better accuracy in uncorrupted settings, however yielding an increased estimation time of about a factor 140 over RipsNet. As for `pointnet`, for $\lambda \geq 0.1$, $\mathrm{Cl}^{\mathrm{NN}}_{\mathbf{RN}}$ surpasses the `pointnet` baseline, whose accuracy decreases sharply for $\lambda \geq 0.25$, at which point $\mathrm{Cl}^{\mathrm{NN}}_{\mathbf{RN}}$ substantially outperforms it.

Generalization capacity to ModelNet40. A final natural question to address is whether RipsNet presents generalization capacity in the following sense: is it capable of producing meaningful topological descriptors on similar—yet unseen—objects to the one it was trained on? To investigate this, we ran the following experiment: after training $\mathbf{RN}$ on `ModelNet10`, we test its performance—along with the one of TopologyNet (Zhou et al., 2021)—on the `ModelNet40` data set; a data set that contains additional types of 3D-shapes. Interestingly, both models were capable of producing good vectorizations on these unseen classes: $\mathbf{RN}$ reaches a MSE of $8 \cdot 10^{-3}$ on this new data set and TopologyNet reaches a MSE of $4 \cdot 10^{-3}$; these quantities being compared to the respective training MSE on `modelNet10` of $4.5 \cdot 10^{-3}$ and $1.9 \cdot 10^{-3}$. See also (de Surrel et al., 2022) for complementary qualitative illustrations of the output produced. While TopologyNet exhibits better quantitative results than RipsNet in this experiment, which is expected as this model takes into account more structure on the input point cloud, this comes at the price of computational efficiency: $\mathbf{RN}$ produces a prediction in 3ms on average, while TopologyNet requires about 780ms to do so.

5 Conclusion

Vectorization of topological features is of central importance for their practical use in machine learning applications. However, the computational complexity of the exact computation of persistence diagrams and their sensitivity to outliers and noise limit their applicability. To address these limitations, we propose RipsNet, a DeepSets-like architecture, which learns to estimate persistence vectorizations of point cloud data, and is theoretically proven to be more robust to outliers and noise than exact vectorizations. Moreover, we substantiate our theoretical findings by numerical experiments on a synthetic data set, as well as on 3D-shape data (and time series data), which exhibit significant improvements in robustness, accuracy scores, and running times.

Several questions remain open for future works. First, $\mathbf{RN}$ is currently limited by its data-driven nature; it requires training beforehand and the architecture should be adapted depending on data sets or tasks. In particular, generalization capacity of $\mathbf{RN}$ may improve by incorporating constraints on the network making it scaling-, translation- and rotation-invariant (e.g. with data augmentation), which we currently handle by manually normalizing the data and PDs. Second, $\mathbf{RN}$ may also be combined with PersLay (Carrière et al., 2020) and parametric families of filtrations, in order to avoid choosing a filtration and an embedding before learning.

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. *Journal of Machine Learning Research*, 18(8), 2017.

Hirokazu Anai, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hiroya Inakoshi, Raphaël Tinarrage, and Yuhei Umeda. DTM-Based Filtrations. In *35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129, pp. 58:1–58:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

Tolga Birdal, Aaron Lou, Leonidas J Guibas, and Umut Simsekli. Intrinsic dimension, persistent homology and generalization in neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

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

Peter Bubenik and Alexander Wagner. Embeddings of persistence diagrams into hilbert spaces. *Journal of Applied and Computational Topology*, 4(3):339–351, 2020.

Mickaël Buchet, Yasuaki Hiraoka, and Ippei Obayashi. Persistent homology and materials informatics. In *Nanoinformatics*, pp. 75–95. Springer-Verlag, 2018.

Pablo Cámara. Topological methods for genomics: present and future directions. *Current Opinion in Systems Biology*, 1:95–101, feb 2017.

Mathieu Carriere and Ulrich Bauer. On the metric distortion of embedding persistence diagrams into separable hilbert spaces. *arXiv preprint arXiv:1806.06924*, 2018.

Mathieu Carrière, Steve Oudot, and Maks Ovsjanikov. Stable topological signatures for points on 3d shapes. In *Computer Graphics Forum*, volume 34, pp. 1–12. Wiley Online Library, 2015.

Mathieu Carrière, Marco Cuturi, and Steve Oudot. Sliced Wasserstein kernel for persistence diagrams. In *International Conference on Machine Learning*, volume 70, pp. 664–673, jul 2017.

Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. PersLay: a neural network layer for persistence diagrams and new graph topological signatures. In *23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020)*, pp. 2786–2796. PMLR, 2020.

Frédéric Chazal, David Cohen-Steiner, Leonidas J Guibas, Facundo Mémoli, and Steve Y Oudot. Gromov-hausdorff stable signatures for shapes using persistence. In *Computer Graphics Forum*, volume 28, pp. 1393–1403. Wiley Online Library, 2009.

Frédéric Chazal, Vin de Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173(1):193–214, 2014.

Frédéric Chazal, Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, and Larry Wasserman. Stochastic convergence of persistence landscapes and silhouettes. *Journal of Computational Geometry*, 6(2):140–161, 2015.

Taco Cohen. Equivariant convolutional networks. 2021.

David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, feb 2009.

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, 2022. URL `https://arxiv.org/abs/2202.01725`.

Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Society, 2010.

Jennifer Gamble and Giseon Heo. Exploring uses of persistent homology for statistical analysis of landmark-based shape data. *Journal of Multivariate Analysis*, 101(9):2184–2199, 2010.

William H Guss and Ruslan Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *arXiv preprint arXiv:1802.04443*, 2018.

Peter Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35 (1):73–101, 1964.

Sara Kališnik. Tropical coordinates on the space of persistence barcodes. *Foundations of Computational Mathematics*, pp. 1–29, jan 2018.

Sara Kališnik. Tropical coordinates on the space of persistence barcodes. *Foundations of Computational Mathematics*, 19(1):101–129, 2019.

Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted Gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, volume 48, pp. 2004–2013, jun 2016.

Tam Le and Makoto Yamada. Persistence Fisher kernel: a Riemannian manifold kernel for persistence diagrams. In *Advances in Neural Information Processing Systems*, pp. 10027–10038, 2018.

Chunyuan Li, Maks Ovsjanikov, and Frédéric Chazal. Persistence-based structural recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2003–2010, jun 2014.

Guido Montúfar, Nina Otter, and Yuguang Wang. Can neural networks learn persistent homology features? *arXiv preprint arXiv:2011.14688*, 2020.

Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. Topology of deep neural networks. *J. Mach. Learn. Res.*, 21(184):1–40, 2020.

Steve Oudot. *Persistence theory: from quiver representations to data analysis*. American Mathematical Society, 2015.

Jose Perea and John Harer. Sliding windows and persistence: an application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 15(3):799–838, 2015.

Cássio MM Pereira and Rodrigo F de Mello. Persistent homology for time series and spatial data clustering. *Expert Systems with Applications*, 42(15-16):6026–6038, 2015.

Charles R. Qi, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017. doi: 10.1109/CVPR.2017.16.

Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Anirudh Som, Hongjun Choi, Karthikeyan Natesan Ramamurthy, Matthew P Buman, and Pavan Turaga. Pi-net: A deep learning approach to extract topological persistence images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 834–835, 2020.

Guillaume Staerman, Pierre Laforgue, Pavlo Mozharovskyi, and Florence d'Alché Buc. When ot meets mom: Robust estimation of wasserstein distance. In *24th International Conference on Artificial Intelligence and Statistics (AISTATS 2021)*, volume 130, pp. 136–144. PMLR, 2021.

Yuhei Umeda. Time series classification via topological data analysis. *Information and Media Technologies*, 12:228–239, 2017.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. pp. 1912–1920, 06 2015. doi: 10.1109/CVPR.2015.7298801.

Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, Yusu Wang, and Chao Chen. Neural approximation of extended persistent homology on graphs. *arXiv preprint arXiv:2201.12032*, 2022.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pp. 3391–3401, 2017.

Chi Zhou, Zhetong Dong, and Hongwei Lin. Learning persistent homology of 3d point clouds. *Computers & Graphics*, 2021.