Topological Autoencoders

Michael Moor [†1 2] **Max Horn** [†1 2] **Bastian Rieck** [‡1 2] **Karsten Borgwardt** [‡1 2]

Abstract

We propose a novel approach for preserving topological structures of the input space in latent representations of autoencoders. Using *persistent homology*, a technique from topological data analysis, we calculate topological signatures of both the input and latent space to derive a topological loss term. Under weak theoretical assumptions, we construct this loss in a differentiable manner, such that the encoding learns to retain multi-scale connectivity information. We show that our approach is theoretically well-founded and that it exhibits favourable latent representations on a synthetic manifold as well as on real-world image data sets, while preserving low reconstruction errors.

1. Introduction

While topological features, in particular multi-scale features derived from persistent homology, have seen increasing use in the machine learning community (Carrière et al., 2019, Guss & Salakhutdinov, 2018, Hofer et al., 2017, 2019a,b, Ramamurthy et al., 2019, Reininghaus et al., 2015, Rieck et al., 2019a,b), employing topology *directly* as a constraint for modern deep learning methods remains a challenge. This is due to the inherently discrete nature of these computations, making backpropagation through the computation of topological signatures immensely difficult or only possible in certain special circumstances (Chen et al., 2019, Hofer et al., 2019a, Poulenard et al., 2018).

This work presents a novel approach that permits obtaining gradients during the computation of topological signatures. This makes it possible to employ topological constraints while training deep neural networks, as well as building topology-preserving autoencoders. Specifically, we make the following contributions:

1. We develop a new topological loss term for autoencoders that helps harmonise the topology of the data space with the topology of the latent space.
2. We prove that our approach is stable on the level of mini-batches, resulting in suitable approximations of the persistent homology of a data set.
3. We empirically demonstrate that our loss term aids in dimensionality reduction by preserving topological structures in data sets; in particular, the learned latent representations are useful in that the preservation of topological structures can improve interpretability.

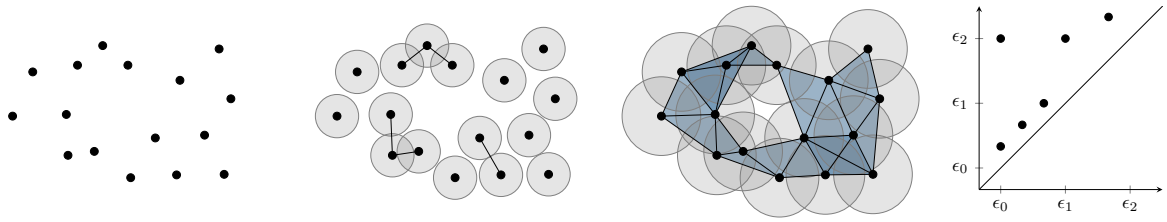
2. Background: Persistent Homology

Persistent homology (Barannikov, 1994, Edelsbrunner & Harer, 2008) is a method from the field of computational topology, which develops tools for analysing topological features (connectivity-based features such as connected components) of data sets. We first introduce the underlying concept of simplicial homology. For a simplicial complex $\mathfrak{K}$, i.e. a generalised graph with higher-order connectivity information such as cliques, simplicial homology employs matrix reduction algorithms to assign $\mathfrak{K}$ a family of groups, the *homology groups*. The d^{th} homology group $\mathrm{H}_d(\mathfrak{K})$ of $\mathfrak{K}$ contains d-dimensional topological features, such as connected components ($d = 0$), cycles/tunnels ($d = 1$), and voids ($d = 2$). Homology groups are typically summarised by their ranks, thereby obtaining a simple invariant "signature" of a manifold. For example, a circle in $\mathbb{R}^2$ has one feature with $d = 1$ (a cycle), and one feature with $d = 0$ (a connected component).

In practice, the underlying manifold $\mathbb{M}$ is unknown and we are working with a point cloud $X := \{x_1, \dots, x_n\} \subseteq \mathbb{R}^d$ and a metric $\mathrm{dist}\colon X \times X \to \mathbb{R}$ such as the Euclidean distance. Persistent homology extends simplicial homology to this setting: instead of approximating $\mathbb{M}$ by means of a *single* simplicial complex, which would be an unstable procedure due to the discrete nature of X, persistent homology tracks changes in the homology groups over *multiple* scales of the metric. This is achieved by constructing a special simplicial complex, the Vietoris–Rips complex (Vietoris, 1927). For $0 \leq \epsilon < \infty$, the Vietoris–Rips complex of X at scale ϵ, denoted by $\mathfrak{R}_\epsilon(X)$, contains all

[†]Equal contribution. [‡]These authors jointly directed this work. [1]Department of Biosystems Science and Engineering, ETH Zurich, 4058 Basel, Switzerland [2]SIB Swiss Institute of Bioinformatics, Switzerland. Correspondence to: Karsten Borgwardt <karsten.borgwardt@bsse.ethz.ch>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.



(a) ϵ_0 (b) ϵ_1 (c) ϵ_2 (d) $\mathcal{D}_d$

Figure 1. The Vietoris–Rips complex $\mathfrak{R}_\epsilon(X)$ of a point cloud X at different scales ϵ_0, ϵ_1, and ϵ_2. As the distance threshold ϵ increases, the connectivity changes. The creation and destruction of d-dimensional topological features is recorded in the d^{th} persistence diagram $\mathcal{D}_d$.

simplices (i.e. subsets) of X whose elements $\{x_0, x_1, \dots\}$ satisfy $\operatorname{dist}(x_i, x_j) \leq \epsilon$ for all i, j. Given a matrix $\mathbf{A}$ of pairwise distances of a point cloud X, we will use $\mathfrak{R}_\epsilon(\mathbf{A})$ and $\mathfrak{R}_\epsilon(X)$ interchangeably because constructing $\mathfrak{R}_\epsilon$ only requires distances. Vietoris–Rips complexes satisfy a nesting relation, i.e. $\mathfrak{R}_{\epsilon_i}(X) \subseteq \mathfrak{R}_{\epsilon_j}(X)$ for $\epsilon_i \leq \epsilon_j$, making it possible to track changes in the homology groups as ϵ increases (Edelsbrunner et al., 2002). Figure 1 illustrates this process. Since X contains a finite number of points, a maximum $\tilde{\epsilon}$ value exists for which the connectivity stabilises; therefore, calculating $\mathfrak{R}_{\tilde{\epsilon}}$ is sufficient to obtain topological features at all scales.

We write $\mathrm{PH}(\mathfrak{R}_\epsilon(X))$ for the persistent homology calculation of the Vietoris–Rips complex. It results in a tuple $(\{\mathcal{D}_1, \mathcal{D}_2, \dots\}, \{\pi_1, \pi_2, \dots\})$ of *persistence diagrams* (1^{st} component) and *persistence pairings* (2^{nd} component). The d-dimensional persistence diagram $\mathcal{D}_d$ (Figure 1d) of $\mathfrak{R}_\epsilon(X)$ contains coordinates of the form (a, b), where a refers to a threshold ϵ at which a d-dimensional topological feature is created in the Vietoris–Rips complex, and b refers to a threshold ϵ' at which it is destroyed (please refer to Supplementary Section A.1 for a detailed explanation). When $d = 0$, for example, the threshold ϵ' indicates at which distance two connected components in X are merged into one. This calculation is known to be related to spanning trees (Kurlin, 2015) and single-linkage clustering, but the persistence diagrams and the persistence pairings carry more information than either one of these concepts.

The d-dimensional persistence pairing contains indices (i, j) corresponding to simplices $s_i, s_j \in \mathfrak{R}_\epsilon(X)$ that create and destroy the topological feature identified by $(a, b) \in \mathcal{D}_d$, respectively. Persistence diagrams are known to be stable with respect to small perturbations in the data set (Cohen-Steiner et al., 2007). Two diagrams $\mathcal{D}$ and $\mathcal{D}'$ can be compared using the *bottleneck distance* $\mathrm{d_b}(\mathcal{D}, \mathcal{D}') := \inf_{\eta\colon \mathcal{D} \to \mathcal{D}'} \sup_{x \in \mathcal{D}} \|x - \eta(x)\|_\infty$, where $\eta\colon \mathcal{D} \to \mathcal{D}'$ denotes a bijection between the points of the two diagrams, and $\|\cdot\|_\infty$ refers to the L_∞ norm. We use $\mathcal{D}^X$ to refer to the *set* of persistence diagrams of a point cloud X arising from $\mathrm{PH}(\mathfrak{R}_\epsilon(X))$.

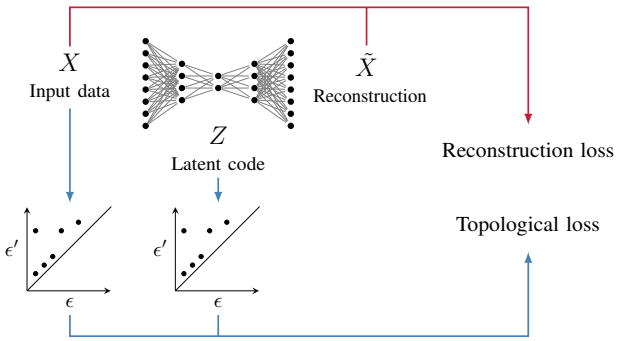


Figure 2. An overview of our method. Given a mini-batch X of data space $\mathcal{X}$, we train an autoencoder to reconstruct X, leading to a reconstruction $\tilde{X}$. In addition to the usual reconstruction loss, we calculate our *topological loss* based on the topological differences between persistence diagrams, i.e. topological feature descriptors, calculated on the mini-batch X and its corresponding latent code Z. The objective of our topological loss term is to constrain the autoencoder such that topological features in the data space are preserved in latent representations.

3. A Topology-Preserving Autoencoder

We propose a generic framework for constraining autoencoders to preserve topological structures (measured via persistent homology) of the data space in their latent encodings. Figure 2 depicts an overview of our method; the subsequent sections will provide more details about the individual steps.

3.1. Vietoris–Rips Complex Calculation

Given a finite metric space $\mathcal{S}$, such as a point cloud, we first calculate the persistent homology of the Vietoris–Rips complex of its distance matrix $\mathbf{A}^{\mathcal{S}}$. It is common practice to use the Euclidean distance for the calculation of $\mathbf{A}^{\mathcal{S}}$, but both the persistent homology calculation and our method are *not* restricted to any particular distance; previous research (Wagner & Dłotko, 2014) shows that even similarity measures that do not satisfy the properties of a metric can be used successfully with $\mathrm{PH}(\cdot)$. Subsequently, let $\epsilon := \max \mathbf{A}^{\mathcal{S}}$ so that $\mathfrak{R}_\epsilon\big(\mathbf{A}^{\mathcal{S}}\big)$ is the corresponding Vietoris–Rips complex

as described in Section 2. Given a maximum dimension[1] of $d \in \mathbb{N}_{>0}$, we obtain a set of persistence diagrams $\mathcal{D}^{\mathcal{S}}$, and a set of persistence pairings $\pi^{\mathcal{S}}$. The d^{th} persistence pairing $\pi_d^{\mathcal{S}}$ contains indices of simplices that are pertinent to the creation and destruction of d-dimensional topological features. We can consider each pairing to represent *edge indices*, namely the edges that are deemed to be "topologically relevant" by the computation of persistent homology (see below for more details). This works because the Vietoris–Rips complex is a *clique complex*, i.e. a simplicial complex that is fully determined by its edges (Zomorodian, 2010).

Selecting indices from pairings The basic idea of our method involves selecting indices in the persistence pairing and mapping them back to a distance between two vertices. We then adjust this distance to harmonise topological features of the input space and the latent space. For 0-dimensional topological features, it is sufficient to consider the indices of *edges*, which are the "destroyer" simplices, in the pairing $\pi_0^{\mathcal{S}}$. Each index corresponds to an edge in the minimum spanning tree of the data set. This calculation is computationally efficient, having a worst-case complexity of $\mathcal{O}\big(m^2 \cdot \alpha\big(m^2\big)\big)$, where m is the batch size and $\alpha(\cdot)$ denotes the extremely slow-growing inverse Ackermann function (Cormen et al., 2009, Chapter 22). For 1-dimensional features, where edges are paired with triangles, we obtain edge indices by selecting the edge with the maximum weight of the triangle. While this procedure, and thus our method, generalises to higher dimensions, our current implementation supports no higher-dimensional features. Since preliminary experiments showed that using 1-dimensional topological features merely increases runtime, the subsequent experiments will focus only on 0-dimensional persistence diagrams. We thus use $\big(\mathcal{D}^{\mathcal{S}}, \pi^{\mathcal{S}}\big)$ to denote the 0-dimensional persistence diagram and pairing of $\mathcal{S}$, respectively.

3.2. Topological Autoencoder

In the following, we consider a mini-batch X of size m from the data space $\mathcal{X}$ as a point cloud. Furthermore, we define an autoencoder as the composition of two functions $h \circ g$, where $g \colon \mathcal{X} \to \mathcal{Z}$ represents the *encoder* and $h \colon \mathcal{Z} \to \mathcal{X}$ represents the *decoder*, denoting latent codes by $Z := g(X)$. During a forward pass of the autoencoder, we compute the persistent homology of the mini-batch in both the data as well as the latent space, yielding two sets of tuples, i.e. $\big(\mathcal{D}^X, \pi^X\big) := \mathrm{PH}(\mathfrak{R}_\epsilon(X))$ and $\big(\mathcal{D}^Z, \pi^Z\big) := \mathrm{PH}(\mathfrak{R}_\epsilon(Z))$. The values of the persistence diagram can be retrieved by subsetting the distance matrix with the edge indices provided by the persistence pairings; we write $\mathcal{D}^X \simeq \mathbf{A}^X\big[\pi^X\big]$ to indicate that the diagram, which is a set, contains the same information as the distances we retrieve with the pairing. We treat $\mathbf{A}^X\big[\pi^X\big]$ as a vector in $\mathbb{R}^{|\pi^X|}$. Informally speaking, the persistent homology calculation can thus be seen as a selection of topologically relevant edges of the Vietoris–Rips complex, followed by the selection of corresponding entries in the distance matrix. By comparing both diagrams, we can construct a topological regularisation term $\mathcal{L}_t := \mathcal{L}_t\big(\mathbf{A}^X, \mathbf{A}^Z, \pi^X, \pi^Z\big)$, which we add to the reconstruction loss of an autoencoder, i.e.

$$\mathcal{L} := \mathcal{L}_r(X, h(g(X))) + \lambda\, \mathcal{L}_t \tag{1}$$

where $\lambda \in \mathbb{R}$ is a parameter to control the strength of the regularisation (see also Supplementary Section A.6).

Next, we discuss how to specify $\mathcal{L}_t$. Since we only select edge indices from π^X and π^Z, the PH calculation represents a selection of topologically relevant *distances* from the distance matrix. Each persistence diagram entry corresponds to a distance between two data points. Following standard assumptions in persistent homology (Hofer et al., 2019a, Poulenard et al., 2018), we assume that the distances are *unique* so that each entry in the diagram has an infinitesimal neighbourhood that only contains a single point. In practice, this can always be achieved by performing (symbolic) perturbations of the distances. Given this fixed pairing and a differentiable distance function, the persistence diagram entries are therefore *also* differentiable with respect to the encoder parameters. Hence, the persistence pairing does not change upon a small perturbation of the underlying distances, thereby guaranteeing the existence of the derivative of our loss function. This, in turn, permits the calculation of gradients for backpropagation.

A straightforward approach to impose the data space topology on the latent space would be to directly calculate a loss based on the selected distances in both spaces. Such an approach will *not* result in informative gradients for the autoencoder, as it merely compares topological features without matching[2] the edges between $\mathfrak{R}_\epsilon(X)$ and $\mathfrak{R}_\epsilon(Z)$. A cleaner approach would be to enforce similarity on the intersection of the selected edges in both complexes. However, this would initially include very few edges, preventing efficient training and leading to highly biased estimates of the topological alignments between the spaces[3]. To overcome this, we account for the *union* of all selected edges in

[1] This means that we do not have to consider higher-dimensional topological features, making the calculation more efficient.

[2] We use the term "matching" only to build intuition. Our approach does not calculate a matching in the sense of a bottleneck or Wasserstein distance between persistence diagrams.

[3] When initialising a random latent space Z, the persistence pairing in the latent space will select random edges, resulting in only 1 expected matched edge (independent of mini-batch size) between the two pairings. Thus, only one edge (referring to one pairwise distance between two latent codes) could be used to update the encoding of these two data points.

X and Z. Our topological loss term decomposes into two components, each handling the "directed" loss occurring as topological features in one of the two spaces remain fixed. Hence, $\mathcal{L}_t = \mathcal{L}_{\mathcal{X}\to\mathcal{Z}} + \mathcal{L}_{\mathcal{Z}\to\mathcal{X}}$, with

$$\mathcal{L}_{\mathcal{X}\to\mathcal{Z}} := \frac{1}{2}\left\|\mathbf{A}^X\left[\pi^X\right] - \mathbf{A}^Z\left[\pi^X\right]\right\|^2 \tag{2}$$

and

$$\mathcal{L}_{\mathcal{Z}\to\mathcal{X}} := \frac{1}{2}\left\|\mathbf{A}^Z\left[\pi^Z\right] - \mathbf{A}^X\left[\pi^Z\right]\right\|^2, \tag{3}$$

respectively. The key idea for both terms is to align and preserve topologically relevant distances from both spaces. By taking the union of all selected edges (and the corresponding distances), we obtain an informative loss term that is determined by at least $|X|$ distances. This loss can be seen as a more generic version of the loss introduced by Hofer et al. (2019a), whose formulation does not take the two directed components into account and optimises the destruction values of all persistence tuples with respect to a uniform parameter (their goal is different from ours and does not require a loss term that is capable of harmonising topological features across the two spaces; please refer to Section 4 for a brief discussion). By contrast, our formulation aims to to align the distances between X and Z (which in turn will lead to an alignment of distances between $\mathcal{X}$ and $\mathcal{Z}$). If the two spaces are aligned perfectly, $\mathcal{L}_{\mathcal{X}\to\mathcal{Z}} = \mathcal{L}_{\mathcal{Z}\to\mathcal{X}} = 0$ because both pairings and their corresponding distances coincide. The converse implication is not true: if $\mathcal{L}_t = 0$, the persistence pairings and their corresponding persistence diagrams are not necessarily identical. Since we did not observe such behaviour in our experiments, however, we leave a more formal treatment of these situations for future work.

Gradient calculation Letting $\boldsymbol{\theta}$ refer to the parameters of the *encoder* and using $\boldsymbol{\rho} := \left(\mathbf{A}^X\left[\pi^X\right] - \mathbf{A}^Z\left[\pi^X\right]\right)$, we have

$$\frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{L}_{\mathcal{X}\to\mathcal{Z}} = \frac{\partial}{\partial\boldsymbol{\theta}}\left(\frac{1}{2}\left\|\mathbf{A}^X\left[\pi^X\right] - \mathbf{A}^Z\left[\pi^X\right]\right\|^2\right) \tag{4}$$

$$= -\boldsymbol{\rho}^\top\left(\frac{\partial\mathbf{A}^Z\left[\pi^X\right]}{\partial\boldsymbol{\theta}}\right) \tag{5}$$

$$= -\boldsymbol{\rho}^\top\left(\sum_{i=1}^{\left|\pi^X\right|}\frac{\partial\mathbf{A}^Z\left[\pi^X\right]_i}{\partial\boldsymbol{\theta}}\right), \tag{6}$$

where $\left|\pi^X\right|$ denotes the cardinality of a persistence pairing and $\mathbf{A}^Z\left[\pi^X\right]_i$ refers to the i^{th} entry of the vector of paired distances. This derivation works analogously for $\mathcal{L}_{\mathcal{Z}\to\mathcal{X}}$ (with π^X being replaced by π^Z). Furthermore, any derivative of $\mathbf{A}^X$ with respect to $\boldsymbol{\theta}$ must vanish because the distances of the input samples do not depend on the encoding by definition. These equations assume infinitesimal perturbations. The persistence diagrams change in a non-differentiable manner during the training phase. However, for any given update step, a diagram is robust to infinitesimal changes of its entries (Cohen-Steiner et al., 2007). As a consequence, our topological loss is differentiable for each update step during training. We make our code publicly available[4].

3.3. Stability

Despite the aforementioned known stability of persistence diagrams with respect to small perturbations of the underlying space, we still have to analyse our topological approximation on the level of mini-batches. The following theorem guarantees that subsampled persistence diagrams are close to the persistence diagrams of the original point cloud.

Theorem 1. *Let X be a point cloud of cardinality n and $X^{(m)}$ be one subsample of X of cardinality m, i.e. $X^{(m)} \subseteq X$, sampled without replacement. We can bound the probability of the persistence diagrams of $X^{(m)}$ exceeding a threshold in terms of the bottleneck distance as*

$$\mathbb{P}\left(\mathrm{d_b}\left(\mathcal{D}^X, \mathcal{D}^{X^{(m)}}\right) > \epsilon\right) \leq \mathbb{P}\left(\mathrm{d_H}\left(X, X^{(m)}\right) > 2\epsilon\right), \tag{7}$$

where $\mathrm{d_H}$ refers to the Hausdorff distance between the point cloud and its subsample.

Proof. See Section A.2 in the supplementary materials. □

For $m \to n$, each mini-batch converges to the original point cloud, so we have $\lim_{m\to n}\mathrm{d_H}\left(X, X^{(m)}\right) = 0$. Please refer to Section A.3 for an analysis of empirical convergence rates as well as a discussion of a worst-case bound. Given certain independence assumptions, the next theorem approximates the expected value of the Hausdorff distance between the point cloud and a mini-batch. The calculation of an exact representation is beyond the scope of this work.

Theorem 2. *Let $\mathbf{A} \in \mathbb{R}^{n\times m}$ be the distance matrix between samples of X and $X^{(m)}$, where the rows are sorted such that the first m rows correspond to the columns of the m subsampled points with diagonal elements $a_{ii} = 0$. Assume that the entries a_{ij} with $i > m$ are random samples following a distance distribution F_D with $\mathrm{supp}(F_D) \in \mathbb{R}_{\geq 0}$. The minimal distances δ_i for rows with $i > m$ follow a distribution F_Δ. Letting $Z := \max_{1\leq i\leq n}\delta_i$ with a corresponding distribution F_Z, the expected Hausdorff distance between*

[4] https://github.com/BorgwardtLab/topological-autoencoders

X and $X^{(m)}$ for $m < n$ is bounded by:

$$\mathbb{E}\Big[\mathrm{d_H}(X, X^{(m)})\Big] = \mathbb{E}_{Z\sim F_Z}[Z] \tag{8}$$

$$\leq \int_0^{+\infty} \Big(1 - F_D(z)^{(n-1)}\Big)\,\mathrm{d}z \tag{9}$$

$$\leq \int_0^{+\infty} \Big(1 - F_D(z)^{m(n-m)}\Big)\,\mathrm{d}z \tag{10}$$

Proof. See Section A.4 in the supplementary materials. □

From Eq. 10, we obtain $\mathbb{E}[\mathrm{d_H}(X, X^m)] = 0$ as $m \to n$, so the expected value converges as the subsample size approaches the total sample size[5]. We conclude that our subsampling approach results in point clouds that are suitable proxies for the large-scale topological structures of the point cloud X.

4. Related Work

Computational topology and persistent homology (PH) have started gaining traction in several areas of machine learning research. PH is often used as as *post hoc* method for analysing topological characteristics of data sets. Thus, there are several methods that compare topological features of high-dimensional spaces with different embeddings to assess the fidelity and quality of a specific embedding scheme (Khrulkov & Oseledets, 2018, Paul & Chalup, 2017, Rieck & Leitte, 2015, 2017, Yan et al., 2018). PH can also be used to characterise the training of neural networks (Guss & Salakhutdinov, 2018, Rieck et al., 2019b), as well as their decision boundaries (Ramamurthy et al., 2019). Our method differs from all these publications in that we are able to obtain gradient information to *update* a model while training. Alternatively, topological features can be integrated into classifiers to improve classification performance. Hofer et al. (2017) propose a neural network layer that learns projections of persistence diagrams, which can subsequently be used as feature descriptors to classify structured data. Moreover, several vectorisation strategies for persistence diagrams exist (Adams et al., 2017, Carrière et al., 2015), making it possible to use them in kernel-based classifiers. These strategies have been subsumed (Carrière et al., 2019) in a novel architecture based on deep sets. The commonality of these approaches is that they treat persistence diagrams as being *fixed*; while they are capable of learning suitable parameters for classifying them, they cannot adjust input data to better approximate a certain topology.

[5]For $m = n$, the two integrals switch their order as $m(n-m) = 0 < n - 1$ (for $n > 1$).

Such topology-based adjustments have only recently become feasible. Poulenard et al. (2018) demonstrated how to optimise real-valued functions based on their topology. This constitutes the first approach for aligning persistence diagrams by modifying input data; it requires the connectivity of the data to be known, and the optimised functions have to be node-based and scalar-valued. By contrast, our method works *directly* on distances and sidesteps connectivity calculations via the Vietoris–Rips complex. Chen et al. (2019) use a similar optimisation technique to regularise the decision boundary of a classifier. However, this requires discretising the space, which can be computationally expensive. Hofer et al. (2019a), the closest work to ours, also presents a differentiable loss term. Their formulation enforces a *single* scale, referred to as η, on the latent space. The learned encoding is then applied to a one-class learning task in which a scoring function is calculated based on the pre-defined scale. By contrast, the goal of our loss term is to support the model in learning a latent encoding that *best preserves* the data space topology in said latent space, which we use for dimensionality reduction. We thus target a different task, and can preserve *multiple* scales (those selected through the filtration process) that are present in the data domain.

5. Experiments

Our main task is to learn a latent space in an unsupervised manner such that topological features of the data space, measured using persistent homology approximations on every batch, are preserved as much as possible.

5.1. Experimental Setup

Subsequently, we briefly describe our data sets and evaluation metrics. Please refer to the supplementary materials for technical details (calculation, hyperparameters, etc.).

5.1.1. Data Sets

We generate a Spheres data set that consists of ten high-dimensional 100-spheres living in a 101-dimensional space that are enclosed by one larger sphere that consists of the same number of points as the total of inner spheres (see Section A.5 for more details). We also use three image data sets (MNIST, Fashion-MNIST, and CIFAR-10), which are particularly amenable to our topology-based analysis because real-world images are known to lie *on* or *near* low-dimensional manifolds (Lee et al., 2003, Peyré, 2009).

5.1.2. Baselines & Training Procedure

We compare our approach with several dimensionality reduction techniques, including UMAP (McInnes et al., 2018), t-SNE (van der Maaten & Hinton, 2008), Isomap (Tenen-

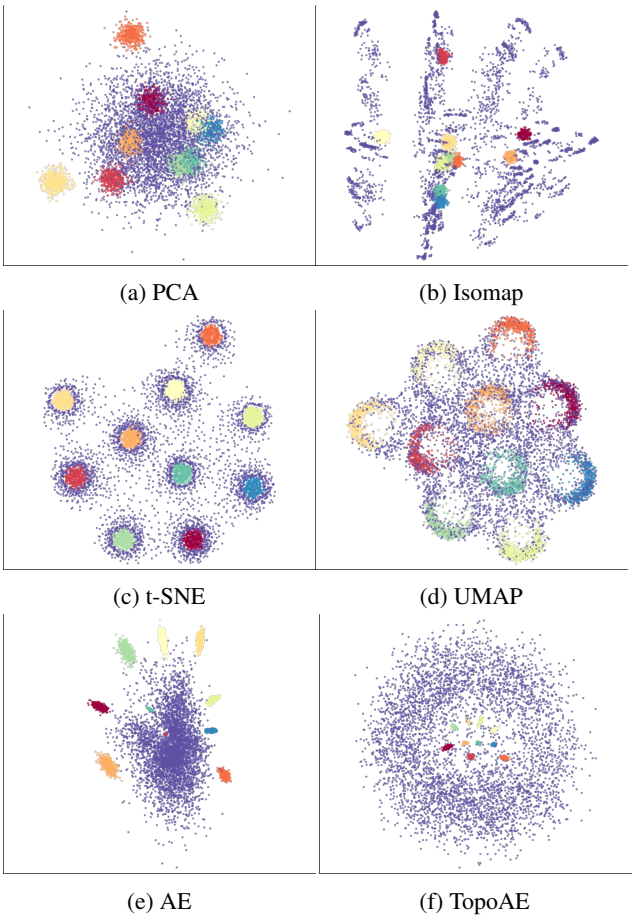
baum et al., 2000), PCA, as well as standard autoencoders (AE). We apply our proposed topological constraint to this standard autoencoder architecture (TopoAE).

For comparability and interpretability, each method is restricted to two latent dimensions. We split each data set into training and testing (using the predefined split if available; 90% versus 10% otherwise). Additionally, we remove 15% of the training split as a validation data set for tuning the hyperparameters. We normalised our topological loss term by the batch size m in order to disentangle λ from it. All autoencoders employ batch-norm and are optimised using ADAM (Kingma & Ba, 2014). Since t-SNE is not intended to be applied to previously unseen test samples, we evaluate this method only on the train split. In addition, significant computational scaling issues prevent us from running a hyperparameter search for Isomap on real-world data sets, so we only compare this algorithm on the synthetic data set. Please refer to Section A.6 for more details on architectures and hyperparameters.

5.1.3. Evaluation

We evaluate the quality of latent representations in terms of (1) low-dimensional visualisations, (2) dimensionality reduction quality metrics (evaluated between input data and *latent* codes), and (3) reconstruction errors (Data MSE; evaluated between input and *reconstructed* data), provided that invertible transformations are available[6]. For (2), we consider several measures (please refer to Section A.7 for more details). First, we calculate KL_σ, the Kullback–Leibler divergence between the density estimates of the input and latent space, based on density estimates (Chazal et al., 2011, 2014b), where $\sigma \in \mathbb{R}_{>0}$ denotes the length scale of the Gaussian kernel, which is varied to account for multiple data scales. We chose minimising $\mathrm{KL}_{0.1}$ as our hyperparameter search objective. Furthermore, we calculate common non-linear dimensionality reduction (NLDR) quality metrics, which use the pairwise distance matrices of the input and the *latent* space (as indicated by the "ℓ" in the abbreviations), namely (1) the *root mean square error* (ℓ-RMSE), which—despite its name—is not related to the reconstruction error of the autoencoder but merely measures to what extent the two distributions of distances coincide, (2) the *mean relative rank error* (ℓ-MRRE), (3) the *continuity* (ℓ-Cont), and (4) the *trustworthiness* (ℓ-Trust) . The reported measures are computed on the test splits (except for t-SNE where no transformation between splits is available, so we report the measures on a random subsample of the train split, preserving the cardinality of the test split).

[6]Invertible transformations are available for PCA and all autoencoder-based methods.



(a) PCA (b) Isomap (c) t-SNE (d) UMAP (e) AE (f) TopoAE

Figure 3. Latent representations of the Spheres data set. Only our method is capable of representing the complicated nesting relationship inherent to the data; t-SNE, for example, tears the original data apart. For TopoAE, we used a batch size of 28. Please refer to Figure A.5 in the supplementary materials for an enlarged version.

5.2. Results

Next to a *quantitative* evaluation in terms of various quality metrics, we also discuss *qualitative* results in terms of visualisations, which are interpretable in case the ground truth manifold is known.

5.2.1. Quantitative Results

Table 1 reports the quantitative results. Overall, we observe that our method is capable of preserving the data density over multiple length scales (as measured by KL). Furthermore, we find that TopoAE displays competitive continuity values (ℓ-Cont) and reconstruction errors (Data MSE). The latter is particularly relevant as it demonstrates that imposing our topological constraints does not result in large impairments when reconstructing the input space.

The remaining classical measures favour the baselines (fore-

Data set	Method	$\mathrm{KL}_{0.01}$	$\mathrm{KL}_{0.1}$	KL_1	ℓ-MRRE	ℓ-Cont	ℓ-Trust	ℓ-RMSE	Data MSE
SPHERES	Isomap	0.181	**0.420**	**0.00881**	**0.246**	**0.790**	**0.676**	10.4	–
	PCA	0.332	0.651	0.01530	0.294	0.747	0.626	11.8	0.9610
	TSNE	**0.152**	0.527	0.01271	**0.217**	0.773	**0.679**	**8.1**	–
	UMAP	0.157	0.613	0.01658	0.250	0.752	0.635	**9.3**	–
	AE	0.566	0.746	0.01664	0.349	0.607	0.588	13.3	**0.8155**
	TopoAE	**0.085**	**0.326**	**0.00694**	0.272	**0.822**	0.658	13.5	**0.8681**
F-MNIST	PCA	**0.356**	**0.052**	**0.00069**	0.057	0.968	0.917	**9.1**	0.1844
	TSNE	0.405	0.071	0.00198	**0.020**	0.967	**0.974**	41.3	–
	UMAP	0.424	0.065	0.00163	0.029	**0.981**	0.959	**13.7**	–
	AE	0.478	0.068	0.00125	**0.026**	0.968	**0.974**	20.7	**0.1020**
	TopoAE	**0.392**	**0.054**	**0.00100**	0.032	**0.980**	0.956	20.5	**0.1207**
MNIST	PCA	0.389	0.163	0.00160	0.166	0.901	0.745	**13.2**	0.2227
	TSNE	**0.277**	**0.133**	0.00214	**0.040**	0.921	**0.946**	22.9	–
	UMAP	**0.321**	0.146	0.00234	**0.051**	**0.940**	**0.938**	**14.6**	–
	AE	0.620	0.155	**0.00156**	0.058	0.913	0.937	18.2	**0.1373**
	TopoAE	0.341	**0.110**	**0.00114**	0.056	**0.932**	0.928	19.6	**0.1388**
CIFAR	PCA	**0.591**	**0.020**	**0.00023**	0.119	**0.931**	0.821	**17.7**	0.1482
	TSNE	0.627	0.030	0.00073	**0.103**	0.903	**0.863**	**25.6**	–
	UMAP	0.617	0.026	0.00050	0.127	0.920	0.817	33.6	–
	AE	0.668	0.035	0.00062	0.132	0.851	**0.864**	36.3	**0.1403**
	TopoAE	**0.556**	**0.019**	**0.00031**	**0.108**	**0.927**	0.845	37.9	**0.1398**

Table 1. Embedding quality according to multiple evaluation metrics (Section 5.1). The hyperparameters of all tunable methods were selected to minimise the objective $\mathrm{KL}_{0.1}$. For each criterion, the winner is shown in bold and underlined, the runner-up in bold. Please refer to Supplementary Table A.2 for more σ scales and variance estimates. The column "Data MSE" indicates the reconstruction error. It is included to demonstrate that applying our loss term has no adverse effects.

most the *train* (!) performance of t-SNE). However, we will subsequently see by inspecting the latent spaces that those classic measures *fail* to detect the relevant structural information, as exemplified with known ground truth manifolds, such as the SPHERES data set.

5.2.2. VISUALISATION OF LATENT SPACES

For the SPHERES data set (Figure 3), we observe that only our method is capable of assessing the nesting relationship of the high-dimensional spheres correctly. By contrast, t-SNE "cuts open" the enclosing sphere, distributing most of its points around the remaining spheres. We see that the KL-divergence confirms the visual assessment that only our proposed method preserves the relevant structure of this data set. Several classical evaluation measures, however, favour t-SNE, even though this method fails to capture the global structure and nesting relationship of the enclosing sphere manifold accounting for half of the data set.

On FASHION-MNIST (Figure 4, leftmost column), we see that, as opposed to AE, which is purely driven by the reconstruction error, our method has the additional objective of *preserving* structure. Here, the constraint helps the regularised autoencoder to "organise" the latent space, resulting in a comparable pattern as in UMAP, which is also topologically motivated (McInnes et al., 2018). Furthermore, we observe that t-SNE tends to fragment certain classes (dark orange, red) into multiple distinct subgroups. This likely does not reflect the underlying manifold structure, but constitutes an artefact frequently observed with this method. For MNIST, the latent embeddings (Figure 4, middle column) demonstrate that the non-linear competitors—mostly by pulling apart distinct classes—lose some of the relationship information *between* clusters when comparing against our proposed method or PCA. Finally, we observe that CIFAR-10 (Figure 4, rightmost column), is challenging to embed in two latent dimensions in a purely unsupervised manner. Interestingly, our method (consistently, i.e. over all runs) was able to identify a *linear substructure* that separates the latent space in two additional groups of classes.

6. Discussion and Conclusion

We presented topological autoencoders, a novel method for preserving topological information, measured in terms of persistent homology, of the input space when learning latent representations with deep neural networks. Under weak theoretical assumptions, we showed how our persistent homology calculations can be combined with backpropagation; moreover, we proved that approximating persistent homology on the level of mini-batches is theoretically justified.

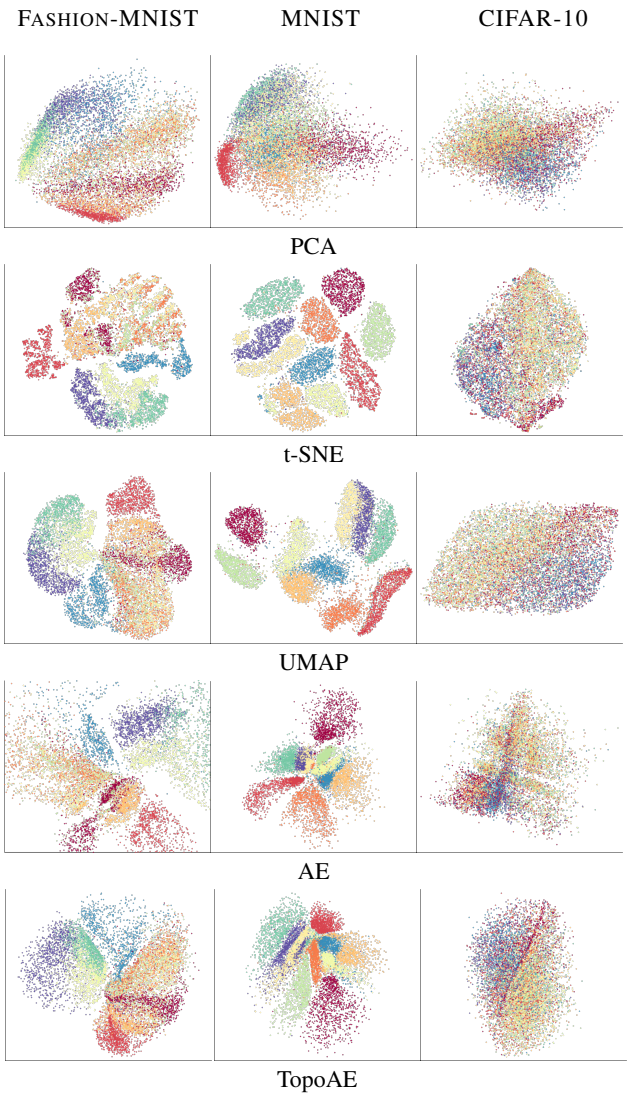


Figure 4. Latent representations of the FASHION-MNIST (left column), MNIST (middle column), CIFAR-10 (right column) data sets. The respective batch size values for our method are $(95, 126, 82)$. Please refer to Figures A.6, A.7, and A.8 in the supplementary materials for enlarged versions.

In our experiments, we observed that our method is uniquely able to capture spatial relationships between nested high-dimensional spheres. This is relevant, as the ability to cope with *several* manifolds in the domain of manifold learning still remains a challenging task. On real-world data sets, we observed that our topological loss leads to competitive performance in terms of numerous quality metrics (such as a density preservation metric), while *not* adversely affecting the reconstruction error. In both synthetic and real-world data sets, we obtain interesting and interpretable representations, as our method does not merely pull apart different classes, but tries to spatially arrange them meaningfully. Thus, we do not observe mere distinct "clouds", but rather entangled structures, which we consider to constitute a more meaningful representation of the underlying manifolds (an auxiliary analysis in Supplementary Section A.10 confirms that our method influences topological features, measured using PH, in a beneficial manner).

Future work Our topological loss formulation is highly generalisable; it only requires the existence of a distance matrix between individual samples (either globally, or on the level of batches). As a consequence, our topological loss term can be directly integrated into a variety of different architectures and is *not* limited to standard autoencoders. For instance, we can also apply our constraint to variational setups (see Figure A.3 for a sketch) or create a topology-aware variant of principal component analysis (dubbed "TopoPCA"; see Table A.2 for more details, as well as Figures A.6, A.7, and A.8 for the corresponding latent space representations). Employing our loss term to more involved architectures will be an exciting route for future work. One issue with the calculation is that, given the computational complexity of calculating $\mathfrak{R}_\epsilon(\cdot)$, for higher-dimensional features, we would scale progressively worse with increasing batch size. However, in our low-dimensional setup, we observed that runtime tends to grow with decreasing batch size, i.e. the mini-batch speed-up still dominates runtime (for more details concerning the effect of batch sizes, see Supplementary Section A.8). In future work, scaling to higher dimensions could be mitigated by approximating the calculation of persistent homology (Choudhary et al., 2018, Kerber & Sharathkumar, 2013, Sheehy, 2013) or by exploiting recent advances in parallelising it (Bauer et al., 2014, Lewis & Morozov, 2015). Another interesting extension would be to tackle classification scenarios with topology-preserving loss terms. This might prove challenging, however, because the goal in classification is to increase class separability, which might be achieved by *removing* topological structures. This goal is therefore at odds with our loss term that tries *preserving* those structures. We think that such an extension might require restricting the method to a subset of scales (namely those that do not impede class separability) to be preserved in the data.

ACKNOWLEDGEMENTS

The authors wish to thank Christian Bock for fruitful discussions and valuable feedback.

This project was supported by the grant #2017-110 of the Strategic Focal Area "Personalized Health and Related Technologies (PHRT)" of the ETH Domain for the SPHN/PHRT Driver Project "Personalized Swiss Sepsis Study" and the SNSF Starting Grant "Significant Pattern Mining" (K.B., grant no. 155913). Moreover, this work was funded in part by the Alfried Krupp Prize for Young University Teachers of the Alfried Krupp von Bohlen und Halbach-Stiftung (K.B.).

References

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

Barannikov, S. A. The framed Morse complex and its invariants. *Advances in Soviet Mathematics*, 21:93–115, 1994.

Bauer, U., Kerber, M., and Reininghaus, J. Distributed computation of persistent homology. In McGeoch, C. C. and Meyer, U. (eds.), *Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pp. 31–38. Society for Industrial and Applied Mathematics, 2014.

Bibal, A. and Frénay, B. Measuring quality and interpretability of dimensionality reduction visualizations. *Safe Machine Learning Workshop at ICLR*, 2019.

Burago, D., Burago, Y., and Ivanov, S. *A course in metric geometry*, volume 33 of *Graduate Studies in Mathematics*. American Mathematical Society, 2001.

Carrière, M., Oudot, S. Y., and Ovsjanikov, M. Stable topological signatures for points on 3D shapes. In *Proceedings of the Eurographics Symposium on Geometry Processing (SGP)*, pp. 1–12, Aire-la-Ville, Switzerland, 2015. Eurographics Association.

Carrière, M., Chazal, F., Ike, Y., Lacombe, T., Royer, M., and Umeda, Y. PersLay: A neural network layer for persistence diagrams and new graph topological signatures. *arXiv e-prints*, art. arXiv:1904.09378, 2019.

Chazal, F., Cohen-Steiner, D., Guibas, L. J., Mémoli, F., and Oudot, S. Y. Gromov–Hausdorff stable signatures for shapes using persistence. *Computer Graphics Forum*, 28(5):1393–1403, 2009.

Chazal, F., Cohen-Steiner, D., and Mérigot, Q. Geometric inference for probability measures. *Foundations of Computational Mathematics*, 11(6):733–751, 2011.

Chazal, F., de Silva, V., and Oudot, S. Y. Persistence stability for geometric complexes. *Geometriæ Dedicata*, 173(1): 193–214, 2014a.

Chazal, F., Fasy, B. T., Lecci, F., Michel, B., Rinaldo, A., and Wasserman, L. Robust topological inference: Distance to a measure and kernel distance. *arXiv e-prints*, art. arXiv:1412.7197, 2014b.

Chazal, F., Fasy, B., Lecci, F., Michel, B., Rinaldo, A., and Wasserman, L. Subsampling methods for persistent homology. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2143–2151. PMLR, 2015a.

Chazal, F., Glisse, M., Labruère, C., and Michel, B. Convergence rates for persistence diagram estimation in topological data analysis. *Journal of Machine Learning Research*, 16:3603–3635, 2015b.

Chen, C., Ni, X., Bai, Q., and Wang, Y. A topological regularizer for classifiers via persistent homology. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pp. 2573–2582. PMLR, 2019.

Choudhary, A., Kerber, M., and Raghvendra, S. Improved topological approximations by digitization. *arXiv e-prints*, art. arXiv:1812.04966, 2018.

Cohen-Steiner, D., Edelsbrunner, H., and Harer, J. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. MIT Press, Cambridge, MA, USA, 3rd edition, 2009.

Edelsbrunner, H. and Harer, J. Persistent homology—a survey. In Goodman, J. E., Pach, J., and Pollack, R. (eds.), *Surveys on discrete and computational geometry: Twenty years later*, number 453 in Contemporary Mathematics, pp. 257–282. American Mathematical Society, Providence, RI, USA, 2008.

Edelsbrunner, H., Letscher, D., and Zomorodian, A. J. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

Gracia, A., González, S., Robles, V., and Menasalvas, E. A methodology to compare dimensionality reduction algorithms in terms of loss of quality. *Information Sciences*, 270:1–27, 2014.

Guss, W. H. and Salakhutdinov, R. On characterizing the capacity of neural networks using algebraic topology. *arXiv e-prints*, art. arXiv:1802.04443, 2018.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313 (5786):504–507, 2006.

Hofer, C., Kwitt, R., Niethammer, M., and Uhl, A. Deep learning with topological signatures. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1633–1643. Curran Associates, Inc., 2017.

Hofer, C., Kwitt, R., Niethammer, M., and Dixit, M. Connectivity-optimized representation learning via persistent homology. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2751–2760. PMLR, 2019a.

Hofer, C. D., Graf, F., Rieck, B., Niethammer, M., and Kwitt, R. Graph filtration learning. *arXiv e-prints*, art. arXiv:1905.10996, 2019b.

Kerber, M. and Sharathkumar, R. Approximate Čech complexes in low and high dimensions. *arXiv e-prints*, art. arXiv:1307.3272, 2013.

Khrulkov, V. and Oseledets, I. Geometry score: A method for comparing generative adversarial networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2621–2629. PMLR, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv e-prints*, art. arXiv:1412.6980, 2014.

Kurlin, V. A one-dimensional homologically persistent skeleton of an unstructured point cloud in any metric space. *Computer Graphics Forum*, 34(5):253–262, 2015.

Lee, A. B., Pedersen, K. S., and Mumford, D. The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision*, 54(1–3):83–103, 2003.

Lee, J. A. and Verleysen, M. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7):1431–1443, 2009.

Lewis, R. and Morozov, D. Parallel computation of persistent homology using the blowup complex. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pp. 323–331. ACM, 2015.

McInnes, L., Healy, J., and Melville, J. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv e-prints*, art. arXiv:1802.03426, 2018.

Mémoli, F. and Sapiro, G. Comparing point clouds. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP)*, pp. 32–40, New York, NY, USA, 2004. Association for Computing Machinery.

Paul, R. and Chalup, S. K. A study on validating non-linear dimensionality reduction using persistent homology. *Pattern Recognition Letters*, 100:160–166, 2017.

Peyré, G. Manifold models for signals and images. *Computer Vision and Image Understanding*, 113(2):249–260, 2009.

Poulenard, A., Skraba, P., and Ovsjanikov, M. Topological function optimization for continuous shape matching. *Computer Graphics Forum*, 37(5):13–25, 2018.

Ramamurthy, K. N., Varshney, K., and Mody, K. Topological data analysis of decision boundaries with application to model selection. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5351–5360. PMLR, 2019.

Reininghaus, J., Huber, S., Bauer, U., and Kwitt, R. 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.

Rieck, B. and Leitte, H. Persistent homology for the evaluation of dimensionality reduction schemes. *Computer Graphics Forum*, 34(3):431–440, 2015.

Rieck, B. and Leitte, H. Agreement analysis of quality measures for dimensionality reduction. In Carr, H., Garth, C., and Weinkauf, T. (eds.), *Topological Methods in Data Analysis and Visualization IV*. Springer, Cham, Switzerland, 2017.

Rieck, B., Bock, C., and Borgwardt, K. A persistent Weisfeiler–Lehman procedure for graph classification. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5448–5458. PMLR, 2019a.

Rieck, B., Togninalli, M., Bock, C., Moor, M., Horn, M., Gumbsch, T., and Borgwardt, K. Neural persistence: a complexity measure for deep neural networks using algebraic topology. In *International Conference on Learning Representations (ICLR)*, 2019b.

scikit-optimize contributers, T. `scikit-optimize/scikit-optimize`: v0.5.2, March 2018.

Sheehy, D. R. Linear-size approximations to the Vietoris–Rips filtration. *Discrete & Computational Geometry*, 49 (4):778–796, 2013.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

van der Maaten, L. J. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

van der Maaten, L. J., Postma, E. O., and van den Herik, H. J. Dimensionality reduction: A comparative review. Technical Report 2009-005, Tilburg University, 2009.

Venna, J. and Kaski, S. Visualizing gene interaction graphs with local multidimensional scaling. In *Proceedings of the 14th European Symposium on Artificial Neural Networks*, pp. 557–562. d-side publishing, 2006.

Vietoris, L. Über den höheren Zusammenhang kompakter Räume und eine Klasse von zusammenhangstreuen Abbildungen. *Mathematische Annalen*, 97(1):454–472, 1927.

Wagner, H. and Dłotko, P. Towards topological analysis of high-dimensional feature spaces. *Computer Vision and Image Understanding*, 121:21–26, 2014.

Yan, L., Zhao, Y., Rosen, P., Scheidegger, C., and Wang, B. Homology-preserving dimensionality reduction via manifold landmarking and tearing. *arXiv e-prints*, art. arXiv:1806.08460, 2018.

Zomorodian, A. J. Fast construction of the Vietoris–Rips complex. *Computers & Graphics*, 34(3):263–271, 2010.

A. Appendix

A.1. Persistent Homology Calculation Details

This section provides more details about the persistent homology calculation; it is more geared towards an expert reader and aims for a concise description of all required concepts.

Simplicial homology To understand persistent homology, we first have to understand simplicial homology. Given a simplicial complex $\mathfrak{K}$, i.e. a high-dimensional generalisation of a graph, let $C_d(\mathfrak{K})$ denote the vector space generated over $\mathbb{Z}_2$ whose elements are the d-simplices in $\mathfrak{K}$[7]. For $\sigma = (v_0, \dots, v_d) \in \mathfrak{K}$, let $\partial_d \colon C_d(\mathfrak{K}) \to C_{d-1}(\mathfrak{K})$ be the boundary homomorphism defined by

$$\partial_d(\sigma) := \sum_{i=0}^{d} (v_0, \dots, v_{i-1}, v_{i+1}, \dots, v_d) \tag{11}$$

for a single simplex and linearly extended to $C_d(\mathfrak{K})$. The d^{th} homology group $\mathrm{H}_d(\mathfrak{K})$ of $\mathfrak{K}$ is defined as the quotient group $\mathrm{H}_d(\mathfrak{K}) := \ker \partial_d / \operatorname{im} \partial_{d+1}$. The rank of the d^{th} homology group is known as the d^{th} Betti number β_d, i.e. $\beta_d(\mathfrak{K}) := \operatorname{rank} \mathrm{H}_d(\mathfrak{K})$. The sequence of Betti numbers $\beta_0, \dots, \beta_d$ of a d-dimensional simplicial complex is commonly used to distinguish between different manifolds. For example, a 2-sphere in $\mathbb{R}^3$ has Betti numbers $(1, 0, 1)$, while a 2-torus in $\mathbb{R}^3$ has Betti numbers $(1, 2, 1)$. Betti numbers are of limited use for analysing real-world data sets, however, because their representation is too coarse and easily affected by small changes in the underlying simplicial complex. In an idealised, platonic setting, this does not pose a problem, because one assumes that the triangulation of a manifold is known a priori; for real-world data sets, however, we are typically dealing with point clouds and have *no* knowledge of the underlying manifold, making the calculation of the "proper" simplicial complex nigh impossible. These disadvantages prompted the development of persistent homology.

Persistent homology Let $\emptyset = \mathfrak{K}_0 \subseteq \mathfrak{K}_1 \subseteq \cdots \subseteq \mathfrak{K}_{m-1} \subseteq \mathfrak{K}_m = \mathfrak{K}$ be a nested sequence of simplicial complexes, called *filtration*. Filtrations can be defined based on different functions; the Vietoris–Rips filtration that we discuss in the paper, for example, is defined by a distance function, such as the Euclidean distance between points of a point cloud. Notice that we may still calculate the simplicial homology of each $\mathfrak{K}_i$ in the filtration. The filtration provides more information, though: the family of boundary operators $\partial(\cdot)$, together with the inclusion homomorphism, induces a homomorphism between corresponding homology groups of the filtration, i.e. $f_d^{i,j} \colon \mathrm{H}_d(\mathfrak{K}_i) \to \mathrm{H}_d(\mathfrak{K}_j)$. This homomorphism yields a sequence of homology groups

$$\begin{aligned} 0 = \mathrm{H}_d(\mathfrak{K}_0) \xrightarrow{f_d^{0,1}} \mathrm{H}_d(\mathfrak{K}_1) \xrightarrow{f_d^{1,2}} \dots \\ \dots \xrightarrow{f_d^{m-2,m-1}} \mathrm{H}_d(\mathfrak{K}_{m-1}) \xrightarrow{f_d^{m-1,m}} \mathrm{H}_d(\mathfrak{K}_m) = \mathrm{H}_d(\mathfrak{K}) \end{aligned}$$

for every dimension d. Given indices $i \leq j$, the d^{th} *persistent homology group* is defined as

$$\mathrm{H}_d^{i,j} := \ker \partial_d(\mathfrak{K}_i) / \left(\operatorname{im} \partial_{d+1}(\mathfrak{K}_j) \cap \ker \partial_d(\mathfrak{K}_i)\right). \tag{12}$$

It can be seen as the homology group that contains all homology classes created in $\mathfrak{K}_i$ that are still *present* ("active") in $\mathfrak{K}_j$. We define the d^{th} persistent Betti number to be the rank of this group, i.e. $\beta_d^{i,j} := \operatorname{rank} \mathrm{H}_d^{i,j}$, which generalises the previous definition for simplicial homology. Persistent homology results in a *sequence* of Betti numbers—instead of a single number—that permits a fine-grained description of topological activity. This activity is typically summarised in a persistence diagram, thus replacing the indices i, j with real numbers based on the function that was used to calculate the filtration.

Persistence diagrams A filtration often has associated values (or weights) $w_0 \leq w_1 \leq \cdots \leq w_{m-1} \leq w_m$, such as the pairwise distances in a point cloud. These values permit the calculation of topological feature descriptors known as *persistence diagrams*: for each dimension d and each pair $i \leq j$, one stores a pair $(a, b) := (w_i, w_j) \in \mathbb{R}^2$ with multiplicity

$$\mu_{i,j}^{(d)} := \left(\beta_d^{i,j-1} - \beta_d^{i,j}\right) - \left(\beta_d^{i-1,j-1} - \beta_d^{i-1,j}\right) \tag{13}$$

in a multiset (typically, $\mu_{i,j}^{(d)} = 0$ for many pairs). The pair (a, b) represents a topological feature that was created at a certain threshold a, and destroyed at another threshold b. In the case of the Vietoris–Rips filtration and connected components, we have $a = 0$ because *all* connected components are present at the beginning of the filtration by definition. Similarly, b will correspond to an edge used in the minimum spanning tree of the data set. In general, the resulting set of points is called the d^{th} *persistence diagram* $\mathcal{D}_d$. Given a point $(a, b) \in \mathcal{D}_d$, the quantity $\operatorname{pers}(x, y) := |a - b|$ is referred to as its *persistence*.

A.2. Proof of Theorem 1

Theorem 1. *Let X be a point cloud of cardinality n and $X^{(m)}$ be one subsample of X of cardinality m, i.e. $X^{(m)} \subseteq X$, sampled without replacement. We can bound the probability of $X^{(m)}$ exceeding a threshold in terms of the bottleneck distance as*

$$\mathbb{P}\left(\mathrm{d_b}\left(\mathcal{D}^X, \mathcal{D}^{X^{(m)}}\right) > \epsilon\right) \leq \mathbb{P}\left(\mathrm{d_H}\left(X, X^{(m)}\right) > 2\epsilon\right), \tag{14}$$

[7] It is also possible to describe this calculation with coefficients in other fields, but the case of $\mathbb{Z}_2$ is advantageous because it simplifies the implementation of all operations.

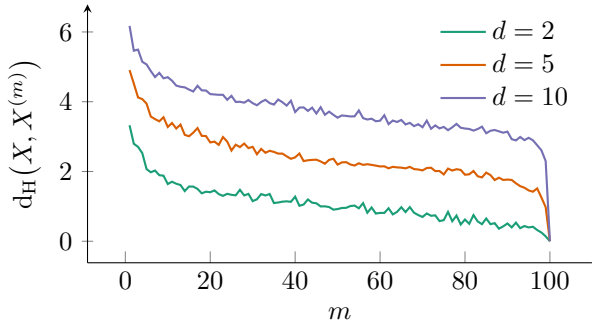


Figure A.1. Empirical convergence rate (mean) of the Hausdorff distance for a subsample of size m of 100 points in a d-dimensional space, following a standard normal distribution.

where $\mathrm{d_H}$ *refers to the Hausdorff distance between the point cloud and its subsample, i.e.*

$$\mathrm{d_H}(X, Y) := \max\{ \sup_{x \in X} \inf_{y \in Y} \mathrm{dist}(x, y), \sup_{y \in Y} \inf_{x \in X} \mathrm{dist}(x, y)\} \tag{15}$$

for a baseline distance $\mathrm{dist}(x, y)$ *such as the Euclidean distance.*

Proof. The stability of persistent homology calculations was proved by Chazal et al. (2014a) for finite metric spaces. More precisely, given two metric spaces X and Y, we have

$$\mathrm{d_b}\big(\mathcal{D}^X, \mathcal{D}^Y\big) \leq 2\, \mathrm{d_{GH}}(X, Y), \tag{16}$$

where $\mathrm{d_{GH}}(X, Y)$ refers to the Gromov–Hausdorff distance (Burago et al., 2001, p. 254) of the two spaces. It is defined as the infimum Hausdorff distance over all isometric embeddings of X and Y. This distance can be employed for shape comparison (Chazal et al., 2009, Mémoli & Sapiro, 2004), but is hard to compute. In our case, with $X = X$ and $Y = X^{(m)}$, we consider both spaces to have the same metric (for Y, we take the canonical restriction of the metric from X to the subspace Y). By definition of the Gromov–Hausdorff distance, we thus have $\mathrm{d_{GH}}(X, Y) \leq \mathrm{d_H}(X, Y)$, so Eq. 15 leads to

$$\mathrm{d_b}\big(\mathcal{D}^X, \mathcal{D}^Y\big) \leq 2\, \mathrm{d_H}(X, Y), \tag{17}$$

from which the original claim from Eq. 14 follows by taking probabilities on both sides. □

A.3. Empirical Convergence Rates of $\mathrm{d_H}\big(X, X^{(m)}\big)$

Figure A.1 depicts the mean of the convergence rate (mean) of the Hausdorff distance for a subsample of size m of 100 points in a d-dimensional space, following a standard normal distribution. We can see that the convergence rate is roughly similar, but shown on different absolute levels that depend on the ambient dimension. While bounding the convergence rate of this expression is feasible (Chazal et al., 2015a,b), it requires more involved assumptions on the measures from which X and $X^{(m)}$ are sampled. Additionally, we can give a simple bound using the *diameter* $\mathrm{diam}(X) := \sup\{\mathrm{dist}(x, y) \mid x, y \in X\}$. We have $\mathrm{d_H}\big(X, X^{(m)}\big) \leq \mathrm{diam}(X)$ because the supremum is guaranteed to be an upper bound for the Hausdorff distance. This worst-case bound does not account for the sample size (or mini-batch size) m, though (see Theorem 2 for an expression that takes m into account).

A.4. Proof of Theorem 2

Prior to the proof we state two observations that arise from our special setting of dealing with finite point clouds.

Observation 1. *Since* $X^{(m)} \subseteq X$, *we have* $\sup_{x' \in X^{(m)}} \inf_{x \in X} \mathrm{dist}(x, x') = 0$. *Hence, the Hausdorff distance simplifies to:*

$$\mathrm{d_H}\Big(X, X^{(m)}\Big) := \sup_{x \in X} \inf_{x' \in X^m} \mathrm{dist}(x, x') \tag{18}$$

In other words, we only have to consider a "one-sided" expression of the distance because the distance from the subsample to the original point cloud is always zero.

Observation 2. *Since our point clouds of interest are finite sets, the suprema and infima of the Hausdorff distance coincide with the maxima and minima, which we will subsequently use for easier readability.*

Hence, the computation of $\mathrm{d_H}(X, X^{(m)})$ can be divided into three steps.

1. Using the baseline distance $\mathrm{dist}(\cdot, \cdot)$, we compute a distance matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ between all points in X and $X^{(m)}$.
2. For each of the n points in X, we compute the minimal distance to the m samples of $X^{(m)}$ by extracting the minimal distance per row of $\mathbf{A}$ and gather all minimal distances in $\boldsymbol{\delta} \in \mathbb{R}^n$.
3. Finally, we return the maximal entry of $\boldsymbol{\delta}$ as $\mathrm{d_H}\big(X, X^{(m)}\big)$.

In the subsequent proof, we require an independence assumption of the samples.

Proof. Using Observations 1 and 2 we obtain a simplified expression for the Hausdorff distance, i.e.

$$\mathrm{d_H}\Big(X, X^{(m)}\Big) := \max_{i, 1 \leq i \leq n} \Big(\min_{j, 1 \leq j \leq m} (a_{ij}) \Big). \tag{19}$$

The minimal distances of the first m rows of $\mathbf{A}$ are trivially 0. Hence, the outer maximum is determined by the

remaining $n-m$ row minima $\{\delta_i \mid m < i \leq n\}$ with $\delta_i = \min_{1\leq j\leq m}(a_{ij})$. Those minima follow the distribution $F_\Delta(y)$ with

$$F_\Delta(y) = \mathbb{P}(\delta_i \leq y) = 1 - \mathbb{P}(\delta_i > y) \tag{20}$$

$$= 1 - \mathbb{P}\left(\min_{1\leq j\leq m} a_{ij} > y\right) \tag{21}$$

$$= 1 - \mathbb{P}\left(\bigcap_j a_{ij} > y\right) \tag{22}$$

$$= 1 - (1 - F_D(y)^m) = F_D(y)^m. \tag{23}$$

Next, we consider $Z := \max_{1\leq i\leq n}\delta_i$. To evaluate the density of Z, we first need to derive its distribution F_Z:

$$F_Z(z) = \mathbb{P}(Z \leq z) = \mathbb{P}\left(\max_{m<i\leq n}\delta_i \leq z\right) \tag{24}$$

$$= \mathbb{P}\left(\bigcap_{m<i\leq n}\delta_i \leq z\right) \tag{25}$$

Next, we approximate Z by Z' by imposing *i.i.d sampling* of the minimal distances δ_i from F_Δ. This is an approximation because in practice, the rows $m+1$ to n are not stochastically independent because of the triangular inequality that holds for metrics. However, assuming i.i.d., we arrive at

$$F_{Z'}(z) = F_\Delta(z)^{n-m}. \tag{26}$$

Since Z' has positive support its expectation can then be evaluated as:

$$\mathbb{E}_{Z'\sim F_{Z'}}[Z'] = \int_0^{+\infty}(1 - F_{Z'}(z))\,\mathrm{d}z \tag{27}$$

$$= \int_0^{+\infty}\left(1 - F_\Delta(z)^{n-m}\right)\mathrm{d}z \tag{28}$$

$$= \int_0^{+\infty}\left(1 - F_D(z)^{m(n-m)}\right)\mathrm{d}z \tag{29}$$

$$\geq \int_0^{+\infty}\left(1 - F_D(z)^{(n-1)}\right)\mathrm{d}z \tag{30}$$

The independence assumption leading to Z' results in *overestimating* the variance of the drawn minima δ_i. Thus, the expected maximum of those minima, $\mathbb{E}[Z']$, is overestimating the actual expectation of the maximum $\mathbb{E}[Z]$, which is why Eq. 27 to Eq. 29 constitute an *upper bound* of $\mathbb{E}[Z]$, and equivalently, an upper bound of $\mathbb{E}\left[\mathrm{d_H}(X, X^{(m)})\right]$. When increasing m, $\mathbb{E}[\mathrm{d_H}(X, X^m)]$ decreases monotonically since for a particular m, we draw $n-m$ samples from the minimal distance distribution F_Δ, and their maximum determines the Hausdorff distance. In contrast, our preliminary upper bound on the left-hand side of Eq. 29 forms a downwards-facing parabola due to the quadratic form in the exponent. This indicates that a tighter bound is achieved for $m \neq n$ by using the minimal subsample size of $m = 1$. □

A.5. Synthetic Data Set

SPHERES consists of eleven high-dimensional 100-spheres living in $101-$dimensional space. Ten spheres of radius $r = 5$ are each shifted in a random direction (by adding the same Gaussian noise vector per sphere). To this end, we draw ten d-dimensional Gaussian vectors following $\mathcal{N}(\mathbf{0}, \mathbf{I}(10/\sqrt{d}))$ for $d = 101$. Crucially, to add interesting topological information to the data set, the ten spheres are enclosed by an additional larger sphere of radius $5r$. The spheres were generated using the library `scikit-tda`.

A.6. Architectures and Hyperparameter Tuning

Architectures for synthetic data set For the synthetically generated data set, we use a simple multilayer perceptron architecture consisting of two hidden layer with 32 neurons each both encoder and decoder and a bottleneck of two neurons such that the sequence of hidden-layer neurons is $32-32-2-32-32$. ReLU non-linearities and batch normalization were applied between the layers excluding the output layer and the bottleneck layer. The networks were fit using mean squared error loss.

Architectures for real world data sets For the MNIST, FASHION-MNIST, and CIFAR-10 data sets, we use an architecture inspired by DeepAE (Hinton & Salakhutdinov, 2006). This architecture is composed of 3 layers of hidden neurons of decreasing size $(1000-500-250)$ for the encoder part, a bottleneck of two neurons, and a sequence of three layers of hidden neurons in decreasing size $(250-500-1000)$ for the decoder. In contrast to the originally proposed architecture, we applied ReLU non-linearities and batch normalization between the layers as we observed faster and more stable training. For the non-linearities of the final layer, we applied the $tanh$ non-linearity, such that the image of the activation matches the range of input images scaled between -1 and 1. Also here, we applied mean squared error loss.

All neural network architectures were fit using Adam and weight-decay of 10^{-5}.

Hyperparameter tuning For hyperparameter tuning we apply random sampling of hyperparameters using the `scikit-optimize` library (scikit-optimize contributers, 2018) with 20 calls per method on all data sets. We select the best model parameters in terms of $\mathrm{KL}_{0.1}$ on the vali-

dation split and evaluate and report it on the test split. To estimate performance means and standard deviations, we repeated the evaluation on an independent test split 5 times by using the best parameters (as identified in the hyperparameter search on the validation split) and refitting the models by resampling the train-validation split.

Neural networks For the neural networks, we sample the learning rate log-uniformly in the range $[10^{-4}, 10^{-2}]$, the batch size uniformly between $[16, 128]$, and for our topological autoencoder method (TopoAE), we sample the regularisation strength λ log-uniformly in the range $[10^{-1}, 3]$. Each model was allowed to train for at most 100 epochs and we applied early stopping with patience $= 10$ based on the validation loss.

Competitor methods For t-SNE, we sample the perplexity uniformly in the range $5 - 50$ and the learning rate log-uniformly in the range $10 - 1000$. For Isomap and UMAP, the number of neighbors included in the computation was varied between $15 - 500$. For UMAP, we additionally vary the min_dist parameter uniformly between 0 and 1.

A.7. Measuring the Quality of Latent Representations

Next to the reconstruction error (if available; please see the paper for a discussion on this), we use a variety of NLDR metrics to assess the quality of our method. Our primary interest concerns the quality of the latent space because, among others, it can be used to visualise the data set. We initially considered classical quality metrics from nonlinear dimensionality reduction (NLDR) algorithms (see Bibal & Frénay (2019), Gracia et al. (2014), van der Maaten et al. (2009) for more in-depth descriptions), namely

(1) the *root mean square error* (ℓ-RMSE) between the distance matrix of the original space and the latent space (as mentioned in the main text, this is not related to the reconstruction error),
(2) the *mean relative rank error* (ℓ-MRRE), which measures the changes in *ranks* of distances in the original space and the latent space (Lee & Verleysen, 2009),
(3) the *trustworthiness* (ℓ-Trust) measure (Venna & Kaski, 2006), which checks to what extent the k nearest neighbours of a point are preserved when going from the original space to the latent space, and
(4) the *continuity* (ℓ-Cont) measure (Venna & Kaski, 2006), which is defined analogously to ℓ-Trust, but checks to what extent neighbours are preserved when going from the *latent* space to the original space.

All of these measures are defined based on comparisons of the original space and the latent space; the reconstructed space is *not* used here. As an additional measure, we calculate the *Kullback–Leibler divergence* between density distributions of the input space and the latent space. Specifically, for a point cloud X with an associated distance dist, we first use the *distance to a measure* density estimator (Chazal et al., 2011, 2014b), defined as $\mathrm{f}_\sigma^{\mathcal{X}}(x) := \sum_{y \in \mathcal{X}} \exp\left(-\sigma^{-1} \mathrm{dist}(x, y)^2\right)$, where $\sigma \in \mathbb{R}_{>0}$ represents a length scale parameter. For dist, we use the Euclidean distance and normalise it between 0 and 1. Given σ, we evaluate $\mathrm{KL}_\sigma := \mathrm{KL}\left(\mathrm{f}_\sigma^X \parallel \mathrm{f}_\sigma^Z\right)$, which measures the similarity between the two density distributions. Ideally, we want the two distributions to be similar because this implies that density estimates in a low-dimensional representation are similar to the ones in the original space.

A.8. Assessing the Batch Size

As we used fixed architectures for the hyperparameter search, the batch size remains the main determinant for the runtime of TopoAE. In Figure A.2, we display trends (linear fits) on how loss measures vary with batch size. Addtionally, we draw runtime estimates. As we applied early stopping, for better comparability, we approximated the epoch-wise runtime by dividing the execution time of a run by its number of completed epochs. Interestingly, these plots suggest that the runtime grows with decreasing batch size (even though the topological computation is more costly for larger batch sizes!). In these experiments, sticking to $0-$dimensional topological features we conclude that the benefit of using mini-batches for neural network training still dominate the topological computations. The few steep peaks most likely represent outliers (the corresponding runs stopped after few epochs, which is why the effective runtime could be overestimated).

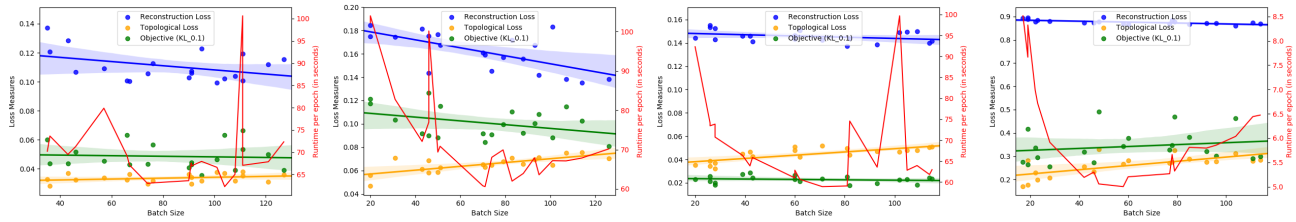
For the loss measures, we see that reconstruction loss tends to *decrease* with increasing batch size, while our topological loss tends to *increase* with increasing batch size (despite normalization). The second observation might be due to larger batch size enabling more complex data point arrangements and corresponding topologies.

A.9. Extending to Variational Autoencoders

In Figure A.3 we sketch a preliminary experiment, where we apply our topological constraint to variational autoencoders for the SPHERES data set. Also here, we observe that our constraint helps identifying the nesting structure of the enclosing sphere.

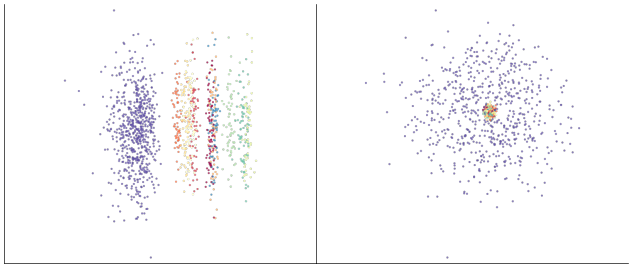
A.10. Topological Distance Calculations

To assess the topological fidelity of the resulting latent spaces, we calculate several topological distances between the test data set (full dimensionality) and the latent spaces obtained from each method (two dimensions). More precisely, we calculate (i) the 1^{st} Wasserstein distance (W_1),



(a) F-MNIST (b) MNIST (c) CIFAR-10 (d) SPHERES

Figure A.2. A scatterplot of batch sizes verses three measures of interest: Topological Loss, Reconstruction Loss, and $\mathrm{KL}_{0.1}$, our objective for the hyperparameter search. Additionally, we draw per-epoch runtime estimates.



(a) VAE (b) TopoVAE

Figure A.3. A depiction of latent spaces obtained for the SPHERES data set with variational autoencoders (VAEs). Here, VAE represents a standard MLP-based VAE, whereas TopoVAE represents the same architecture plus our topological constraint.

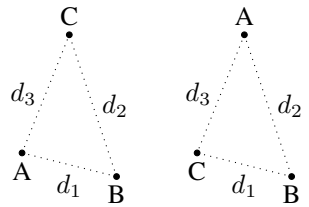
(ii) the 2nd Wasserstein distance (W_2), and (iii) the bottleneck distance (W_∞) between the persistence diagrams obtained from the test data set of the SPHERES data and their resulting 2D latent representations. Even though our loss function is *not* optimising this distance, we observe in Table A.1 that the topological distance of our method ("TopoAE") is always the lowest among all the methods. In particular, it is *always* smaller than the topological distance of the latent space of the autoencoder architecture; this is true for all distance measures, even though W_∞, for example, is known to be susceptible to outliers. Said experiment serves as a simple "sanity check" as it demonstrates that the changes induced by our method are beneficial in that they reduce the topological distance of the latent space to the original data set. For a proper comparison of topological features between the two sets of spaces, a more involved approach would be required, though.

A.11. Alternative Loss Formulations

Our choice of loss function was motivated by the observation that *only* aligning the persistence diagrams between mini-batches of $\mathcal{X}$ and $\mathcal{Z}$ can lead to degenerate or "meaningless" latent spaces. As a simple example (see Figure A.4 for a visualisation), imagine three non-collinear points in the input space and the triangle they are forming. Now assume

Method	W_1	W_2	W_∞
Isomap	4.32±0.037	0.477±0.0045	0.165±0.00096
PCA	4.42±0.053	0.476±0.0046	0.158±0.00108
t-SNE	4.38±0.038	0.478±0.0045	0.164±0.00094
UMAP	4.47±0.042	0.478±0.0045	0.160±0.00092
AE	3.99±0.037	0.469±0.0053	0.154±0.00128
TopoAE	3.73±0.076	0.459±0.0055	0.152±0.00268

Table A.1. Topological distances between the test data set and the corresponding latent space. We used subsamples of size $m = 500$ and 10 repetitions (obtaining a mean and a standard deviation).



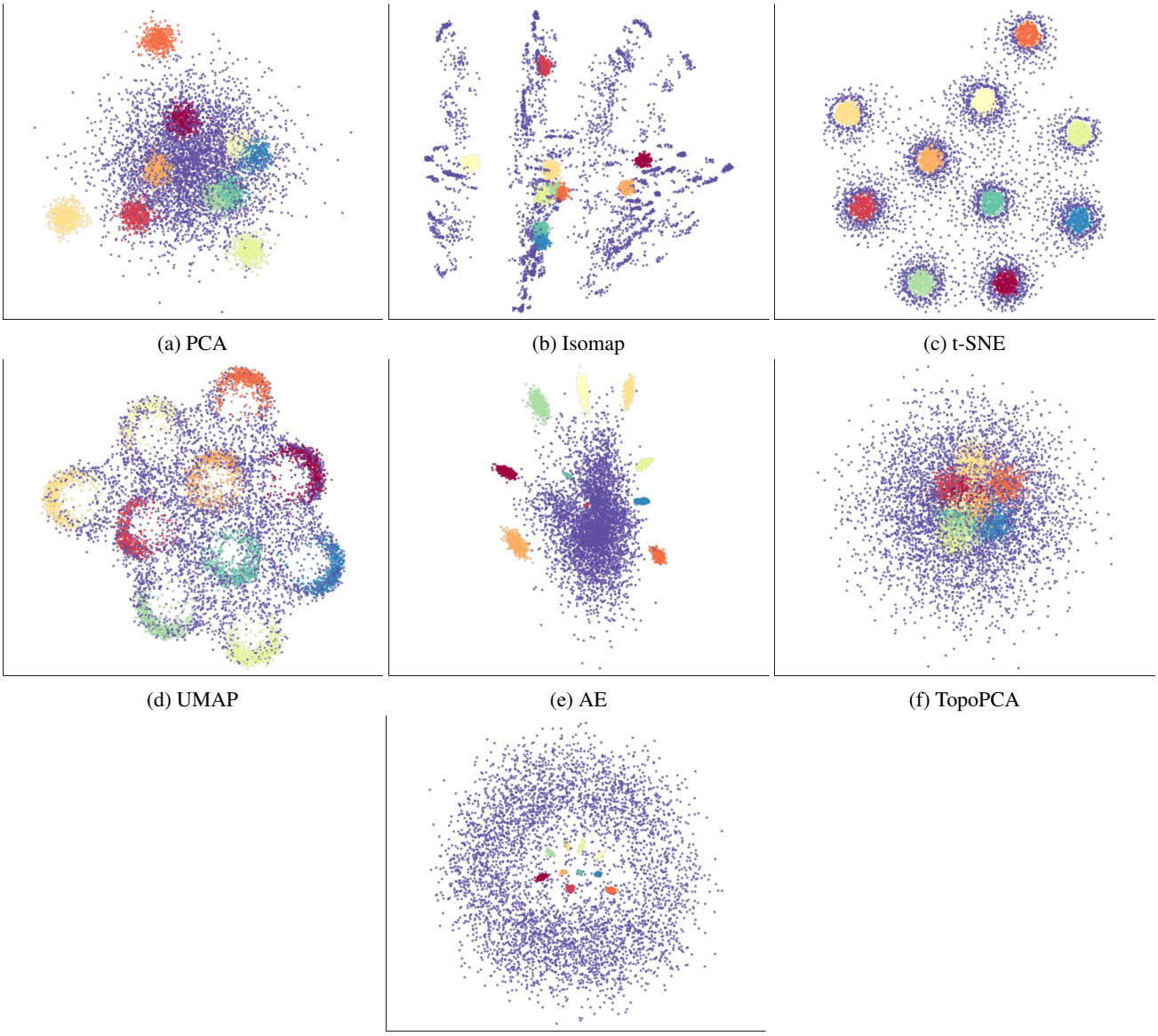
(a) $\mathcal{X}$ (b) $\mathcal{Z}$

Figure A.4. An undesirable configuration of the latent space of three non-collinear points, resulting in equal persistence diagrams for $\mathcal{X}$ and $\mathcal{Z}$. Pairwise distances are shown as dotted lines. We prevent this by not *explicitly* minimising the distances between persistence diagrams but by including persistence pairings.

that the latent space consists of the same triangle (in terms of its side lengths) but with permuted labels. A loss term of the form

$$\mathcal{L}' := \left\| \mathbf{A}^X \left[\pi^X \right] - \mathbf{A}^Z \left[\pi^Z \right] \right\|^2 \tag{31}$$

only measures the distance between persistence diagrams (which would be zero in this situation) and would not be able to penalise such a configuration.

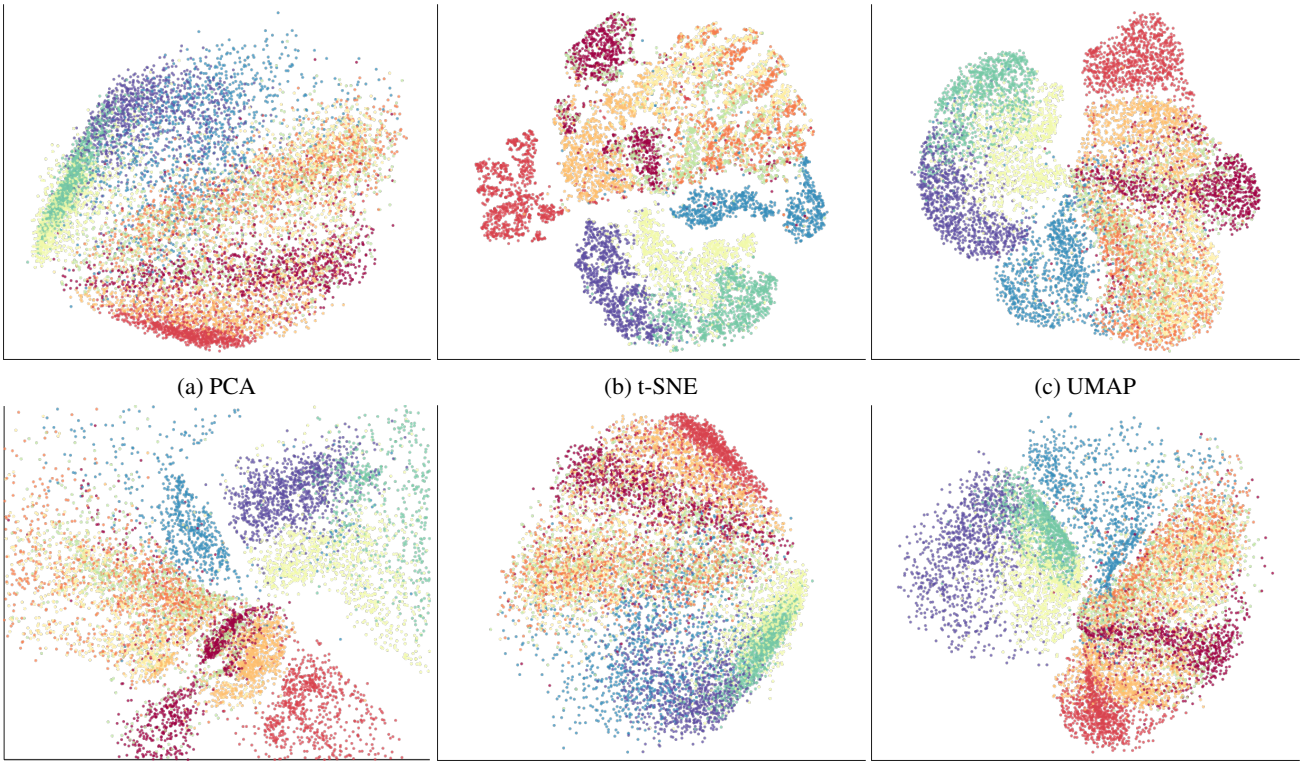


(a) PCA (b) Isomap (c) t-SNE

(d) UMAP (e) AE (f) TopoPCA

(g) TopoAE

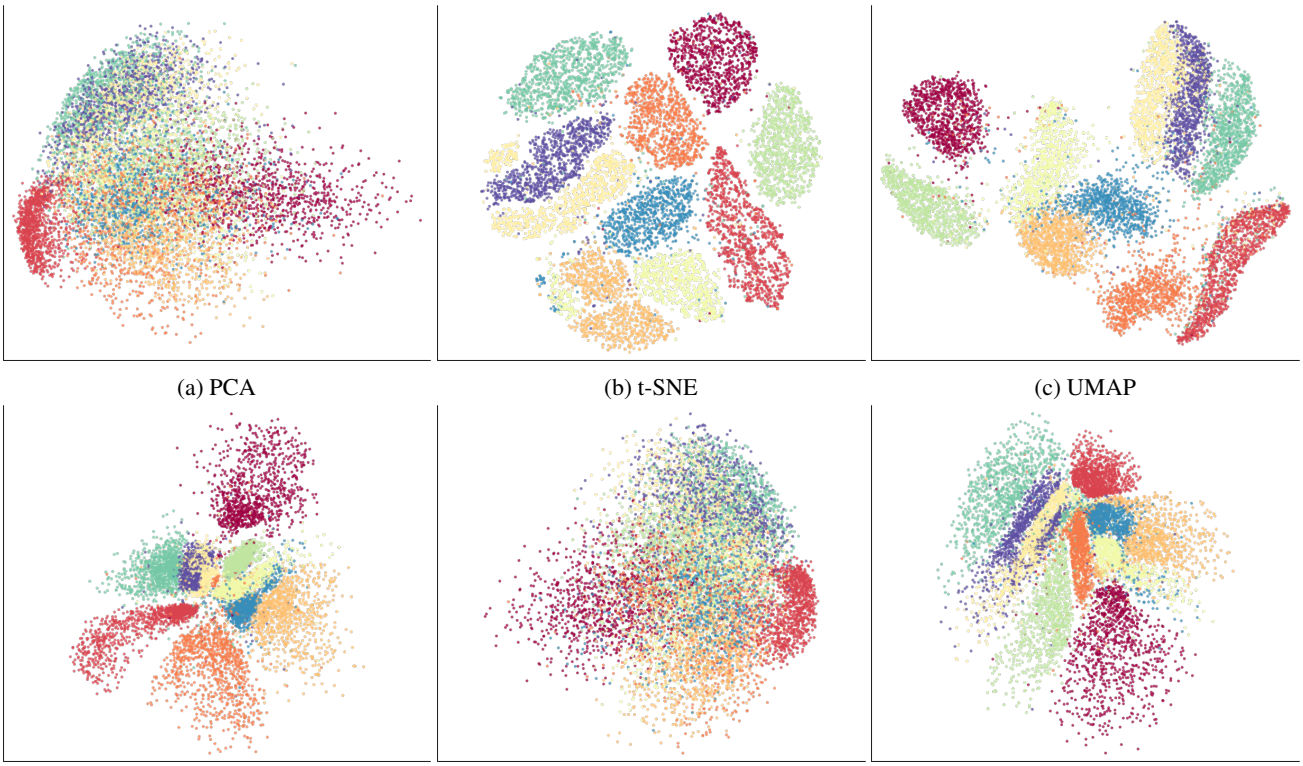
Figure A.5. A depiction of *all* latent spaces obtained for the SPHERES data set. TopoAE used a batch size of 28. This is an enlarged version of the figure shown in Section 5.2.



(a) PCA (b) t-SNE (c) UMAP

(d) AE (e) TopoPCA (f) TopoAE

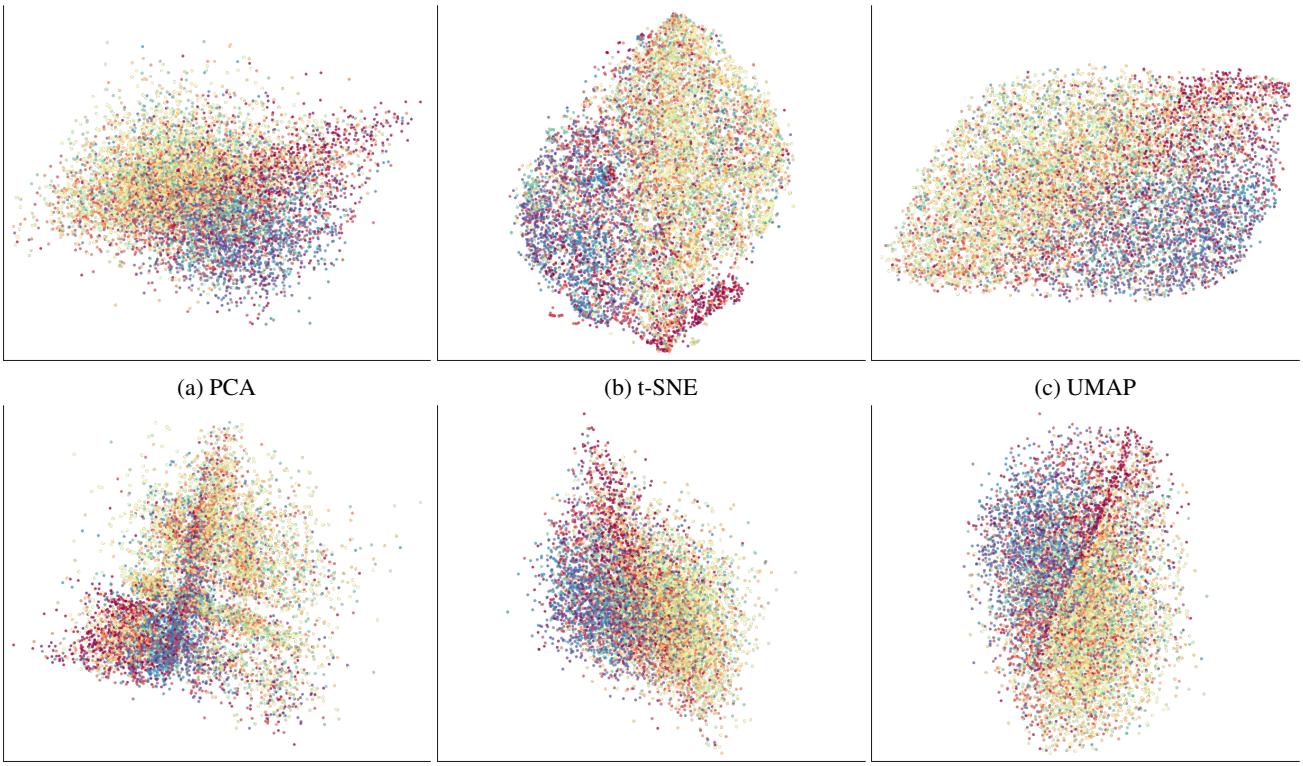
Figure A.6. Latent representations of the FASHION-MNIST data set. TopoAE used a batch size of 95. This is a larger extension of the figure shown in Section 5.2.



(a) PCA (b) t-SNE (c) UMAP

(d) AE (e) TopoPCA (f) TopoAE

Figure A.7. Latent representations of the MNIST data set. TopoAE used a batch size of 126. This is a larger extension of the figure shown in Section 5.2.



(a) PCA (b) t-SNE (c) UMAP

(d) AE (e) TopoPCA (f) TopoAE

Figure A.8. Latent representations of the CIFAR-10 data set. TopoAE used a batch size of 82. This is a larger extension of the figure shown in Section 5.2.

Data set	Method	$\mathrm{KL}_{0.001}$	$\mathrm{KL}_{0.01}$	$\mathrm{KL}_{0.1}$	KL_{1}	KL_{10}	ℓ-Cont	ℓ-MRRE	ℓ-Trust	ℓ-RMSE	Data MSE
SPHERES	Isomap	0.53095±0.01929	0.18096±0.02547	0.42048±0.00559	0.00881±0.00020	0.000089±0.000002	**0.79027±0.00244**	**0.24573±0.00158**	**0.67643±0.00323**	10.37188±0.22856	–
	PCA	**0.22445±0.00691**	0.33231±0.00552	0.65121±0.00256	0.01530±0.00010	0.000159±0.000001	0.74740±0.00140	0.29402±0.00108	0.62557±0.00066	11.76482±0.01460	0.96103±0.00029
	TSNE	**0.22794±0.00722**	**0.15228±0.00805**	0.52722±0.03261	0.01271±0.00058	0.000133±0.000006	0.77300±0.00513	**0.21740±0.00472**	**0.67862±0.00474**	**8.05018±0.11057**	–
	UMAP	0.24752±0.01917	0.15687±0.00599	0.61326±0.00752	0.01658±0.00028	0.000178±0.000003	0.75153±0.00360	0.24968±0.00094	0.63483±0.00185	**9.27009±0.03417**	–
	Vanilla	0.28432±0.02165	0.56571±0.02864	0.74588±0.04323	0.01664±0.00115	0.000172±0.000013	0.60663±0.01685	0.34918±0.00903	0.58843±0.00475	13.33061±0.05198	**0.81545±0.00106**
	TopoPCA	0.43344±0.01823	0.17837±0.00888	**0.39816±0.01178**	**0.00866±0.00025**	**0.000087±0.000003**	0.77320±0.00135	0.32765±0.00158	0.62260±0.00251	11.91542±0.56134	0.97305±0.00067
	TopoAE	0.62765±0.05415	**0.08504±0.01270**	**0.32572±0.02050**	**0.00694±0.00055**	**0.000069±0.000006**	**0.82200±0.01813**	0.27239±0.01108	0.65775±0.01428	13.45753±0.04177	**0.86812±0.00074**
F-MNIST	PCA	0.22559±0.00011	**0.35594±0.00011**	**0.05205±0.00004**	**0.00069±0.00000**	**0.000007±0.000000**	0.96777±0.00001	0.05744±0.00001	0.91681±0.00003	**9.05121±0.00041**	0.18439±0.00000
	TSNE	**0.03516±0.00226**	0.40477±0.01251	0.07095±0.00962	0.00198±0.00026	0.000023±0.000003	0.96731±0.00268	**0.01962±0.00073**	**0.97405±0.00070**	41.25460±0.53671	–
	UMAP	**0.05069±0.00238**	0.42362±0.00609	0.06491±0.00161	0.00163±0.00005	0.000019±0.000001	**0.98126±0.00016**	0.02867±0.00034	0.95874±0.00060	**13.68933±0.02896**	–
	Vanilla	0.17177±0.13603	0.47798±0.09567	0.06791±0.00700	0.00125±0.00017	0.000014±0.000002	0.96849±0.00372	**0.02562±0.00217**	**0.97418±0.00119**	20.70674±3.56861	**0.10197±0.00222**
	TopoPCA	0.18857±0.00197	**0.36201±0.00186**	**0.05296±0.00045**	**0.00083±0.00002**	**0.000009±0.000000**	0.97030±0.00013	0.05584±0.00022	0.91790±0.00034	20.88881±0.29929	0.18315±0.00002
	TopoAE	0.11039±0.02948	0.39204±0.03264	0.05353±0.00959	0.00100±0.00015	0.000011±0.000002	**0.97998±0.00194**	0.03156±0.00253	0.95612±0.00391	20.49122±0.93206	**0.12071±0.00238**
MNIST	PCA	0.16754±0.00051	0.38876±0.00146	0.16301±0.00059	0.00160±0.00001	**0.000016±0.000000**	0.90084±0.00016	0.16582±0.00022	0.74546±0.00048	**13.17437±0.00216**	0.22269±0.00002
	TSNE	**0.03767±0.00140**	**0.27695±0.05266**	**0.13266±0.02362**	0.00214±0.00041	0.000024±0.000004	0.92101±0.00288	**0.03953±0.00129**	**0.94624±0.00147**	22.89261±0.24373	–
	UMAP	**0.07214±0.00091**	**0.32063±0.00320**	0.14568±0.00207	0.00234±0.00004	0.000027±0.000001	**0.93992±0.00066**	**0.05109±0.00022**	**0.93770±0.00039**	**14.61535±0.04332**	–
	Vanilla	0.44690±0.08540	0.61993±0.11742	0.15542±0.02203	**0.00156±0.00023**	0.000016±0.000002	0.91293±0.00564	0.05828±0.00353	0.93699±0.00262	18.18105±0.21459	**0.13732±0.00160**
	TopoPCA	0.16138±0.00716	0.39157±0.01283	0.15556±0.00313	0.00162±0.00007	0.000017±0.000001	0.90301±0.00057	0.16297±0.00049	0.75040±0.00091	17.33353±0.82592	0.22477±0.00009
	TopoAE	0.32427±0.03312	0.34069±0.03056	**0.11012±0.01069**	**0.00114±0.00010**	**0.000012±0.000001**	**0.93210±0.00132**	0.05553±0.00044	0.92844±0.00142	19.57784±0.01812	**0.13884±0.00066**
CIFAR	PCA	0.27320±0.00014	0.59073±0.00004	**0.01961±0.00001**	**0.00023±0.00000**	**0.000002±0.000000**	**0.93130±0.00000**	0.11921±0.00005	0.82117±0.00002	**17.71567±0.00084**	0.14816±0.00000
	TSNE	**0.04451±0.00222**	0.62733±0.01427	0.03014±0.00333	0.00073±0.00007	0.000009±0.000001	0.90300±0.00611	**0.10265±0.00242**	**0.86325±0.00151**	**25.61099±0.11551**	–
	UMAP	**0.06934±0.00202**	0.61673±0.00052	0.02562±0.00019	0.00050±0.00001	0.000006±0.000000	0.92045±0.00013	0.12680±0.00028	0.81668±0.00019	33.57785±0.00796	–
	Vanilla	0.37737±0.06507	0.66834±0.02992	0.03458±0.00448	0.00062±0.00021	0.000007±0.000002	0.85072±0.00429	0.13204±0.00316	**0.86359±0.00442**	36.26827±0.56159	**0.14030±0.00190**
	TopoPCA	0.27207±0.00619	**0.58401±0.00515**	0.01973±0.00040	**0.00024±0.00001**	**0.000003±0.000000**	0.92439±0.00115	0.12607±0.00133	0.81551±0.00139	30.73848±1.35231	0.15002±0.00076
	TopoAE	0.20877±0.00951	**0.55642±0.00412**	**0.01879±0.00051**	0.00031±0.00002	0.000003±0.000000	**0.92691±0.00100**	**0.10809±0.00210**	0.84514±0.00359	37.85914±0.03303	**0.13975±0.00171**

Table A.2. Extended version of the table from the main paper, showing more length scales and variance estimates.