






# TopoMap++: A faster and more space efficient technique to compute projections with topological guarantees

Vitoria Guardieiro , Felipe Inagaki de Oliveira , Harish Doraiswamy , Luis Gustavo Nonato , Claudio Silva 

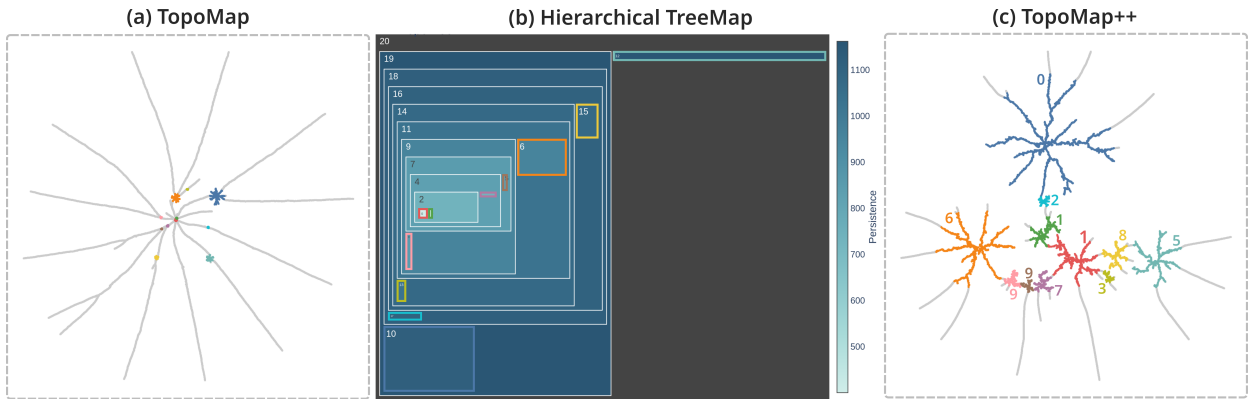


Fig. 1: Representations of the MNIST database of handwritten digits. **(a)** This data is projected using TopoMap [13]. Note the presence of long branches with star-based cluster ensembles (colored) barely identifiable unless one zooms into the respective regions of interest. **(b)** The hierarchy defined by the process of topological simplification is visualized as a TreeMap. Each leaf of this tree corresponds to the smallest simplified component with a user-defined minimum number of points. There are eleven such components which are automatically selected as indicated by the colored borders. Each box in the TreeMap is colored-coded based on the persistence of the corresponding component. The largest box is colored gray to indicate its infinite persistence. **(c)** The TopoMap++ representation of the same data where the eleven components selected by the TreeMap are highlighted. As can be seen, TopoMap++ makes much more efficient use of the space compared to TopoMap, thus allowing users to easily analyze the relationships between the different clusters.

**Abstract**— High-dimensional data, characterized by many features, can be difficult to visualize effectively. Dimensionality reduction techniques, such as PCA, UMAP, and t-SNE, address this challenge by projecting the data into a lower-dimensional space while preserving important relationships. TopoMap is another technique that excels at preserving the underlying structure of the data, leading to interpretable visualizations. In particular, TopoMap maps the high-dimensional data into a visual space, guaranteeing that the 0-dimensional persistence diagram of the Rips filtration of the visual space matches the one from the high-dimensional data. However, the original TopoMap algorithm can be slow and its layout can be too sparse for large and complex datasets. In this paper, we propose three improvements to TopoMap: 1) a more space-efficient layout, 2) a significantly faster implementation, and 3) a novel TreeMap-based representation that makes use of the topological hierarchy to aid the exploration of the projections. These advancements make TopoMap, now referred to as TopoMap++, a more powerful tool for visualizing high-dimensional data which we demonstrate through different use case scenarios.

**Index Terms**—Topological data analysis, Computational topology, High-dimensional data, Projection.

## 1 INTRODUCTION

Dimensionality reduction has long been a main visualization resource for analyzing and exploring high-dimensional datasets from various domains. Over the years, numerous dimensionality reduction methods have been proposed to translate high-dimensional data into a visual representation [4, 10, 19], most of which were designed to preserve geometric properties such as Euclidean distance between data points or distributions derived from it (e.g., see [33, 50]). An important chal-

lenge common to these methods is that the preservation of geometric properties can only be assured under specific conditions. Consequently, errors and distortions are highly probable in the resultant mapping. For example, structures present in the point cloud resulting from a projection, such as neighborhood relations, may not correspond to those in the original data, potentially misleading inexperienced practitioners and leading to incorrect conclusions.

To enable more robust and reliable analysis, recent dimensionality reduction methodologies incorporate theoretical guarantees into the mapping process. Ensuring the preservation of topological properties is a main trend in this context (e.g., [12, 23, 55]).

In this work, we focus on one such approach, TopoMap [13], which provides strong topological guarantees. Specifically, TopoMap ensures that the 0-cycles obtained by the Rips filtration of the projection are the same as in the original high-dimensional space. However, the consequence of providing such a guarantee results in TopoMap having two weaknesses. First, it makes inefficient use of the visual space especially for large datasets. Second, the computational cost can be exorbitant when the data size and/or dimension is high.

This work presents TopoMap++, an adaptation of the TopoMap [13] algorithm that renders the TopoMap layout more effectively in terms of visual space usage. The idea here is to understate the problematic

- Vitoria Guardieiro, Felipe Inagaki de Oliveira, and Claudio Silva are with New York University. E-mail: vitoria.guardieiro, felipe.oliveira, csilva@nyu.edu.
- Luis Gustavo Nonato is with ICMC-USP, São Carlos, Brazil. E-mail: gnonato@icmc.usp.br.
- Harish Doraiswamy is with Microsoft Research India. E-mail: harish.doraiswamy@microsoft.com.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxx/TVCG.201x.xxxxxx

points that are the cause of the inefficient visual space usage. We also provide a TreeMap [44] based exploratory mechanism that allows users to analyze high-dimensional data in a more robust and effective manner. The TreeMap is used to visualize the topological hierarchy of the high-dimensional data which allows for an intuitive exploration of the data set, thus making the analysis of the two-dimensional layout produced by TopoMap easier. The layout improvement mechanism when combined with the TreeMap-based interactive exploration facilitates the visualization of complex high-dimensional data structures by visually emphasizing the high-dimensional patterns found by the former. Note that, to the best of our knowledge, this the first projection approach that also allows for an interactive exploration.

Additionally, we present an approximation scheme that makes TopoMap more computationally efficient. This scheme drastically speeds up the most time-consuming step of TopoMap, the computation of the Euclidean minimum spanning tree.

In summary, the contributions of this work are:

- TopoMap++, a layout improvement scheme to highlight important structures in the TopoMap layout where the structures are identified using the notion of topological simplification;
- A novel topology-guided TreeMap-based exploratory mechanism that facilitates the analysis of complex high-dimensional data;
- An approximation scheme that makes TopoMap more computationally efficient. We show that this approximation preserves the topology of the input while attaining at least two orders of magnitude speedup.

We also present case studies that demonstrate the effectiveness of our approach in analyzing several high-dimensional datasets.

## 2 RELATED WORK

In order to better contextualize our contribution, we focus the related work discussion on techniques that explicitly rely on topology to perform dimensionality reduction. More comprehensive discussions can be found in a number of surveys approaching different aspects of dimensionality reduction, including general overviews [18, 33, 39], quantitative and qualitative comparisons [2, 19, 56], interaction tasks [42], layout enrichment [43, 46], model specificities [1, 6, 10, 53], and computational performance [52].

*Isomap* [47] is a pioneering technique that employs topological mechanisms for dimensionality reduction. It resorts to a graph representation that captures the topological structure of the data and enables the estimation of geodesic distances. Variants of Isomap have emerged to speed up computation [45], allow out-of-sample projections [3], and to handle spatio-temporal data [26]. Lee and Verleysen [29] improved the classical Isomap by building a graph representation that preserves non-contractable loops, enabling loop-preserving unfolding. Yan et al. [57] target the preservation of cycles present in the original data by selecting landmarks based on a topological scheme that captures the structure of 1-dimensional homology groups, which are ideally preserved during the dimensionality reduction. However, 0-homology groups are not considered, which consequently remain unpreserved. Gerber et al. [21, 22] build upon Yan et al. [57] to propose projection methods that are guided by a network derived from the maximum dimension cells of the Morse-Smale complex. However, rather than preserving topological structures, Gerber methods aim to encode information tailored for regression tasks.

The well-known UMAP [31] and t-SNE [49] techniques rely on KNN-graphs to capture the topological structure of high-dimensional data. While UMAP builds upon category theory and uses a force-directed scheme to project data to a visual space, t-SNE relies on the theoretical foundations of SNE [24], minimizing the KL-divergence between distance distributions defined in the original and visual spaces. The main difference between t-SNE and SNE is the distance distribution defined in the visual space, which is assumed to be a t-student distribution in t-SNE. UMAP and t-SNE techniques have been widely used in a variety of applications, but they in no way guarantee the preservation of topological properties. Doppalapudi et al. [12] proposed a topology-oriented force-directed layout that first generates an initial layout using the maximal spanning tree of a KNN graph of the high-dimensional

data. Users then select 0 and 1-dimensional topological features in the corresponding persistence barcodes. Nodes belonging to selected 0-dimensional cycles are attracted to emphasize the component while nodes in selected 1-cycles are arranged in an elliptical shape using tailored forces.

Based on a terrain metaphor, Weber et al. [55] proposed a method to depict, in a two-dimensional layout, topological features of a 3D scalar field. Specifically, they carefully design a 2D terrain whose elevation contour tree is guaranteed by construction to match the original data’s contour tree. However, Weber’s method ignores metric information, thus it may project 3D topological features that are far apart in the original data space closer to each other in the 2D layout. In the same line, Harvey and Wang [23] proposed a methodology to generate a terrain ensemble, each one with the same contour tree as the original data. Nonetheless, their method has the same shortcomings as Weber et al.’s [55] approach. Oesterling et al. [35] extended the terrain-based metaphor to high-dimensional data, relying on the Gabriel graph [20] to build a simplicial representation of the data and on a kernel density estimation to derive a scalar field that faithfully captures the high-dimensional point cloud organization. The authors also proposed several improvements on the original approach [34, 36], showing flexibility for application in different scenarios [37].

The literature also brings topology-inspired regularization schemes to enforce soft topological constraints in the dimensionality reduction process [32, 51, 54], but those methods do not provide any guarantee as to topological properties preservation. Topological tools have also been employed to evaluate and compare dimensionality reduction techniques [40, 41].

In contrast to the aforementioned techniques, the TopoMap [13] method offers theoretical guarantees as to the preservation of 0-dimensional homology groups, thereby ensuring that the connected components depicted in the projection layout match those in the original high-dimensional data. More specifically, TopoMap ensures, by construction, that the 0-dimensional persistence diagram of the Rips filtration of the projected data precisely mirrors that of the high-dimensional dataset. The topological guarantee provided by TopoMap renders it quite reliable as an analytical tool. In the present work, we tackle two main drawbacks of TopoMap, namely, computational cost and the inefficient use of visual space, thus substantially improving the analytical power of TopoMap.

## 3 BACKGROUND: TOPOMAP

In this section, we briefly introduce TopoMap and the related topological concepts. For an extended discussion over the topology topics, we refer the readers to Edelsbrunner and Harer’s book [17] and Chazal and Michel’s introductory article [7]. More details about TopoMap can be found in Doraiswamy et al. [13].

Consider a set  $P = \{p_1, p_2, \dots, p_n\}$  of  $n$  high-dimensional points in  $\mathbb{R}^d$ . Let  $\delta \geq 0$  be a threshold parameter. The **Victoris-Rips complex** [17] (also called Rips complex)  $Rips_\delta(P)$  is the set of all simplices  $K \subset P$ , such that  $d(p_i, p_j) \leq \delta$ ,  $\forall p_i, p_j \in K$ . Geometrically, given  $\delta$ , consider adding a  $d$ -dimensional ball of diameter  $\delta$  around each point  $p_i$ . The Vietoris-Rips complex is equivalent to the set of all simplices formed by the points whose balls intersect. The threshold  $\delta$  can be seen as the resolution we use to see the data set  $P$  [7].

The **Rips filtration** is defined as the nested sequence of subcomplexes  $K_i$  formed by increasing  $\delta$  from 0 to  $\infty$ . It is an ordered set of subcomplexes  $\mathbb{K} = \{K_0 = \emptyset, K_1, \dots, K_m\}$ , where  $K_i \subseteq K_{i+1}$  for all  $i \in [0, m-1]$ . Additionally, let  $\delta_i$  be the smallest threshold such that  $K_i \in Rips_\delta(P)$ . Then, for all  $i, j \in [0, m]$  with  $i < j$ , we have  $\delta_i \leq \delta_j$ .

As the threshold is varied, new simplices get added to the previous subcomplex. This addition can modify the topology of the subcomplex, where the topological features are  $k$ -dimensional cycles. Here, a 0-cycle is a connected component, 1-cycle is a loop, 2-cycle is a void, and so forth. The topology is modified by creating or destroying one (or more) of such cycles. For a given  $k$ -cycle, we define  $\delta_c$  and  $\delta_d$  as thresholds at which the cycle is created and destroyed, respectively. The **topological persistence** [16] of the given  $k$ -cycle is the difference between  $\delta_d$  and  $\delta_c$ . If a  $k$ -cycle is never destroyed, we define its persistence as infinite.

The creation and destruction thresholds of the topological cycles are used to define the **persistence diagram** [9] of the Rips filtration. Specifically, the persistence diagram is a scatter plot where each point is a cycle created during the filtration, with the coordinates being the creation ( $x$ -axis) and destruction ( $y$ -axis) thresholds. The vertical distance between the point and the identity line ( $x = y$ ) corresponds to the persistence of the cycle. We denote by  $PD_P^k$  the persistence diagram containing only the  $k$ -cycles of the Rips filtration over the set  $P$ .

TopoMap’s [13] projection approach tackles the following problem: given a set  $P = \{p_1, \dots, p_n\}$  in  $\mathbb{R}^d, d > 2$ , find a corresponding set of points  $P' = \{p'_1, p'_2, \dots, p'_n\}$  in  $\mathbb{R}^2$  such that  $PD_P^0 = PD_{P'}^0$  with point correspondences between the 0-cycles (that is, if  $p_i$  belongs to a certain 0-cycle in  $PD_P^0$ , then  $p'_i$  must belong to the corresponding 0-cycle in  $PD_{P'}^0$ ). TopoMap accomplishes this by using the following result about the *topology changing edges* (i.e., edges that merge two disconnected components into a single component) of the Rips filtration:

**Lemma 1 (Doraiswamy et al. [13], Lemma 2)** *Let  $\mathbb{K}_0 = \{e_1, e_2, \dots, e_{n-1}\}$  be the ordered set of topology changing edges of  $P$ . Then,  $\mathbb{K}_0$  is exactly the set of edges of the Euclidean distance minimum spanning tree ( $E_{mst}$ ) of the points  $P$  in increasing order of length.*

where the Euclidean Minimum Spanning Tree is defined as follows:

**Definition 1** *Consider a set of points  $P \in \mathbb{R}^d, |P| = n$ . Let  $K_E^n$  be the complete graph over  $P$ , where each edge has a weight equal the Euclidean distance between its end points. The **Euclidean Minimum Spanning Tree**  $E_{mst}$  of  $P$  is defined as the minimum spanning tree computed over the complete graph  $K_E^n$ .*

This lemma is then used to derive an iterative approach (Algorithm 1, black colored lines) to compute the projected points  $P'$ . The algorithm builds the projection one connected component at a time, where the connected components are processed in the order in which they are formed during the filtration defined by the  $E_{mst}$  (see Figure 4 in [13]). Note that this is equivalent to the Rips filtration, due to Lemma 1.

---

#### Algorithm 1 TopoMap++

---

**Require:** Set of points  $P = \{p_1, \dots, p_n\}$ ; **Set of Components**  $\mathbb{C} = \{C_1^*, C_2^*, \dots, C_k^*\}$

- 1: Compute the Euclidean minimum spanning tree  $E_{mst}$  of  $P$
- 2: Let  $E_{mst} = \{e_1, \dots, e_{n-1}\}$  be the edges ordered by length
- 3: Let  $l_{max}$  be the maximum edge length in  $E_{mst}$
- 4:  $P' = \{p'_1, \dots, p'_n\}$ , where  $p'_i = (0, 0), \forall i$
- 5: Let  $C_i = \{p'_i\}$  be the initial set of components
- 6: **for** each  $i \in [1, n - 1]$  **do**
- 7:   Let  $(p_a, p_b)$  be the end points of edge  $e_i$
- 8:   Let  $C_a, C_b$  be the components containing  $p_a, p_b$ , respectively
- 9:   Let  $l$  be the length of  $e_i$
- 10:   Place  $C_a$  and  $C_b$  in  $\mathbb{R}^2$  s.t.  $\min_{p'_j \in C_a, p'_k \in C_b} d(p'_j, p'_k) = l$
- 11:   Let  $C' = C_a \cup C_b$
- 12:   **if**  $C' \in \mathbb{C}$  **then**
- 13:     ScaleComponent( $C', l_{max}$ )
- 14:   **end if**
- 15:   Remove  $C_a$  and  $C_b$  from the set of components, and add  $C'$
- 16: **end for**

---

## 4 TOPOMAP++

The projections generated by TopoMap are star-shaped ensembles with (often long) branches. Most of the points are usually concentrated quite densely in the center of such stars, with the branches taking up most of the visual space of the projections. To analyze this projection, users must zoom into the centers of the different star shapes (e.g., see Figures 7, 8, and 9 of Doraiswamy et al. [13]). This introduces two important issues. First, users can often miss features of interest if the star shapes are too small (relative to the area taken by the entire

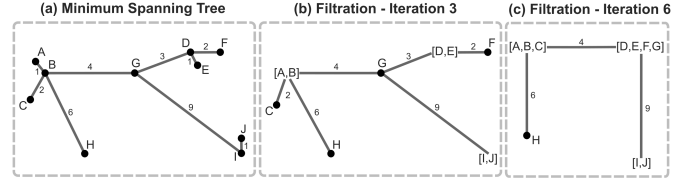


Fig. 2: Computing the hierarchical tree based on the Rips filtration defined by the minimum spanning tree. (a) MST over an input with ten points (labeled from A to J). The edges weights (length) are specified for each edge. (b) The set of components after the filtration has processed 3 edges of the MST. The merged components now become a single node labeled using “[ ]”. (c) The set of components after the filtration has processed 5 edges of the MST.

projection). Second, it is often hard to analyze multiple features at once, since when zooming into one, others often move out of frame.

This is primarily due to topological components with a single point (or components with very few points) that have high persistence during the Rips filtration. Such components spread the projection over the 2D space in order to maintain the topological consistency that is required by TopoMap. A consequence of this is that larger components with similar or lower persistence take up a significantly smaller fraction of the area (used by the projection), thus making it difficult to easily identify such features.

Our goal in designing TopoMap++ is to allow such dense components to also be emphasized in the projection. The main idea is to first identify such components, which can then be reflected in the layout. At the same time, we also aim to provide users with flexibility in the exploration of the topological components. In Section 4.1, we first describe how these components of interest can be identified using the notion of topological simplification. We then present, in Section 4.2, an alternate projection layout that adapts the original TopoMap layout to emphasize/highlight features of interest. This approach takes as input components of interest and generates a projection that focuses on the points present in these components. Finally, in Section 4.3, we propose using a TreeMap visual to allow users to interact and explore high dimensional data sets.

### 4.1 Identifying Components of Interest

Consider the Euclidean minimum spanning tree  $E_{mst}$  corresponding to a point set  $P$ . Let the edges of this tree be sorted in increasing order of edge weight. As mentioned in the previous section, these edges correspond precisely to the edges of the Rips filtration that merge two components (0-cycles). During this filtration, as each edge is added, new components are created by merging the two components that correspond to the end points of that edge. This creates a parent-child relationship between the components—the parent component is the union of the two child sub-components that are merged when an edge is processed. This can be represented as a tree, where each node corresponds to a component, and its children are the sub-components that are merged. Since only two components can merge for any edge, this tree becomes a binary tree.

For example, consider an example  $E_{mst}$  as shown in Figure 2(a). This was built on a point set  $P$  with  $|P| = 10$ . Figures 2(b) and (c) show the components that are created at different stages of the filtration. The binary tree representing the hierarchy formed by the filtration is shown in Figure 3(a). Each node in the binary tree corresponds to a component that is created during the Rips filtration.

Our goal is to choose a disjoint set of components that are formed during the filtration to be highlighted during the projection. Furthermore, we want to choose only components that are reasonably large, i.e., the size of the component (number of points) satisfies a minimum size criteria  $\eta$ . Thus, this set would not include the smaller high persistence components that are cause of the “long branches”.

We adapt the notion of topological simplification [5] to identify components that satisfy the above property. Specifically, given a size threshold  $\eta$ , we simplify the hierarchical binary tree to identify only



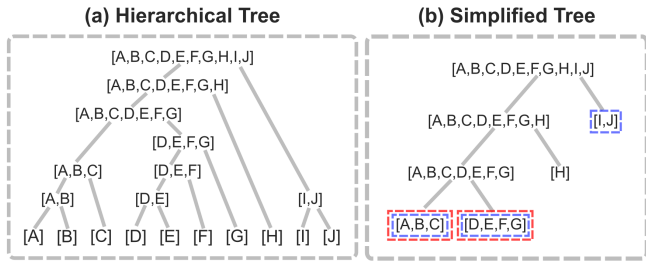


Fig. 3: **(a)** The hierarchy of the components formed during the filtration in Figure 2 is represented as a hierarchical binary tree. **(b)** Simplified tree when  $\eta = 2$  (as well as when  $\eta = 3$ . The components chosen when  $\eta = 2$  are shown by a blue border, while those chosen when  $\eta = 3$  are shown by a red border.

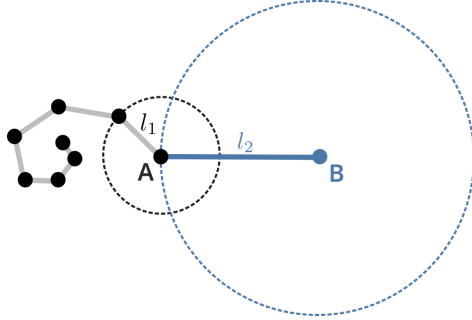


Fig. 4: Illustration of how the edge lengths impact the point density. The black points form a component and the segments correspond to the edges processed to form said component. Point A has two edges in the MST – one connecting it to the black component (with distance  $l_1$ ) and the other connecting it with point B ( $l_2$ ). Since  $l_1 < l_2$ , A is the only point inside the ball centered at it with radius  $l_1$ . Similarly, B is the only point inside the ball around it with radius  $l_2$ . Since all edges between the points in the component are way smaller than  $l_2$ , their density in the visual space is higher than the density around point B.

components that have size at least  $\eta$ . This is accomplished by repeatedly merging leaf nodes in the binary tree that have size less than  $\eta$  until no more leaf nodes can be merged. Note that a leaf node can be merged (simplified) if and only if the node it is merging with is also a leaf. At the end of this procedure, each leaf node of the simplified hierarchical binary tree with size greater than or equal to  $\eta$  corresponds to a component that contains a dense set of points as defined by the filtration. Figure 3(b) shows the simplified tree, when  $\eta = 2$  and  $\eta = 3$ , for the tree shown in Figure 3(a). Note that, when  $\eta = 3$ , the component  $[I, J]$  cannot be merged since its sibling node is not a leaf. However, when choosing the components of interest, this node is not considered.

## 4.2 Space Efficient Layout

The goal of our projection approach is enable the simultaneous highlighting of multiple features of the projection while still allowing users to visually infer the topological properties.

Suppose that we have a list  $\mathbf{C} = \{C_1^*, \dots, C_k^*\}$  of  $k$  disjoint components of interest. That is,  $C_i^* \cap C_j^* = \emptyset, \forall i \neq j$ . To highlight these components, our idea is to let these components take up more visual space when compared to other components. To accomplish this, we propose *scaling* their projections to increase the area used by these components. Formally, for each component  $C_i^*, i \in [1, \dots, k]$ , our goal is to find a scalar  $\alpha_i > 1$  such that for each  $j \in C_i^*$ , we replace  $p_j'$  by  $\alpha_i \cdot p_j'$  (wlog., assume that the center of this component is the origin). The TopoMap algorithm is modified to perform this scaling as soon as the component of interest is created (lines 11–13 in Algorithm 1).

To determine the scalar  $\alpha_i$ , we consider the edges in the MST that were processed to form the component. For a given data point  $j \in [1, \dots, n]$ , let  $l_j$  be the length of the shortest edge in the MST containing  $j$ . In the projected space, there will be no other point inside the 2-dimensional ball of radius  $l_j$  centered in  $p_j'$ . Therefore, points with

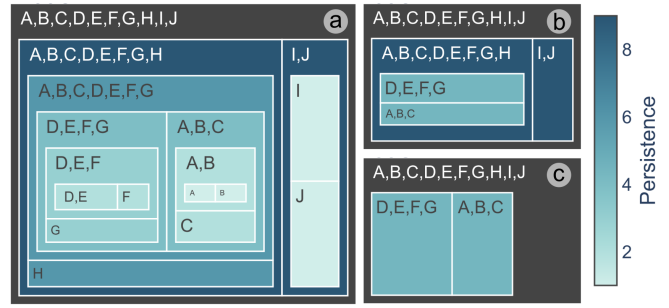


Fig. 5: TreeMaps corresponding to **(a)** the unsimplified hierarchical tree; **(b)** simplified tree with  $\eta = 2$ ; and **(c)** simplified tree with  $\eta = 3$ . The gray color represents the component with infinite persistence.

large shortest edges will be in sparse regions of the visual space, forcing the points with shorter edges to be cramped together (see Figure 4). To highlight the components of interest, we scale the corresponding edges to increase their size, thus allowing the points to be placed farther apart. This is illustrated in Algorithm 2. Specifically, let  $L_i, i \in [1, \dots, k]$  be the average edge length of component  $C_i$ . We would like  $L_i$  to be at least as large as the biggest  $l_j$  for any point  $j$ . In other words, we want  $L_i \geq l_{max}$ , where  $l_{max}$  is the longest edge of  $E_{mst}$ . With that, we set the scaling factor  $\alpha_i$  to be  $\alpha_i = c \frac{l_{max}}{L_i}$ , where  $c \geq 1$  is a constant that can be used to control this scaling. Note that we also allow users to optionally impose an upper bound on the scaling parameter  $\alpha_i$  so that no single component takes up a disproportionate fraction of the visual space.

### Algorithm 2 ScaleComponent

**Require:** Component  $C^*$ ; Max edge length  $l_{max}$ ; Constants  $c \geq 1$  and  $\alpha_{max} \geq 1$

- 1: Let  $L$  be the average edge length in  $C^*$
- 2: Set  $\alpha = \min \left\{ c \frac{l_{max}}{L}, \alpha_{max} \right\}$
- 3: **for**  $p \in C^*$  **do**
- 4:     Set  $p = \alpha p$
- 5: **end for**

While this approach will no longer have the strong topological guarantees of the original TopoMap projection, it still retains the topological guarantees in the local neighborhood of the highlighted components. That is, the filtration corresponding to a highlighted component is the same (up to a constant scale factor) in both the input high-dimensional space as well as the project 2D space.

As an example, consider the MNIST dataset [?] that is composed of images of handwritten digits. Figure 1(a) shows the original TopoMap projection of this data. As can be seen from the figure, the long branches result in an inefficient use of the visual space. The TreeMap++ layout that highlights the components that remain after simplifying the corresponding hierarchical tree using a value of  $\eta$  set to be 1% of the size of the dataset is shown in Figure 1(c). Note the more efficient use of the visual space using this proposed layout.

## 4.3 TreeMap-based Exploration

The above approach chooses all components that satisfy the  $\eta$  threshold. To provide more flexibility in data exploration to the user, we also aim to allow users to choose the components of interest. This allows the user to focus on fewer components, which in turn allows relatively more space to the components of interest.

To accomplish this, we propose to use the TreeMap [44] visual that can visualize the simplification hierarchy represented by the hierarchical binary tree. Such a TreeMap provides an abstract hierarchical representation of the topology of the high-dimensional space. However, it can be visually cluttered if all components in the hierarchy are represented. Thus, we choose to represent only the hierarchy present in the simplified binary tree to generate this visualization (using the  $\eta$  parameter specified by the user). We modify the original TreeMap layout to



include boxes for a component *if and only if* this component satisfies the threshold criteria. This enables users to focus only on components identified after simplification (see Figure 5). Furthermore, we color the boxes of the TreeMap to indicate properties of the components, such as persistence. By inspecting and interacting with the TreeMap, the user can define which component(s) to highlight in the projection.

## 5 APPROXIMATE MST

Recall that TopoMap’s theoretical guarantees stem from the equivalence between the topology-changing edges in the Rips filtration and the Euclidean distance minimum spanning tree (EMST). As mentioned in [13], the minimum spanning tree could also be computed using a different distance metric, such as cosine distance, and used for the projection. For the rest of this section, we assume that the distance metric is the Euclidean distance.

Computing the EMST, however, can be prohibitively expensive for large and/or high-dimensional datasets. To address this problem, we propose to use an approximation of the EMST instead of the actual EMST. To do this, our idea is to first reduce the complete graph defined by the input points to a substantially smaller subgraph  $G'$  and then compute the MST of  $G'$ . The key-insight to compute this smaller subgraph  $G'$  is inspired by the state-of-the-art approximate nearest neighbor (ANN) algorithms. In this section, we first describe the Vamana graph which provides us with the necessary subgraph  $G'$  to compute the approximate EMST. We then evaluate this approximation, demonstrating its effectiveness with respect to the quality of approximation and its efficiency compared to computing the exact EMST.

### 5.1 Vamana graph

A *Relative Neighborhood Graph* (RNG) is an undirected graph constructed on a set of points  $P$  such that there is an edge between two points  $u, v$  if and only if there is no point  $p \in P$  that is closer to both  $u$  and  $v$  than they are to each other. More formally, for a set of points  $P$  in a metric space with distance  $d$ , the RNG of  $P$  is a graph with vertex set  $P$  and set of edges equal to those pairs  $(u, v)$  such that  $d(u, v) \leq \max_{p \in P \setminus \{u, v\}} (d(u, p), d(v, p))$ . An RNG allows for an efficient identification of the nearest neighbors corresponding to a query point. However, since computing the RNG is expensive, especially for higher dimensional data sets, several ANN algorithms use an approximate variant of the RNG.

By definition, an EMST is a subgraph of the RNG. Given this, our idea is to use the *Vamana graph* [25], which is a sparse approximation of the RNG, to compute an approximate EMST. For completeness, we briefly describe the Vamana graph construction algorithm next. We refer the reader to the DiskANN paper [25] for more details.

The Vamana indexing algorithm constructs the graph iteratively as follows. It takes 3 parameters as input: a distance threshold factor  $\alpha$  that determines the diameter of the approximate RNG; an upper bound on the out degree of each node,  $R$ ; and the allowed search list size  $L$  used for doing a greedy ANN search during the graph build process. It first computes a random  $R$ -regular directed graph  $G'$  over the input set of points  $P$ . Then, during each iteration, it searches for the approximate nearest neighbors of a random point in  $p_i \in P$  that is not yet processed and updates the out-neighbors of this point  $p_i$  based on the search results. Additional pruning is done during each step to ensure that the out-degree of the graph  $G'$  is within the upper bound  $R$ .

To compute the EMST, we first compute the Vamana graph  $G'$  and then compute the EMST using this graph.

### 5.2 Evaluation

To compute the approximate EMST, denoted as *AMST*, we first compute the Vamana Graph  $G'$  with  $\alpha = 1.3$ ,  $L = 100$  and  $R = 100$  and then extract the MST of  $G'$ . We found that this set of parameters provided a good tradeoff between accuracy and running time. The exact EMST used for comparison was computed using the Dual-Tree algorithm [30] that was implemented as part of the MLPACK library [11]. Note that this was the same implementation that was also used in [13].

Table 1 presents the number of data points, dimensions, and classes of the datasets used in our evaluation. The first 5 datasets are well

Table 1: Data sets used in our experiments.

Data set	# Points	Dimension	# Classes
Iris [15]	150	4	3
Seeds [15]	210	7	3
Mfeat [15]	2000	64	10
MNIST [?]	60000	784	10
BIGANN [28]	100000	128	not labeled
LLM	6669	4096	2
Urban	17520	6	not labeled
StreetAware	363134	768	3

Table 2: Comparing time to compute the EMST and the AMST. Note that as the data size/dimension increases, we are able to achieve a significant speedup in the running times.

Dataset	Running Time (sec)		Speedup
	EMST	AMST	
Iris	0.003	0.02	0.15
Seeds	0.002	0.012	0.17
MFeat	0.332	0.302	1.1
MNIST	9010	28	321.8
BIGANN	2317	20	115.8
LLM	542	1.3	416.9
Urban	0.18	1.03	0.17
StreetAware	92631	156	593.8

known open datasets and are used purely for the quantitative evaluation, while the last three datasets are used for both the quantitative evaluation as well as the case studies discussed in the next Section. All experiments were run on a machine with Intel (R) Core(TM) i9-12900KF running at 3.19GHz and 32 GB of memory.

#### 5.2.1 Efficiency

Table 2 shows the performance improvement of our proposed approach for computing the AMST. We note that for small low-dimensional datasets, building the Vamana graph incurs a small overhead. However, given that the total running time itself is very small, this is insignificant. However, as the data sizes/dimensions increase, we note that our approach provides a significant speedup over computing the exact MST, attaining over **two orders of magnitude** speedup.

#### 5.2.2 Approximation Quality

We use the following two metrics to evaluate the approximation quality of our AMST approach:

1. **Bottleneck Distance:** This measure is used to assess the topological similarity between the filtration defined by the AMST and the EMST (i.e., the original Rips filtration). It is computed as the bottleneck distance between the persistence diagrams defined by the two filtrations. Note that during these computations, the persistence diagrams are normalized to ignore any scaling effects.
2. **Relative Weight Error (RWE):** This measure is used to assess the quality of the approximation attained by the AMST. It is defined as the difference between the total weights of the AMST and EMST normalized by the weight of the EMST:  $RWE(AMST, EMST) = \frac{W(AMST) - W(EMST)}{W(EMST)}$ . Here,  $W(\cdot)$  is the total weight of a given weighted tree.

Table 3 compares the above two metrics for the different datasets shown in Table 1. Note that for all datasets, the relative weight error between the AMST and EMST is very small (in the order of  $10^{-1}\%$  or less). We notice that, for smaller datasets, the AMST is exactly the same as the EMST (RWE = 0). Even in cases where the AMST does not match the EMST, we notice that the bottleneck distance between the persistence diagrams is still small, thus ensuring that the topology of the projection is still mostly preserved with significantly lower computational effort.

Table 3: Approximation quality of the AMST.

Dataset	Bottleneck Distance	RWE
Iris	0	0
Seeds	0	0
Mfeat	0	0
MNIST	$2.4 \times 10^{-2}$	$1.86 \times 10^{-4}$
BIGANN	$2.6 \times 10^{-2}$	$6.39 \times 10^{-4}$
LLM	$7.5 \times 10^{-2}$	$2.01 \times 10^{-3}$
Urban	$1.3 \times 10^{-2}$	$4.55 \times 10^{-3}$
StreetAware	$6.0 \times 10^{-2}$	$7.74 \times 10^{-4}$

## 6 CASE STUDIES

In this section, we first discuss the layout and interpretation of TopoMap++. We then describe three use case scenarios that uses TopoMap++ to analyze datasets. Note that, unless otherwise mentioned, we use  $\eta = 1\%$  of the dataset size as the simplification threshold.

### 6.1 TopoMap++: Layout and Interpretation

TopoMap’s layout consists of star-shaped ensembles with branches connecting and emanating from them. The center of the star shapes are denser than the branches and contain points that are closer together in the original space. These correspond to dense topological components that could be of interest in the analysis. For example, the centers in Figure 1(a) make up points of the same digit.

Since the projected points satisfy the topological filtration guarantee, the branches connecting any two such components indicate the order in which these components merge during the filtration. This could be used to understand the “connection relationship” – the points that are responsible for connecting these components. Returning to the MNIST example, we see that such a branch connecting two components typically contains points that gradually change patterns between the components – in this case we see a transition between the digits that are connected (e.g., see Figure 7 of Doraiswamy et al. [13]).

A common trait seen in various datasets is that they often also contain several small components (sometimes with just 1 point) that have high persistence. When there are a lot of high-persistent small-sized components, the layout tends to make inefficient use of the visual space. This makes it difficult to (1) identify components of interest; and (2) understand the relationship between these components. Both these shortcomings are addressed using TopoMap++, as we show next using the case studies. Using TopoMap++, selected components are enlarged and may be emphasized with colored regions in the projection plot. Since we make use of the notion of topological simplification to identify components of interest, TopoMap++ allows for a better analysis of the topological components themselves.

In the MNIST example, we can barely notice the presence of most of the star centers using the TopoMap layout (Figure 1(a)). In contrast, the TopoMap++ projection (Figure 1(c)) more clearly displays these centers, thus allowing the user to see all of the points that form each of these components. At the same time, since such components are already identified and emphasized using TopoMap++, it now becomes possible to also study the relationship between them much more easily. However, care should be taken while analyzing TopoMap++ layout to remember that the filtration (distances up to a constant scaling) is consistent within each emphasized component but not across them.

### 6.2 Case Study 1: Unlabeled Urban Data

In the first case study, we directly compare TopoMap++ with TopoMap for the same unlabeled urban data set used in Doraiswamy et al. [13](Section 4.2), with the goal to assess: (1) the ease of exploring the projection using our proposed approach; and (2) the quality of the features identified/explored when compared to a manual exploration.

The data set contains six features for a 100-meter radius region in Times Square (precipitation, temperature, wind speed, count of taxi pickups, average fare, and average distance) for hourly intervals during 2014 and 2015 (for a total of 17,520 six-dimensional data points). In Doraiswamy et al. [13], this data set is manually explored through the

TopoMap projections. Here, we show that TopoMap++ significantly improves the unlabeled data exploration compared to TopoMap.

As discussed earlier, generating the TopoMap++ projection first computes and simplifies the hierarchical binary tree. This layout is shown in Figure 6(c), where each component has a different color. The corresponding TreeMap is shown in Figure 6(b). Figure 6(a) shows the TopoMap projection for the same dataset in which we can barely see some of the components (for example, those in pink and gray) — these components could easily be missed during the visual exploration.

Next, we analyze how well the automatically selected components compare to the features found manually by Doraiswamy et al. [13]. Figure 7 shows the histograms of the month and hour of the day for each of the eight components. Most of the patterns found in [13] match the components found automatically here: the biggest component (in orange) corresponds to intervals during the spring and summer months during the day (between 8 am and 2 am). This pattern corresponds to the points shown in [13], Fig. 8(b); the component in red corresponds to night intervals from spring and summer (matching cluster [13], Fig. 8(f)); the teal component contains day intervals during the winter (matching [13], Fig. 8(d)); yellow has day intervals during the spring and fall (matching [13], Fig. 8(e)).

Beyond matching the clusters from [13], the automatic selection proposed in this paper also found additional patterns. The green component finds mostly the night hours during Winter (with a few points bleeding from Fall and into Spring). The purple one contains the periods during the early hours of the day after midnight for the fall and winter months, complementing the red component in terms of season. The pink component has summer periods during the night (mostly before midnight) when there are more taxi pickups than usual during the night. This ability to quickly and easily find features of interest can greatly help in the analysis of high dimensional datasets.

In addition to easily finding such interesting components, we can now further examine their structure, such as neighborhoods, to better understand the relationship between them. As mentioned above, components closer merge earlier in the filtration. For example, we notice that the red component has two components in its neighborhood—pink and purple. The pink component corresponds to the same monthly intervals as the red one, while differing in the hours of the day that these points represent. On the other hand, the purple component represents the same hours of the day as the pink component while differing in the months (seasons) that it represents.

Note that the only component from Doraiswamy et al. [13] that was not automatically identified here is the set of periods with rainfall ([13], Fig. 8(g)). This component has a small size but is high persistent, which is basically the type of component that gets simplified by our approach. While such components can be obtained by manually exploring the projection, in future, it would be interesting to identify alternate approaches to automatically highlight such components as well.

### 6.3 Case Study 2: Analyzing LLM Embeddings

The analysis of high dimensional embeddings (or hidden states) of large machine learning models, in particular large language models (LLM), is a current hot topic in high dimensional analysis. By analyzing these embeddings, we can further understand how those models interpret text, answer questions, and perform tasks. In this case study, we employ TopoMap++ to explore patterns in the embedding space of a large language model when answering multiple-choice questions. We use the AI2 Reasoning Challenge (ARC) data set [8], which contains grade-school level questions split into Easy and Challenge sets. To answer those questions, we used the Llama 2 model with 7B parameters [48]. Ten sample questions were provided, with answers, to the model as examples (i.e., a fewshot learning). Then, we asked the model the questions from the ARC question set and extracted the embeddings that resulted in the model’s answer. In particular, we looked at the embeddings after the last attention layer of the model. For each of the 6,669 questions<sup>1</sup>, we have one embedding of dimension 4,096. For

<sup>1</sup>We selected the ARC questions whose options were letters (A, B, C, or D) from the Train and Test sets for the Easy and Challenge levels. The examples

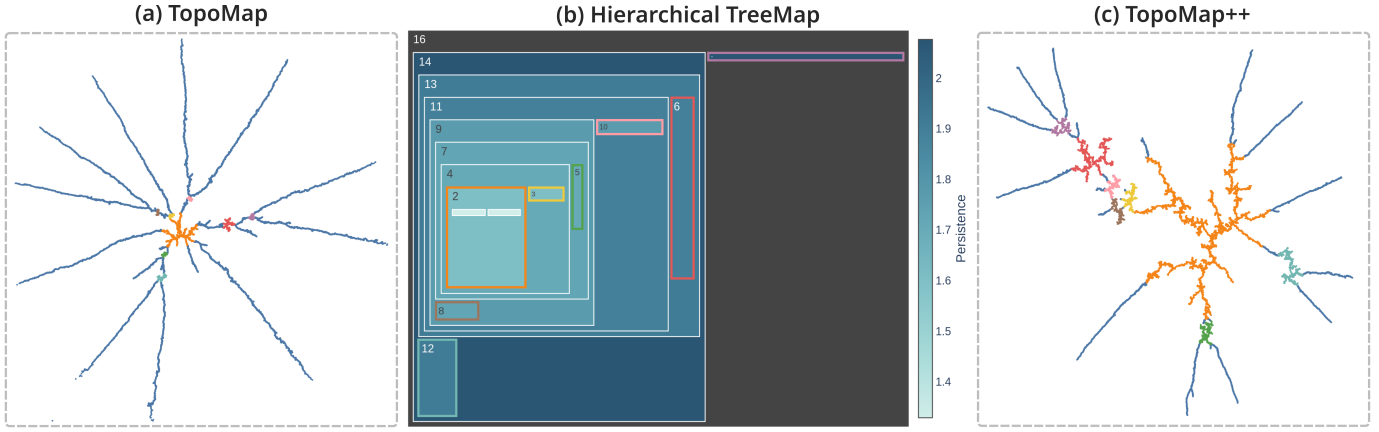


Fig. 6: **(a)** TopoMap projection, **(b)** Hierarchical TreeMap, and **(c)** TopoMap++ projection generated using the urban data set from Case Study 1. The selected components in **(b)** are used to emphasize the clusters in the TopoMap++ projection **(c)**. The same points are also colored with the corresponding colors in the original TopoMap projection **(a)**. Note that these components end up being small due to the inefficient use of the visual space by the original algorithm. Using our proposed approach, it becomes easy to identify and analyze such features in the data.

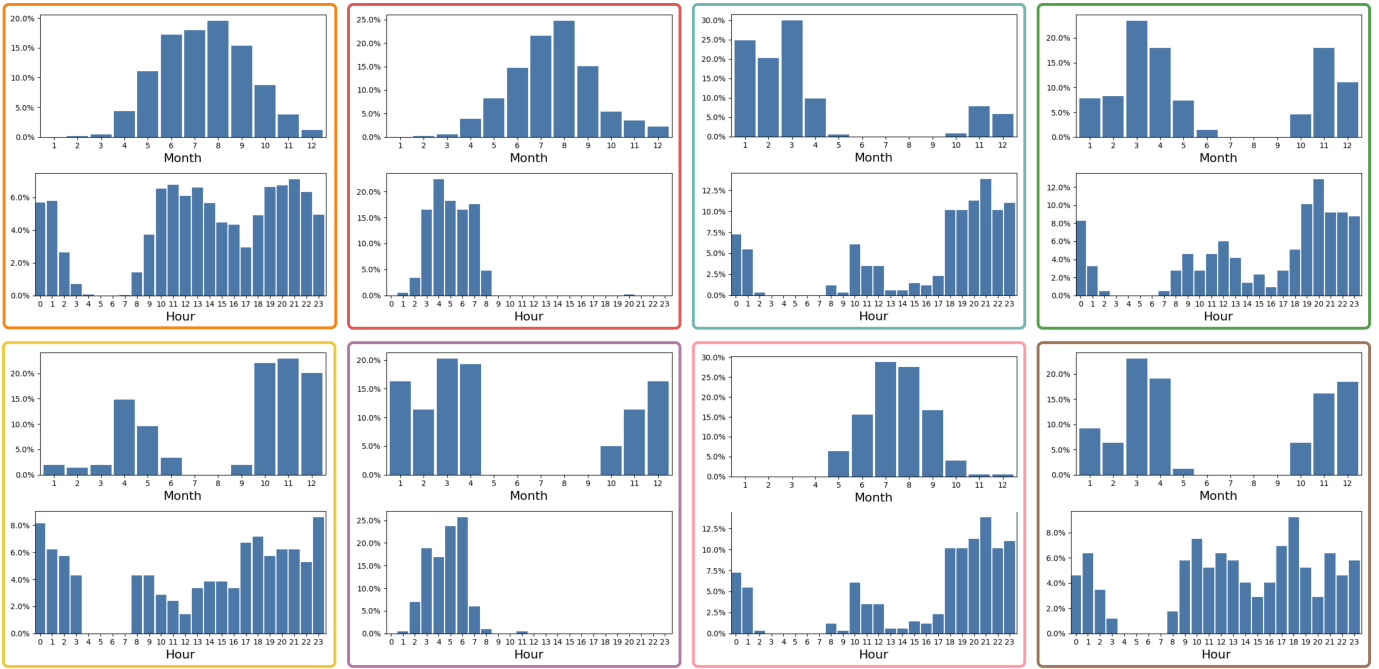


Fig. 7: Histograms of the temporal components (month and hour) for each cluster highlighted in Figure 6. The colored boxes correspond to these clusters. The components are ordered (from left to right, top to bottom) according to their volume (i.e., number of points).

Table 4: Number of correct and model answers from each option.

Choice	Correct Answer	Model Answer
A	1,589	1,411
B	1,721	2,066
C	1,745	1,785
D	1,614	1,407

reference, 71.6% of the questions were answered correctly, with an accuracy of 77.6% for the Easy set and 59.7% for the Challenge set. Also, Table 4 shows the number of correct answers of each possible choice and the number of answers the model gave.

We computed the TopoMap++ projection for this data set setting  $\eta$  as 50 (1% of the data size)—this resulted in five large disjoint components as shown in Figure 8(a). Since the questions had four possible answer options, we first explored these five components to assess how they associate with the four answer options. Here, the highlighted components are enclosed within grey convex hulls, and the points are

provided were randomly selected from the Dev sets.

colored based on the LLM’s answer. In Figure 8(b), we see that each of the four outer components corresponds to one of the four choices made by the LLM. The fifth component is a central component whose points correspond to a mix of all four answer options but with more B and C answers than the others.

We then analyzed whether the LLM provided the correct answer or not for the questions in each of the five components, as shown by the blue bars in Figure 8(c). It is interesting to note that all the answers made by the LLM for questions in the outer four components were correct, while the central region consists of questions where the LLM mostly answered incorrectly. The latter also holds for the points in the long branches connected to this central component. Figure 8(c) also shows, as green bars, the proportion of questions from the Easy set for each component. Note that this is very close to the proportion of correct answers, implying that each outer component corresponds to only easy questions.

Furthermore, the presence of this central component indicates a contiguous region in the higher dimensional space where the LLM frequently makes mistakes. Based on the projection, it is also clear



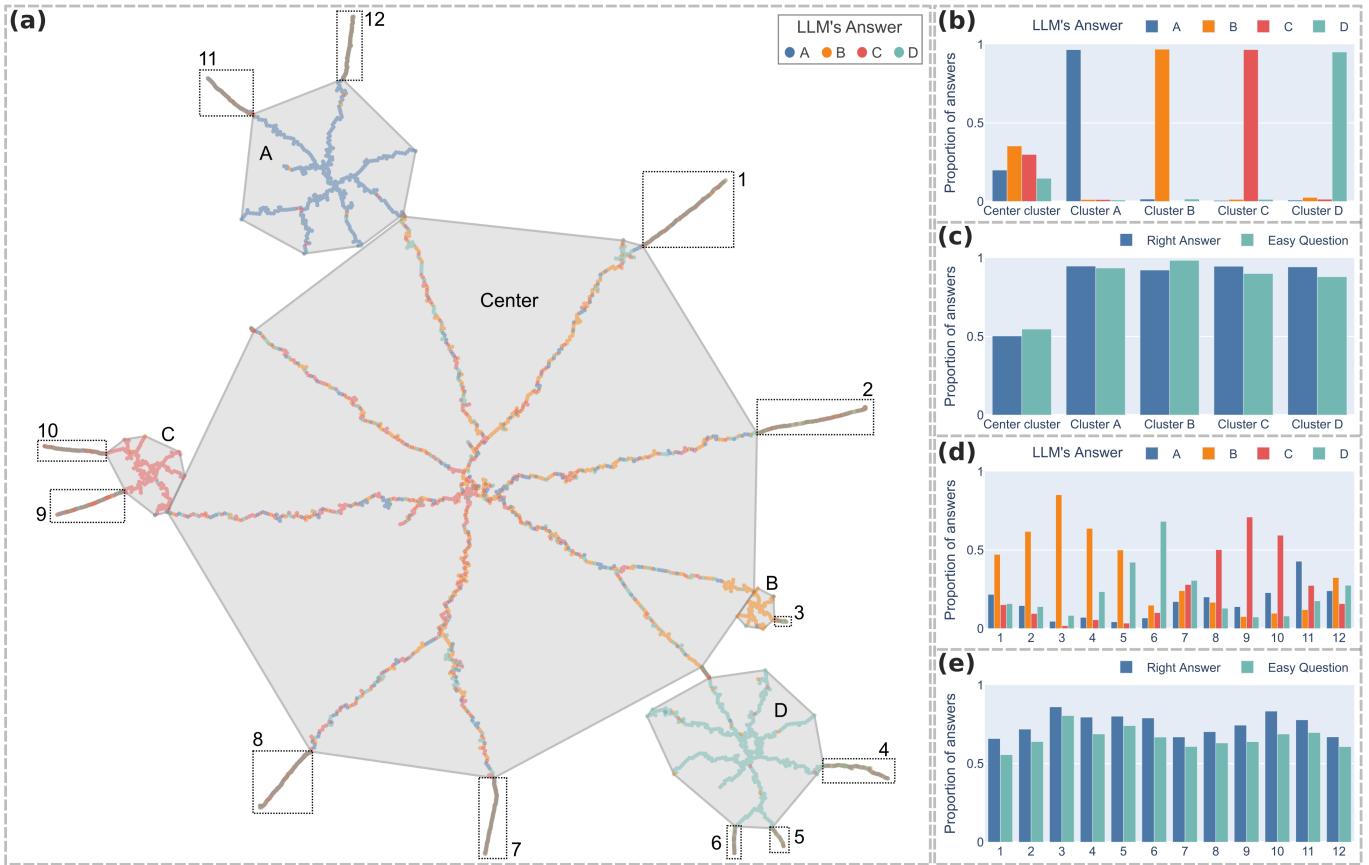


Fig. 8: (a) TopoMap++ projection of the LLM embeddings dataset explored in Case Study 2. The grey regions indicate the convex hull of the five highlighted components. There are also twelve branches besides the highlighted components, which are indicated with bounding boxes and are numbered clockwise. The proportion of each option answered by the LLM is shown in (b) and (d) for the highlighted components and the numbered branches, respectively.

that this region separates other regions that mostly answer correctly. This knowledge can potentially be used for improving both the answer quality (e.g., by augmenting the inference with additional information when the embedding falls within such region), as well as fine tuning the LLM by focusing on such regions.

Beyond the components themselves, the projection in Figure 8(a) also displays twelve branches that are not contained by any component. Four of those branches (numbered 1, 2, 7, and 8) are directly connected to the central component, while the other eight are connected to one of the outer components. In Figure 8(e), we see the proportion of correct answers (in blue, ranging from 66% to 86%) and Easy questions (in green, ranging from 55% to 80%) for each branch. Compared to the central component, all branches have higher accuracy and more Easy questions. However, those values are smaller than the ones corresponding to the outer components. The branches directly connected to the central component show a slightly lower accuracy and also contain a smaller proportion of Easy questions than the other branches.

Lastly, Figure 8(d) shows the proportion of answer options in each branch. We notice that the branches connected to an outer component tend to have more answers of the corresponding option (branch 3 has mostly B answers, branches 6 and 7 mostly D, branches 9 and 10 mostly C, and branch 11 mostly A). However, the branches close to the B component (from 1 to 4) have mostly B answers, even though branch 4 is connected to component D. A similar pattern happens from branches 8 to 10, which have mostly C answers. The only branches with mostly A or D answers are branches 11 and 6, respectively. This is consistent with our previous observation that the B and C answers are scattered across the embedding space instead of being closer together as we see with answers A and D. This might indicate that the LLM might be biased towards choices B and C, and they therefore become the “default” choices when the model does not know the answer.

#### 6.4 Case Study 3: Vision Transformer Embeddings

Examining the embeddings of Vision Transformer (ViT) provides useful information on how neural networks handle visual data, assisting in the identification of objects and comprehension of scenes. Understanding these embeddings elevates the transparency and durability of the model, which is critical for dependable AI systems. In this case study, we use TopoMap++ to analyze the embeddings of ViT [14]. For this study, we use the StreetAware dataset [38], which is a dataset with fully anonymized high-resolution videos from urban environments. From this dataset, we selected a 46-minute video stream that was recorded using a fixed camera in an activity-rich traffic intersection and then utilized the YOLOv8 [27] detection model to detect bounding boxes for each object identified as ‘car’ or ‘person’ in every video frame. We then crop the images using these bounding boxes and compute the ViT model embedding for each of the cropped images. So, for each ViT embedding, there’s a YOLOv8 score and label.

The TopoMap++ projection generated by using a value of  $\eta = 3600$  (approximately 1% of the number of data points) resulted in highlighting 6 components as shown in Figure 10(a). Figure 10(b) shows the same visualization but with the points colored based on its labels. Note that each component corresponds to a single label. Interestingly, each label is split into multiple sub-components. To analyze this, we look at the images corresponding to these points. Figure 10(c) contains sample images from each sub-component where we can see that these components form a pattern within each category. For example, the cyan and green components consist of embeddings related to two static cars, respectively, that were parked throughout the duration of the video. The images are part of the frames when the car was not completely obstructed by traffic. The lilac and yellow colored components correspond to cars that show a specific profile. Specifically,

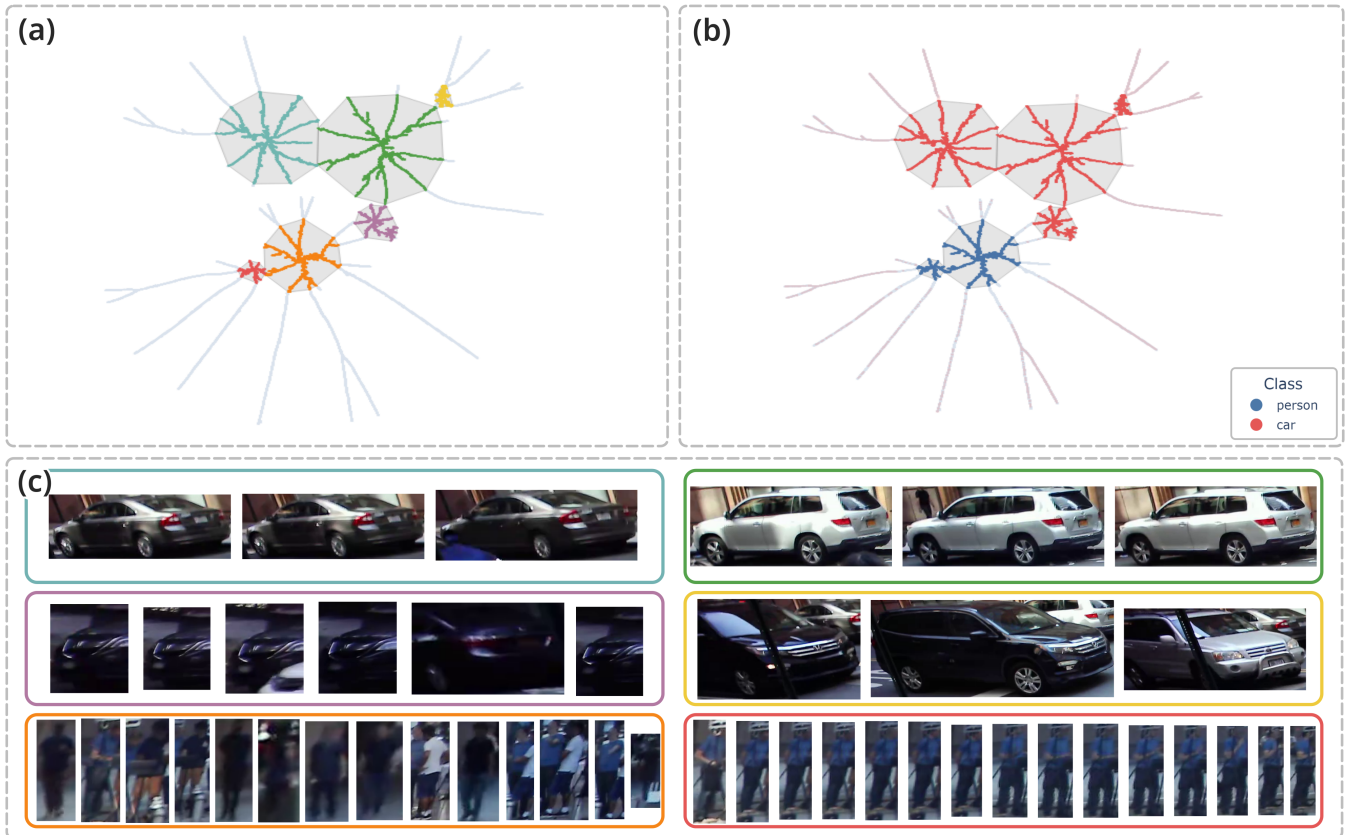


Fig. 10: TopoMap++ projections of the Street Aware embeddings data set. The grey regions indicate the convex hull of the highlighted components. (a) Projection the StreetAware data using TopoMap++ highlighting six prominent components. (b) The points in the projection are colored based on the YOLOv8 classification label. Note that even within the same class, there are sub-clusters. (c) Sample images corresponding to the different highlighted clusters are shown. We can see that the sub-clusters within a given class correspond to different variations of the data.

the yellow component consists of different cars that were captured by the camera and correspond to cars that stopped at the traffic light.

The red component has images of the same person positioned at the same place. Note that this person was present throughout the video. On the other hand, the orange component shows a greater diversity of people passing through the camera’s view. One can also see that the path from the red component (clear image of a person) to the green component (clear image of a car) is via the orange and lilac components, both of which correspond to more blurry images, which indicates that the model clearly differentiates between the images when the objects in them are clear.

## 7 LIMITATIONS AND DISCUSSIONS

As seen in the previous Section, the addition of the nested TreeMap for exploration and the ability to highlight features of interest in TopoMap++ help users easily analyze high-dimensional datasets. At the same time, there are some limitations to these techniques that we aim to tackle going forward.

**Improving TreeMap Interaction.** Recall from Section 4.3 that the rectangles for the TreeMap are created only when a component has at least  $\eta$  points. Thus, a parent box is composed of points that are within its child boxes together with these outlier points. One shortcoming of this is that it does not allow users to select such outliers. One way we are thinking of incorporating this is to add an explicitly marked “outlier box” which can then be used to explore such points.

**Parameter identification.** Using TopoMap++ requires users to specify a few parameters. First, there is  $\eta$  which defines the simplification threshold, which is set  $\eta$  to be approximately 1% of the input size. While this gave us good results for the data used in this paper, this might not always be the case. Second is the parameter  $c$ , which is used to compute the scale factor  $\alpha$ . In this paper, we fix the value of  $c$  to be

2. But, we could instead try to use the amount of white space present in the original TopoMap visualization to compute  $c$  such that the resulting TopoMap++ visualization minimizes the amount of wasted space.

## 8 CONCLUSION

In this paper, we presented TopoMap++, that improves the visual space usage of the TopoMap [13] projection. This is accomplished by selectively scaling, and thus highlighting, topological features of interest. This reduces the space usage of the outliers that were the primary cause of the wasted space in the original TopoMap projection. While the resulting global layout no longer provides the topological guarantees of the original projection, these guarantees are still locally preserved within each highlighted component. We also make use of the hierarchy formed by topological simplification to design a nested TreeMap-based visualization that allows users to easily interact with and analyze high-dimensional datasets using the TopoMap++ projection. We also propose an approximation scheme to compute the Rips filtration inspired by state-of-the-art ANNS algorithms. We show that this approximation preserves the topology of the data with at least a two orders of magnitude improvement in the computational cost.

This work still focuses on the topology of the 0-cycles of the Rips filtration. However, given the complex topologies that are possible in higher dimensions, in the future, we intend to explore techniques that would enable TopoMap/TopoMap++ to also preserve/portray 1-cycles.

## ACKNOWLEDGMENTS

This work was supported by DARPA ASKEM, DARPA PTG, Capital One, and the NSF award CSSI 2411221. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, Capital One, or NSF.

## REFERENCES

- [1] M. Ashraf, F. Anowar, J. H. Setu, A. I. Chowdhury, E. Ahmed, A. Islam, and A. Al-Mamun. A survey on dimensionality reduction techniques for time-series data. *IEEE Access*, 11:42909–42923, 2023. doi: 10.1109/ACCESS.2023.3269693 2
- [2] S. Ayesha, M. K. Hanif, and R. Talib. Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Information Fusion*, 59:44–58, 2020. doi: 10.1016/j.inffus.2020.01.005 2
- [3] Y. Bengio, J.-f. Paiement, P. Vincent, O. Delalleau, N. Roux, and M. Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In S. Thrun, L. Saul, and B. Schölkopf, eds., *Advances in Neural Information Processing Systems*, vol. 16. MIT Press, 2003. 2
- [4] I. Borg and P. Groenen. Modern multidimensional scaling: Theory and applications. *Journal of Educational Measurement*, 40:277–280, 06 2006. doi: 10.1111/j.1745-3984.2003.tb01108.x 1
- [5] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *IEEE Visualization 2004*, pp. 497–504, 2004. doi: 10.1109/VISUAL.2004.96 3
- [6] G. Chao, Y. Luo, and W. Ding. Recent advances in supervised dimension reduction: A survey. *Machine Learning and Knowledge Extraction*, 1(1):341–358, 2019. doi: 10.3390/make1010020 2
- [7] F. Chazal and B. Michel. An introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence*, 4, 2021. doi: 10.3389/frai.2021.667963 2
- [8] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Taffjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. 6
- [9] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *Proceedings of the Twenty-First Annual Symposium on Computational Geometry*, SCG '05, 9 pages, p. 263–271. Association for Computing Machinery, New York, NY, USA, 2005. doi: 10.1145/1064092.1064133 3
- [10] J. P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015. 1, 2
- [11] R. R. Curtin, M. Edel, O. Shrit, S. Agrawal, S. Basak, J. J. Balamuta, R. Birmingham, K. Dutt, D. Eddebuettel, R. Garg, S. Jaiswal, A. Kaushik, S. Kim, A. Mukherjee, N. G. Sai, N. Sharma, Y. S. Parihar, R. Swain, and C. Sanderson. mlpack 4: a fast, header-only c++ machine learning library. *Journal of Open Source Software*, 8(82):5026, 2023. doi: 10.21105/joss.05026 5
- [12] B. Doppalapudi, B. Wang, and P. Rosen. Untangling force-directed layouts using persistent homology. In *2022 Topological Data Analysis and Visualization (TopoInVis)*, pp. 81–91, 2022. doi: 10.1109/TopoInVis57755.2022.00015 1, 2
- [13] H. Doraiswamy, J. Tierny, P. J. S. Silva, L. G. Nonato, and C. Silva. Topomap: A 0-dimensional homology preserving projection of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):561–571, 2021. doi: 10.1109/TVCG.2020.3030441 1, 2, 3, 5, 6, 9
- [14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 8
- [15] D. Dua and C. Graff. UCI machine learning repository, 2017. 5
- [16] Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & computational geometry*, 28:511–533, 2002. doi: 10.1007/s00454-002-2885-2 2
- [17] H. Edelsbrunner and J. Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010. 2
- [18] D. Engel, L. Hüttenberger, and B. Hamann. A Survey of Dimension Reduction Methods for High-dimensional Data Analysis and Visualization. In C. Garth, A. Middel, and H. Hagen, eds., *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG 1131 Workshop 2011*, vol. 27 of *Open Access Series in Informatics (OASIs)*, pp. 135–149. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012. doi: 10.4230/OASIs.VLUDS.2011.135 2
- [19] M. Espadoto, R. M. Martins, A. Kerren, N. S. T. Hirata, and A. C. Telea. Toward a quantitative survey of dimension reduction techniques. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2153–2173, 2021. doi: 10.1109/TVCG.2019.2944182 1, 2
- [20] K. R. Gabriel and R. R. Sokal. A New Statistical Approach to Geographic Variation Analysis. *Systematic Biology*, 18(3):259–278, 09 1969. doi: 10.2307/2412323 2
- [21] S. Gerber, P.-T. Bremer, V. Pascucci, and R. Whitaker. Visual exploration of high dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1271–1280, 2010. doi: 10.1109/TVCG.2010.213 2
- [22] S. Gerber, O. Rübél, P.-T. Bremer, V. Pascucci, and R. T. Whitaker. Morse–smale regression. *Journal of Computational and Graphical Statistics*, 22(1):193–214, 2013. PMID: 23687424. doi: 10.1080/10618600.2012.657132 2
- [23] W. Harvey and Y. Wang. Topological landscape ensembles for visualization of scalar-valued functions. *Computer Graphics Forum*, 29(3):993–1002, 2010. doi: 10.1111/j.1467-8659.2009.01706.x 1, 2
- [24] G. E. Hinton and S. Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, eds., *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, 2002. 2
- [25] S. Jayaram Subramanya, F. Devvrit, H. V. Simhadri, R. Krishnawamy, and R. Kadekodi. Diskann: Fast accurate billion-point nearest neighbor search on a single node. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. 5
- [26] O. C. Jenkins and M. J. Mataric. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, p. 56. Association for Computing Machinery, New York, NY, USA, 2004. doi: 10.1145/1015330.1015357 2
- [27] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics yolov8, 2023. 8
- [28] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: Re-rank with source coding. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 861–864, 2011. doi: 10.1109/ICASSP.2011.5946540 5
- [29] J. A. Lee and M. Verleysen. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing*, 67:29–53, 2005. Geometrical Methods in Neural Networks and Learning. doi: 10.1016/j.neucom.2004.11.042 2
- [30] W. B. March, P. Ram, and A. G. Gray. Fast euclidean minimum spanning tree: algorithm, analysis, and applications. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, 10 pages, p. 603–612. Association for Computing Machinery, New York, NY, USA, 2010. doi: 10.1145/1835804.1835882 5
- [31] L. McInnes, J. Healy, N. Saul, and L. Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861 2
- [32] B. J. Nelson and Y. Luo. Topology-preserving dimensionality reduction via interleaving optimization. *CoRR*, abs/2201.13012, 2022. 2
- [33] L. G. Nonato and M. Aupetit. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2650–2673, 2019. doi: 10.1109/TVCG.2018.2846735 1, 2
- [34] P. Oesterling, C. Heine, H. Janicke, G. Scheuermann, and G. Heyer. Visualization of high-dimensional point clouds using their density distribution's topology. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1547–1559, 2011. doi: 10.1109/TVCG.2011.27 2
- [35] P. Oesterling, C. Heine, H. Jänicke, and G. Scheuermann. Visual analysis of high dimensional point clouds using topological landscapes. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 113–120, 2010. doi: 10.1109/PACIFICVIS.2010.5429601 2
- [36] P. Oesterling, C. Heine, G. H. Weber, and G. Scheuermann. Visualizing nd point clouds as topological landscape profiles to guide local data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):514–526, 2013. doi: 10.1109/TVCG.2012.120 2
- [37] P. Oesterling, G. Scheuermann, S. Teresiak, G. Heyer, S. Koch, T. Ertl, and G. H. Weber. Two-stage framework for a topology-based projection and visualization of classified document collections. In *2010 IEEE Symposium on Visual Analytics Science and Technology*, pp. 91–98, 2010. doi: 10.1109/VAST.2010.5652940 2
- [38] Y. Piadyk, J. Rulff, E. Brewer, M. Hosseini, K. Ozbay, M. Sankaradas, S. Chakradhar, and C. Silva. Streetaware: A high-resolution synchronized multimodal urban scene dataset. *Sensors*, 23(7), 2023. doi: 10.3390/s23073710 8



- [39] P. Ray, S. S. Reddy, and T. Banerjee. Various dimension reduction techniques for high dimensional data analysis: a review. *Artificial Intelligence Review*, 54(5):3473–3515, June 2021. doi: [10.1007/s10462-020-09928-0](https://doi.org/10.1007/s10462-020-09928-0) 2
- [40] B. Rieck and H. Leitte. Persistent homology for the evaluation of dimensionality reduction schemes. *Computer Graphics Forum*, 34(3):431–440, 2015. doi: [10.1111/cgf.12655](https://doi.org/10.1111/cgf.12655) 2
- [41] B. Rieck and H. Leitte. Agreement analysis of quality measures for dimensionality reduction. In H. Carr, C. Garth, and T. Weinkauff, eds., *Topological Methods in Data Analysis and Visualization IV*, pp. 103–117. Springer International Publishing, Cham, 2017. 2
- [42] D. Sacha, L. Zhang, M. Sedlmair, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):241–250, 2017. doi: [10.1109/TVCG.2016.2598495](https://doi.org/10.1109/TVCG.2016.2598495) 2
- [43] M. Sedlmair, T. Munzner, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2634–2643, 2013. doi: [10.1109/TVCG.2013.153](https://doi.org/10.1109/TVCG.2013.153) 2
- [44] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, 8 pages, jan 1992. doi: [10.1145/102377.115768](https://doi.org/10.1145/102377.115768) 2, 4
- [45] V. Silva and J. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, and K. Obermayer, eds., *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, 2002. 2
- [46] J.-T. Sohns, M. Schmitt, F. Jirasek, H. Hasse, and H. Leitte. Attribute-based explanation of non-linear embeddings of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):540–550, 2022. doi: [10.1109/TVCG.2021.3114870](https://doi.org/10.1109/TVCG.2021.3114870) 2
- [47] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. doi: [10.1126/science.290.5500.2319](https://doi.org/10.1126/science.290.5500.2319) 2
- [48] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. Canton-Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hossain, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. 6
- [49] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 2
- [50] L. van der Maaten, E. Postma, and H. Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research - JMLR*, 10, 01 2007. 1
- [51] R. Vandaele, B. Kang, J. Lijffijt, T. D. Bie, and Y. Saeyns. Topologically regularized data embeddings. In *International Conference on Learning Representations*, 2022. 2
- [52] S. Velliangiri, S. Alagumuthukrishnan, and S. I. Thankumar josph. A review of dimensionality reduction techniques for efficient computation. *Procedia Computer Science*, 165:104–111, 2019. 2nd International Conference on Recent Trends in Advanced Computing ICRTAC -DISRUP -TIV INNOVATION , 2019 November 11-12, 2019. doi: [10.1016/j.procs.2020.01.079](https://doi.org/10.1016/j.procs.2020.01.079) 2
- [53] V. M. Vu, A. Bibal, and B. Fréney. Integrating constraints into dimensionality reduction for visualization: A survey. *IEEE Transactions on Artificial Intelligence*, 3(6):944–962, 2022. doi: [10.1109/TAI.2022.3204734](https://doi.org/10.1109/TAI.2022.3204734) 2
- [54] A. Wagner, E. Solomon, and P. Bendich. Improving metric dimensionality reduction with distributed topology. *CoRR*, abs/2106.07613, 2021. 2
- [55] G. Weber, P.-T. Bremer, and V. Pascucci. Topological landscapes: A terrain metaphor for scientific data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1416–1423, 2007. doi: [10.1109/TVCG.2007.70601](https://doi.org/10.1109/TVCG.2007.70601) 1, 2
- [56] J. Xia, Y. Zhang, J. Song, Y. Chen, Y. Wang, and S. Liu. Revisiting dimensionality reduction techniques for visual cluster analysis: An empirical study. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):529–539, 2022. doi: [10.1109/TVCG.2021.3114694](https://doi.org/10.1109/TVCG.2021.3114694) 2
- [57] L. Yan, Y. Zhao, P. Rosen, C. Scheidegger, and B. Wang. Homology-preserving dimensionality reduction via manifold landmarking and tearing. *CoRR*, abs/1806.08460, 2018. 2