INTERNSHIP REPORT

Dimensionality Reduction under Topological Constraints

Summer 2025

Emeric PAYER, supervized by Dr. Julien TIERNY

ÉCOLE POLYTECHNIQUE
INSTITUT POLYTECHNIQUE DE PARIS
LIP6
SORBONNE UNIVERSITÉ
cnrs

Abstract

This report presents research conducted at the Computer Science Laboratory of Paris 6 (LIP6) on developing efficient dimensionality reduction techniques that preserve topological structure. The primary objective was to implement and optimize topological autoencoders that maintain the essential structural properties of high-dimensional data when projecting into lower dimensions. We developed three approaches: a baseline MSE-only autoencoder, a classical topological autoencoder preserving Minimum Spanning Tree (MST) structure through full computation at each epoch, and an accelerated version using edge pair ordering constraints to avoid expensive MST computations. Our implementation in Python and PyTorch demonstrates that the accelerated approach significantly reduces computational complexity from $O(n^2 \log n)$ to $O(n)$ per epoch. On simple synthetic datasets with known ground truth, both topological methods substantially outperform the MSE-only baseline, with improvements around 200× in geometric MST distance. The accelerated method achieves performance comparable or sometimes superior to that of the classical method on these simple datasets. However, on complex real-world datasets (*3blobs*, *3rings*, *2cavities*), performance varies with topological complexity, and neither topological method fully converges to zero loss, indicating limitations in handling highly complex nested structures. Computational results show that the accelerated method achieves dramatic speedups, particularly on larger datasets: for the *3rings* dataset with 3,210 points, the accelerated approach is approximately 36× faster per epoch than the classical method, for *2cavities* with 4,002 points, it is approximately 40× faster per epoch, and for a huge dataset of 10,000 points, it is faster by over 400×. Despite limitations on complex topological structures, our work demonstrates that topological constraints can be efficiently incorporated into neural dimensionality reduction, making topology-preserving autoencoders practical for larger-scale applications.

CONTENTS

1 Introduction **4**

2 Context of Internship **4**
2.1 Institution Overview 4
2.2 TORI Project 5
2.3 Internship Objectives 5

3 Background and Related Work **6**
3.1 Topological Data Analysis 6
3.2 Persistent Homology 6
3.3 Existing Approaches for Dimensionality Reduction 7

4 Methodology **8**
4.1 Data Generation 8
4.2 Implementation Architecture 9
4.3 Classical Topological Autoencoder 10
4.4 Accelerated Topological Autoencoder 11
4.5 Training Configuration 11

5 Results **12**
5.1 Topology Preservation Quality 12
5.2 Computational Performance 14
5.3 Loss Evolution Analysis 17

6 Discussion **18**
6.1 Effectiveness of Topological Constraints 18
6.2 Accelerated Approach Performance 19
6.3 Limitations and Challenges 19

7 Conclusion and Future Work **20**
7.1 Summary of Contributions 20
7.2 Limitations and Open Challenges 20
7.3 Future Research Directions 20
7.4 Concluding Remarks 21

1 INTRODUCTION

Dimensionality reduction is a fundamental challenge in machine learning and data analysis, particularly when dealing with high-dimensional datasets where visualization and interpretation become difficult. While traditional methods like Principal Component Analysis (PCA) and standard autoencoders focus primarily on preserving variance or minimizing reconstruction error, they often fail to maintain the intrinsic topological structure of the data, leading to representations that may distort important geometric relationships.

Topological Data Analysis (TDA) has emerged as a powerful framework for understanding the shape and structure of complex datasets. By incorporating topological constraints into dimensionality reduction techniques, we can preserve essential structural properties that traditional methods might lose. This preservation is particularly important in applications where the connectivity and relationships between data points carry significant meaning, such as in biological networks, social networks, sensor data analysis, and manifold learning problems where the underlying geometric structure encodes critical information about the data generation process.

This internship focused on developing and optimizing topological autoencoders that preserve the Minimum Spanning Tree (MST) structure of data during dimensionality reduction. The MST captures the essential connectivity of a dataset and serves as a computationally tractable proxy for preserving 0-dimensional persistent homology. However, computing the MST in the latent space at every training epoch presents a significant computational bottleneck with complexity $O(n^2 \log n)$, severely limiting the scalability of classical topological autoencoders to large datasets.

To address this challenge, we developed an accelerated approach that approximates topology preservation using edge pair ordering constraints from the input MST, eliminating the need for full MST computation at each iteration. This novel method maintains the quality of topological preservation while significantly reducing computational overhead through a clever sampling strategy that preserves local topological relationships. By selecting only 10% of MST edge pairs and enforcing ordering constraints between them, we achieve a dramatic reduction in computational complexity while maintaining the essential topological structure, making topological autoencoders practical for larger datasets and real-world applications.

2 CONTEXT OF INTERNSHIP

2.1 Institution Overview

The internship was conducted at the Laboratoire d'Informatique de Paris 6 (LIP6), a joint research unit (UMR 7606) of the Centre National de la Recherche Scientifique (CNRS) and Sorbonne University. Founded in January 1997 through the fusion of three research laboratories, LIP6 stands as one of the direct heirs of the Blaise Pascal Institute of CNRS, the very first research structure in France dedicated to computer science, established in 1946.

LIP6 is one of the largest computer science research laboratories in France, with approximately 400-450 members including over 170 permanent researchers and around 130 PhD candidates. The laboratory's research activities span nearly the entire spectrum of computer science, organized through 18 teams across four main research axes: Artificial Intelligence and Data Science, Architecture and Systems and Networks, Security and

Safety and Reliability, and Theory and Mathematical Tools for Computer Science. This broad scope enables interdisciplinary collaboration and positions LIP6 at the forefront of computer science research in Europe.

2.2 TORI Project

The internship was supervised by Dr. Julien Tierny within the TORI project (https://erc-tori.github.io/), an ERC-funded initiative focusing on Topological Data Analysis. The TORI team specializes in high-performance computation of topological data representations and the analysis of large-scale datasets through their topological properties. This team has developed the Topology ToolKit (TTK), a comprehensive open-source library for topological data analysis that serves as a reference implementation for many TDA algorithms and has been widely adopted by the research community.

The TORI project aims to develop novel theoretical foundations and practical algorithms for topological data analysis, with particular emphasis on scalability and applicability to real-world problems. The team's research covers both fundamental aspects of computational topology and applied research in various domains including scientific visualization, materials science, and biomedical data analysis.

2.3 Internship Objectives

The internship focused on accelerating topological autoencoders for dimensionality reduction while preserving topological structure. The TopoAE approach computes the Minimum Spanning Tree (MST) in both input and latent spaces at every training epoch, but computing the latent MST at each iteration incurs significant computational overhead ($O(n^2 \log n)$), making the approach impractical for large datasets.

Our specific objectives were the following:

1. **Basic Tools Development:** Develop MST computation tools using Kruskal's algorithm and implement comparison metrics (geometric MST distance, weight difference) to quantitatively measure topology preservation between spaces.
2. **Baseline Implementation:** Implement the classical TopoAE methodology in Python and PyTorch, computing MSTs in both input and latent spaces at each epoch. This serves as our accuracy reference and performance baseline.
3. **Acceleration Strategy:** Design and implement an accelerated approach that approximates topological preservation *without* computing the latent MST at each epoch. Here, we use edge pair ordering constraints from the input MST, enforcing that relative edge lengths are preserved in the latent space through ranking losses. This reduces per-epoch complexity from $O(n^2 \log n)$ to $O(k)$, where k is a small constant number of edge pairs.
4. **Validation and Comparison:** Systematically evaluate three approaches (MSE-only baseline, classical TopoAE, accelerated TopoAE) on:
   - Synthetic datasets with known ground truth (2D planes embedded in 3D)
   - Complex external datasets (3blobs, 3rings, 2cavities)

   We measure both the topology preservation accuracy and the computational efficiency.
5. **Documentation:** Create a well-documented, reproducible codebase to facilitate future research in topological machine learning.

The central research question is: *Can we achieve comparable topology preservation to the classical TopoAE approach while significantly reducing computational cost through edge pair ordering constraints?*

3
BACKGROUND AND RELATED WORK

3.1 Topological Data Analysis

Topological Data Analysis (TDA) is a field at the intersection of mathematics, statistics, and computer science that applies concepts from algebraic topology to analyze the shape and structure of data. Unlike traditional statistical methods that focus on metrics like mean and variance, TDA captures qualitative geometric features that are invariant under continuous deformations, providing robust descriptors particularly valuable when dealing with noisy or high-dimensional datasets.

The fundamental principle underlying TDA is that data has shape, and this shape carries meaningful information about the underlying phenomena. Consider a dataset $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{R}^d$ sampled from an unknown manifold $\mathcal{M}$. Traditional methods might analyze the distribution of points through statistical moments, but TDA seeks to understand the topological invariants of $\mathcal{M}$ - properties that remain unchanged under homeomorphisms. These invariants include connected components (0-dimensional features), loops (1-dimensional features), and voids (higher-dimensional features), providing a multi-scale description of the data's structure.

By studying these topological invariants through the lens of persistent homology, TDA provides insights that complement traditional analytical approaches. These invariants exhibit stability properties under small perturbations, making them particularly suitable for real-world data that often contains noise and uncertainties. The stability theorem for persistence diagrams, proven by Cohen-Steiner, Edelsbrunner, and Harer, guarantees that small changes in the input data lead to small changes in the topological descriptors, providing theoretical foundations for the robustness of TDA methods.

3.2 Persistent Homology

Persistent homology is the mathematical foundation of TDA, providing a multi-scale approach to studying topological features in data. It tracks how topological features appear and disappear as we vary a scale parameter, typically represented through a filtration - a nested sequence of simplicial complexes built from the data.

Mathematical foundations: Let K be a simplicial complex. The d-dimensional chain group $C_d(K)$ is the vector space over $\mathbb{Z}_2$ generated by the d-simplices of K. The boundary operator $\partial_d : C_d(K) \to C_{d-1}(K)$ is defined on a simplex $\sigma = (v_0, ..., v_d)$ as:

$$\partial_d(\sigma) = \sum_{i=0}^{d} (v_0, ..., \hat{v}_i, ..., v_d)$$

where $\hat{v}_i$ denotes the omission of vertex v_i, and we work over $\mathbb{Z}_2$ so signs are omitted. The d-th homology group is then defined as:

$$H_d(K) = \ker(\partial_d)/\mathrm{im}(\partial_{d+1})$$

This quotient group captures d-dimensional topological features: $\ker(\partial_d)$ represents d-cycles (closed d-dimensional features) while $\mathrm{im}(\partial_{d+1})$ represents d-boundaries (features that bound $(d+1)$-dimensional simplices).

Persistence and filtrations: Given a filtration $\emptyset = K_0 \subseteq K_1 \subseteq ... \subseteq K_m = K$ with associated parameter values $t_0 < t_1 < ... < t_m$, persistent homology tracks topological features across scales. The inclusion maps

$K_i \hookrightarrow K_j$ for $i \leq j$ induce homomorphisms $f_d^{i,j} : H_d(K_i) \to H_d(K_j)$. The d-dimensional persistent homology group is:

$$H_d^{i,j} := \ker \partial_d(K_i) / (\text{im } \partial_{d+1}(K_j) \cap \ker \partial_d(K_i))$$

This represents the d-dimensional homology classes born at or before time t_i that have not yet become boundaries by time t_j.

0-Dimensional persistent homology and MST: In this work, we focus specifically on 0-dimensional persistent homology, which captures the connectivity structure of data through connected components. For a point cloud dataset with Euclidean distance, 0-dimensional persistence tracks how points merge into connected components as we increase a distance threshold ϵ.

The Vietoris-Rips complex at scale ϵ, denoted $\mathcal{R}_\epsilon(X)$, contains all simplices whose vertices have pairwise distances at most ϵ:

$$\mathcal{R}_\epsilon(X) = \{\sigma \subseteq X : \text{diam}(\sigma) \leq \epsilon\}$$

As ϵ increases from 0 to ∞, we obtain a filtration that captures the multi-scale connectivity of the data.

Crucially, the 0-dimensional persistence can be efficiently computed and represented through the Minimum Spanning Tree (MST) of the dataset. For 0-dimensional features, the persistence pairing π_0 contains indices corresponding to edges in the MST. If we sort the MST edges $e_1, e_2, \ldots, e_{n-1}$ by weight, the corresponding persistence diagram contains $(n-1)$ finite points of the form $(0, w(e_i))$ where $w(e_i)$ is the weight of edge e_i, representing when the two connected components joined by e_i merge. This correspondence makes the MST an ideal structure for preserving 0-dimensional topological features during dimensionality reduction, as it encodes exactly when connected components merge in the filtration.

3.3 Existing Approaches for Dimensionality Reduction

Topological Autoencoders (2019): Moor et al. introduced Topological Autoencoders, extending classical autoencoders by incorporating topological constraints into the loss function. Their approach adds topology-preserving terms to the standard reconstruction loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \lambda_1 \mathcal{L}_{\text{I2L}} + \lambda_2 \mathcal{L}_{\text{L2I}}$$

where $\mathcal{L}_{\text{I2L}}$ preserves distances of MST edges from the input space in the latent representation, and $\mathcal{L}_{\text{L2I}}$ ensures the latent MST structure aligns with input distances. Specifically:

$$\mathcal{L}_{\text{I2L}} = \sum_{(i,j) \in \text{MST}_{\text{input}}} \|d_{\text{input}}(i,j) - d_{\text{latent}}(i,j)\|^2 \quad , \quad \mathcal{L}_{\text{L2I}} = \sum_{(i',j') \in \text{MST}_{\text{latent}}} \|d_{\text{latent}}(i',j') - d_{\text{input}}(i',j')\|^2$$

While effective for topology preservation, this approach requires computing the full MST in the latent space at every training epoch, resulting in $O(n^2 \log n)$ computational overhead that severely limits scalability.

TopoMap (2020): another method is TopoMap, preserving 0-dimensional homology through a different approach. Instead of using MST directly, TopoMap employs a Vietoris-Rips complex construction and optimizes for preservation of persistence diagrams using the bottleneck distance as a loss term. While providing theoretical guarantees on topology preservation, the computational complexity remains high for large datasets, and the discrete nature of the bottleneck distance can lead to optimization challenges.

RTD and recent methods (2023 – 2025): Chen and Gel (2023) introduced Robust Topological Descriptors (RTD) that combine multiple topological features including persistence landscapes and persistence images for more comprehensive structure preservation. Recent work on Topological Autoencoders++ (2025) extends these ideas with attention mechanisms and multi-scale topology preservation. However, these methods generally focus on accuracy improvements rather than addressing the fundamental computational efficiency challenges, making them impractical for large-scale applications.

4 METHODOLOGY

4.1 Data Generation

To evaluate our approaches, we built synthetic datasets, and used more complex ones from the TTK repository.

Synthetic datasets: We generated five synthetic datasets of increasing complexity to evaluate dimensionality reduction and topology preservation. Each dataset originated as a 2D point distribution with distinct topological features, then was linearly projected into 3D using $z = \alpha x + \beta y$, where α and β are dataset-specific coefficients (chosen randomly). This provides a known 2D ground truth while creating a realistic 3D $\rightarrow$ 2D reduction task.

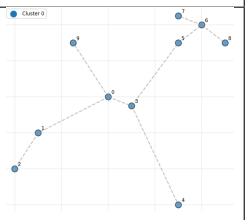
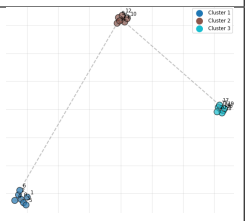
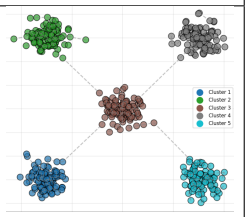
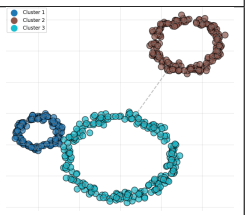
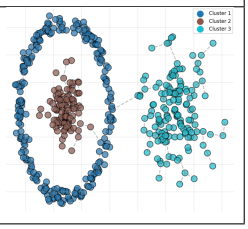
Dataset Name	# Points	Description	Ground Truth
`points_1`	10	Minimal test case for debugging and validation; simple structure for verifying topology-preserving methods.	
`points_2`	21	Medium-scale dataset with 3 well-separated clusters; useful for evaluating clustering and separation performance.	
`points_3`	500	Contains 5 distinct clusters with no overlap; designed to test scalability and cluster resolution.	
`points_4`	520	Complex topological structure with 3 rings; challenges topology-preserving embeddings.	
`points_5`	500	Nested structure with a ring containing a blob plus a separate cluster; tests ability to capture nested relationships.	

Table 1: Overview of the synthetic datasets.

The datasets range from minimal test cases with 10 points (points_1) for debugging and validation, to medium-scale datasets with well-separated clusters (points_2 with 21 points in 3 clusters, points_3 with 500 points in 5 clusters), to complex topological structures including rings (points_4 with 520 points forming 3 rings) and nested structures (points_5 with 500 points forming a ring containing a blob plus a separate cluster). Each dataset presents unique challenges for topology preservation, from simple connectivity to complex nested relationships.

External datasets: We also evaluated our methods on complex datasets from the TTK repository: 3blobs ($\approx$ 800 points in 3 well-separated clusters), 3rings ($\approx$ 3,200 points forming 3 interlocking rings), and 2cavities ($\approx$ 4,000 points with two topological voids). These datasets provide realistic test cases with known topological features commonly encountered in real-world applications.

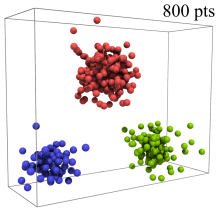
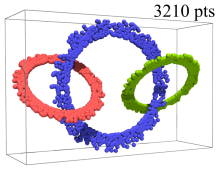
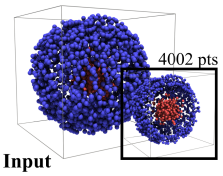
Dataset Name	# Points	Description	Ground Truth
`3blobs`	800	Three well-separated 3D Gaussian clusters; serves as a baseline for clustering and manifold preservation in low-noise conditions.	
`3rings`	3210	Three interlocking rings in 3D space; tests topology-aware embeddings and the ability to preserve linked structures.	
`2cavities`	4002	Nested spherical structure with a small solid ball enclosed inside a hollow sphere; evaluates preservation of cavities and nested topology.	

Table 2: Overview of the more complex 3D datasets from TTK.

4.2 Implementation Architecture

The first step is to establish the core tools and foundations that enable the construction of a topological autoencoder.

MSE-Only baseline autoencoder: We first implemented a standard autoencoder as a baseline, using only Mean Squared Error (MSE) for reconstruction:

$$\mathcal{L}_{\text{MSE}} = \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2$$

where x_i is the input data point and $\hat{x}_i = h(g(x_i))$ is its reconstruction through encoder $g : \mathcal{X} \to \mathcal{Z}$ and decoder $h : \mathcal{Z} \to \mathcal{X}$. The architecture uses fully connected layers with ReLU activations and batch normalization, trained using the Adam optimizer with weight decay for regularization.

MST computation tools: We developed efficient tools for MST computation and validation using Kruskal's algorithm. Given a point cloud $X = \{x_1, ..., x_n\}$, we first compute the complete distance matrix $D \in \mathbb{R}^{n \times n}$ where $D_{ij} = \|x_i - x_j\|_2$. The algorithm maintains a disjoint-set data structure and iteratively adds edges in order of increasing weight, skipping edges that would create cycles. This yields the MST in $O(n^2 \log n)$ time, where the bottleneck is sorting the $O(n^2)$ edges.

Evaluation tools: For MST comparison, we implemented two complementary metrics.

1. The geometric MST distance compares sorted edge lengths between MSTs:

$$d_{\text{geometric}} = \sum_i |l_i^{\text{input}} - l_i^{\text{latent}}|$$

   where edges are sorted by length.

2. The MST weight difference captures scale differences:

$$d_{\text{weight}} = |\sum e \in \text{MST}_{\text{input}} - \sum e \in \text{MST}_{\text{latent}}|$$

These metrics provide quantitative measures of topology preservation quality. One can prove that when the geometric distance between two MSTs tends to 0, then the MSTs are identical.

4.3 Classical Topological Autoencoder

Following the TopoAE paper, we implemented the classical approach with three loss components that jointly optimize reconstruction quality and topology preservation:

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{MSE}} + \lambda_2 \mathcal{L}_{\text{I2L}} + \lambda_3 \mathcal{L}_{\text{L2I}}$$

The input-to-latent loss $\mathcal{L}_{\text{I2L}}$ preserves the structure of the input MST by minimizing distance discrepancies:

$$\mathcal{L}_{\text{I2L}} = \sum_{(i,j) \in \text{MST}_{\text{input}}} \|d_{\text{input}}(i,j) - d_{\text{latent}}(i,j)\|^2$$

This ensures that edges important for connectivity in the input space maintain similar distances in the latent space.

The latent-to-input loss $\mathcal{L}_{\text{L2I}}$ performs the complementary operation:

$$\mathcal{L}_{\text{L2I}} = \sum_{(i',j') \in \text{MST}_{\text{latent}}} \|d_{\text{input}}(i',j') - d_{\text{latent}}(i',j')\|^2$$

This bidirectional approach ensures that both the input topology influences the latent space and the latent topology aligns with input distances, preventing degenerate solutions where the latent space collapses to a single point.

The critical computational bottleneck lies in computing $\text{MST}_{\text{latent}}$ at each training iteration. For a batch of size m, this requires $O(m^2)$ distance computations followed by $O(m^2 \log m)$ for MST construction, making the approach prohibitively expensive for large datasets or batch sizes.

4.4 Accelerated Topological Autoencoder

To address the computational bottleneck, we developed an accelerated approach using edge pair ordering constraints that approximate topology preservation without computing the latent MST.

Edge pair selection strategy: During initialization, we compute the input MST once and select $N = 0.5 \times |\text{MST}_{\text{edges}}|$ edge pairs. For each edge e_0 in the sorted MST, we identify adjacent edges (sharing a vertex) and select edges e_1 where $\text{weight}(e_1) > \text{weight}(e_0)$. This creates pairs (e_0, e_1) that capture local topological relationships. The adjacency constraint ensures that the selected pairs represent meaningful topological constraints rather than arbitrary edge combinations.

Ordering loss formulation: For each selected edge pair (e_0, e_1), we enforce that their relative ordering is preserved in the latent space using a margin ranking loss:

$$\mathcal{L}_{\text{ordering}} = \sum_{(e_0,e_1)\in\text{pairs}} \max(0, d_{\text{latent}}(e_0) - d_{\text{latent}}(e_1) + \text{margin})$$

where margin (typically 0.1) provides robustness against numerical errors and ensures a clear separation between edge lengths. This loss is zero when the ordering is correctly preserved with sufficient margin, and increases linearly when violated.

Periodic validation: To prevent accumulation of approximation errors, every 30 epochs we compute the full latent MST and add a validation loss term. This infrequent but comprehensive check ensures long-term stability while maintaining computational efficiency. The complete accelerated loss becomes:

$$\mathcal{L}_{\text{total}} = \lambda_1\mathcal{L}_{\text{MSE}} + \lambda_2\mathcal{L}_{\text{I2L}} + \lambda_3\mathcal{L}_{\text{ordering}} + \lambda_4\mathcal{L}_{\text{validation}}$$

where $\mathcal{L}_{\text{validation}}$ is only computed every 30 epochs.

This approach reduces per-epoch complexity from $O(n^2 \log n)$ to $O(N)$ where $N \ll n^2$, typically around 10% of the total number of edges edge pairs, providing a good speedup while maintaining topology preservation quality through careful selection of topologically significant edge pairs.

4.5 Training Configuration

All models were trained with consistent hyperparameters to ensure fair comparison. We used the AdamW optimizer with learning rate 0.001 and weight decay 10^{-5} for regularization. Training proceeded for 10,000 epochs with early stopping based on validation loss plateau detection (patience of 100 epochs with relative improvement threshold of 10^{-4}).

The network architecture was adapted based on input dimension: for low-dimensional inputs ($d \leq 10$), we used 2 hidden layers with sizes $(\max(16, 2d), \max(8, d))$, while for higher-dimensional inputs ($d > 10$), we employed 3 hidden layers of sizes $(64, 32, 16)$. All layers except the final decoder output use ReLU activations and batch normalization.

Loss weights were initially set equal ($\lambda_1 = \lambda_2 = \lambda_3 = 1.0$) with optional adaptive balancing that adjusts weights every 10 epochs based on loss magnitudes to prevent any single component from dominating. The adaptive mechanism computes target weights inversely proportional to recent loss values, then applies gradual updates to maintain stability. In practice, weights were manually tuned to minimize geometric distance between predictions and ground truth.

5 RESULTS

5.1 Topology Preservation Quality

We evaluated the three approaches (MSE-only, Classical, Accelerated) on both synthetic and complex datasets, using geometric MST distance as our primary metric for topology preservation quality.

Results for the Synthetic Datasets:

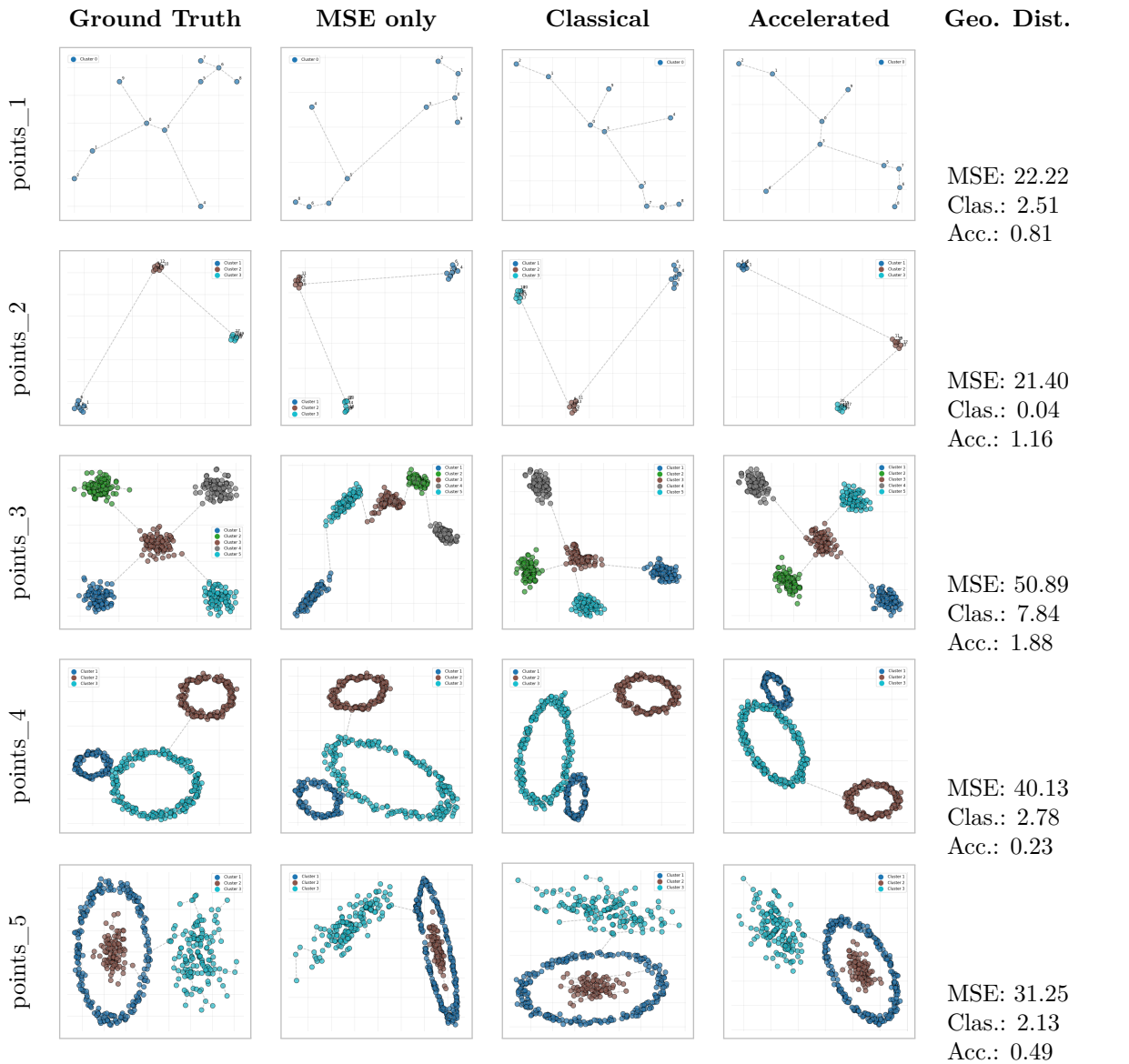


Table 3: Topology Preservation Quality on our Synthetic Datasets

Results for the Complex Datasets:

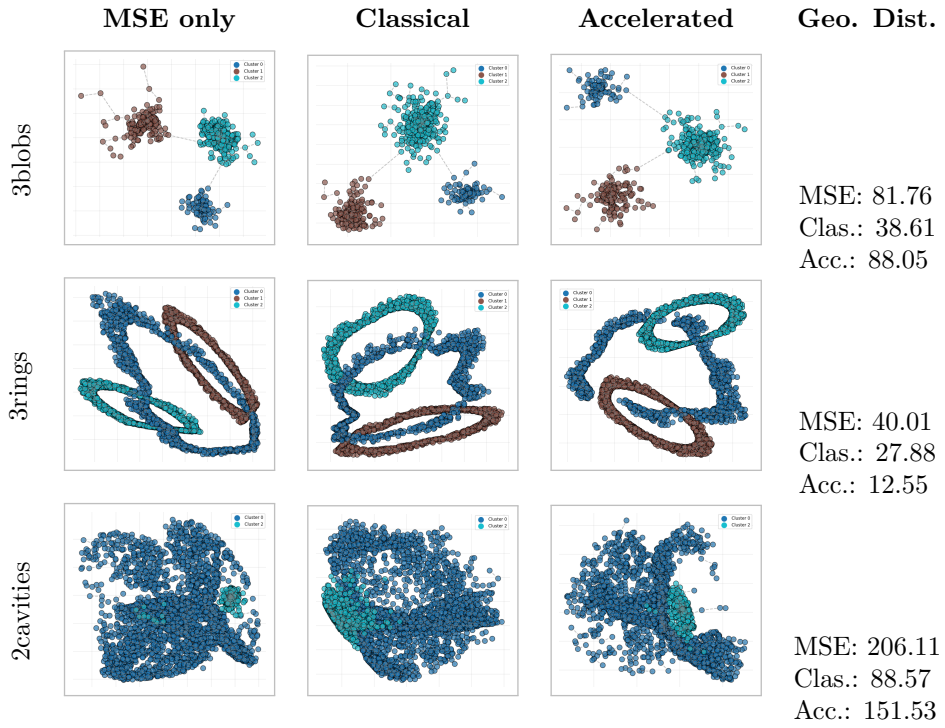


Table 4: Topology Preservation Quality on TTK's Complex Datasets

Results Analysis:

The following table summarizes the results for the synthetic datasets, where the total MST weight difference is the difference in total edge weight between the ground-truth MST and the MST from the latent space.

Table 5: Performance comparison on synthetic datasets with ground truth

Dataset	Geometric MST Distance ↓			Total MST Weight Difference ↓		
	MSE-only	Classical	Accelerated	MSE-only	Classical	Accelerated
points_1 (10)	22.22	2.51	0.81	20.05	1.33	0.31
points_2 (21)	21.40	0.04	1.16	21.26	0.02	0.51
points_3 (500)	50.89	7.84	1.88	21.12	0.83	0.67
points_4 (520)	40.13	2.78	0.23	30.06	1.46	0.00
points_5 (500)	31.25	2.13	0.49	22.43	0.551	0.12

This second table summarizes the results obtained from the external complex datasets (from TTK).

Table 6: Performance on complex external datasets

Dataset	Geometric MST Distance ↓			Topology Type
	MSE-only	Classical	Accelerated	
3blobs (800)	81.76	38.62	88.05	Clusters
3rings (3,210)	40.01	27.88	12.55	Rings
2cavities (4,002)	206.11	88.57	151.53	Voids

The results show that, on simple datasets with a ground-truth (our synthetic datasets), both topological methods substantially outperform the MSE-only baseline in preserving geometric structure. We can even observe that on these simple datasets, the accelerated approach seems to slightly outperform the classical pipeline, probably because there is no complicated topological structure to preserve.

Yet, we must note that the performance strongly depends on the random initialization: when the code is run multiple times, the resulting geometric distances vary considerably, and either the classical or the accelerated approach can outperform the other. The only consistent trend is that both topological approaches clearly and significantly outperform the MSE-only baseline. Overall, the geometric MST distance ranges from approximately 7× (*points_3 with the classical approach*) to over 500× (*points_2* with the classical approach).

However, on more complex datasets, the classical pipeline appears to outperform our accelerated pipeline, with the exception of the *3rings* dataset. We observe that our model's performance varies notably with the topological complexity of the data.

Regardless of the method used for the *3blobs* or *2cavities* datasets (whether MSE-only, the classical topological approach, or the accelerated one), the model consistently struggles with convergence. For the *3rings* dataset, the accelerated topological method shows a substantial improvement over the MSE baseline, while the classical approach performs in between the two. In contrast, for the *2cavities* dataset, the classical topology-based method achieves significant gains compared to the MSE baseline, whereas the accelerated pipeline's performance lies between the two methods. However, the model fails to converge to zero loss in all three cases.

Overall, we clearly observe that cluster structures are better preserved when topological information is incorporated, aligning with our original objective. However, despite these notable improvements, our topological-based methods are not yet robust enough to handle highly complex datasets effectively. We would need to incorporate additional topological features and tools to address this limitation (cf. Discussion section).

5.2 Computational Performance

Basic Performance Data

The primary goal of this internship was to develop a method to approximate the L2I component of the loss function, avoiding the need to compute it at every epoch. This motivated the design of the accelerated pipeline, whose key advantage is to reduce computational complexity, making it feasible to apply to larger datasets.

We have recorded the training time for each method in the following table:

Table 7: Training time (in seconds) comparison across datasets

Dataset	MSE only		Classical		Accelerated	
	Total time	Time per epoch	Total time	Time per epoch	Total time	Time per epoch
points_1	8.7689	0.0009	20.133	0.0020	13.477 $^{(8,779)}$	0.0015
points_2	8.9188	0.0009	36.672	0.0037	6.8501 $^{(3,235)}$	0.0021
points_3	137.22	0.0137	253.55 $^{(1,445)}$	0.1755	138.31 $^{(5,850)}$	0.0236
points_4	147.40	0.0147	1605.74 $^{(7,594)}$	0.2114	246.63	0.0247
points_5	138.46	0.0138	1982.13	0.1982	172.38 $^{(7,691)}$	0.0224
3blobs	204.18	0.0204	1030.34 $^{(1,730)}$	0.5956	108.16 $^{(3,841)}$	0.0282
3rings	818.75	0.0819	34675.27 $^{(3,661)}$	9.4716	133.77 $^{(1,012)}$	0.1322
2cavities	1036.14	0.1036	35082.36 $^{(2,231)}$	15.725	884.55 $^{(5,393)}$	0.1640

The above table reports both the total time to convergence and the time per epoch. The number in parentheses next to the total time indicates the number of epochs required for convergence (by default 10,000), as our model employs an early stopping criterion that terminates training once this condition is satisfied. As such, evaluating the time per epoch provides a more pertinent basis for comparison between the approaches.

Our first observation is that the MSE-only approach achieves the fastest time per epoch, while the classical topological approach is consistently the slowest, requiring over 15 seconds per epoch on the *2cavities* dataset. The accelerated method is consistently only slightly slower than the MSE approach (expected since part of the accelerated loss is derived from the MSE), but it becomes exponentially faster than the classical approach.

For a better analysis, we can visualize the data using a vertical bar chart:

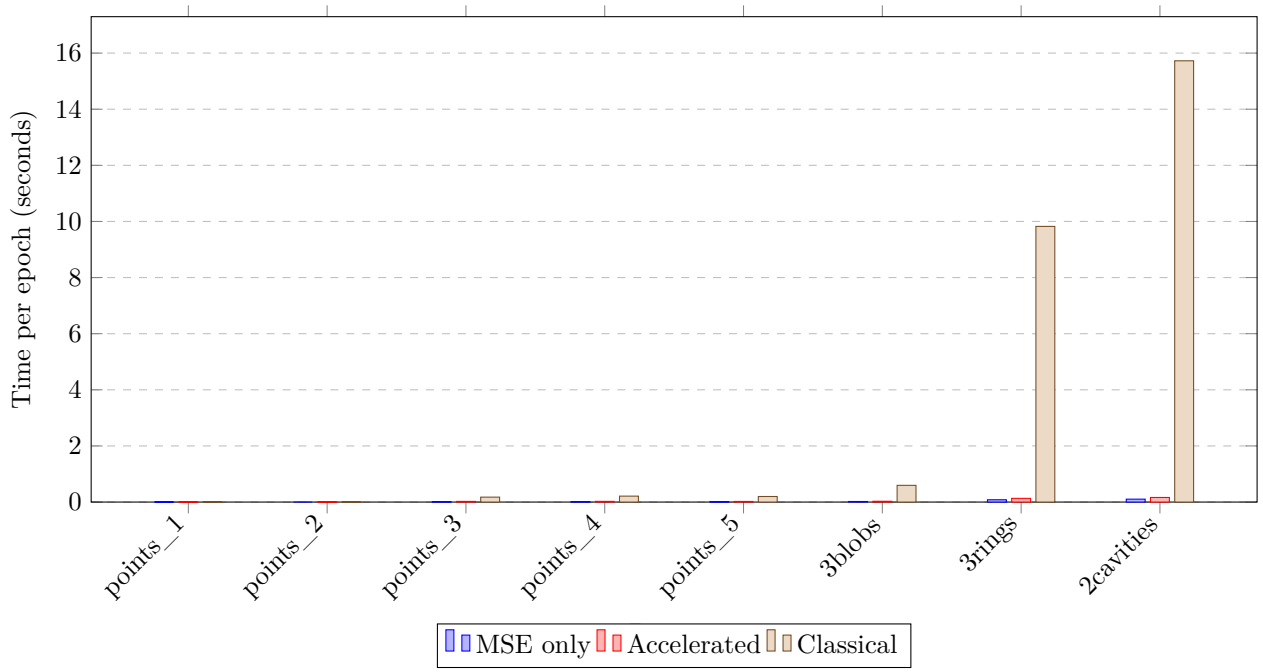


Figure 1: Training time per epoch comparison across datasets

This shows that the time per epoch grows exponentially with respect to the size of the dataset. However, since the results for the smaller datasets are difficult to interpret, we created a separate bar chart for each dataset to better compare the computational performance of the different methods within the same dataset.

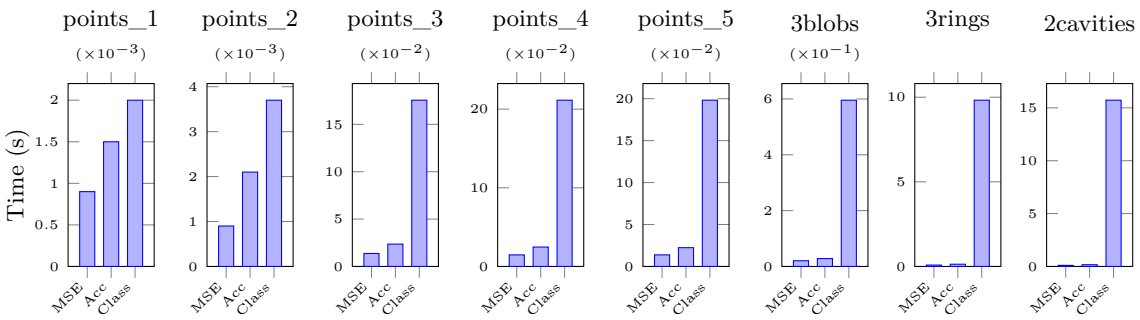


Figure 2: Training time per epoch comparison across datasets

The above chart demonstrates that the accelerated approach consistently achieves intermediate performance between the MSE-only and classical methods across all datasets, positioning significantly closer to the MSE-only baseline than to the classical topological approach.

This behavior validates the correctness of the accelerated implementation, confirming that it effectively reduces the computational overhead of topology preservation while retaining the topological constraints that are absent in the MSE-only baseline. Moreover, it achieves this within a computational time much closer to that of the MSE approach than to the classical topological method. The exponential growth in the performance gap with increasing dataset size is expected, as previously discussed: the classical approach has a per-epoch complexity of $\mathcal{O}(n^2 \log n)$, whereas the accelerated approach scales linearly with $\mathcal{O}(n)$.

Detailed Comparison between the Topological and Accelerated approaches

Our primary focus is on large datasets, where the $O(n^2 \log n)$ complexity of the L2I component becomes significant. Currently, we compute the full latent MST every 30 epochs for monitoring purposes; however, this computation is not practically necessary. Therefore, in the following table, we report results without performing this repeated computation. This allows for a more objective comparison between the two approaches.

Table 8: Execution time comparison (seconds per 1000 epochs)

Dataset	Classical	Accelerated	Speedup	Relative Time
3blobs (800)	589.16	27.97	21.06×	4.75%
3rings (3,210)	9504.23	264.66	35.91×	2.78%
2cavities (4,002)	15768.10	386.83	40.76×	2.45%

We observe that for datasets with approximately 1,000 points, the speedup factor is approximately at 20 times. For datasets with 3,000 to 4,000 points, the accelerated approach achieves a speedup of 35 to 40 times. It is important to note, however, that this speedup is not linear. As the number of data points increases, the advantage of our pipeline becomes more pronounced, since it has a computational complexity of $O(N)$ per epoch, with $N \approx 0.5n$, compared to $O(n^2 \log n)$ for the standard approach. Consequently, as n grows, the time required by the standard method increases superlinearly, while the accelerated method scales linearly with n. We can even increase the speedup factor by taking less pair edges (instead of 50% of the dataset, we can take for example 10-15%, which will again divide by approximately 2 the training time).

To test this claim, we generated a very large dataset, *points_mega*, consisting of 10,000 points representing two rings. As before, the data were first generated on a 2D plane, after which a z-component was added as a linear combination of the x and y coordinates to obtain a 3D dataset. We then trained the autoencoder on this dataset, using the original 2D data as the ground truth. Due to the computational cost, the autoencoder was trained for only 10 epochs.

Table 9: Execution time comparison (seconds per 10 epochs)

Dataset	Classical	Accelerated	Speedup	Relative Time
points_mega (10,000)	1464.55	3.47	422.06×	0.24%

Note that since this dataset was trained for only 10 epochs, direct comparison of reconstruction results would not have meaningful. However, we can still compare execution times. We observe that the speedup factor now exceeds 400×, with the accelerated approach requiring only 0.24% of the time taken by the classical method. This result further demonstrates that the execution time of our accelerated pipeline decreases exponentially relative to dataset size.

5.3 Loss Evolution Analysis

In this section, we present the training loss. While this is not the central focus of the project, it is important to comment on it. We observe that, regardless of the approach used, the model typically converges relatively quickly, which explains why the early stopping criterion is frequently triggered during training (cf. above).

The MSE-only approach serves as a baseline, using mean squared error alone without topological information, providing a reference to evaluate the impact of topological enhancements on convergence and cluster separation.

MSE Only approach:

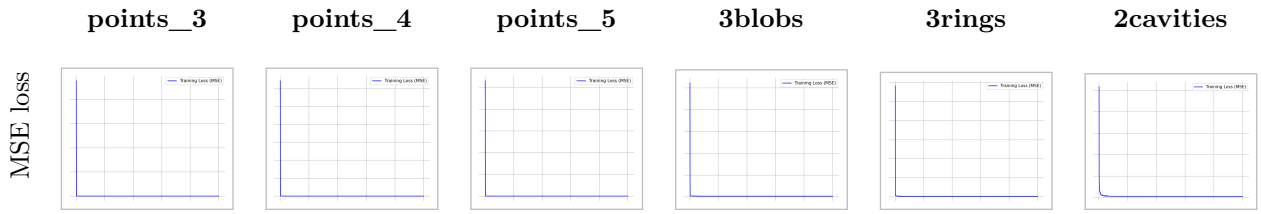


Table 10: MSE only training loss

We can now perform a similar analysis for the classical topological-based loss function. Recall that the classical approach combines MSE, L2I, and I2L components. In this section, we will present three types of plots: the total combined loss (i.e., the weighted sum of all three components), the MSE component alone, and a combined graph showing the L2I and I2L contributions.

Classical Topological approach:

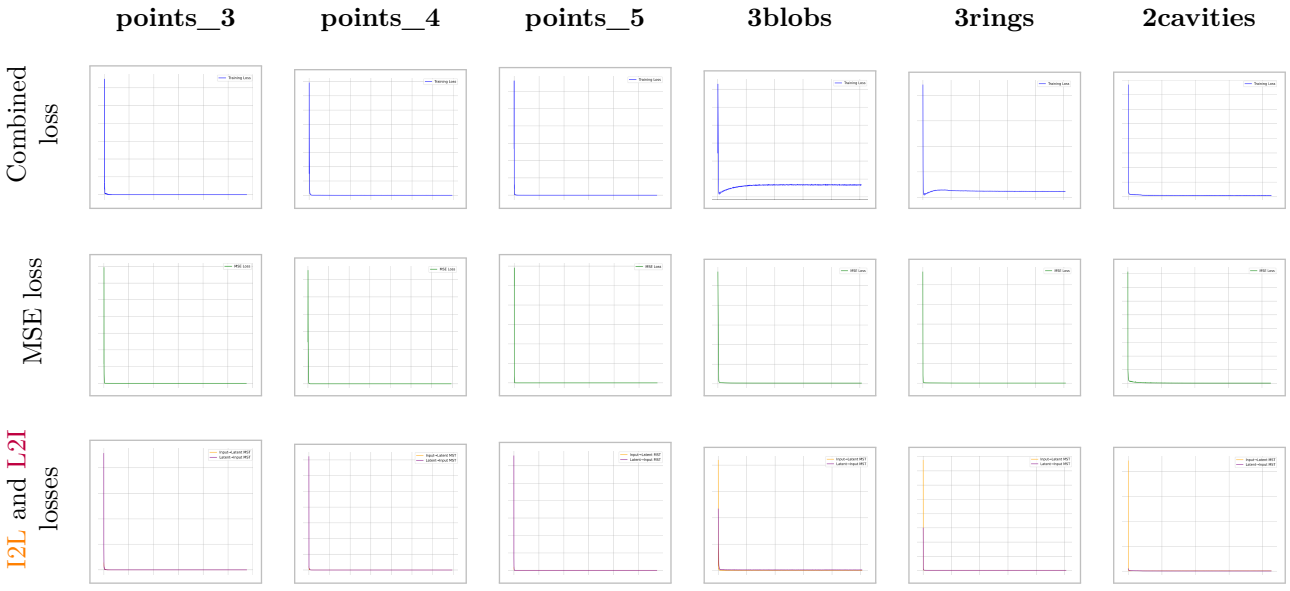


Table 11: Topological Classical results

And we can perform the same analysis with the accelerated approach.

Accelerated Topological approach:

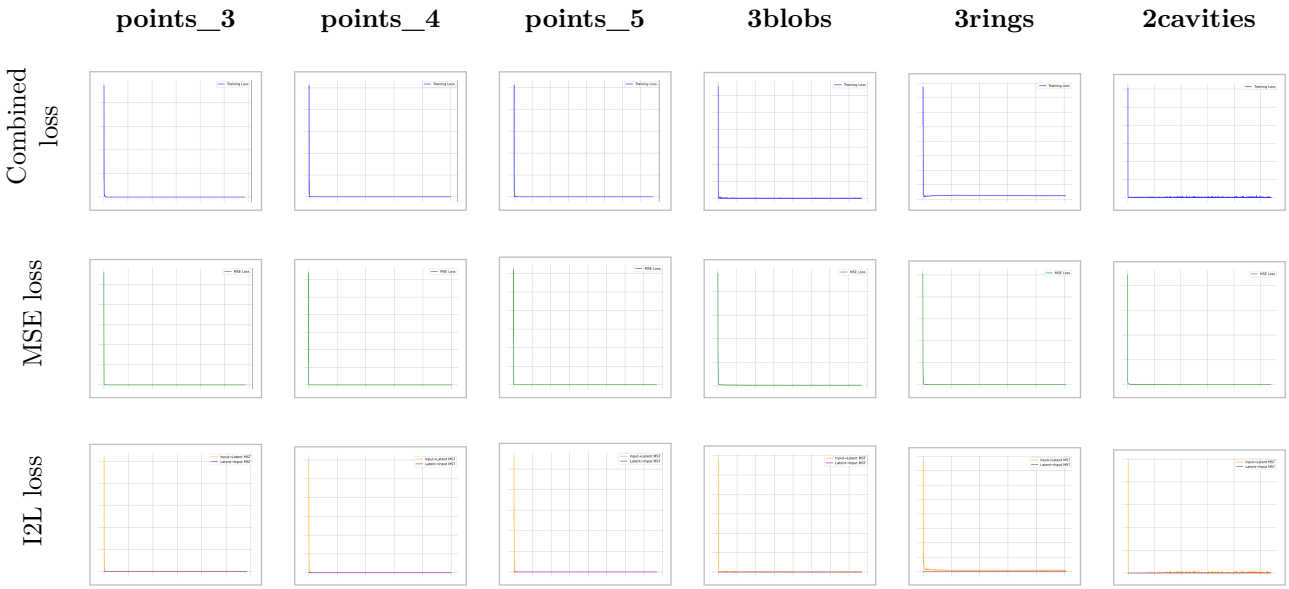


Table 12: Topological Accelerated results

For the accelerated approach, the I2L component is never computed due to its high computational cost, and therefore this curve is fixed at zero. Consequently, for the accelerated approach, we will plot the total combined loss, the MSE component independently, and the L2I component (with I2L always set to zero) for comparison. This visualization allows us to directly assess the impact of each loss component on the overall training dynamics and to compare the classical and accelerated approaches.

We may note that, although the loss curves are generally smooth, occasional minor fluctuations can occur. These are mainly due to the "automatic weight adjustment mode," which re-calibrates the weights of the loss function every 30 epochs. This mode automatically increases the weight of the largest loss component, resulting in the slight bumps observed in the training loss curves.

6 DISCUSSION

6.1 Effectiveness of Topological Constraints

Our results demonstrate that incorporating topological constraints significantly improves structure preservation on simple datasets. On synthetic datasets (*points_1* through *points_5*), both topological approaches substantially outperform the MSE-only baseline, with improvements in geometric MST distance ranging from 7× to over 500× depending on the dataset and initialization. Visual inspection confirms that topological methods maintain clear cluster separation and connectivity patterns.

However, performance on complex datasets reveals significant limitations. On *3blobs* (800 points), the classical method achieves geometric MST distance of 38.62 versus 81.76 for MSE-only. For *3rings* (3,210 points), the accelerated method performs best (12.55) compared to classical (27.88) and MSE-only (40.01). Yet for *2cavities*

(4,002 points), all methods struggle significantly, with distances of 88.57 (classical), 151.53 (accelerated), and 206.11 (MSE-only).

Critically, none of the methods converge to zero loss on complex datasets, regardless of approach. This indicates that MST-based topology preservation captures only part of the topological information in datasets with nested components or intricate connectivity patterns.

6.2 Accelerated Approach Performance

The accelerated approach successfully achieves very important computational speedups while maintaining competitive topology preservation quality. On smaller datasets (10-21 points), speedup is modest ($< 1\times$), but as dataset size grows, the advantage becomes substantial: $\approx 10\times$ for 500-520 points, $21\times$ for 800 points, $36\times$ for 3,210 points, and $41\times$ for 4,002 points. A test on 10,000 points demonstrated $400\times$ speedup, with the accelerated method requiring only 0.24% of the classical method's time.

The effectiveness validates our hypothesis that local edge ordering relationships encode much of the global MST structure. By sampling adjacent edge pairs (50% of MST edges) and enforcing relative length preservation through margin ranking loss, we maintain connectivity patterns without full MST recomputation.

On simple synthetic datasets, the accelerated method often even outperforms the classical approach, though this varies with initialization. The consistent trend is that both topological approaches clearly outperform the MSE-only baseline.

6.3 Limitations and Challenges

Complex Topological Structures

Both methods struggle with nested or hierarchically organized structures. The *2cavities* dataset proved particularly challenging, with high geometric MST distances (88.57 classical, 151.53 accelerated) even after extensive training. This was confirmed with the original TopoAE implementation, suggesting fundamental rather than implementation issues.

The reliance on MST as the sole topological descriptor is a key limitation. MSTs capture 0-dimensional homology (connected components) but cannot encode loops, voids, or higher-dimensional features. For datasets where 1-dimensional homology (loops in *3rings*) or 2-dimensional homology (voids in *2cavities*) are important, MST-based approaches are fundamentally insufficient.

Optimization Challenges

Training exhibits significant sensitivity to random initialization. Multiple runs with identical hyperparameters produce considerably varying geometric distances, with either classical or accelerated methods outperforming the other depending on the seed. The optimization landscape contains multiple local minima where networks preserve most but not all MST edges correctly.

The balance between reconstruction and topological loss is delicate. Equal weighting ($\lambda_1 = \lambda_2 = \lambda_3 = 1.0$) works reasonably well, but optimal values vary by dataset and often require manual tuning. The automatic weight adjustment mode (re-calibrating every 30 epochs) helps but causes minor training fluctuations.

7
CONCLUSION AND FUTURE WORK

7.1 Summary of Contributions

We implemented a complete topological autoencoder framework in Python and PyTorch, including the classical approach and a novel accelerated approach using edge pair ordering constraints. The accelerated method reduces per-epoch complexity from $O(n^2 \log n)$ to $O(n)$, achieving practical speedups of $\approx 10\times$ on medium datasets (500-520 points), 35-40$\times$ on larger datasets (3,000-4,000 points), and over 400$\times$ on a 10,000-point test case.

Experimental evaluation demonstrates clear benefits over MSE-only approaches on simple datasets (usually around 200$\times$ improvement in terms of geometric MST distance) while revealing important limitations on complex nested structures.

7.2 Limitations and Open Challenges

The restriction to 0-dimensional homology through MST preservation is a significant constraint. Many datasets contain important higher-dimensional features (loops, voids, higher-dimensional structures) that MSTs do not necessarily capture. Results on *3rings* and *2cavities* show that while MST-based methods improve over baselines, they cannot fully preserve complex topology.

Significant sensitivity to initialization and multiple local minima also represent a fundamental challenge. Because of the variable quality across runs with identical hyperparameters, we are faced with a limited practical applicability that sometimes necessitates multiple training attempts before obtaining a satisfying result.

7.3 Future Research Directions

Richer Topological Descriptors

The most natural next step would be to incorporate other topological features that we wish to conserve throughout the training. For example, we could preserve Rips filtrations, which will add more topological features on connected components. While computing full Rips filtrations at each epoch would be expensive, approximation strategies similar to our edge pair approach could work. We could also preserve higher-dimensional topological features through persistent homology, including loops, voids, and higher-dimensional holes.

Accelerated Approach Improvements

Currently, we sample 50% of edge pairs uniformly during training. A potential improvement would be to select edge pairs more strategically from the start, aiming to capture the most informative relationships. For example, edges derived from the original minimum spanning tree or other structurally important subsets could provide stronger learning signals. This approach might reduce redundancy among sampled pairs and lead to faster, more stable convergence. Further exploration of dynamic weighting or alternative constraint formulations, such as ranking over larger edge sets, soft penalties, or optimal transport methods, could also improve optimization efficiency and robustness.

Alternative Architectures

Another direction would be to explore models that explicitly capture the underlying graph structure. For instance, Graph Neural Networks operating on the minimum spanning tree could help preserve the overall topology. Variational autoencoders with topological constraints might combine probabilistic modeling with structural awareness, improving reconstruction quality. Incorporating attention mechanisms could allow the model to focus on regions that are topologically important. Finally, training strategies such as curriculum learning or multi-stage optimization could help avoid poor local minima and lead to more stable convergence.

7.4 Concluding Remarks

This work demonstrates that topological constraints can be efficiently incorporated into neural dimensionality reduction. The accelerated approach achieves dramatic speedups (over 400× on a 10,000-points dataset) while maintaining competitive quality on simple datasets, where topological methods show up to 500× improvement over MSE-only baselines.

However, significant challenges remain: current methods struggle with nested structures and are limited to 0-dimensional homology. These limitations highlight the need for richer topological representations (particularly Rips filtrations), more robust optimization strategies, and stronger theoretical foundations.

By incorporating comprehensive topological descriptors capturing loops and voids, developing adaptive multi-scale approaches, and establishing theoretical guarantees, future work can create dimensionality reduction methods that reliably preserve essential structural properties of complex data. This work provides both a practical framework achieving substantial speedups and a foundation identifying specific directions for advancing topological autoencoders.

ACKNOWLEDGMENTS

I would like to thank Dr. Julien Tierny for his supervision and guidance throughout this internship, the TORI team at LIP6 for their support and help, and the broader topological data analysis community for their foundational work that made this research possible.

REFERENCES

[1] Edelsbrunner, H. and Harer, J. (2010). *Computational Topology: An Introduction*. American Mathematical Society. Available at: https://github.com/lotzma/TDA/raw/master/docs/Edels_Book.pdf

[2] Zomorodian, A. J. (2010). Topology for computing. In M. J. Atallah and M. Blanton (Eds.), *Algorithms and Theory of Computation Handbook* (Second Edition), Chapter 3, pp. 82–112. CRC Press. Available at: https://julien-tierny.github.io/stuff/algorithmsAndTheoryOfComputationHandbook.pdf

[3] Tierny, J. *Topological Data Analysis Class*. Available at: https://julien-tierny.github.io/topologicalDataAnalysisClass.html

[4] Moor, M., Horn, M., Rieck, B., and Borgwardt, K. (2019). Topological Autoencoders. *arXiv preprint arXiv:1906.00722*. Available at: https://arxiv.org/abs/1906.00722

[5] Doraiswamy, H. and Natarajan, V. (2020). TopoMap: A 0-dimensional homology preserving projection of high-dimensional data. *arXiv preprint arXiv:2009.01512*. Available at: https://arxiv.org/abs/2009.01512

[6] Chen, Y. and Gel, Y. R. (2023). RTD: Robust Topological Descriptor for Machine Learning Applications. *arXiv preprint arXiv:2302.00136*. Available at: https://arxiv.org/abs/2302.00136

[7] Anonymous (2025). Topological Autoencoders++. *arXiv preprint arXiv:2502.20215*. Available at: https://arxiv.org/abs/2502.20215

[8] Tierny, J., Favelier, G., Levine, J. A., Gueunet, C., and Michaux, M. (2018). The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1), 832-842.

[9] Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1), 48-50.

[10] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.