Topological Autoencoders

Kelan Gray, Bernat Jordà Carbonell

May 26, 2024

Abstract

We introduce topological autoencoders, proposed by [7]. This model adapts the reconstruction loss of traditional autoencoders to encourage the preservation of topological features between the input and latent spaces. We demonstrate that topological autoencoders achieve competitive results on synthetic and real image datasets compared to state-of-the-art non-linear dimension reduction (NLDR) algorithms. However, we identify limitations with this approach and find that purely probabilistic models can often capture topological features as effectively as topological autoencoders.

1 Introduction

Tools from topological data analysis, particularly persistent homology, are increasingly used in machine learning [2]. However, persistent homology computations are inherently discrete, making direct application as a constraint in machine learning challenging due to the need for differentiability. In dimensionality reduction, it is desirable to preserve topological features, which algorithms like UMAP achieve [6]. Autoencoders, a deep learning generalization of principal component analysis (PCA) [1], often fail to preserve topological features during training.

Topological autoencoders [7] address these issues by incorporating a differentiable topological loss, enabling consideration of topological information during deep neural network training. This approach employs persistent homology to identify "topologically relevant simplices," which are used to measure how the thresholds at which these simplices appear change between the original data space and the latent space of the autoencoder. By leveraging the resilience of persistence diagrams to noise, the model ensures a differentiable topological loss, facilitating its integration into the training process. This topological loss is stable under batch processing, making it suitable for large-scale training in deep learning.

In addition to reviewing the work of [7], our contributions are:

- We deepen the theoretical framework and identify limitations of the topological loss with a counterexample.
- We show that topological autoencoders do not uniquely capture the SPHERES dataset's topological structures.
- We identify limitations of the Bottleneck and Sliced Wasserstein distances for evaluating dimensionality reduction algorithms.

2 Background

We present foundational theory on autoencoders and persistent homology necessary for understanding topological autoencoders.

2.1 Autoencoders

Autoencoders are neural networks designed for unsupervised learning tasks aimed at reconstructing input data. They encode the input into a lower-dimensional latent space and decode it to approximate the original input. The encoding and decoding processes are performed by neural networks known as the encoder $f : \mathbb{R}^n \to \mathbb{R}^d$ and decoder $g : \mathbb{R}^d \to \mathbb{R}^n$, respectively. The objective is to minimise the expected reconstruction loss

$$\arg\min_{f,g} \mathbb{E}_{\mathbf{x}\sim D}[\mathcal{L}_r(\mathbf{x}, g \circ f(\mathbf{x}))], \tag{1}$$

where d is the distribution of $\mathbf{x}$, and $\mathcal{L}_r$ is the reconstruction loss. If f and g consist of linear operations without non-linear activation functions, autoencoders resemble Principal Component Analysis (PCA) in terms of latent space representation. Thus, autoencoders generalize PCA [1]. Care must be taken to avoid $g \circ f$ collapsing to the identity operator, which regularization techniques address. For further details, refer to [1].

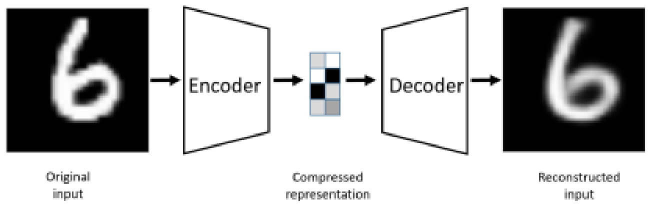


Figure 1: Visualisation of the autoencoder architecture. The image is encoded into the latent space and then decoded back to create a new image. (Image source: [1]).

2.2 Persistent Homology

Persistent homology calculates the birth and death of topological features in a dataset. The algorithm orders simplices by appearance and uses a matrix of the ordered simplices to determine when these features are born and die.

Topological autoencoders leverage this by storing persistence pairings that identify the simplices creating and destroying topological features, denoted as b and d respectively. Each

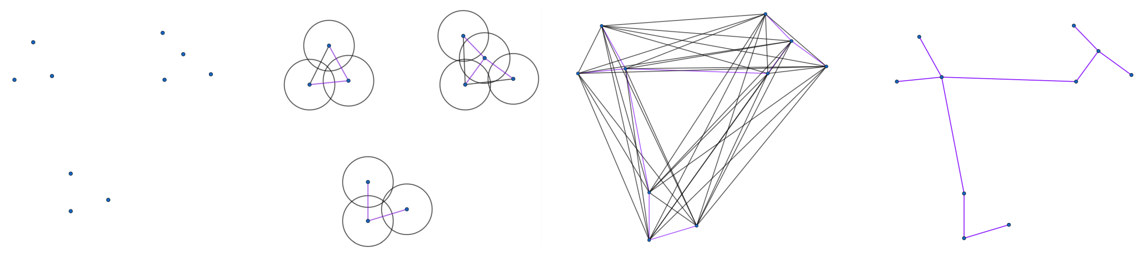

(a) ϵ_0 (b) ϵ_1 (c) ϵ_2 (d) Topologically relevant features

Figure 2: Vietoris-Rips adds edges and identifies those that alter the number of connected components (highlighted in pink). In the final picture, these edges form the minimum spanning tree, representing the death of the 0-dimensional features.

topological feature is parameterized by two simplices and represented in the persistence diagram as the point (ϵ_b, ϵ_d), where ϵ_x refers to the value at which the simplex x appears.

If A^X is the distance matrix assigning appearance thresholds to simplices, and π^X contains the pairings characterizing topological features, then the persistence diagram $\mathbb{D}^X$ is $A^X[\pi^X]$, assigning to each topological feature the thresholds at which it is born and dies.

To focus on "topologically relevant" edges, we can fix a maximum dimension d and examine the topological features of dimension $\leq d$. For dimension 0 features, TopoAE observes the edges that connect previously separated connected components. This is equivalent to constructing a minimum spanning tree, which represents the closest distances that bridge over connected components. This tree, with its weights, provides an intuitive representation of the dataset's topological structure. Figure 2 illustrates this process.

3 Topological Autoencoders

In this section, we present the theory behind topological autoencoders, including the formulation of the topological loss and its stability results. Figure 3 illustrates the topological autoencoder framework.

3.1 Topological Loss

Topological autoencoders compute the persistent homology of mini-batches in both data and latent spaces, comparing the resulting diagrams to calculate the topological loss. This loss is added to the autoencoder loss, encouraging the autoencoder to preserve topological features.

The direct approach would be to compute the Bottleneck or another distance between the two diagrams. However, this raises problems because some edges may be topologically significant in one diagram but not in the other, leading to potentially irrelevant feature matching. Additionally, considering only edges present in both diagrams is impractical due to the large number of edges.

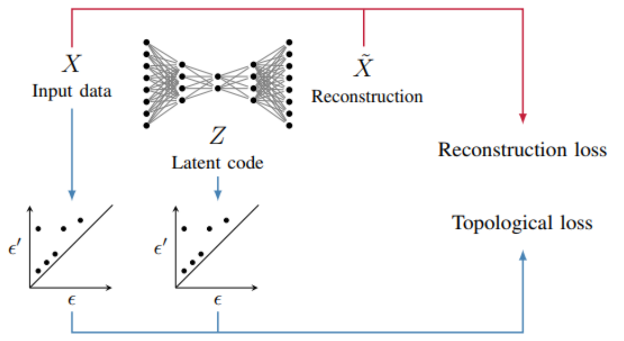


Figure 3: Architecture of the topological autoencoder. The autoencoder is trained to reconstruct the input data $\tilde{X}$ with a specific reconstruction loss, while persistence diagrams for the input and latent spaces are used to compute the topological loss. (Image source: [7]).

These issues arise because finding the best matching between features from different spaces can be computationally intensive and may not reflect true topological relationships. Instead of finding the best matching between potentially unrelated features, we focus on the topologically relevant features and observe how they change. We must consider the edges that are relevant in both the original and latent spaces. The topological loss cn then be decomposed into

$$\mathcal{L}_t = \mathcal{L}_{\mathcal{X}\to\mathcal{Z}} + \mathcal{L}_{\mathcal{Z}\to\mathcal{X}},$$

where

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

, and

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

$\mathbf{A}^X$ and $\mathbf{A}^Z$ represent the distance matrices in the original and latent spaces, respectively. π^X and π^Z represent the persistence pairings containing the indices of the topologically relevant edges in the original and latent spaces, respectively.

Both terms measure the change in the topologically relevant distances of each space. Minimizing these values aims to preserve these distances, ensuring three key aspects: the matching aligns distances between the two spaces, slight perturbations to the persistence diagram leverage its noise resilience to isolate each point, and the gradient can be computed efficiently.

3.2 Gradient Computation

Let $\boldsymbol{\theta}$ denote the parameters of the encoder, and define $\boldsymbol{\rho} := \left(\mathbf{A}^X\left[\pi^X\right] - \mathbf{A}^Z\left[\pi^X\right]\right)$. Then, the gradient of the loss $\mathcal{L}_{\mathcal{X}\to\mathcal{Z}}$ with respect to $\boldsymbol{\theta}$ is given by

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

$$= -\boldsymbol{\rho}^\top\left(\frac{\partial \mathbf{A}^Z\left[\pi^X\right]}{\partial \boldsymbol{\theta}}\right) \tag{3}$$

$$= -\boldsymbol{\rho}^\top\left(\sum_{i=1}^{|\pi^X|}\frac{\partial \mathbf{A}^Z\left[\pi^X\right]_i}{\partial \boldsymbol{\theta}}\right), \tag{4}$$

where $|\pi^X|$ represents the cardinality of a persistence pairing, and $\mathbf{A}^Z\left[\pi^X\right]_i$ denotes the i-th entry of the vector of paired distances. A similar derivation applies to $\mathcal{L}_{\mathcal{Z}\to\mathcal{X}}$ (with π^X replaced by π^Z).

3.3 Stability

Autoencoders train on mini-batches rather than the entire dataset to improve speed and overcome memory constraints. However, it is uncertain if subsampled persistence diagrams approximate the persistence diagram of the original data accurately. The following theorem confirms this approximation.

Theorem 1. *Let X be a point cloud with $|X| = n$, and $X^{(m)}$ be a subsample with $|X^{(m)}| = m$. Let $\mathcal{D}^X$ and $\mathcal{D}^{X^{(m)}}$ be their respective persistence diagrams. Then:*

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

where d_H is the Hausdorff distance [7].

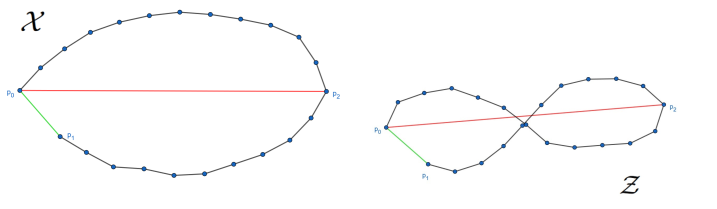
Moreover,

$$\lim_{m\to n}\mathbb{E}[d_H(\mathcal{D}^X, \mathcal{D}^{X^{(m)}})] = 0$$

[7]. Therefore, persistence diagrams computed on batches provide good approximations of the original data's persistence diagram. This confirms that our topological loss is well-suited for computation in mini-batches.

3.4 Topological Loss Limitations

The topological loss discussed does not consider all edges for computationally feasible, but in exchange there is some information loss. It is important to remark that perfect alignment of the two spaces results in $\mathcal{L}_{\mathcal{X}\to\mathcal{Z}} = \mathcal{L}_{\mathcal{Z}\to\mathcal{X}} = 0$ 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 may not be identical. To illustrate this, we provide a simple counterexample in figure 4.



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

Figure 4: An illustrative example depicts data in the input space and its latent space representation. Despite significant changes in the data's topology between input and latent spaces, the distances of topologically relevant edges in $\mathcal{X}$ do not change, resulting in $\mathbf{A}^X\left[\pi^X\right] = \mathbf{A}^Z\left[\pi^X\right]$. Thus $\mathcal{L}_{\mathcal{X}\to\mathcal{Z}} = 0$.

4 Experiments

The goal of this section is to empirically compare topological autoencoders with state-of-the-art NLDR algorithms.

4.1 Experimental Setup

We first discuss the datasets used in this experiment, followed by the NLDR models considered, their training procedures, and hyperparameter tuning. We conclude with the evaluation methods for the models.

4.1.1 Datasets

We consider the SPHERES dataset, consisting of ten 100-dimensional spheres within a 101-dimensional sphere. The larger sphere contains twice as many datapoints as the smaller spheres combined [7]. This configuration facilitates assessing our models' ability to capture dataset topology. We also consider the MNIST handwritten images dataset, which is relevant since real-world images are known to lie on or near low-dimensional manifold [8].

4.1.2 Models

We compare t-SNE [9], UMAP [6], and a standard Autoencoder(AE) against our topological autoencoder(topoAE), which integrates a topological loss term into the AE framework. We utilize two latent dimensions for visualization purposes and partition the data into 60-20-20 training, validation, and testing sets. Hyperparameters for each model are tuned using the validation set.

4.1.3 Model Architecture and training

For the SPHERES dataset, both AE and topoAE employ the same architecture, consisting of two hidden layers with 32 neurons each in both the encoder and decoder, resulting in a 32-32-2-32-32 configuration. Conversely, for the MNIST dataset, a three-layer architecture is utilized, with decreasing and then increasing sizes of hidden neurons in the encoder and decoder, respectively, forming a configuration of 1000-500-250-2-25-500-1000. ReLU activation functions are applied after each layer, followed by batch normalization. The training procedure involves minimizing the mean-squared error loss. Both AE and topoAE models are optimized using the ADAM optimizer [5]. We fix the batch size to 128 and normalise our topological loss term by the batch size in order to disentangle λ from it [7]. Training is conducted for 20 epochs on both datasets.

For t-SNE and UMAP, default parameters are maintained, with only their respective hyperparameters, *perplexity*, and *n_ neighbors*, adjusted accordingly.

4.1.4 Hyperparameter Tuning

Bayesian optimization is employed to optimize hyperparameters on the validation set, with the objective function being the KL divergence $KL_{\sigma=0.1}$. The KL divergence is computed using kernel density estimation for both input and latent data, with a bandwidth parameter σ. For t-SNE, the *perplexity* hyperparameter is varied from 5 to 50 in the dataset, while for UMAP, the parameter *n_ neighbors* is varied over 15 to 500. In topoAE, the hyperparameter λ is varied from 1 to 3. Hyperparameter tuning is not performed for the fixed architecture of the AE.

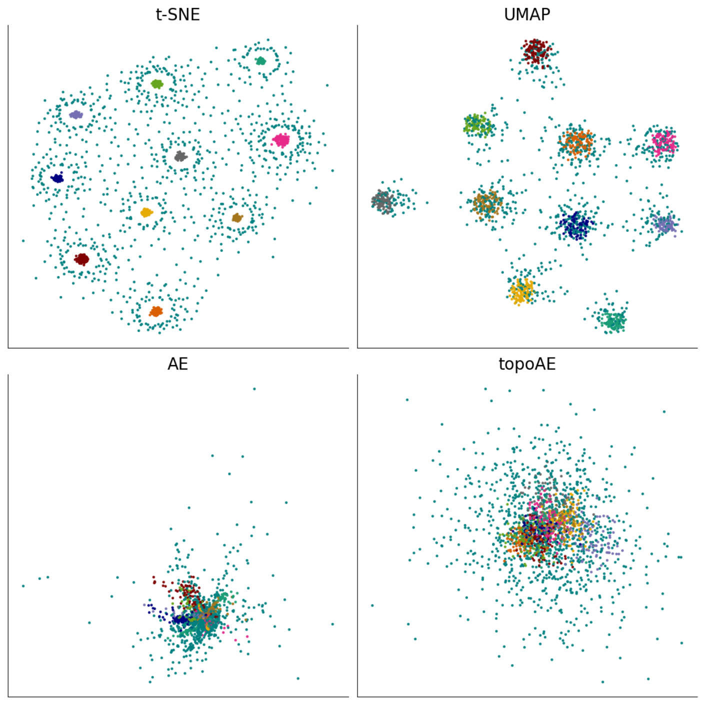


Figure 5: Latent representation of SPHERES dataset. The topoAE method successfully captures significant aspects of the original topology in its latent representation. Competitor methods like t-SNE and UMAP tend to fragment the larger sphere, failing to preserve the original topology.

4.1.5 Evaluation

The evaluation of our model comprises three main aspects: local metrics, global (topological) metrics, and 2D visualizations.

Local Metrics: These metrics assess the fidelity of the model's latent representation at a local level.

- *Root-Mean-Squared Error (RMSE):* This metric quantifies the discrepancy between the distance matrices of the input space and the latent space.
- *Trustworthiness:* A measure that penalizes unexpected nearest neighbors in the latent space in proportion to their rank in the input space [4].
- *KL-Divergence (KL_σ):* Calculated for $\sigma = 0.01, 0.1, 1$, this metric assesses the difference between probability distributions estimated using kernel density estimation from the input and latent spaces.

Global Metrics: These metrics evaluate the preservation of the overall topology of the data. We do this by considering the persistence diagrams in the input and latent space.

- *Bottleneck Distance:* This measures the maximum distance between corresponding points in the two persistence diagrams after a one-to-one matching has been established.
- *Sliced Wasserstein Distance:* This provides an approximation to the Wasserstein distance between persistence diagrams, which calculates the minimum cost needed to transform one diagram into the other [3].

4.2 Results

We present quantitative results comparing models' latent representations using standard NLDR metrics and topological distance metrics. Additionally, we examine qualitative results through 2D latent visualizations.

4.2.1 Quantitative results

Table 1 presents the quantitative results. TopoAE excels in the topological distance metrics, consistently ranking either first or second compared to the competitor models. It also has the lowest RMSE on both datasets, indicating that adding topological constraints to our autoencoder is not hindering the reconstruction error. We note that while TopoAE demonstrates robust KL-divergence results across multiple scales, the calculation of KL-divergence is highly sensitive to the bandwidth parameter, warranting cautious interpretation of these values.

The authors in [7] use 1st and 2nd Wasserstein distances and the Bottleneck distance to demonstrate how their method reduces the topological distance between the latent space and the original dataset. However, our closer examination revealed that the differences in these metrics between methods were smaller than expected (Table 1, last two columns). To gain a deeper understanding, we applied t-SNE to the SPHERES dataset with varying perplexity values and visualized the persistence diagrams. The results are shown in Figure 6.

The discrepancy between vastly different diagrams yielding similar results lies in the scale. Whereas the persistence diagram of the original dataset exhibits features that vanish at a threshold of approximately 25, the latest vanishing point in the other three diagrams occurs at a threshold of 2. Consequently, the Bottleneck distance measures the distance from

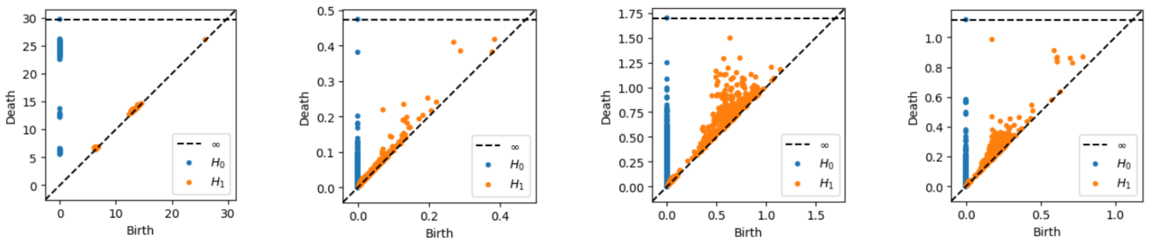

(a) Original (b) perplexity 5 (c) perplexity 255 (d) perplexity 750

Figure 6: The difference in scale between the original and processed data is significant. Due to this, even though the persistence diagrams appear quite similar across different perplexity levels, the Sliced-Wasserstein and Bottleneck distances show only minimal changes.

the highest point of the original diagram to the diagonal. Moreover, despite slight fluctuations in the Wasserstein distance, the matches being made are between points that are entirely unrelated: none of the points on either the original dataset diagram or the latent space diagrams have a point from the other diagram closer than the diagonal, implying that every single point is matched with the diagonal. This is the case with the case with different result of t-SNE at different levels of perplexity as you can see in 6 and we observed a very similar result with both UMAP and TopoAE.

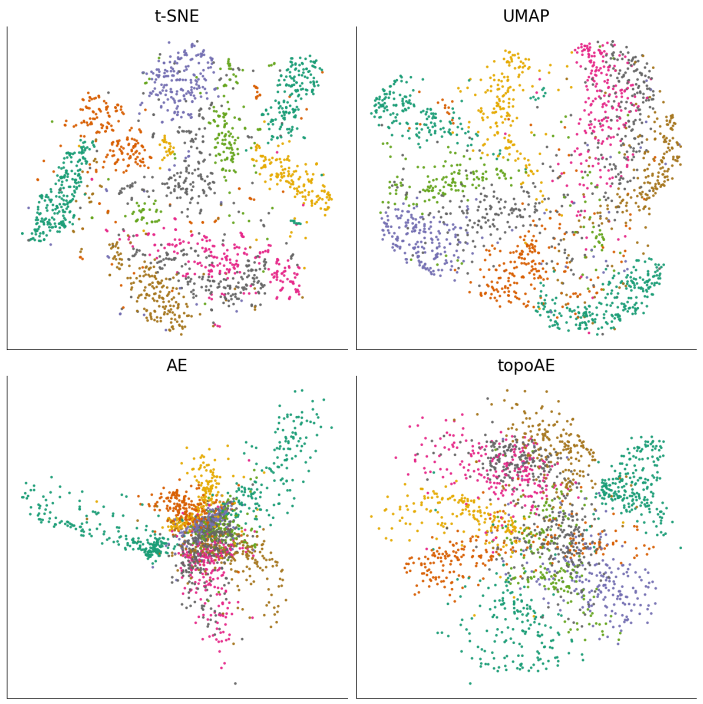


Figure 7: Latent representations on MNIST dataset. We observe that topoAE is able to identify more structure than the standard AE through it's regulariation term, yielding results similar to UMAP.

4.2.2 Qualitative results

Figure 5 shows the 2D latent representations of each model on the SPHERES dataset. TopoAE uniquely captures the topological properties of the original data, correctly identifying that the 10 smaller clusters are encapsulated by a larger grey sphere. In contrast, t-SNE and UMAP tend to distort the larger sphere, while AE compresses it into a cluster similar in size to the smaller ones. Although TopoAE introduces some noise and does not achieve a perfect circular structure as reported in [7], this is likely due to computational limitations rather than the method itself.

We initially restricted *perplexity* to a range of 5 to 50. However, as noted in [9], increasing perplexity can help capture the global structure of the data. Inspired by this, we ran t-SNE with a larger perplexity value of 750 on the SPHERES dataset. On the left of Figure 8, t-SNE clearly captures the complex topology of the SPHERES dataset. On the right of Figure 8, we plot the Sliced Wasserstein distance between persistence diagrams in the input and latent spaces against increasing perplexity values. For sufficiently large perplexity, the Sliced Wasserstein distance decreases, indicating that t-SNE becomes topology-informed.

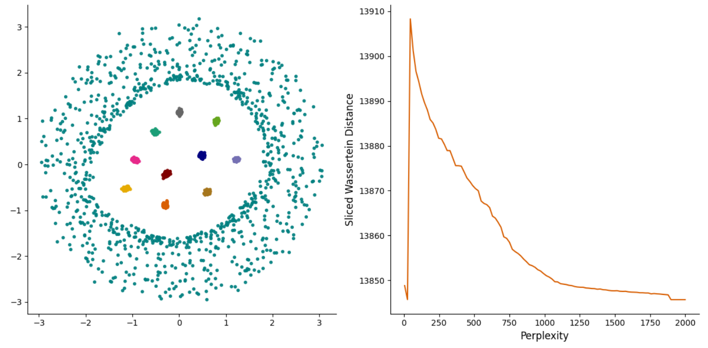


Figure 8: Left: Latent representation of SPHERES dataset via t-SNE (perplexity: 750) accurately captures original data topology. Right: Increasing perplexity reduces topological distances in Sliced Wasserstein distance between input and latent space persistence diagrams.

Figure 7 shows the latent representations of our models on the MNIST dataset. TopoAE provides competitive visualizations compared to the state-of-the-art algorithm UMAP, displaying clearly distinguishable clusters. While AE tends to compress clusters, resulting in less meaningful representations, TopoAE regularizes the autoencoder to preserve the structure of the original MNIST data. Additionally, t-SNE creates distinct clusters due to its repulsion effects [9]. However, this clustering effect is likely artificial and not reflective of the underlying manifold structure.

Dataset	Model	$KL_{0.01}$	$KL_{0.1}$	KL_1	Trustworthiness	RMSE	d_b	d_{SW}
SPHERES	t-SNE	$\underline{\mathbf{0.001391}}$	**0.1343**	0.1865	**0.6639**	8.461	$\underline{\mathbf{13.0615}}$	14035.74009
	UMAP	0.02501	$\underline{\mathbf{0.1046}}$	$\underline{\mathbf{0.004590}}$	$\underline{\mathbf{0.6934}}$	7.564	13.1304	13794.1151
	AE	**0.02144**	0.42840	0.2454	0.5842	**2.789**	13.095550	**13974.2799**
	topoAE	0.04614	0.2601	**0.05162**	0.5477	$\underline{\mathbf{2.049}}$	**13.09084**	$\underline{\mathbf{13732.88160}}$
MNIST	t-SNE	$\underline{\mathbf{0.00040908}}$	$\underline{\mathbf{0.03612}}$	**0.07139**	$\underline{\mathbf{0.9720}}$	11.50	$\underline{\mathbf{14.033036}}$	15616.526
	UMAP	0.01188	**0.08211**	$\underline{\mathbf{0.02113}}$	**0.9104**	23.79	$\underline{\mathbf{14.033036}}$	**15537.018**
	AE	0.01430	0.2949	0.2189	0.8191	**2.142**	**14.033038**	15546.01406
	topoAE	**0.008810**	0.1436	0.07642	0.8570	$\underline{\mathbf{1.493}}$	$\underline{\mathbf{14.033036}}$	**15539.73464**

Table 1: Comparison of latent space quality in SPHERES and MNIST datasets between models using several standard NLDR metrics and two topological distance metrics. For each metric, the winner is highlighted in bold and underlined, while the runner-up is emphasized in bold.

5 Conclusion

We presented topological autoencoders, as introduced by [7]. This paper augments the autoencoder's loss function with a topological loss term to measure topological information loss between the input and latent spaces. The topological loss is differentiable and thus supports backpropagation. Our results indicate that topological autoencoders perform competitively with state-of-the-art models such as UMAP.

5.1 Limitations and Future Work

TopoAE's definition of topological loss is effective for dimension 0 but does not easily generalize to higher dimensions. We saw that relying solely on the defined topological loss can result in unnoticed collapses of topological features, which was illustrated in section 3.4. Additionally, focusing on edges that create or destroy higher-dimensional features introduces computational inefficiencies due to noise. We propose that an implementation considering not only the pairing but the entire generators of persistent topological features in higher dimensions would yield better results.

We observed that TopoAE is not the only method capable of capturing the intricate topology of the SPHERES dataset; t-SNE can also achieve similar results with sufficiently large perplexity values. Further empirical investigation revealed that significantly increasing t-SNE's perplexity can reduce topological distance metrics, suggesting that t-SNE can learn topology with adequate perplexity. Future research could explore how purely probabilistic models effectively capture complex topologies. Additionally, constructing new synthetic datasets where t-SNE fails to capture the topology while TopoAE succeeds would be an intriguing area for study.

Upon closer examination, we found that the Bottleneck and Wasserstein distances may not serve as reliable measures for comparing topological fidelity, as they overlook the scale of the diagrams without additional processing. This discrepancy arises due to a significant scale difference between the persistence diagrams of the original data and the processed results, leading to matchings between unrelated features. To address this issue, we advocate for a thorough investigation into this phenomenon, taking into consideration the variations in distances across different dimensions. Additionally, we propose exploring the potential of scaling the processed diagrams to achieve better alignment and uncover additional insights.

References

[1] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pages 353–374, 2023.

[2] Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *International Conference on Artificial Intelligence and Statistics*, pages 2786–2796. PMLR, 2020.

[3] Mathieu Carriere, Marco Cuturi, and Steve Oudot. Sliced wasserstein kernel for persistence diagrams. In *International conference on machine learning*, pages 664–673. PMLR, 2017.

[4] Samuel Kaski, Janne Nikkilä, Merja Oja, Jarkko Venna, Petri Törönen, and Eero Castrén. Trustworthiness and metrics in visualizing similarity of gene expression. *BMC bioinformatics*, 4:1–13, 2003.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[7] Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. Topological autoencoders. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 7045–7054. PMLR, 2020.

[8] Gabriel Peyré. Manifold models for signals and images. *Computer vision and image understanding*, 113(2):249–260, 2009.

[9] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.