Joint UMAP for Visualization of Time-Dependent Data

Yun Zou, Benchao Li, and Ruisheng Ran(✉)

Chongqing Normal University, Chongqing 401331, China
rshran@cqnu.edu.cn

Abstract. With the development of data collection and storage technologies, high-dimensional time series data has be-come increasingly prevalent in various scenarios. Time series data visualization, which intuitively depicts the dynamic changes in such data through graphical representations, has become an important research direction in data science and information visualization. Existing static data visualization methods have limitations, such as inconsistent projection results, while current time series visualization techniques, including Joint t-SNE, suffer from high computational complexity. To address these issues, we propose Joint UMAP. We first introduce a Graphlet Frequency Distribution (GFD) to capture similarities between data points in adjacent time frames and then integrate a new vector constraint into UMAP's loss function to preserve inherited structures in the high-dimensional data space, ensuring consistency in subsequent projections. Using both synthetic and real-world datasets, we demonstrate that Joint UMAP outperforms existing techniques in terms of Local Coherence Error (LCE) and Cross-Entropy (CE). By eliminating the global dependency of projection results during optimization, Joint UMAP is particularly advantageous in dynamic scenarios where datasets change over time.

Keywords: Dimensionality Reduction · High-dimensional Data · Visualization · Time-oriented Data · UMAP

1 Introduction

In the fields of data science and machine learning, high-dimensional data visualization [18] is a crucial task. Some traditional data visualization techniques [8], such as pie charts, graphs, maps, and heat maps, are suitable for both conventional data and large datasets to some extent. However, as the dimensions of data increase, more specialized visualization tools are required for advanced visualization of massive datasets. High-dimensional data visualization techniques map high-dimensional data to lower-dimensional spaces, simplifying the complexity of the data. This allows us to intuitively observe patterns, relationships, and anomalies within the data in a low-dimensional space, thereby helping us to understand the structure and patterns of the data more clearly. Consequently,

P. Didyk and J. Hou (Eds.): CVM 2025, LNCS 15665, pp. 266–288, 2025.
https://doi.org/10.1007/978-981-96-5815-2_14

researchers have proposed various dimensionality reduction visualization methods, such as PCA (Principal Component Analysis) [15], LLE (Locally Linear Embedding) [26], and MDS (Multidimensional Scaling) [5], t-SNE(t-Stochastic Neighbor Embedding) [19], UMAP(Uniform Manifold Approximation and Projection) [21], etc. These algorithms have demonstrated excellent performance in visualizing high-dimensional data, revealing the inherent structure and patterns within the data.

With the advancement of data collection and storage technologies, various time series data have emerged in numerous fields, such as financial market data [29], sensor data [16], medical monitoring data [28]. These datasets are vast in quantity and update rapidly. Time series data visualization has become increasingly important and has evolved into a significant research direction in data science and information visualization. The goal of time series data visualization is to use time as a dimension to showcase the trends and patterns in the data, often through intuitive graphical representations that depict the dynamic changes in time series data, such as dynamic graph drawing [17,34]. In recent years, researchers have proposed many effective visualization techniques to enable users to quickly understand the dynamic characteristics of time-oriented data. According to [1], current time-oriented data visualization techniques are diverse and can be broadly categorized into abstract or spatial, univariate or multivariate, linear or cyclical, instantaneous or interval-based, static or dynamic, and 2D or 3D visualizations. Our research primarily focuses on abstract, multivariate, and instantaneous time-oriented visualizations.

UMAP has been widely used in recent years for dimensionality reduction and visualization of high-dimensional data, with significant advantages in computational efficiency, enabling it to process larger datasets more quickly. Compared to other dimensionality reduction methods (such as PCA and t-SNE), UMAP is better at preserving both global and local structures of data. This makes UMAP particularly suitable for visualizing large datasets, and it has gained wide recognition in various application areas, such as image processing and genomics. Especially in large-scale data processing, UMAP's time complexity is significantly lower than that of methods like t-SNE, allowing it to generate high-quality projections in a shorter time.

However, existing UMAP methods still face challenges when dealing with dynamic datasets, such as inconsistencies in projection results and inaccuracies in capturing temporal dynamics. This is because UMAP processes each time frame of dynamic data independently during its stochastic optimization process, leading to varying projection results across different time frames, and thus failing to accurately reflect the temporal evolution of the data. To address this issue, we propose a new projection technique called Joint UMAP (JUMAP). The main idea of JUMAP is to selectively preserve the topological structure between projections across different time frames based on the topological similarity of the projections, ensuring consistency in the results. Specifically, JUMAP first uses Graphlet Frequency Distribution (GFD) [24] to capture the frequency distribution of various small subgraphs (graphlets) in the graph, representing the

local structural characteristics around each point. Then, a vector constraint is introduced during UMAP's optimization process to ensure coherence in the projection results across different time frames. This improvement allows JUMAP to effectively capture both the global and local structures of the data, while also accurately reflecting its temporal dynamics. In various evaluations, JUMAP has shown excellent performance in metrics such as Local Coherence Error (LCE) and Cross-Entropy (CE), demonstrating higher stability and projection consistency, with significant potential for applications in dynamic data visualization and analysis.

This paper is organized into six main sections: Sect. 1 serves as the Introduction, outlining the significance of the research and key concepts. Section 2 provides an overview of the Related Work, discussing existing literature on dimensionality reduction techniques, particularly focusing on static and dynamic projection methods. Section 3 delves into the Research Background, introducing the UMAP algorithm and the concept of Graphlet Frequency Distribution (GFD). Section 4 details the Joint UMAP Algorithm, emphasizing its design and unique contributions. Section 5 presents the Experimental Design and Results Analysis, where various algorithms are evaluated on different datasets, detailing the methods used and the results obtained. Finally, Sect. 6 concludes the paper by summarizing the research findings and suggesting future work directions.

2 Related Works

Projection [23] is a data visualization technique that maps high-dimensional data to a low-dimensional space and is widely applied in fields such as data mining [3], machine learning [12], computer vision [30], and bioinformatics [4]. Static projection methods are primarily used to display a one-time low-dimensional representation of data, while dynamic projection methods are employed to continuously capture and visualize the process of data changing over time. As the scale and complexity of datasets grow, dynamic projection methods play an increasingly important role in the analysis and visualization of time-series data. This section first discusses common static dimensionality reduction methods, followed by a detailed exploration of dynamic projection techniques and their development.

2.1 Techniques for Static Dimensionality Reduction

Static projection techniques have matured and are widely used for processing static data. Common methods include linear techniques such as PCA [15] and nonlinear methods like t-SNE [19] and UMAP [21]. PCA emphasizes the global structure of the data and is suitable for linear data, but it struggles with complex nonlinear structures. t-SNE highlights the clustering characteristics of data by preserving local structures, but its distortion of global structures can lead to misleading results. UMAP combines topology and geometry to balance the

preservation of local and global structures while improving computational efficiency. Although these methods excel in visualizing static data, they cannot handle the dynamic changes of data over time. Several reviews have discussed these techniques in detail [6,27,35], covering comparisons and analyses from linear to nonlinear methods.

2.2 Techniques for Dynamic Dimensionality Reduction

Dynamic projection techniques focus on time-series data, aiming to capture the evolution trajectory of data over time and provide consistent, temporally continuous low-dimensional projections. Unlike static projections, dynamic projection methods not only need to preserve the structural characteristics of high-dimensional data in low-dimensional space but also must ensure that the projection results remain consistent across consecutive time frames, avoiding distortion. Given that the dynamic changes in data can be highly complex, these techniques hold significant importance in the visualization of time-series data but also face numerous technical challenges.

Dominik Jäckle et al. proposed Temporal MDS (Temporal Multidimensional Scaling) [14], one of the earlier dynamic projection methods. It extends the classic MDS approach to capture the temporal evolution trajectory of high-dimensional data, particularly in the absence of prior knowledge, meaning that there is no need to know the temporal information of the data in advance. The method reduces multivariate data in each time frame to one-dimensional slices and aligns these slices along the time axis using a flipping heuristic method. However, the method's accuracy in alignment is limited when facing rapid changes, and without prior knowledge, it struggles to adapt to fast-evolving data structures, making it less effective at capturing the dynamic nature of the data.

Paulo E. Rauber et al. extended t-SNE by developing Dynamic t-SNE [25] to handle time-series data. This method maintains projection consistency between time frames during the projection process through a smoothing mechanism and the addition of loss terms, allowing for a more intuitive tracking of data changes. However, Dynamic t-SNE often proves too rigid when faced with significant changes in the data, leading to projection distortion and an inability to adapt flexibly to rapid changes in data structure, making it less effective at capturing the dynamic evolution of data in such cases.

Yinqiao Wang et al. further improved upon Dynamic t-SNE by proposing Joint t-SNE [31], which reinforces projection consistency and accuracy to some extent by introducing constraint conditions. Compared to Dynamic t-SNE, Joint t-SNE achieves a better balance between visual continuity and distortion reduction; however, it still faces challenges with high computational complexity when dealing with large-scale streaming data, particularly in real-time processing, where it encounters significant computational bottlenecks.

It is also worth mentioning that Fujiwara et al. proposed a PCA-based method for handling streaming data [7]. This type of method dynamically adjusts low-dimensional projections to accommodate the characteristics of streaming data, making it suitable for continuously updating data flows. However, due to

the inability to predict future data changes, this method may face certain limitations in optimizing low-dimensional representations, as it cannot guarantee projection consistency or preserve global structures effectively in highly dynamic scenarios.

Other research has also explored the use of dimensionality reduction techniques to visualize time-series data. For instance, Hu et al. employed Self-Organiz-ing Maps [13] to create 2D trajectory plots that capture the dynamic changes in human motion data. Similarly, Mao et al. applied PCA to analyze the evolution of text features [20], demonstrating the dynamic changes in text sequences.

However, these methods often struggle to balance projection consistency, global structure preservation, and computational efficiency when dealing with dynamic data. While each method has its advantages, they still face significant limitations when dealing with time-dependent high-dimensional data. On the one hand, t-SNE-based methods (including Dynamic t-SNE and Joint t-SNE) are highly sensitive to initial conditions and suffer from high computational complexity, making it difficult to effectively preserve global structures. On the other hand, although UMAP demonstrates high computational efficiency and the ability to preserve both global and local structures in static data, it often produces inconsistent projections when applied to dynamic data and struggles to capture the dynamic changes over time. To address these issues, this paper introduces the Joint UMAP method.

3 Research Background

3.1 UMAP

UMAP, as a nonlinear dimensionality reduction algorithm based on manifold learning, is often used for projection. UMAP is constructed based on the theoretical framework of Riemannian geometry and algebraic topology. Its main idea is to build a fuzzy topological representation between high-dimensional data and low-dimensional embeddings, and to optimize the low-dimensional embeddings so that their fuzzy topological representation is similar to that of the high-dimensional data, thereby achieving dimensionality reduction results, as shown in the Fig. 1. The overall process of UMAP can be classified as the following two stages:

Graph Construction. Given a high-dimensional dataset $X = \{x_1, ..., x_n\}$, we calculate the distance or dissimilarity between any two data points in the high-dimensional space, denoted as $d : X \times X \to \mathbb{R}_{\geq 0}$. To model the local structure around each data point, we use the conditional probability $p_{i|j}$ to measure the similarity between x_i and x_{i_j}, which represents the probability of data point x_i choosing x_{i_j} as its neighbor. The expression for this conditional probability is:

$$p_{i|j} = \exp\left(\frac{-\max\left(0, d\left(x_i, x_{i_j}\right) - \rho_i\right)}{\sigma_i}\right). \tag{1}$$

Here, ρ_i represents the distance from data point x_i to its nearest neighbor, ensuring that at least one edge weight of 1 connects to x_i. The parameter σ_i reflects the distance distribution among the nearest neighbors of x_i. Specifically, σ_i is defined as the maximum distance between x_i and its nearest neighbors. It describes the range of relative distances among x_i's nearest neighbors, capturing the spatial distribution of the local structure around x_i. ρ_i and σ_i are computed using function 2 and 3, respectively.

$$\rho_i = \min\left\{d\left(x_i, x_{i_j}\right) | 1 \leq j \leq k, d\left(x_i, x_{i_j}\right) > 0\right\}. \tag{2}$$

$$\sum_{j=1}^{k} \exp\left(\frac{-\max\left(0, d\left(x_i, x_{i_j}\right) - \rho_i\right)}{\sigma_i}\right) = \log_2(k). \tag{3}$$

Since the conditional probability $p_{i|j}$ is not symmetric, it is necessary to calculate the symmetric joint probability p_{ij}:

$$p_{ij} = p_{i|j} + p_{j|i} - p_{i|j}p_{j|i}. \tag{4}$$

Finally, let $Y = \{y_1, \ldots, y_n\}$ denote the low-dimensional data points, and the similarity between data points y_i and y_j in the low-dimensional space is given by the following equation:

$$q_{ij} = \left(1 + a\left(\|y_i - y_j\|_2\right)^{2b}\right)^{-1}. \tag{5}$$

Here, q_{ij} represents the similarity between y_i and y_j in the low-dimensional space, and the parameters a and b control the expansion rate and curvature of the similarity function. By default, UMAP sets $a \approx 1.93$ and $b \approx 0.79$, but these parameters can be adjusted to modify how distances in the high-dimensional space are mapped to similarities in the low-dimensional space. By adjusting a and b, UMAP can better balance the local and global structures in the low-dimensional projection.

Graph Embedding. The precomputed fuzzy topological representations are optimized so that the high-dimensional and low-dimensional representations are as similar as possible. UMAP uses stochastic gradient descent to optimize the low-dimensional embedding, finding the appropriate values for y_i and y_j (i.e., the positions of data points in the low-dimensional space) to minimize the difference between the high-dimensional similarity p_{ij} and the low-dimensional similarity q_{ij}. To measure this difference, UMAP uses cross-entropy to evaluate the similarity between the two representations. The UMAP loss function is defined as follows:

$$CE = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \log\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right)\right). \tag{6}$$

UMAP defines a loss function to describe this optimization process, and by adjusting the positions of the data points to minimize the loss function, it generates the projection results in the low-dimensional space. The attraction and repulsion forces are the core components of UMAP, responsible for maintaining the relationships between data points in the low-dimensional space. Attraction refers to the pull between neighboring points during the optimization process. The attraction in the UMAP algorithm is defined as follows:

$$\frac{-2ab\left\|y_i - y_j\right\|_2^{2(b-1)}}{1+\left\|y_i - y_j\right\|_2^2} p_{ij}\left(y_i - y_j\right) \tag{7}$$

where y_i and y_j represent the corresponding data points in the low-dimensional space.

Repulsion refers to the push that separates non-neighboring points in the low-dimensional space during the optimization process. The repulsion in the UMAP algorithm is defined as follows:

$$\frac{2b}{\left(\varepsilon + \left\|y_i - y_j\right\|_2^2\right)\left(1+\alpha\left\|y_i - y_j\right\|_2^{2b}\right)}\left(1-p_{ij}\right)\left(y_i - y_j\right) \tag{8}$$

where ε is a small constant added to prevent division by zero, with a default value of 0.001.

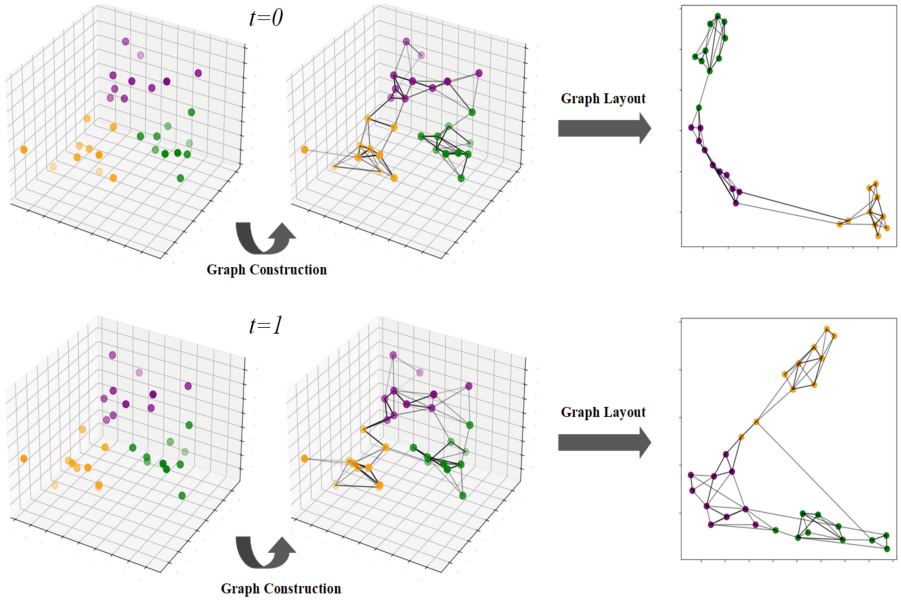

Fig. 1. Temporal Graph Construction and Layout Representation at t = 0 and t = 1

3.2 Graphlet

Graphlets refer to non-isomorphic substructures in a graph with k nodes, which are the fundamental building blocks of a graph. Any large graph can be decomposed into these subgraph structures, making them useful for representing and describing the local structure of the graph. In Fig. 2, all substructures with $k \in \{3, 4, 5\}$ nodes are shown, indicating that they cannot be transformed into another substructure through simple node rearrangement. The frequency of graphlets, meaning the occurrence frequency of each distinct graphlet (i.e., the number of times various graphlets appear in a graph), can serve as a fingerprint to differentiate between graphs. This fingerprint is defined as the Graphlet Frequency Distribution (GFD). GFD [24], also known as Graphlet concentration or Graphlet statistics, was proposed by Pržulj et al. It has become a widely used tool for graph analysis and has been applied in various domains such as image classification [36], biological network comparison [11], and disease gene identification [22], among many other fields.

The main idea of JUMAP is to selectively preserve the topological structure between projections of high-dimensional data into low-dimensional space, specifically focusing on the topological similarity across different time frames. This edge similarity is captured using the Graphlet Frequency Distribution (GFD), which measures the structural similarities and differences between adjacent time frames. By comparing the GFDs of various graphs representing different time frames, we can gain insights into the dynamic changes in the data, allowing GFD to serve as a distinctive feature akin to a fingerprint for identifying and differentiating between various graphs, similar to how human fingerprints are used to identify individuals.

Although GFD is an effective method, counting graphlets is a computationally expensive task, especially for large graphs. This paper adopts the idea of combining GFD with GUISE [2], a graph-oriented uniform sampling method. GUISE uses the Markov Chain Monte Carlo (MCMC) method to approximate the GFD of large graphs, capturing the local topological structure around high-dimensional data points. This approach significantly accelerates the computation, greatly reducing the associated costs.

Graphlet kernel is an improved kernel method for measuring graph similarity. A kernel, also known as a generalized inner product, is a function that measures the similarity between two vectors $x, y \in \mathbb{R}^m$ by computing their corresponding inner product in a higher-dimensional feature space $\phi(x), \phi(y) \in \mathbb{R}^n$. The mapping function ϕ transforms x and y into the feature space $\mathbb{R}^n$.

The similarity between two graphs G_0 and G_1 using the graphlet kernel k_g is defined as the inner product of their feature vectors:

$$k_g(G_0, G_1) = \langle f_{G_0}, f_{G_1} \rangle \tag{9}$$

where the feature vectors f are based on the normalized GFD. Common kernel functions used in this context include the Gaussian kernel, Laplacian kernel, and cosine similarity.

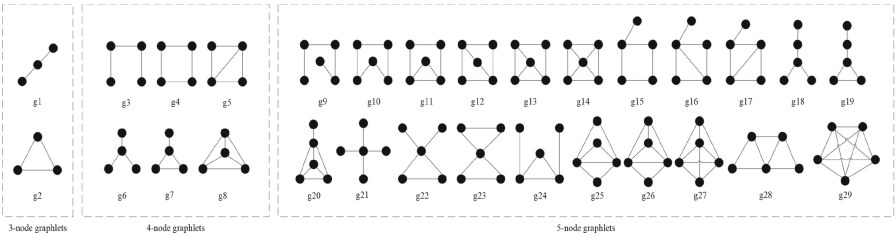

Fig. 2. All graphlets of 3, 4, and 5 nodes

4 Joint UMAP

The objective of JUMAP is to generate low-dimensional embeddings that can consistently visually track the evolution of data over time, as shown in the Fig. 3. In most time series datasets, data points exhibit both continuity and variability within each time frame. Therefore, in order to achieve the JUMAP's goal, it is crucial not only to identify and analyze the dynamic structural changes between different time frames in the data space, but also to maintain the inherent structures of the data as it exists in different time frames within the projections. This ensures that the embeddings can effectively distinguish between time-varying structures and consistent structures. Additionally, it is important to ensure the accuracy of the projections, meaning that the low-dimensional space should accurately reflect the structures and relationships present in the high-dimensional space. By addressing these sub-goals, JUMAP creates low-dimensional representations that capture the temporal evolution of data while preserving significant local inherited structures, thereby facilitating visual exploration and analysis.

Traditional dimensionality reduction algorithms, such as PCA, t-SNE, and UMAP, are primarily designed to project individual time frames or datasets without temporal dependencies, focusing on the accuracy of each standalone projection. However, when projecting multiple time frames simultaneously, these algorithms struggle to generate aligned layouts, resulting in inconsistent projections of the same classes across different time frames, as illustrated in Fig. 1. To address this issue, JUMAP employs a sliding window mechanism with a window length of 2 to handle adjacent time-frame data. This mechanism generates a coherent projection Y_1 for the new dataset X_1, combining it with the existing projection Y_0. This combination preserves the topological structure of the items with high similarity between Y_0 and Y_1 in the data space. Figure 3 illustrates this process, clearly reflecting the relationships between the two time frames.

To preserve the topological structure of items with high similarity and detect local changes, JUMAP first specifies a parameter k, for the number of neighbors to be considered. For each dataset, an undirected k-Nearest Neighbors(kNN) graph $G(V, E)$ is constructed in high-dimensional space, where $V = \{v_1, v_2, ..., v_n\}$ and $E = \{e_1, e_2, ..., e_n\}$ represent the set of vertices and edges in the graph, respectively. To associate datasets with different dimensions, the original datasets are not used directly. Instead, the relationship is calculated

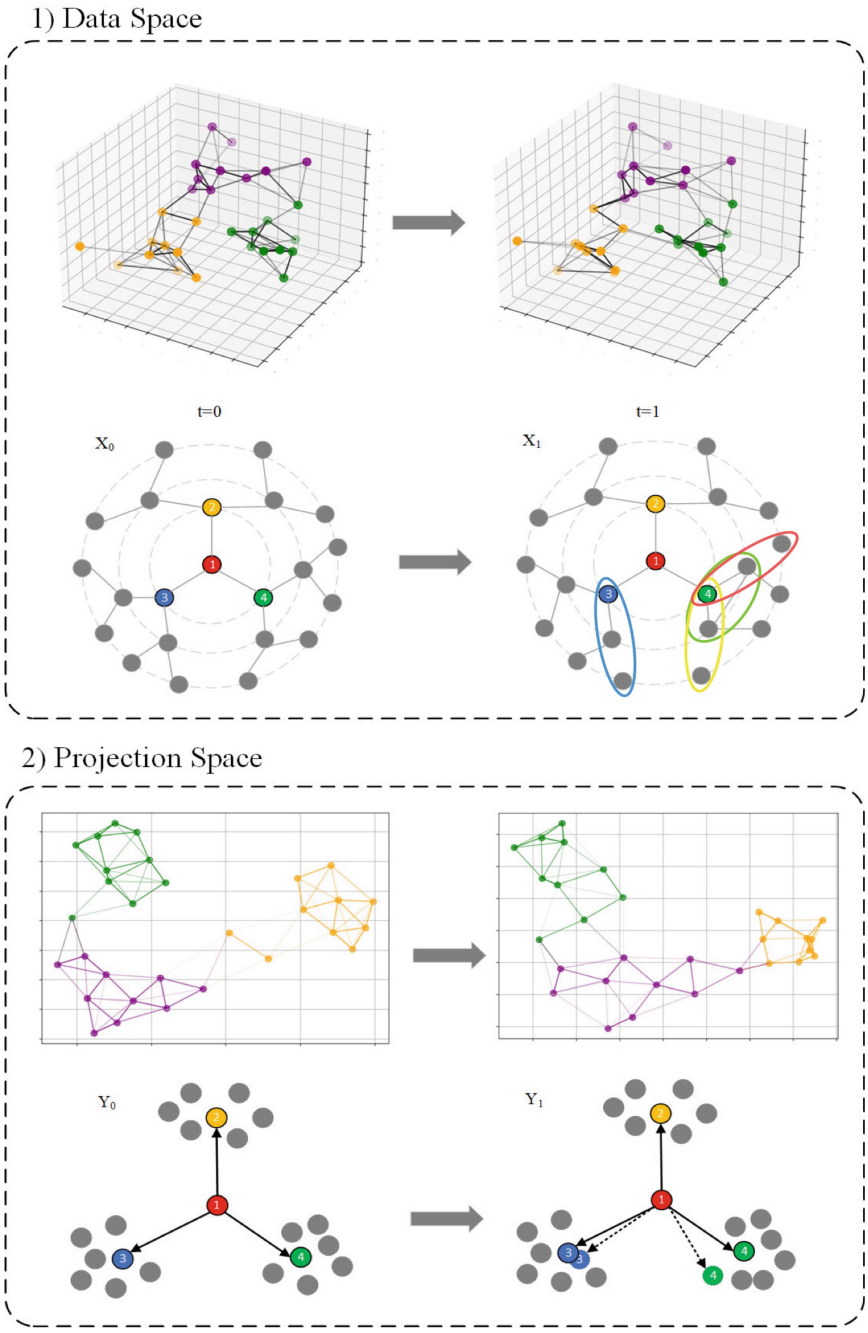

Fig. 3. Technical Illustration of Joint UMAP

through two graphs. For the two datasets X_0 and X_1, kNN undirected graphs G_0 and G_1, are constructed using a distance metric. An edge e_{ij} exists if and only if $v_i \in kNN\left(v_j\right)$ or $\nu_j \in kNN\left(\nu_i\right)$.

To capture the structural similarity between two graphs, we introduce two types of similarity: point similarity and edge similarity. **Point similarity** S_{p_i} refers to the cosine similarity $\langle f_{v_i^0}, f_{v_i^1} \rangle$ calculated between the normalized feature vectors $f_{v_i^0}$ and $f_{v_i^1}$, which are derived from the Graphlet Frequency Distribution (GFD) of nodes v_i^0 and v_i^1, respectively. This cosine similarity is then multiplied by the proportion of their common neighbors to assess their overall similarity. We utilize the graphlet kernel to measure the similarity value for each node, capturing changes in the topological structure of a node and its neighbors from one time frame to the next.

In this context, Fig. 3 illustrates the comparison between the data and projection spaces, clearly reflecting how structural similarities between different time frames are preserved in the projection. Specifically, some changes occurred between X_0 and X_1, with several points breaking their neighborhood relationships with the original cluster. Joint UMAP detects these changes by measuring the similarity of local structures and calculating edge similarities ($Se_{12} > Se_{13} > Se_{14}$). It uses edge similarities as weights for the corresponding vector constraints, generating projection Y_1 while preserving the relative positions of points in Y_0.

Ultimately, this method, which combines cosine similarity and the proportion of common neighbors, better reflects the dynamic changes between nodes. The expression for point similarity is as follows:

$$S_{p_i} = \frac{\left|kNN\left(G_0, \nu_i^0\right) \cap kNN\left(G_1, \nu_i^1\right)\right|}{k} \cdot \left\langle f\nu_i^0, f\nu_i^1 \right\rangle. \tag{10}$$

The edge similarity between $e_{0_{ij}}$ and $e_{1_{ij}}$ is defined as the product of the point similarities between their two endpoints. Specifically, this refers to the similarity of the common edges between the k-nearest neighbors (kNN) undirected graphs G_0 and G_1, which are constructed from the datasets X_0 and X_1. This means that for an edge $e_{0_{ij}}$ connecting nodes v_i^0 and v_j^0 in graph G_0, and an edge $e_{1_{ij}}$ connecting nodes v_i^1 and v_j^1 in graph G_1, the edge similarity is given by the product of the point similarities of the node pairs (v_i^0, v_i^1) and (v_j^0, v_j^1). This allows us to measure the overall similarity of corresponding edges in two kNN graphs. The vector constraint, introduced to preserve temporal consistency between time frames, is defined as:

$$S_{e_{ij}} = S_{p_i} \cdot S_{p_j}. \tag{11}$$

where S_{p_i} and S_{p_j} are the similarity scores for each data point. These similarity scores represent the degree of relationship between two data points in the original high-dimensional space and are crucial for maintaining the importance of local structures, especially when projections of data points change across time frames.

To effectively integrate the dimensionality reduction results from different time frames into subsequent time frames, and to ensure that the projections in

the later frames remain roughly consistent with those from earlier frames, we introduce a vector constraint. Inspired by edge vector constraint methods [32,33], which have shown better performance in preserving local structures compared to other methods, we incorporate this constraint into the UMAP loss function. This vector constraint helps maintain structural consistency between consecutive time frames by preserving the relative positions between pairs of points. Specifically, the vector constraint guides the optimization process by weighting each edge with its similarity score, ensuring that the projections across different time frames remain consistent while preserving the local topological structure. The loss function that includes this constraint aims to minimize the relative position difference between the initial time frame (y_i^0 and y_j^0) and the subsequent time frames (y_i^1 and y_j^1), while also considering the smoothness of the projection positions across time frames. The vector constraint can be expressed as:

$$\arg\min_{Y_1} C_1 = \frac{\gamma}{M} \sum_{i \neq j} S_{e_{ij}} \cdot \left\| \left(y_i^0 - y_j^0\right) - \left(y_i^1 - y_j^1\right) \right\|^2 + \lambda \sum_i \|y_i^1 - y_i^0\|^2. \quad (12)$$

where λ is the smoothing penalty coefficient that controls the range of node movement across time frames, ensuring that projections transition smoothly between adjacent time frames. M represents the number of common edges between G_0 and G_1, and γ is the user-defined weight for the vector constraint, with a default value of 0.1.

The gradient in Eq. 13 represents the vector constraint term's gradient with respect to the projections in the subsequent time frame. This gradient will be used in the optimization process to update the positions of data points. The gradient is given by:

$$\frac{\partial C_1}{\partial y_i^1} = -\frac{2\gamma}{M} \sum_{i \neq j} S_{e_{ij}} \left((y_i^0 - y_j^0) - (y_i^1 - y_j^1) \right) + 2\lambda (y_i^1 - y_i^0) \quad (13)$$

This gradient update ensures that the optimization process accounts for the structural consistency between the common edges in both graphs. It drives the optimization process by minimizing the difference between the initial and subsequent time frame projections, while also smoothing the transition between frames using the smoothing penalty coefficient λ. This helps maintain structural consistency and stability between time frames.

5 Experiments

To evaluate JUMAP, we compare it to the original UMAP (UMAP), Equal-initialization UMAP (EUMAP), the original t-SNE (t-SNE), Equal-initialization t-SNE (ETSNE) and Joint t-SNE (JTSNE) techniques that can be used to generate projections and visualizations are compared quantitatively and qualitatively. Among these, UMAP and t-SNE served as the original algorithms with random initial layouts to project each dataset. In contrast, EUMAP and ETSNE projected each dataset D_i using UMAP and t-SNE, respectively, but both used the

same initial layout which was also randomly generated. We validate and compare these algorithms using synthetic datasets for which the true structure is known and the fidelity and consistency between projections can be accurately assessed through operations such as transformations.

5.1 Datasets

The four datasets used in the experiments are: (1) the 5-Gaussian dataset, (2) the 8-Gaussian dataset, (3) the 10-Gaussian dataset [25], and (4) the MNIST dataset. These four datasets are described below.

The 5-Gaussian dataset consists of 1000 data points in a 100-dimensional space, sampled from 5 isotropic Gaussian distributions, with each distribution containing 200 data points. The variance of the Gaussian distributions is set to 0.1, indicating equal variance across all dimensions. The centers of these distributions are randomly selected between the standard basis vectors of $\mathbb{R}^{100}$ (i.e., vectors similar to $(1, 0, 0, ..., 0)$ and $(0, 1, 0, ..., 0)$). This means that the centroids are located at certain vertices or along some edges of the unit hypercube in $\mathbb{R}^{100}$. To simulate temporal dependence, a progressive contraction operation is applied, where each data point is moved towards the center of its respective Gaussian distribution by a distance equal to 10% of the remaining distance to the center. This operation is repeated three times, generating a new time frame with each iteration, resulting in a final dataset comprising four time frames. The 8-Gaussian dataset consists of 1600 data points in a 100-dimensional space, similarly sampled from 8 isotropic Gaussian distributions. The 10-Gaussian dataset comprises 2000 data points in a 100-dimensional space, sampled from 10 isotropic Gaussian distributions. Both the 8-Gaussian and 10-Gaussian datasets share similar parameters with the 5-Gaussian dataset: each Gaussian distribution contains 200 data points with a variance set to 0.1. The same progressive contraction operation of 10% is applied three times to each cluster, resulting in datasets spanning ten frames. The MNIST dataset consists of 60,000 gray-scale images of handwritten digits $(0 - 9)$ with each image represented as a 28×28 pixel array.

5.2 Evaluation Metrics

In order to quantitatively compare JUMAP with other techniques that can be used for data visualization, we used two metrics:

(1) Local Coherence Error (LCE)

LCE (Local Clustering Error) is used to quantify the changes in the local structure of the same clusters in two projections Y_0 and Y_1, by calculating the Euclidean distance between the edge vectors of each pair of points in the two projections.

$$LCE\,(Y_0, Y_1) = \frac{1}{n(n-1)} \sum_{\substack{C_k^0 \subset Y_0 \\ C_k^1 \subset Y_1}} \sum_{\substack{y_i^0, y_j^0 \in C_k^0 \\ y_i^1, y_j^1 \in C_k^1 \\ i<j}} \left\| \left(y_i^0 - y_j^0\right) - \left(y_i^1 - y_j^1\right) \right\|^2 \qquad (14)$$

where n is the number of sample points, $C_k^0 \subset Y_0$ and $C_k^1 \subset Y_1$ in C_k^0 and C_k^1 are the corresponding clusters in Y_0 and Y_1. Points $y_i^0, y_j^0 \in C_k^0$ refer to a pair of points within cluster C_k^0 in projection Y_0, while $y_i^0, y_j^0 \in C_k^0$ refer to the corresponding pair of points within cluster C_k^1 in projection Y_1.

(2) Cross Entropy (CE)

CE evaluates the effect of dimensionality reduction by calculating the cross entropy between the similarity distributions in the high and low dimensional spaces and quantifying the difference between the similarity distributions in the high and low dimensional spaces. See function 6.

5.3 Evaluation Results and Comparison

On the 5-Gaussian, 8-Gaussian, and 10-Gaussian datasets, we projected the data at different time frames $\left(t = 0, 1, 2, 3\right)$ using the JUMAP algorithm. The experimental results indicate that the alignment of projections deteriorates as the number of clusters increases. To provide a better comparison of the projections, Fig. 4 illustrates the visualization results on the 10-Gaussian dataset using the UMAP, EUMAP, JTSNE, and JUMAP algorithms. It is evident that the projection quality of JUMAP is comparable to that of JTSNE, with good alignment across different time frames. In contrast, the positions of the data classes using UMAP and EUMAP show noticeable shifts across different time frames, indicating poor alignment.

To gain a deeper understanding of the effectiveness of our algorithm in alignment projection, we measured and compared the performance of UMAP, EUMAP, t-SNE, ETSNE, JTSNE, and JUMAP algorithms on different datasets using the LCE. Tables 1 and 3 present the comparison results of LCE and running time for these six algorithms across three different datasets. Additionally, Table 2 shows the results of CE evaluation metrics for UMAP, EUMAP, and JUMAP.

In terms of LCE, a smaller LCE indicates better preservation of local structures in the projected data, meaning the proximity relationships of the data are better maintained during dimensionality reduction. As shown in Table 1, for the 5-Gaussian, 8-Gaussian, and 10-Gaussian datasets, the JUMAP's LCE is significantly lower than that of the other five algorithms, indicating that JUMAP has a substantial advantage in preserving local structures and aligning data across different time frames to generate more consistent projections.

In terms of CE, a smaller CE value indicates a higher match between the reduced-dimensional data distribution and the original distribution, reflecting better preservation of the global structure. As shown in Table 2, despite JUMAP introducing an additional vector constraint, its dimensionality reduction performance still surpasses the other two algorithms. Specifically, JUMAP is more effective in retaining the global structure and class information of the original high-dimensional data, resulting in more precise alignment and classification.

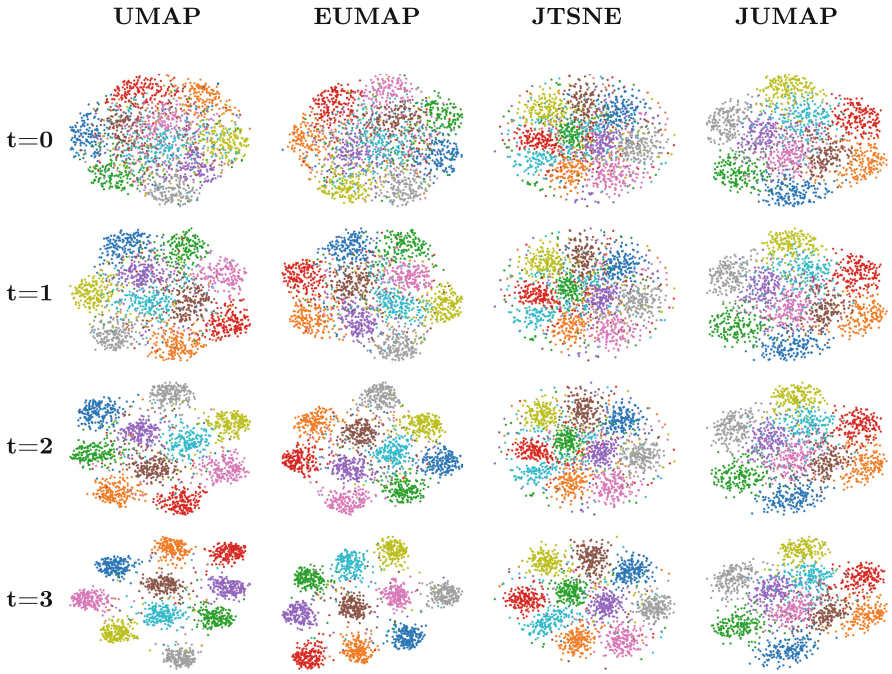

Fig. 4. The effect of 10-Gaussian low-dimensional embedding

Table 1. LCE for the Joint UMAP

Methods	5-Gaussian	8-Gaussian	10-Gaussian
UMAP	15.57	17.55	19.84
EUMAP	11.83	12.91	15.72
t-SNE	66.85	107.28	257.66
ETSNE	131.81	468.67	719.44
JTSNE	2.28	6.89	8.38
JUMAP	**0.18**	**0.19**	**0.19**

From Table 3, it is evident that JUMAP, compared to t-SNE, ETSNE, and JTSNE, significantly reduced running time while maintaining good embedding quality. JUMAP increases running time compared to UMAP and EUMAP due to the additional vector constraints considered during optimization. The time complexity of the JUMAP algorithm is primarily influenced by three key components: initial nearest-neighbor search, layout optimization, and the gradient descent step. Overall, the time complexity of JUMAP is $O(n^2 T)$, where n is the number of data points and T is the number of iterations. This is higher than standard UMAP ($O(n \log n)$) due to the additional temporal alignment constraints.

Table 2. CEs for the Joint UMAP

Methods	5-Gaussian	8-Gaussian	10-Gaussian	MNIST
UMAP	1.03	1.22	1.31	0.68
EUMAP	1.03	1.22	1.31	0.67
JUMAP	**0.97**	**1.15**	**1.31**	**0.66**

Table 3. Running time of Joint UMAP

Methods	5-Gaussian	8-Gaussian	10-Gaussian
UMAP	21.11	30.56	37.62
EUMAP	14.48	23.22	30.48
t-SNE	689.72	1759.56	2645.31
ETSNE	572.11	1519.69	2318.13
JTSNE	823.97	2034.23	3105.30
JUMAP	**152.66**	**388.17**	**552.81**

Specifically, the time complexity of the initial nearest-neighbor search is $O(n \log n)$, utilizing efficient search techniques such as KD-Trees or Ball Trees to find the nearest neighbors for each data point. The layout optimization process involves computing pairwise distances between data points during each iteration, resulting in a per-iteration time complexity of $O(n^2)$. Finally, the gradient descent step also has a time complexity of $O(n^2)$, as it involves computing the gradients of the loss function with respect to the data layout. Despite the higher time complexity, JUMAP maintains efficient performance for moderately-sized datasets, as evidenced by experimental results showing a balance between runtime and projection quality, particularly when compared to t-SNE and ETSNE, which have even higher time complexities.

5.4 Consistency of Projections on Synthetic Datasets

To better evaluate the fidelity and consistency of JUMAP, we conducted another experiment. The 5-Gaussian dataset, created by progressively contracting each cluster through moving each data point toward the center of its respective Gaussian distribution, simulates time-dependent changes. To explore richer types of transformations beyond the contraction operation, we applied three different transformations to three different time frames $\big(t = 0, 1, 2, 3\big)$ of the dataset. The three transformations were: translation, splitting, and overlapping. Similarly, before applying each transformation, we contracted all clusters by 10%. At $t = 1$, we applied the translation transformation to the first cluster (the blue cluster in Fig. 5). Specifically, we shifted all the data points in this cluster by 0.08 units in the positive direction in each dimension, resulting in a slight overall shift in one direction. At $t = 2$, we performed a splitting transformation on the third

cluster (the green cluster in Fig. 5), which we divided into two subclusters using the k-means clustering algorithm [10] and then moved these two sub-clusters by +0.08 in opposite directions along each dimension. Despite the split, this cluster is still considered as one. At $t = 3$, we performed an overlapping transformation on the second and fourth clusters (the orange and red clusters in Fig. 5). By making their means the same, we translated these clusters to a new position where they partially overlapped with each other. These transformations simulated the evolution of data over time, allowing us to evaluate the performance of different algorithms in handling data with various transformation patterns.

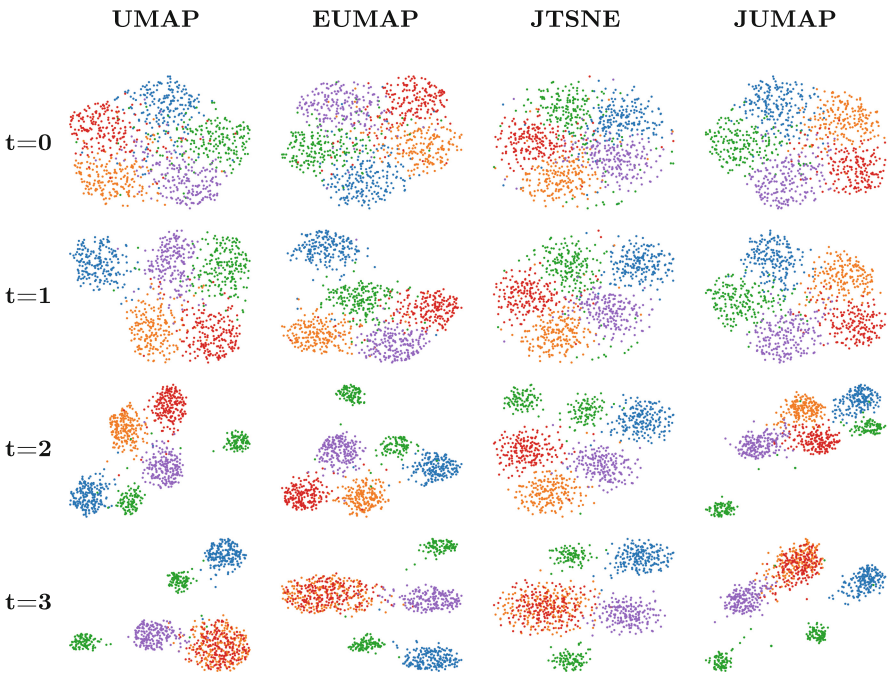

Fig. 5. Comparison plot of 4 different algorithms for projection of 5-Gaussian dataset

At $t = 1$, as shown in Fig. 5, all projection results place this cluster far from the others. However, the projections from UMAP and EUMAP exhibit misalignment. UMAP shifts the red cluster from the top left corner at $t = 0$ to the bottom right corner at $t = 1$, while EUMAP moves the red cluster from the top right corner at $t = 0$ to the bottom right corner at $t = 1$, resulting in projection inconsistency. This misalignment cannot be easily corrected using Procrustes transformation [9], which involves rotation, scaling, and translation to minimize shape differences. If one tries to rotate UMAP's projection at $t = 1$ to align the red cluster with its previous position, the green and orange clusters would still be misaligned. However, although JTSNE is able to move the blue cluster

from its position at $t = 0$ to a new location, the projection of JTSNE exhibits larger fluctuations between different time steps, causing significant changes in the position of the blue cluster and resulting in poor alignment between clusters. In contrast, JUMAP is better at maintaining the relative positions of the clusters, with its projection results showing smaller shifts and preserving the topological consistency of the data. This suggests that JUMAP provides more stable projection results when handling cluster translations.

At $t = 2$, the green cluster splits into two subclusters. While UMAP and EUMAP effectively separate the two subclusters, the projection's consistency and fidelity degrade due to the split. In contrast, JUMAP accurately handles the clustering split by considering the topological similarity between datasets from adjacent time frames. JUMAP compares the neighborhood structures of data points between $t = 1$ and $t = 2$, identifying topological changes and adjusting the projection accordingly. This enables JUMAP to more faithfully reflect the true structural changes between clusters compared to the other algorithms. At $t = 3$, the orange and red clusters completely overlap. Although JTSNE can capture overlapping trends, there is still a certain degree of distortion and overlap in its projection results, which leads to inaccurate cluster distribution in the projection. JUMAP effectively manages this complete overlap conversion. By identifying significant changes in the neighborhood structure of data points between time frames, JUMAP accurately projects fully overlapping clusters, avoiding partial overlap or distortion. This ability enables JUMAP to perform excellently in handling significant topological changes.

The experiment concludes that JUMAP is more accurate in dealing with complex transformations such as cluster splitting and overlapping by taking into account topological similarities between time points. JUMAP can identify and adapt to changes in the neighborhood structure of data points, ensuring that the projection results more faithfully reflect actual cluster transformations. JUMAP consistently performs well in projection tasks across different time frames, particularly excelling in alignment. Compared to UMAP and EUMAP, JUMAP shows a significant advantage in providing more consistent and accurate projection results when dealing with complex cluster transformations. The projections generated by JTSNE also indicate that JTSNE exhibits larger fluctuations in data point positions across different time frames compared to JUMAP, resulting in slightly poorer structural coherence.

5.5 Consistency of Projections on Real Datasets

Due to the Gaussian dataset itself being synthetic, created by generating points that follow a Gaussian distribution, in this experiment we use the real dataset MNIST to evaluate the consistency of projections generated by four different projection algorithms (UMAP, EUMAP, JTSNE, and JUMAP). The MNIST dataset is a well-known dataset of handwritten digit images. For this experiment, we extracted a subset of the MNIST dataset, selecting images of five different digits $\big(0, 1, 2, 3, 4\big)$. We randomly selected 100 images for each digit without repetition, resulting in a total of 500 images as the initial dataset.

In the experiment, we applied two transformations to these images that did not alter the pixel values but rather replaced the images. First, considering the similarity in shape between digits 9 and 4, we replaced all 100 images of digit 0 with images of digit 9. It was hypothesized that in the projection, the cluster of digit 4 and the replaced cluster of digit 9 would become closer. Second, we replaced the 100 images of digit 1 with a new set of 100 images of digit 3. These new digit 3 images were different from the initial 100 digit 3 images, being randomly selected without repetition from the remaining digit 3 images in the dataset. Since both sets of images represent digit 3, we hypothesized that the two clusters would completely overlap in the projection.

As shown in Fig. 6, at $t = 0$, digit 0 is represented by the blue cluster, digit 1 by the orange cluster, digit 2 by the green cluster, digit 3 by the red cluster, and digit 4 by the purple cluster. At $t = 1$, the projection results after applying the two transformations illustrate how different algorithms perform with the changed data. For UMAP, although the new cluster of digit 3 (the orange cluster) nearly overlaps with the original cluster of digit 3 (the red cluster), at $t = 0$, the original red cluster is located at the top left of digit 2 (the green cluster). However, after the data update at $t = 1$, the red cluster moves to the upper right of the green cluster, failing to maintain cluster position stability and exhibiting obvious projection inconsistencies, thereby not achieving the desired projection effect. EUMAP's performance is also suboptimal. The cluster of digit 3, which should have been at the upper right of digit 2, drifts to its lower right, while the cluster of digit 4 shifts from the lower right of digit 2 to the upper left. These projection results not only disrupt the local structural relationships between clusters but also indicate that EUMAP and UMAP struggle to maintain stable relative positions of clusters when handling temporal data updates. Moreover, these algorithms lack an effective mechanism to handle temporal smoothing between time steps, leading to discontinuous and inconsistent representations of dynamic data changes.

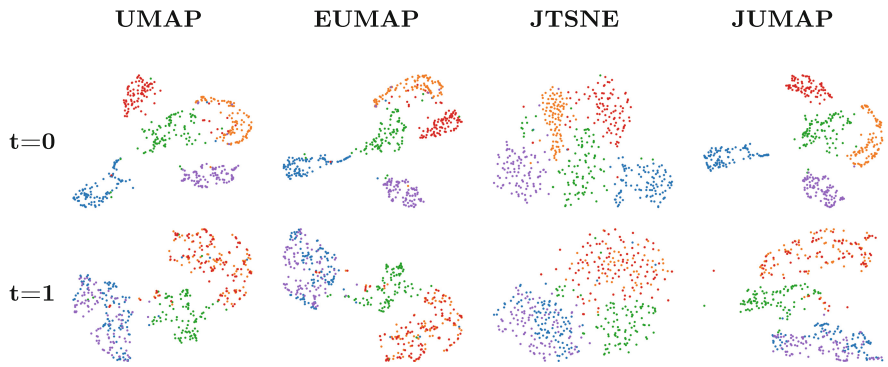

Fig. 6. Comparison plot of projections on MNIST dataset for 4 different algorithms

At $t = 0$, the projection generated by JUMAP shows compact and well-separated clusters for the five digits, clearly distinguishing each category in the low-dimensional space. At $t = 1$, after the data transformations, JUMAP maintains the relative positions of clusters effectively. The new cluster of digit 9 moves noticeably closer to the cluster of digit 4, aligning with the expected pattern from the transformation. Meanwhile, the new cluster of digit 3 almost perfectly overlaps with its original cluster, preserving both spatial consistency and cluster compactness. This demonstrates JUMAP's strong capability to maintain projection stability and coherence over time.

In contrast, JTSNE struggles to maintain local consistency across time points. The same categories of data points exhibit less cohesive clustering in the projection space, with clusters becoming fragmented and dispersed after data transformations. This inconsistency indicates that JTSNE is less effective at capturing temporal dynamics and preserving the inherent structural relationships within evolving datasets compared to JUMAP.

When applied to the real-world dataset MNIST, JUMAP demonstrates clear advantages in both quantitative metrics and visualization results compared to UMAP, EUMAP, and JTSNE. As shown in Table 2, the CE comparison highlights that JUMAP achieves the best dimensionality reduction performance on MNIST. It excels at preserving the relative positions of clusters before and after data transformations, resulting in more compact clusters for data points of the same category and clearer separation between different categories. These findings indicate that JUMAP provides superior stability and accuracy when handling complex data transformations.

6 Conclusions

This paper proposes Joint UMAP, an advanced visualization technique designed to handle dynamic datasets. By introducing kernel-based similarity measures and a novel vector constraint, Joint UMAP effectively addresses the challenge of dimensionality reduction for high-dimensional temporal data, generating consistent projections across multiple time steps. These innovations alleviate the inconsistencies faced by traditional methods, such as UMAP and t-SNE, in sequential projections. Experimental results demonstrate that Joint UMAP excels at producing projections with high consistency and fidelity. Tests conducted on both synthetic and real-world datasets show that Joint UMAP maintains superior projection consistency across datasets, significantly outperforming existing UMAP-based solutions. Compared to Dynamic t-SNE and Joint t-SNE, Joint UMAP exhibits greater robustness to data transformations, shorter runtime, and substantially reduced computational costs, making it better suited for handling large-scale and complex dynamic datasets.

Future work will explore validating this method on more complex datasets, such as financial time series and sensor network data, to further enhance its practical value. Additionally, integrating automated parameter selection strategies and lightweight neural networks (such as GCN or Transformer) could

improve adaptability and performance when handling high-dimensional nonlinear structures. Exploring its potential for real-time data stream projection in IoT and dynamic network analysis scenarios remains a promising avenue for future research.

Acknowledgments. This study was funded by Chongqing Municipal Education Commission (grant KJZD-K202100505).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aigner, W., Miksch, S., Schumann, H., Tominski, C.: Visualization of Time-Oriented Data, vol. 4. Springer, London (2011)
2. Bhuiyan, M.A., Rahman, M., Rahman, M., Al Hasan, M.: Guise: uniform sampling of graphlets for large graph analysis. In: 2012 IEEE 12th International Conference on Data Mining, pp. 91–100. IEEE (2012)
3. Boggust, A., Carter, B., Satyanarayan, A.: Embedding comparator: visualizing differences in global structure and local neighborhoods via small multiples. In: Proceedings of the 27th International Conference on Intelligent User Interfaces, pp. 746–766 (2022)
4. Chen, X., Xie, D., Wang, L., Zhao, Q., You, Z.H., Liu, H.: BNPMDA: bipartite network projection for MiRNA-disease association prediction. Bioinformatics **34**(18), 3178–3186 (2018)
5. Cox, T.F., Cox, M.A.: Multidimensional Scaling. CRC Press, Boca Raton (2000)
6. Fodor, I.K.: A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States) (2002)
7. Fujiwara, T., Chou, J.K., Shilpika, S., Xu, P., Ren, L., Ma, K.L.: An incremental dimensionality reduction method for visualizing streaming multidimensional data. IEEE Trans. Visual Comput. Graphics **26**(1), 418–428 (2019)
8. Gandhi, P., Pruthi, J.: Data visualization techniques: traditional data to big data. In: Data Visualization: Trends and Challenges Toward Multidisciplinary Perception, pp. 53–74 (2020)
9. Goodall, C.: Procrustes methods in the statistical analysis of shape. J. Roy. Stat. Soc.: Ser. B (Methodol.) **53**(2), 285–321 (1991)
10. Hartigan, J.A., Wong, M.A.: Algorithm as 136: a k-means clustering algorithm. J. Roy. Stat. Soc. Ser. C (Appl. Stat.) **28**(1), 100–108 (1979)
11. Hayes, W., Sun, K., Pržulj, N.: Graphlet-based measures are suitable for biological network comparison. Bioinformatics **29**(4), 483–491 (2013)
12. Hidaka, A., Kurita, T.: Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks. In: Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and Its Applications, vol. 2017, pp. 160–167 (2017)
13. Hu, Y., Wu, S., Xia, S., Fu, J., Chen, W.: Motion track: visualizing variations of human motion data. In: 2010 IEEE Pacific Visualization Symposium (PacificVis), pp. 153–160. IEEE (2010)

14. Jäckle, D., Fischer, F., Schreck, T., Keim, D.A.: Temporal MDS plots for analysis of multivariate data. IEEE Trans. Visual Comput. Graphics **22**(1), 141–150 (2015)
15. Jolliffe, I.T., Cadima, J.: Principal component analysis: a review and recent developments. Philos. Trans. Roy. Soc. A Math. Phys. Eng. Sci. **374**(2065), 20150202 (2016)
16. Khan, N., Iqbal, K., Martini, M.G.: Time-aggregation-based lossless video encoding for neuromorphic vision sensor data. IEEE Internet Things J. **8**(1), 596–609 (2020)
17. Leydesdorff, L., Schank, T.: Dynamic animations of journal maps: indicators of structural changes and interdisciplinary developments. J. Am. Soc. Inform. Sci. Technol. **59**(11), 1810–1818 (2008)
18. Liu, S., Maljovec, D., Wang, B., Bremer, P.T., Pascucci, V.: Visualizing high-dimensional data: advances in the past decade. IEEE Trans. Visual Comput. Graphics **23**(3), 1249–1268 (2016)
19. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(11) (2008)
20. Mao, Y., Dillon, J., Lebanon, G.: Sequential document visualization. IEEE Trans. Visual Comput. Graphics **13**(6), 1208–1215 (2007)
21. McInnes, L., Healy, J., Melville, J.: UMAP: uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018)
22. Milenković, T., Memišević, V., Ganesan, A.K., Pržulj, N.: Systems-level cancer gene identification from protein interaction network topology applied to melanogenesis-related functional genomics data. J. R. Soc. Interface **7**(44), 423–437 (2010)
23. Paulovich, F.V., Oliveira, M.C.F., Minghim, R.: The projection explorer: a flexible tool for projection-based multidimensional visualization. In: XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2007), pp. 27–36. IEEE (2007)
24. Pržulj, N., Corneil, D.G., Jurisica, I.: Modeling interactome: scale-free or geometric? Bioinformatics **20**(18), 3508–3515 (2004)
25. Rauber, P.E., Falcao, A.X., Telea, A.C., et al.: Visualizing time-dependent data using dynamic t-SNE (2016)
26. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
27. Sorzano, C.O.S., Vargas, J., Montano, A.P.: A survey of dimensionality reduction techniques. arXiv preprint arXiv:1403.2877 (2014)
28. Tscholl, D.W., Rössler, J., Said, S., Kaserer, A., Spahn, D.R., Nöthiger, C.B.: Situation awareness-oriented patient monitoring with visual patient technology: a qualitative review of the primary research. Sensors **20**(7), 2112 (2020)
29. Tuarob, S., et al.: Davis: a unified solution for data collection, analyzation, and visualization in real-time stock market prediction. Finan. Innov. **7**, 1–32 (2021)
30. Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E.: Deep learning for computer vision: a brief review. Comput. Intell. Neurosci. **2018**(1), 7068349 (2018)
31. Wang, Y., Chen, L., Jo, J., Wang, Y.: Joint t-SNE for comparable projections of multiple high-dimensional datasets. IEEE Trans. Visual Comput. Graphics **28**(1), 623–632 (2021)
32. Wang, Y., et al.: Revisiting stress majorization as a unified framework for interactive constrained graph visualization. IEEE Trans. Visual Comput. Graphics **24**(1), 489–499 (2017)
33. Wang, Y., et al.: Structure-aware fisheye views for efficient large graph exploration. IEEE Trans. Visual Comput. Graphics **25**(1), 566–575 (2018)

34. Xu, K.S., Kliger, M., Hero, A.O.: A regularized graph layout framework for dynamic network visualization. Data Min. Knowl. Disc. **27**, 84–116 (2013)
35. Yin, H.: Nonlinear dimensionality reduction and data visualization: a review. Int. J. Autom. Comput. **4**, 294–303 (2007)
36. Zhang, L., Han, Y., Yang, Y., Song, M., Yan, S., Tian, Q.: Discovering discriminative graphlets for aerial image categories recognition. IEEE Trans. Image Process. **22**(12), 5071–5084 (2013)