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 dth 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 dth homology group is known as the dth 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 dth *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 dth 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 dth *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.

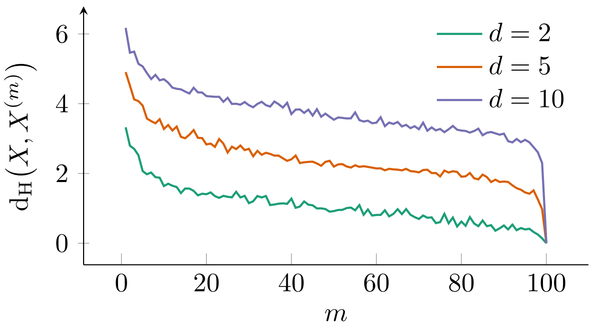


Figure A.1. Empirical convergence rate (mean) of the Hausdorff distance for a subsample of size m of 100 points in a d-dimensional space, following a standard normal distribution.

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

$$\mathrm{d_b}\left(\mathcal{D}^X, \mathcal{D}^Y\right) \le 2\,\mathrm{d_{GH}}(X, Y), \tag{16}$$

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 $\mathrm{d_{GH}}(X,Y) \le \mathrm{d_H}(X,Y)$, so Eq. 15 leads to

$$\mathrm{d_b}\left(\mathcal{D}^X, \mathcal{D}^Y\right) \le 2\,\mathrm{d_H}(X, Y), \tag{17}$$

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

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

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 $\mathrm{d_H}\left(X, X^{(m)}\right) \le \mathrm{diam}(X)$ 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:*

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

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 $\mathrm{d_H}\left(X, X^{(m)}\right)$.

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.

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

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 $m = 1$. $\square$

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 validation 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 $\mathrm{f}_\sigma^{\mathcal{X}}(x) := \sum_{y \in \mathcal{X}} \exp\left(-\sigma^{-1} \operatorname{dist}(x, y)^2\right)$, 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).

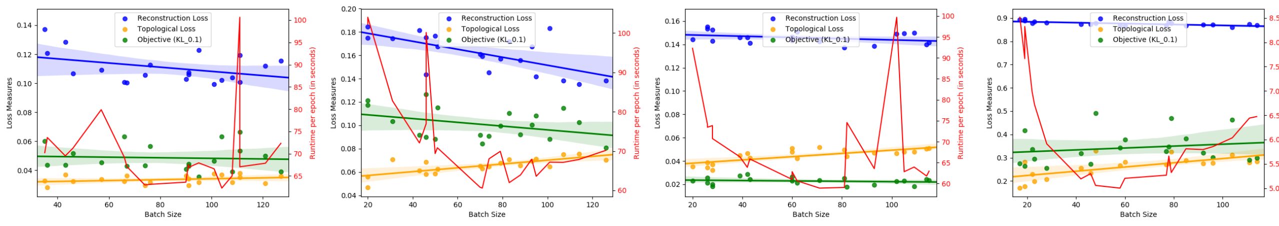
For the loss measures, we see that reconstruction loss tends to *decrease* with increasing batch size, while our topological loss tends to *increase* with increasing batch size (despite normalization). The second observation might be due to larger batch size enabling more complex data point arrangements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

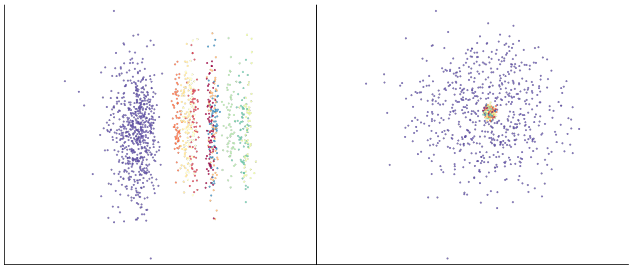
A.10. Topological Distance Calculations

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 1st 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.

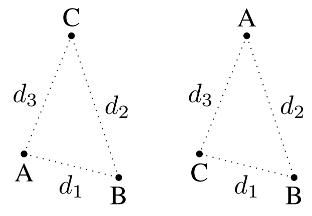


(a) VAE (b) TopoVAE

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.

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.

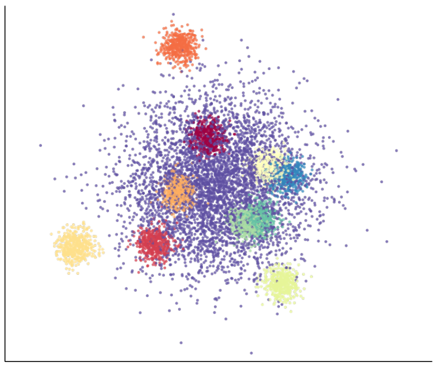
(ii) the 2nd Wasserstein distance (W_2), and (iii) the bottleneck distance (W_∞) between the persistence diagrams obtained from the test data set of the SPHERES data and their resulting 2D latent representations. Even though our loss function is *not* optimising this distance, we observe in Table A.1 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.

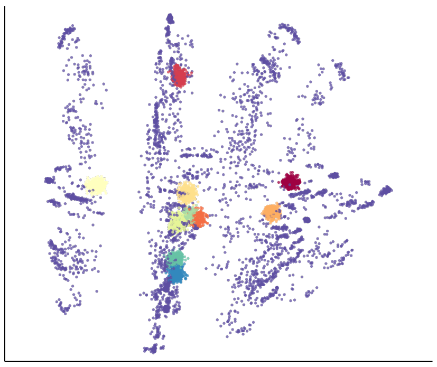
A.11. Alternative Loss Formulations

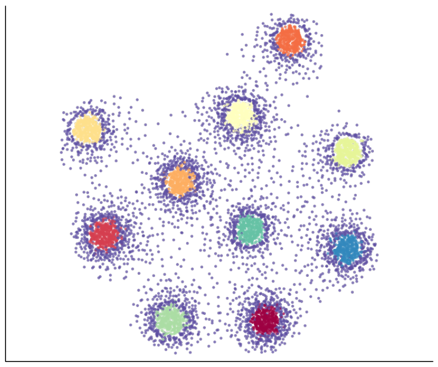
Our choice of loss function was motivated by the observation that *only* aligning the persistence diagrams between mini-batches of $\mathcal{X}$ and $\mathcal{Z}$ can lead to degenerate or "meaningless" latent spaces. As a simple example (see Figure A.4 for a visualisation), imagine three non-collinear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

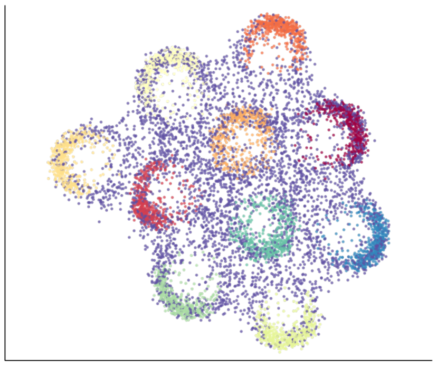
$$\mathcal{L}' := \left\| \mathbf{A}^X\left[\pi^X\right] - \mathbf{A}^Z\left[\pi^Z\right] \right\|^2 \tag{31}$$

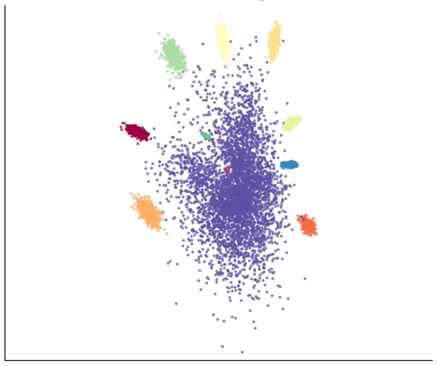
only measures the distance between persistence diagrams (which would be zero in this situation) and would not be able to penalise such a configuration.

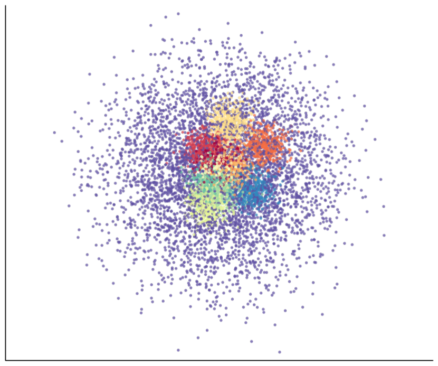

(a) PCA


(b) Isomap


(c) t-SNE


(d) UMAP


(e) AE


(f) TopoPCA

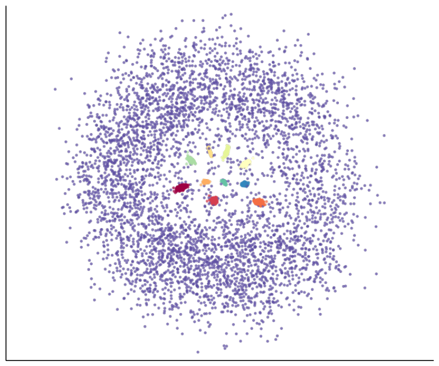
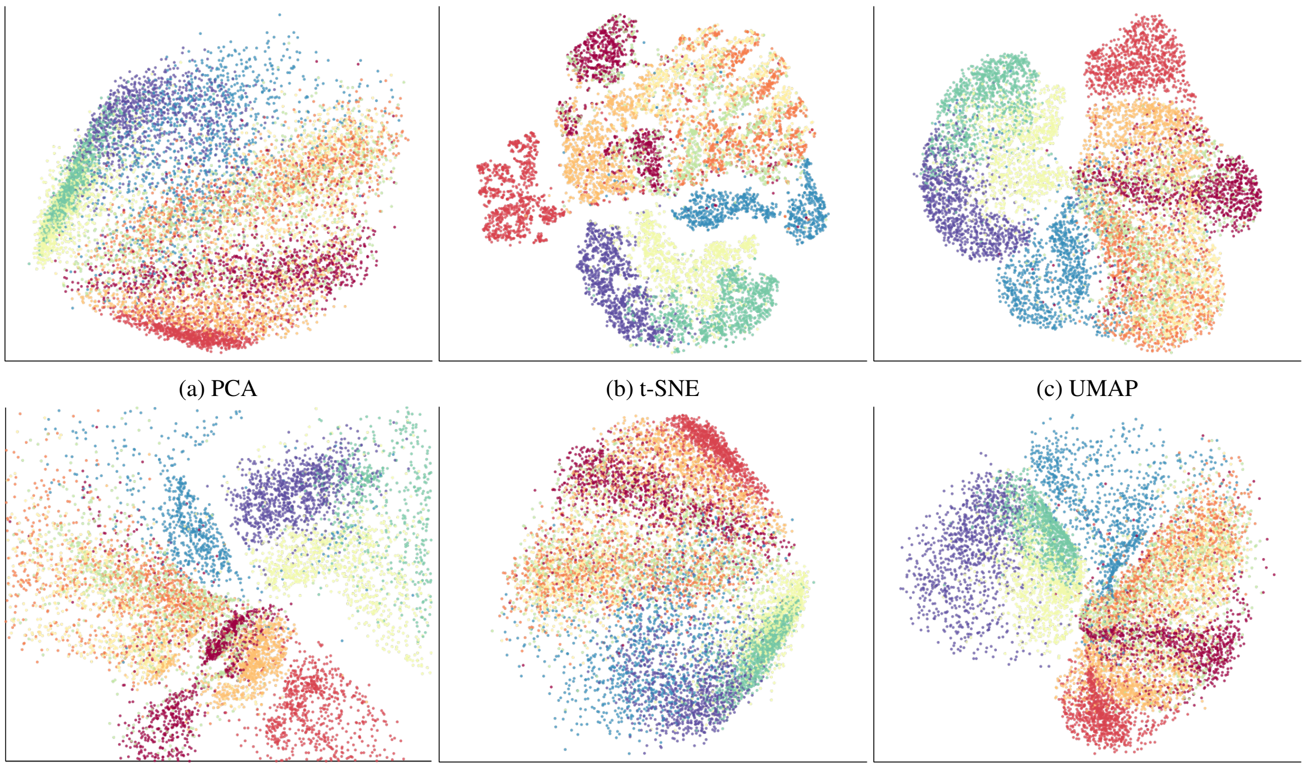

(g) TopoAE

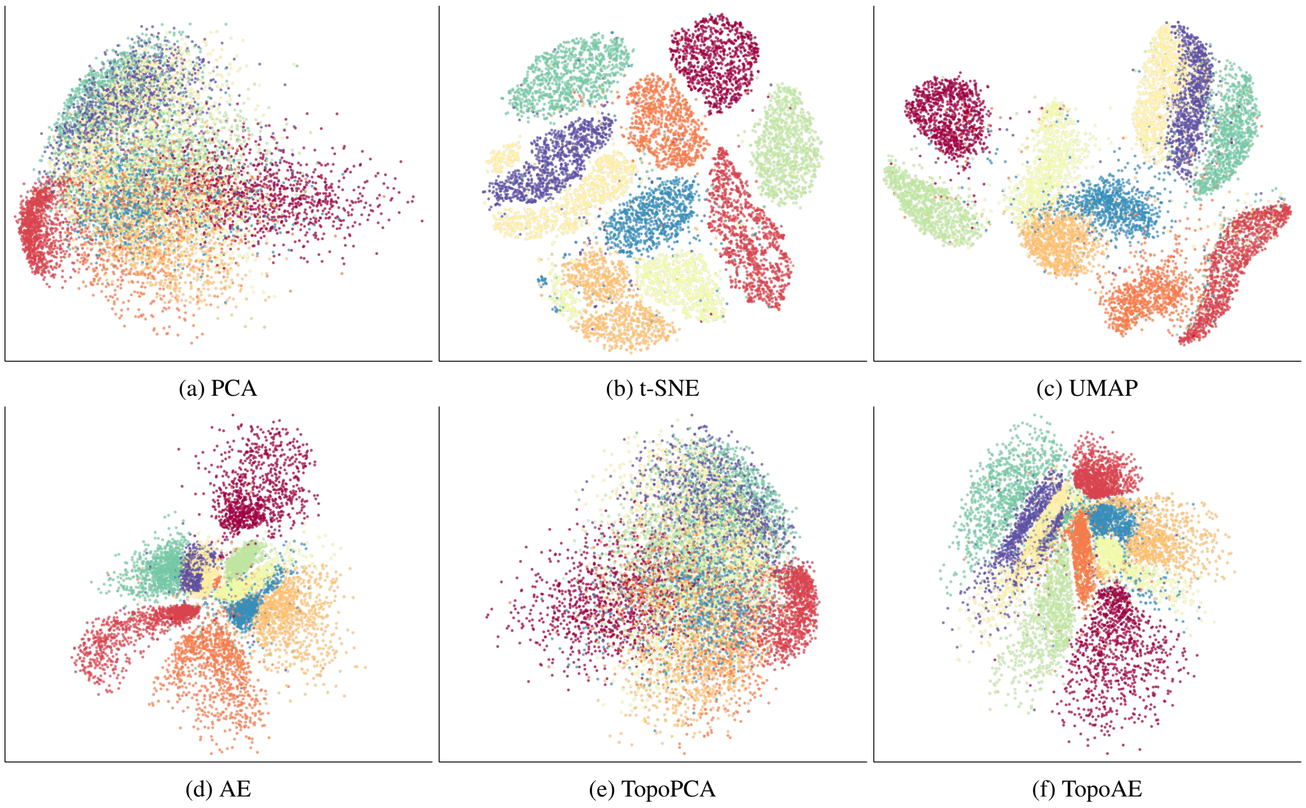
Figure A.5. A depiction of *all* latent spaces obtained for the SPHERES data set. TopoAE used a batch size of 28. This is an enlarged version of the figure shown in Section 5.2.



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

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

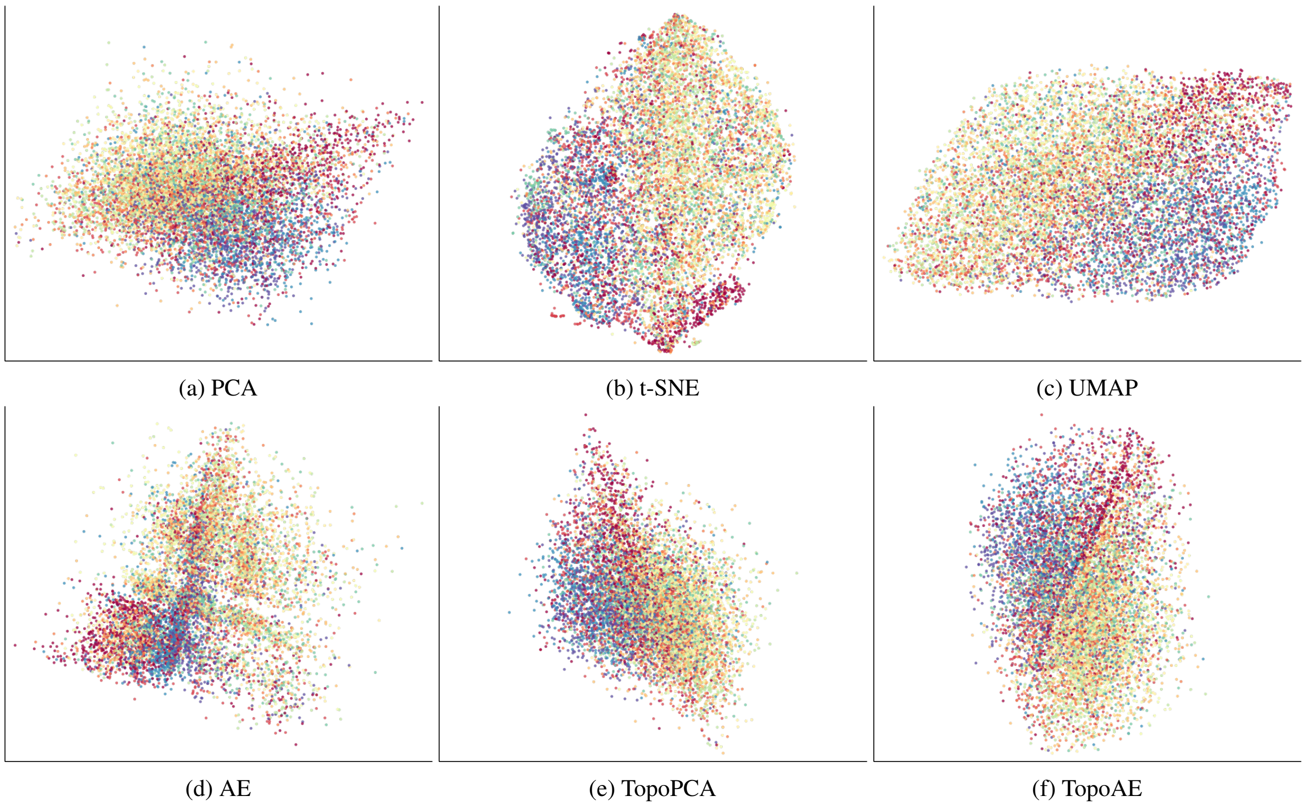
Figure A.6. Latent representations of the FASHION-MNIST data set. TopoAE used a batch size of 95. This is a larger extension of the figure shown in Section 5.2.



(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.