This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114765, IEEE Transactions on Visualization and Computer Graphics

Joint t-SNE for Comparable Projections of Multiple High-Dimensional Datasets

Yinqiao Wang, Lu Chen, Jaemin Jo, Yunhai Wang

Abstract— We present Joint t-Stochastic Neighbor Embedding (Joint t-SNE), a technique to generate comparable projections of multiple high-dimensional datasets. Although t-SNE has been widely employed to visualize high-dimensional datasets from various domains, it is limited to projecting a single dataset. When a series of high-dimensional datasets, such as datasets changing over time, is projected independently using t-SNE, misaligned layouts are obtained. Even items with identical features across datasets are projected to different locations, making the technique unsuitable for comparison tasks. To tackle this problem, we introduce edge similarity, which captures the similarities between two adjacent time frames based on the Graphlet Frequency Distribution (GFD). We then integrate a novel loss term into the t-SNE loss function, which we call vector constraints, to preserve the vectors between projected points across the projections, allowing these points to serve as visual landmarks for direct comparisons between projections. Using synthetic datasets whose ground-truth structures are known, we show that Joint t-SNE outperforms existing techniques, including Dynamic t-SNE, in terms of local coherence error, Kullback-Leibler divergence, and neighborhood preservation. We also showcase a real-world use case to visualize and compare the activation of different layers of a neural network.

Index Terms—High-dimensional data, projection, embedding, t-stochastic neighbor embedding

1 Introduction

We aim to generate comparable t-Stochastic Neighbor Embedding (t-SNE) [26] projections of a series of multidimensional datasets. Although multidimensional projection (MDP) techniques, such as t-SNE, play an essential role in high-dimensional data analysis by giving the users the ability to inspect high-dimensional data with lower-dimensional (e.g., 2D) representations, most of them are designed for visualizing a single high-dimensional dataset. Yet, a common data analysis task is to explore the evolution of a high-dimensional dataset, which requires comparing the projections between adjacent frames. One real-world example is a dataset consisting of the activation in each layer of a deep neural network. Given n input images, we record the internal representation of each image on layer i having f_i dimensions, which yields a dataset D_i consisting of $n \cdot f_i$ scalar values. One may project each dataset D_i and compare the projections to understand how each layer transforms the internal representations. Note that in this example, the number of items in datasets (i.e., n) does not change, but the number of dimensions (i.e., f_i) can vary.

Since most MDP techniques are designed to visualize one dataset, there have been a few workarounds to apply them to multiple datasets and use the results for comparison. The simplest workaround is to start with the same initial layout when optimizing the projection for each dataset. However, this straightforward method does not consider the relationship between adjacent time frames, and as the optimization process usually resorts to a stochastic process, the alignment between projections is not guaranteed. Thus, even the items that are invariant across the datasets can be placed in different locations, making the projections lack visual landmarks that can be used to facilitate comparison tasks [15].

Another workaround common in bioinformatics [14] is to concatenate all datasets into a single dataset and compute the projection of the combined dataset. Then, the projection can be visualized as a conventional scatterplot with an additional visual encoding for the source of data points (i.e., the dataset from which a data point comes), for example, by adopting color-coding [1] or by connecting the data points for the same item [3]. However, this workaround does not scale well visually nor computationally since the number of data points to be shown increases by a factor of the number of the datasets, which leads to visual clutters and long computation time.

The most relevant technique to ours is Dynamic t-SNE [35]. In this technique, the projections for multiple time frames are optimized together with an extra loss term that penalizes the length of the locus of each data point across projections. The additional loss term makes each data point stays in a similar position across projections. However, we identified three major drawbacks of Dynamic t-SNE as follows:

1. **Smoothing effects**: Dynamic t-SNE tends to keep projections too rigid, which leads to inaccurate projections when there are abrupt changes. For example, assume a point p lies inside a cluster A at time t but moves to another cluster B at $t+1$. Although the local structure around p has been completely changed at $t+1$, the extra loss term pulls back p to cluster A to minimize the locus of p between time t and $t+1$, which gives users an illusion that p is gradually moving from cluster A to cluster B.

2. **Long-range Interference**: Since multiple projections are optimized together, the projection at t even reflects future changes that will happen after t and past changes happened before t, leading to a less faithful projection.

3. **Monolithic**: All datasets must be available when Dynamic t-SNE is performed, and this makes the technique unsuitable for datasets generated periodically. To compute the projection P_{t+1} for a new dataset at $t+1$, all the previous projections ($P_1 \cdots P_t$) should be updated, which can invalidate the findings made previously and prevent incremental analysis.

To address these limitations, we present *Joint t-SNE*, a novel multidimensional projection technique that generates coherent projections of multiple high-dimensional datasets. The main idea of Joint t-SNE is to preserve the topology between projections selectively based on their topological similarity. To this end, we first capture the topological characteristics around each point by employing the Graphlet Frequency Distribution (GFD) [34]. We then introduce a novel loss term, *vector constraints*, that guides the optimization process to preserve edge vectors between projected points across two-time frames. In Joint t-SNE, a projection at $t+1$ is joined by the previous projection at t to preserve the edge vectors weighted by their topological similarity.

- *Y.Q. Wang, L. Chen, and Y.H. Wang are with Shandong University, CN. Email: {InfamyWong,chenlu.scien,cloudseawang}@gmail.com.*
- *J. Jo is with Sungkyunkwan University, KR. E-mail: jmjo@skku.edu.*
- *Y.H. Wang and J. Jo are joint corresponding authors*
- *Y.Q. Wang and L. Chen contribute equally to this work.*

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.xxxx/TVCG.201x.xxxxxxx

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114765, IEEE Transactions on Visualization and Computer Graphics

Through benchmarks on synthetic and real-world datasets, we found that Joint t-SNE can generate more consistent projections compared with the existing workarounds and Dynamic t-SNE. We also found that, compared with Dynamic t-SNE projections, Joint t-SNE projections are not only more faithful in terms of quality metrics, such as local coherence error or kNN preservation but also robust to distortion, such as the smoothing effect. Finally, Joint t-SNE breaks the global dependency between projections in the optimization process and naturally lends itself to dynamic scenarios where datasets arrive over time.

In summary, our contributions are:

- We introduce and implement Joint t-SNE that can generate consistent and faithful projections of multiple datasets by introducing kernel-based similarity measures and a novel loss term, vector constraints.
- We quantitatively and qualitatively evaluate Joint t-SNE and show that Joint t-SNE outperforms the existing practices and techniques in terms of visual consistency and projection fidelity.

2 Related Work

In this section, we discuss previous studies targeting comparable projections and review the related work of a key technical module in Joint t-SNE, comparing graph similarity using Graphlet Frequency Distributions (GFD).

2.1 Comparable Projections

A projection [30] is a mapping that projects or embeds high-dimensional data into a perceptible lower-dimensional space (usually two or three dimensions) while striving to maintain the relationships among data points in the original space. Formally, a projection algorithm takes in a high-dimensional dataset $X = \{x_1, x_2, ..., x_n\}$, where each x_i is an m-dimensional data point, and computes the projection of X in a lower-dimensional space, $Y = \{y_1, y_2, ..., y_n\}$, where each y_i is an injective mapping of x_i. Projections are now widely used in various fields like data mining, machine learning [5, 17], and bioinformatics [8] for the superiority in revealing the underlying distribution and topology of high-dimensional data, making it possible for people to make analyses and interpretations.

Comparable projections, as extensions of traditional projections, deal with the problem of projecting a sequence of dynamic high-dimensional datasets $X_1, X_2, \cdots, X_T$, where each X_i is captured at a particular time frame into a low-dimensional space, resulting in an equal-sized sequence of projections $Y_1, Y_2, \cdots, Y_T$. A simple method would be to project each X_i independently. But due to the stochastic and unpredictable optimization process of many projection techniques, such method often introduces undesirable variations, such as misalignment of identical data points. Thus, the common goal of comparable projections is to achieve visual consistency between sequential projections while maintaining projection reliability. More specifically, we expect the algorithm to automatically find and highlight the inheritances and variations of data across different time frames to provide viewers with accessible observations into the evolving pattern of streaming data.

Previous attempts for comparable projections can be classified into two categories based upon the type of data evolution: incremental and time-dependent method. First, in the scenario for incremental projection methods, datasets update or expand by an incremental and cumulative pattern in each frame. For example, Alencar et al. [1] proposed a technique based on least square projection and a backward strategy for creating content-based document maps to visualize temporal changes in document collections. Takanori et al. [13] augmented an existing incremental PCA algorithm [36] by applying an affine transformation to find the best overlap of common data points in two adjacent projections and presenting a position estimation algorithm to support adding data points with a non-uniform number of dimensions.

On the other hand, in contrast to incremental evolution, where previous data usually remain static, time-dependent projection methods deal with dynamic datasets in which the features of all data points can change over different time frames. With the providers of data points fixed in consecutive frames, this scenario can be found almost everywhere in today's information society, such as tracking the physical status of all patients in a hospital or the traffic volume of all crossroads in a city. Jäckle et al. [18] proposed temporal MDS plots which reduce the multivariate data in each time frame to a 1D slice and align these slices along the time axis by adopting a flipping heuristic to achieve comparability. However, reducing the dimensionality to one discards too much information, making it nearly impossible for the users to analyze in-depth. Ali et al. [2] treated data points in a frame as a high-dimensional vector and projected the whole time-series dataset into a single image. Although such a method can provide users with instant discovery of anomalies and clusters, it is not scalable to datasets which contain hundreds and thousands of data sources, and could not be used in dynamically updated scenarios.

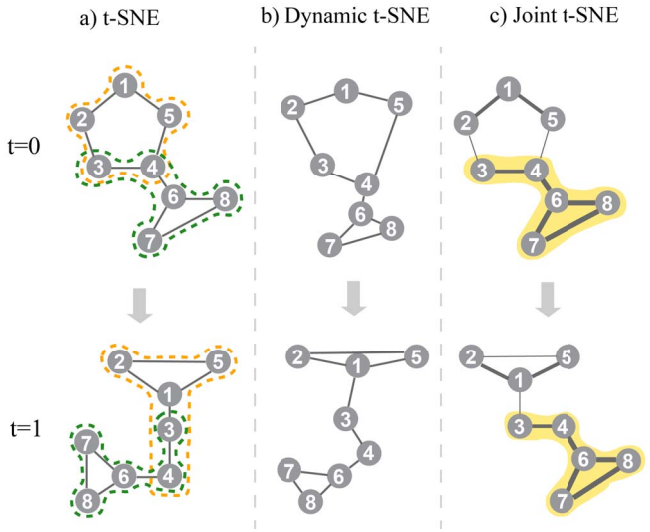


Fig. 1. An illustrative example of t-SNE, Dynamic t-SNE, and Joint t-SNE results. Two intersected 5-node structures, one with changes in topology and one without, are projected by three methods. a) t-SNE projects both structures differently, making it hard for viewers to distinguish between changed and unchanged structures. b) Although Dynamic t-SNE smooths the transition between projections, it also introduces distortions that cannot be ignored. c) Joint t-SNE preserves identical structures based on measuring edge similarities (encoded as thickness in the figure) in data space. The unchanged structure is well-preserved and could serve as a visual landmark for comparison.

Dynamic t-SNE developed by Rauber et al. [35] enhanced the conventional t-SNE by introducing an additional loss term, which penalizes the movement of all data points in between multiple projections, into the objective function of t-SNE and optimizing the position of all data points across projections at the same time. Although the purpose of maintaining visual consistency is achieved, distortions frequently occur due to its rigid constraints on the absolute position of every single point, as seen in Fig. 1. And since dynamic t-SNE takes in the full sequence of datasets at once and optimizes all projections as a whole, it introduces a heavy computational burden and poses a significant challenge to the memory, thus not suitable for projecting streaming data.

Our Joint t-SNE is designed for time-dependent data evolution but can also be applied in incremental evolution scenarios with a simple modification as discussed in Sect. 6. By applying vector constraints to adjacent projections based on similarities measured in data space, we provide consistent projections suitable for comparison tasks.

2.2 Measuring the Similarity between Graphs

Measuring the similarity between graphs, a subarea of graph mining, has gained popularity for many years for its significant value in various fields like chemistry [39], bioinformatics [12], and computer vision [44].

One straightforward approach to measure graph similarity is to compare *node sequences* by traversing the sequence of vertices or edges

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114765, IEEE Transactions on Visualization and Computer Graphics

such as graph edit distance [7]. These methods have low computational complexity and are easy to implement but can hardly capture the topological structure within the graph. A more complicated approach is to compare *feature maps*, which are specific distinguishable attributes extracted from graphs, like fingerprints to humans. Research has shown the remarkable performance of this method as a number of different graphs can be distinguished and identified accurately by referring to feature maps such as degree distribution [33], shortest paths [6], subtrees [22], and most importantly graphlets [34].

Graphlets, first proposed by Pržulj et al. [34], are defined as induced non-isomorphic k-node subgraph patterns, where $k \in \{3,4,5\}$. Graphlet Frequency Distribution (GFD), also known as graphlet concentrations or graphlet statistics, is a widely used feature for analyzing graphs in image category recognition [44], biological network comparison [16], disease gene identification [28] and a host of other areas.

Based on statistics of elementary substructures, GFD is able to reflect the differences in the topological structure of graphs. Milenković and Pržulj [29] generalized the concept of node degree to graphlet degree, which is defined as the number of graphlets connected to one node. By using the vector of graphlet degrees to represent and analyze protein function on protein-protein interaction (PPI) networks, it is proven to be a robust algorithm in comparing the topological structure of a local area. The graphlet spectrum [21] presented by Kondor et al. applied a group-theoretic approach to enhance the capability of graph kernels, enabling them to capture the relative positions of subgraphs besides numbers. Recently, Kwon et al. [23] used a supervised machine learning method to learn the relationship between topological features of existing graphs measured by GFD and their layouts. The trained model could show the most topologically similar graphs in different layouts rapidly to provide users with a quick visual perception of the input graph.

However, one of the biggest challenges of using GFD in practice is the computational complexity of algorithms since the detection and enumeration of graphlets are costly computations. A multitude of work has been done to accelerate such computation. By adopting a Markov Chain Monte Carlo sampling method, Bhuiyan et al. [4] proposed GUISE for approximating GFDs of large networks, which provided significant speedup compared with the brute-force counting method. In this work, we employ GFD and GUISE to capture the local topological structures around high-dimensional data points.

3 Background

3.1 t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) [26], a commonly used nonlinear projection method, has been widely acknowledged for its outstanding performance in learning the underlying structure of high-dimensional data at different scales. The overall process of t-SNE can be divided into the following three stages:

Computing Probability Distribution of High-dimensional Data t-SNE starts by calculating the pairwise distance $d(x_i,x_j)$ (e.g., the Euclidean distance) of data points in high-dimensional space, resulting in a distance matrix, which is then converted into a probability distribution P using a Gaussian kernel to model the local structure around each data point. The conditional probability $p_{j|i}$, which measures the pairwise similarity between point x_i and x_j, can be intuitively interpreted as the probability of data point x_i selecting x_j as its neighbor. A symmetrized joint probability p_{ij} is computed and used to form distribution P.

$$p_{j|i} = \frac{\exp\left(-d\left(x_i,x_j\right)^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-d\left(x_i,x_k\right)^2/2\sigma_i^2\right)}, \qquad p_{i|i}=0. \tag{1}$$

$$p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}. \tag{2}$$

Computing Probability Distribution of Projected Points In the low-dimensional space, t-SNE first randomly initializes the projection and then calculate the joint probability distribution Q from pairwise distance matrix, same as it did in high-dimensional space, only this time using a heavy-tailed Student's t-distribution.

$$q_{ij} = \frac{\left(1+\left\|y_i-y_j\right\|^2\right)^{-1}}{\sum_{k\neq l}\left(1+\left\|y_k-y_l\right\|^2\right)^{-1}}, \qquad q_{ii}=0. \tag{3}$$

Minimizing Mismatch between the Two Distributions Finally, t-SNE minimizes the Kullback-Leibler divergence between distribution P and Q to conserve local structures between high and low-dimensional spaces, using a gradient descent method that updates the position of each projected point iteratively.

$$C = KL(P\|Q) = \sum_{i\neq j} p_{ij}\log\frac{p_{ij}}{q_{ij}} \tag{4}$$

$$\frac{\partial C}{\partial y_i} = 4\sum_{i\neq j}(p_{ij}-q_{ij})(y_i-y_j)(1+\|y_i-y_j\|^2)^{-1} \tag{5}$$

Dynamic t-SNE To adapt t-SNE for comparable projections, Dynamic t-SNE chooses to penalize the overall movement of each point across projections by adding an extra loss term that restricts projected points to stay in the same position as much as possible. Formally, for a series of high-dimensional datasets $X_1, X_2, \cdots, X_T$, the position of each point y_i^t, where $t \in [1,T]$, in the corresponding projections $Y_1, Y_2, \cdots, Y_T$ is computed by minimizing the function below:

$$C = \sum_{t=1}^{T} C_t + \frac{\lambda}{2N}\sum_{i=1}^{N}\sum_{t=1}^{T-1}\left\|y_i^t - y_i^{t+1}\right\|^2$$

where C_t is the t-SNE cost of projecting X_t to Y_t.

3.2 Graphlet Kernels

Graphlets are non-isomorphic substructures in graphs with k nodes, typically $k \in 3,4,5$ (see Fig. 2). Any large graph could be disassembled to these basic structures, thus the frequencies of graphlets could be used as a fingerprint to differentiate various graphs. Such fingerprint is defined as Graphlet Frequency Distribution (GFD) and is widely used to characterize and compare the structure of different graphs.

While GFD is an effective method, counting graphlets is a computationally expensive task, especially on large graphs. The time complexity for enumerating k-node graphlets exhaustively on a graph $G(V,E)$ is $O(|V|^k)$, which forces sampling methods to be introduced into the pipeline.

One work towards the uniform sampling of graphlets is GUISE [4], an efficient algorithm that uses a Markov Chain Monte Carlo (MCMC) method to approximate GFD for large graphs. Formally, given a graph G, assume S_G is a set that contains all the graphlets in G. The key to uniform sampling is to ensure the probability of selecting each one of the graphlets g_i in S_G is $1/|S_G|$. The MCMC algorithm implemented in GUISE performs a random walk on sample space S_G with a transition probability matrix in such a manner that the stationary distribution of the random walk aligns with the independent and identical distribution.

Graphlet Kernels Graphlet kernels are adaptations of kernel methods for measuring graph similarities. A kernel, also known as generalized dot product, is a function that measures the similarity between two vectors, $\mathbf{x},\mathbf{y} \in \mathbb{R}^m$, by computing the dot product of their counterparts, $\varphi(\mathbf{x}),\varphi(\mathbf{y}) \in \mathbb{R}^n$, in a high-dimensional feature space, where φ is a mapping that brings $\mathbf{x}$ and $\mathbf{y}$ into the feature space $\mathbb{R}^n$.

Formally, a graphlet kernel k_g which measures similarity between graph G_0 and G_1 is defined as the dot product of their feature vectors.

$$k_g(G_0,G_1) = < f_{G_0}, f_{G_1} >$$

where feature vector f is usually based on normalized GFD. Frequently used kernels include Gaussian kernel, Laplacian kernel and cosine similarity.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114765, IEEE Transactions on Visualization and Computer Graphics

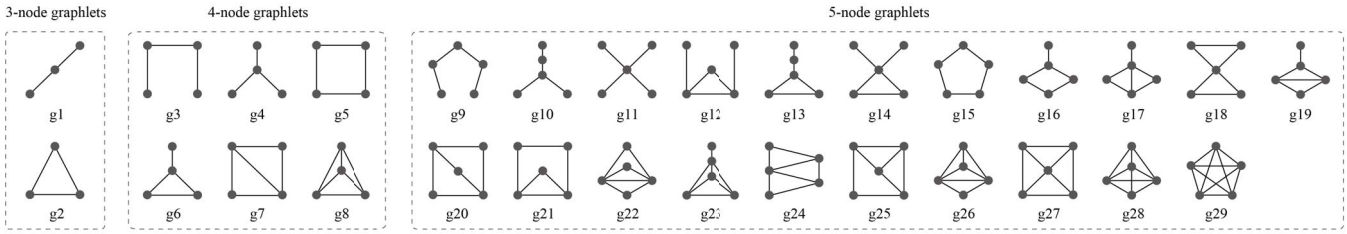


Fig. 2. All graphlets of 3, 4, or 5 nodes.

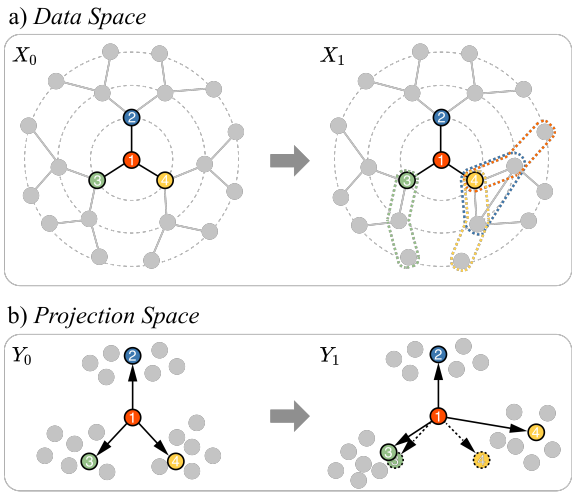


Fig. 3. Technical Illustration of Joint t-SNE. Note that we only consider 3-node graphlets for simplicity. a) Some changes happened between X_0 and X_1. Several points broke the neighborhood relationship with the original cluster. Joint t-SNE measures the similarity of local structures to find such changes and computes edge similarities ($S_{e_{12}} > S_{e_{13}} > S_{e_{14}}$). b) Using edge similarity as the weight of the corresponding vector constraint, Joint t-SNE generates projection Y_1, which keeps the relative position of points in Y_0 accordingly.

4 Joint t-Stochastic Neighbor Embedding

In this section, we elaborate on Joint t-SNE. We first investigate the design considerations and introduce the basic idea of our solution. Then we give an overview of the algorithm pipeline followed by a detailed description of two technical highlights of Joint t-SNE, graphlet kernel-based edge similarity and edge vector constrained t-SNE.

4.1 Design Considerations

The aim of Joint t-SNE is to generate low-dimensional embeddings that can provide users with consistent visual tracking of data evolution in the high-dimensional space. For most time-series datasets, there are both inheritances and variations of data points in each time frame during the evolution. We expect those inherited structures to be preserved in the projection space to serve as visual landmarks, whereas the varied ones to be distinguished from the others. Therefore, the problem of generating comparable projections can be reduced to realizing the following three goals:

1. **G1**: Detecting changes of local structures happening between time frames in data space,
2. **G2**: Preserving inherited structures in projection space, and
3. **G3**: Ensuring projection fidelity.

Note that G1 is the basis of G2, and G2 may be conflicted with G3 since distortions may be introduced.

Conventional dimensionality reduction methods such as PCA and t-SNE are designed for projecting a single dataset, with G3 as their only concern. When used for projecting datasets from multiple time frames, such methods would generate misaligned layouts. A common idea in dealing with time-series data is the sliding window mechanism. The term "window" refers to an imaginary box holding a batch of data from consecutive time frames. A window of length l takes in and processes data from l frames simultaneously and slides to the next position after finishing the process of current data inside. We adopt such mechanism and set $l = 2$ to acquire datasets from adjacent time frames, X_0 and X_1, at a time, as well as to avoid the effect of long-range interference. Using one dataset, say X_0, whose projection Y_0 has already been computed as a reference, we project X_1, taking both G2 and G3 into consideration, resulting in Y_1. Note that we will state this process as "Y_1 is joined by Y_0."

To meet G2, the major challenge is how to incorporate the DR results of the previous frames into the next frames. Dynamic t-SNE uses the absolute positions of the previous frames, which cannot effectively characterize the change of local structures. Recent work [41, 42] has shown that edge-vector-based constraints outperform previous methods in preserving local structures. Inspired by that, We develop *vector constraints*, which try to keep the relative position of point-pairs, and integrate such constraint to the loss function of t-SNE. The goal is to preserve structures that are inherited in data space and make their appearances in the later projection the same as in the former. To this end, we set the weight of each vector as the *edge similarity*, which measures the variation of the corresponding edge between high-dimensional data points, to guide the optimization process.

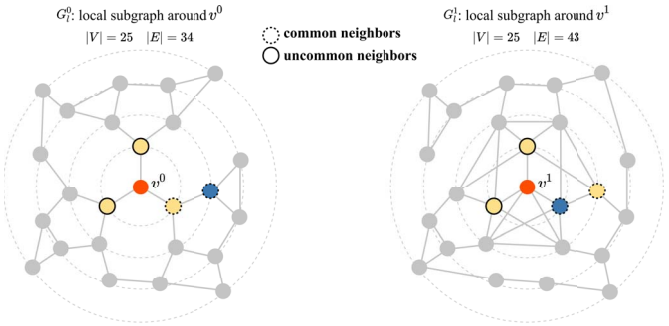
In order to detect local changes and further measure edge similarities, we first construct a kNN graph in high-dimensional space for each dataset to model the neighborhood relationship among data points. Then we measure the similarity of the local subgraph around each node between two graphs with the identity of nodes taken into account. To this end, we use a graphlet kernel to measure such similarity, resulting in one value for each point which represents how much its surrounding topology, as well as neighbors, changed from the former time frame to the latter. Note that high edge similarity indicates minor changes in local structure, while low similarity reflects major changes.

4.2 Algorithm Pipeline

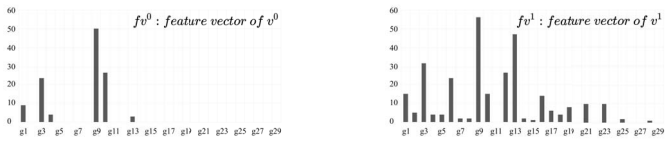
Given two datasets X_0 and X_1, the computation pipeline (see Fig. 3) of Joint t-SNE is as follows:

1. We construct G_0 and G_1, the undirected k-Nearest Neighbor (kNN) graphs of X_0 and X_1, respectively.
2. We compute the Graphlet Frequency Distribution(GFD) vector of local graph structure around each node in G_0 and G_1, fv_i^0 for $v_i^0 \in V(G_0)$ and fv_i^1 for $v_i^1 \in V(G_1)$, respectively (see Fig. 4).
3. For each node pair $(v_i^0, v_i^1) \in \{(v_i^0, v_i^1) \mid v_i^0 \in V(G_0), v_i^1 \in V(G_1)\}$, we compute its point similarity (Eq. 6) using the feature vectors of v_i^0 and v_i^1.
4. Then for each edge pair common in both graphs, i.e., $(e_{ij}^0, e_{ij}^1) \in \{(e_{ij}^0, e_{ij}^1) | e_{ij}^0 = (v_i^0, v_j^0) \in E(G_0), e_{ij}^1 = (v_i^1, v_j^1) \in E(G_1)\}$, we compute its edge similarity (Eq. 7) using the point similarity of its two endpoints.
5. We compute Y_0, the t-SNE projection of X_0, using normal t-SNE (Sect. 3.1). Note that this is the case at the very beginning; when

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114765, IEEE Transactions on Visualization and Computer Graphics



a) *common neighborhood ratio* $= 2/3$



b) $< fv^0, fv^1 >= 0.731$

Fig. 4. An illustrative example of computing point similarity. a) Node v loses a neighbor during the transition from time $t = 0$ to $t = 1$. b) The difference between feature vector fv^0 and fv^1 reflects the changes in the local structures of node v.

the sliding window moves on to subsequent time frames, the latest computed projection would be taken as the new Y_0 directly.

6. We compute Y_1, the t-SNE projection of X_1, joined by Y_0, incorporating vector constraints of common edges between G_0 and G_1 (Sect. 4.4).

4.3 Graphlet Kernel based Edge Similarity

Given a hyperparameter k, which indicates the number of neighbors to consider, Joint t-SNE starts with constructing G_0 and G_1, the kNN graphs of two datasets X_0 and X_1, respectively, using a distance metric (e.g., Euclidean). We consider the graphs to be undirected, i.e., an edge e_{ij} is present if and only if $v_i \in kNN(v_j)$ or $v_j \in kNN(v_i)$. Note that hereafter we use the two graphs for computing point similarities instead of the two raw datasets, which allows us to correlate datasets with different numbers of dimensions.

Then, we introduce the **feature vector** of a node v, which characterizes the local structure around v in a kNN graph. Graphlets consist of 29 unique connected structures of 3 to 5 nodes, and the frequency distribution of graphlets (GFD) can be used as a measurement of graph topology. We count the number of each structure in the kNN graph that includes v, normalize the counts by dividing each count by the sum, and use the normalized counts as the feature vector f_v of v.

Since counting all graphlets is computationally expensive, especially when k is large, we embed the sampler, the core module of GUISE, into Joint t-SNE and develop an algorithm that computes the feature vectors of nodes using uniformly sampled graphlets from a global scale. The pseudo-code of both the original GUISE algorithm and our modified version is included in our supplementary material.

To capture the structural similarity between two graphs, we present two types of similarity, **point similarity** and **edge similarity**. The point similarity between v_i^0 and v_i^1 is defined as the cosine similarity between their feature vectors multiplied by the fraction of common neighbors between v_i^0 and v_i^1. Since we assumed the two datasets have the same items, we can compute the point similarity of every item i between the two datasets.

$$S_{p_i} = \frac{|kNN(G_0, v_i^0) \cap kNN(G_1, v_i^1)|}{k} \cdot < fv_i^0, fv_i^1 > \tag{6}$$

Although the cosine similarity of feature vectors captures the structural similarity between the neighbors of two nodes, it does not consider their identity. For example, suppose an item i lies within cluster A in G_0 but moves to cluster B in G_1. Even if the two clusters have identical structures, the point similarity between v_i^0 and v_i^1 should be zero since they do not have common neighbors. Therefore, we penalize the cosine similarity by multiplying the fraction of common kNN items between v_i^0 and v_i^1. Fig. 4 gives an example of computing point similarity.

The edge similarity between e_{ij}^0 and e_{ij}^1 is defined as the multiplication of the point similarities between their two endpoints:

$$S_{e_{ij}} = S_{p_i} \cdot S_{p_j} \tag{7}$$

Note that edge similarities can be computed only for the common edges that exist in both G_0 and G_1 while point similarities can be computed for all items.

During projecting the datasets separately, we will try to keep the edge vectors between projected points with high edge similarities in the data space invariant across projections.

4.4 Edge Vector Constrained t-SNE

Given the two datasets, X_0 and X_1, we project X_0 using the conventional t-SNE and get Y_0, the projection of X_0 in a low-dimensional space, that minimizes the Kullback-Leibler divergence between a joint probability distribution in the high-dimensional space, p_{ij}^0, and a joint probability distribution in the low-dimensional space, q_{ij}^0 as follows:

$$\arg\min_{Y_0} C_0 = \sum_{i \neq j} p_{ij}^0 \log \frac{p_{ij}^0}{q_{ij}^0} \tag{8}$$

For X_1, Joint t-SNE generates a coherent projection Y_1 joined by Y_0, preserving the topological structures of items that possess high similarity in data space between Y_0 and Y_1. To this end, we introduced a novel constraint to the cost function of t-SNE: **vector constraints**. The cost function for X_1 is as follows:

$$\begin{aligned} \arg\min_{Y_1} C_1 = & \sum_{i \neq j} p_{ij}^1 \log \frac{p_{ij}^1}{q_{ij}^1} \\ & + \frac{\gamma}{M} \sum_{i \neq j} S_{e_{ij}} \cdot \|(y_i^0 - y_j^0) - (y_i^1 - y_j^1)\|^2 \end{aligned} \tag{9}$$

where $S_{e_{ij}}$ is the similarities of common edges between graph G_0 and G_1 constructed from X_0 and X_1 respectively, M is the number of those common edges, γ is the weight for vector constraints set by users.

The gradient is computed as follows:

$$\begin{aligned} \frac{\partial C_1}{\partial y_i^1} = & 4 \sum_{i \neq j} (p_{ij}^1 - q_{ij}^1)(y_i^1 - y_j^1)(1 + \|y_i^1 - y_j^1\|^2)^{-1} \\ & - \frac{2\gamma}{M} \sum_{i \neq j} S_{e_{ij}} ((y_i^0 - y_j^0) - (y_i^1 - y_j^1)) \end{aligned} \tag{10}$$

For each common edge $(e_{ij}^0, e_{ij}^1) \in (G_0, G_1)$, edge constraints penalize Y_1 if the two vectors calculated by endpoints of common edges are distant in the Euclidean space, i.e., aim to maintain the relative positions between points. Vector constraints are weighted by a hyperparameter γ. The impacts of the hyperparameter are demonstrated in Sect. 5.

5 Evaluation

In this section, we quantitatively and qualitatively compare four techniques for generating comparable projections, *the original t-SNE* [26] (OT), *Equal-initialization t-SNE* (ET), *Dynamic t-SNE* [35] (DT), and our *Joint t-SNE* (JT). OT served as a baseline; we projected each dataset D_i individually using the original t-SNE algorithm with a random initial layout. For ET, we applied the original t-SNE algorithm to each dataset individually but used the same initial layout, which was generated randomly. For DT, we used the implementation from the original paper [35]. We describe the hyperparameter settings used for evaluation in the supplementary materials.

JT requires two hyperparameters, γ and k, while DT requires λ. k was set to 3 in all experiments in our paper and supplementary materials. Since both loss terms of JT and DT are the averaged displacements of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114765, IEEE Transactions on Visualization and Computer Graphics

edges or points, we set $\gamma = \lambda$ for direct comparison between JT and DT. In our paper, they were set to 0.1 by default which was the value used in a previous experiment [35].

To compare the techniques quantitatively, we used the following three metrics:

1. **Local Coherence Error (LCE)** of specific clusters between frames: We use LCE to quantify the visual consistency of local structures across consecutive projections. LCE is defined as the sum of Euclidean distances between every two edge vectors in clusters C_k in projections Y_0 and Y_1, whose ground-truth topology is known to be unchanged.

$$LCE(Y_0, Y_1) = \sum_{\substack{C_k^0 \subset Y_0 \\ C_k^1 \subset Y_1}} \sum_{\substack{y_i^0, y_j^0 \in C_k^0 \\ y_i^1, y_j^1 \in C_k^1 \\ i<j}} \|(y_i^0 - y_j^0) - (y_i^1 - y_j^1)\|^2 \quad (11)$$

2. *k*NN **Preservation (KNN)** for a single frame: We use KNN to measure the preservation of local structures when conducting dimensionality reduction following a previous paper [20]. KNN is defined as the proportion of k-nearest neighbors of points in the original high-dimensional space X that are preserved in the projection space X_{prj}.

$$KNN(X) = \frac{|kNN(X) \cap kNN(X_{prj})|}{|kNN(X)|} \quad (12)$$

3. **KL Divergence (KLD)** for a single frame: KLD measures the faithfulness of modeling between high- and low-dimensional spaces. See function 4.

Although these metrics are useful in measuring how well projections preserve local structures, they are not enough to understand the fidelity and consistency between projections for different frames. To understand them, we also conduct the qualitative evaluation by investigating projections before and after a certain data transformation. Designing the experiments, we identified that there is a general trade-off between the internal and external validity in our evaluation. For example, if we experiment on the techniques using synthetic high-dimensional data and transformations whose ground-truth meanings are known (e.g., changing the distance between two hypothetical spheres), we can accurately assess the faithfulness and consistency between projections, but the external validity is hurt. Otherwise, if we use two real-world datasets, which are externally valid, we cannot judge whether the projections preserve the structures between the frames faithfully since it is rare to have the ground truth on the topological changes happening between the two real-world datasets.

To balance the trade-off, we conduct three experiments with different levels of abstraction. In the first experiment (Sect. 5.1), we use two synthetic datasets that samples from random Gaussian clusters: *the 10-Gaussian dataset* [35] and *the 5-Gaussian dataset*. To simulate time-dependent changes, we perturb the datasets by applying known data transformations on clusters, such as translation, split, and overlapping. This design is the most internally valid; we can assess the faithfulness and consistency of projections since the structures of datasets, and the meaning of data transformations are known. To seek the balance between internal and external validity, in the second experiment (Sect. 5.2), we use a real-world dataset, *the MNIST dataset*, with synthetic data transformations, i.e., replacing digits in images (e.g., "1" to "3"). Finally, in the third experiment (Sect. 5.3), we apply the four techniques to a real-world dataset, the activation tensors from the VGG-16 network [38], i.e., *the VGG dataset*, to understand how high-dimensional representations are transformed on each layer. In this experiment, even the meaning of the transformation that happens between two adjacent datasets is unknown.

5.1 Joint t-SNE on Synthetic Datasets

The goal of the first experiment was to assess the faithfulness and consistency of the four projection techniques when applied to generate comparable projections.

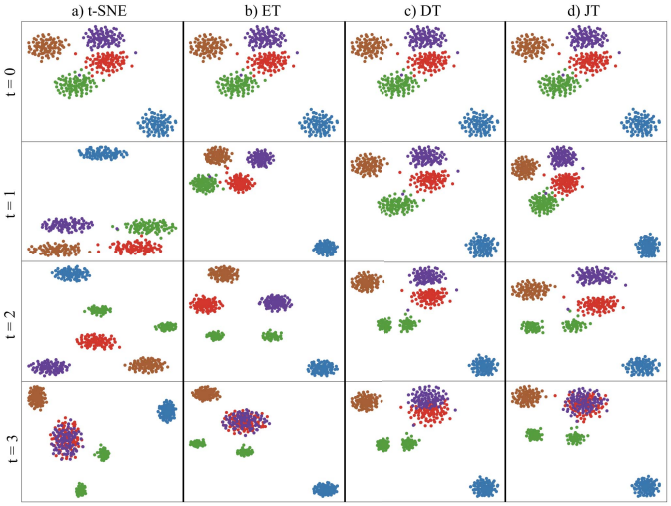


Fig. 5. Comparison of the 5-Gaussian dataset projection of four different t-SNE methods. a) t-SNE produced misaligned layouts all across four time frames. b) Equal-initialization t-SNE provides better visual consistency than t-SNE but there are still unnecessary movements of clusters. c) Dynamic t-SNE showed smoothing effect by distorting projections at $t = 2$ and 3. d) Joint t-SNE generated coherent and reliable projections that reflected the ground-truth transformations of clusters.

Datasets and Transformation We first employed the 10-Gaussian dataset that was used to evaluate Dynamic t-SNE [35]. The dataset consisted of 2,000 points in a 100-dimensional space that were sampled from ten isotropic Gaussian distributions with a variance of 0.1. The centers of the distributions were chosen randomly between the standard basis vectors for R^{100}, i.e., the centers were equidistant from each other.

The authors of the Dynamic t-SNE paper simulated time-dependent changes by contracting each cluster stepwisely. Specifically, they moved each data point towards the center of the corresponding Gaussian distribution by 10% of the remaining distance to the center. They repeated this contraction operation nine times to generate nine frames (i.e., $t = 1 \cdots 9$), obtaining the final datasets of ten frames.

To further experiment on richer types of transformations in addition to the contraction operation, we generated a dataset, the 5-Gaussian dataset. We started by sampling 500 points from five isotropic Gaussian distributions (100 for each) in a 100-dimensional space. The centers of the distributions were chosen randomly between the standard basis vectors with a variance of 0.05 (the first row of Fig. 5).

We then generated three more frames ($t = 1, 2, 3$) by applying one of the following transformations: translation, splitting, and overlapping, and before applying each transformation, we contracted all clusters by 10% as in the 10-Gaussian dataset. At $t = 1$, we translated the points in the first cluster (the blue cluster in Fig. 5) by +0.15 in all dimensions (i.e., $3 \cdot variance$). At $t = 2$, we split the second cluster (the green cluster in Fig. 5) by half. We used the k-means clustering algorithm to make two subclusters ($k = 2$) and translate the subclusters either by ± 0.15 (i.e., $\pm 1.5 \cdot variance$) in all dimensions to the opposite direction. At $t = 3$, we overlapped the third and fourth clusters (the red and purple clusters in Fig. 5) by translating them to have the same mean.

Results and Discussion Since the 5-Gaussian dataset illustrates a richer set of data transformations, we report the projection results of the 5-Gaussian dataset in Fig. 5. the result for the 10-Gaussian dataset can be found in the supplementary materials. In Fig. 5, note that for ease of comparison, we used the Dynamic t-SNE projection at $t = 0$ as the common projection for all four methods even though it suffered from the long-range interference problem; the blue cluster is already distant from the other clusters even at $t = 0$ since it *will* become distant.

At $t = 1$, a cluster (i.e., the blue cluster) was pushed away from the other clusters. All the projection results manifested this transformation by placing the cluster distantly from the other clusters, but the original

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114765, IEEE Transactions on Visualization and Computer Graphics

Table 1. Averaged Local Coherence Error

Datasets \ Methods	t-SNE	ET	DT	**JT**
5-Gaussian	1,972.38	1,468.45	30.21	**13.66**
10-Gaussian	7,365.81	6,822.01	249.88	**61.03**

Table 2. Quantitative measurement for projection fidelity

Datasets \ Methods	kNN preservation				KL divergence			
	t-SNE	ET	DT	**JT**	t-SNE	ET	DT	**JT**
5-Gaussian	0.30	0.30	0.32	**0.34**	1.00	1.00	1.05	**1.03**
10-Gaussian	0.19	0.19	0.16	**0.23**	1.62	1.62	1.69	**1.69**
MNIST	0.26	0.26	0.21	**0.24**	1.00	1.00	1.16	**1.05**
VGG	0.57	0.56	0.48	**0.55**	0.60	0.60	1.01	**0.65**

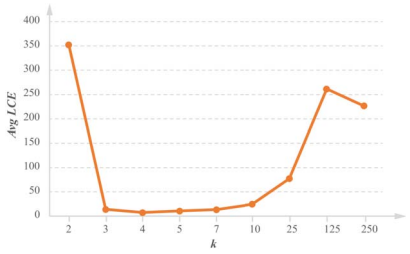
t-SNE produced a misaligned projection; the blue cluster was near the bottom-right corner at $t = 0$, but it moved to the top at $t = 1$. Note that this misalignment cannot be fixed by a simple Procrustes transform; one may want to rotate the projection at $t = 1$ to align the position of the blue cluster with its previous position, but then the red and green clusters will become misaligned. In contrast, DT and JT generated more consistent projections; they preserved the positions of the four untouched clusters.

At $t = 2$, a cluster (i.e., the green cluster) was split into two subclusters. We could see that the separation between the subclusters was relatively small in DT compared with the other three techniques due to the smoothing effect; the two clusters were less separated to minimize the movement of points. The smoothing effect of DT became clearer at $t = 3$. Here, we overlapped two clusters (i.e., the red and purple clusters) completely, but they overlapped each other only partially in DT. In contrast, JT alleviated the smoothing effect by considering the topological similarity of points between two datasets; the two clusters could be placed faithfully since the neighborhood of points in the clusters was drastically changed, resulting in small edge similarity.

In terms of Local Coherence Error, JT outperformed the other three techniques (Table 1) for both the 10-Gaussian and 5-Gaussian datasets, suggesting that JT produced more consistent projections. Regarding projection fidelity (Table 2), JT outperformed other techniques in terms of kNN preservation, but it produced higher KL divergence than OT and ET. This is because JT considered an extra loss term (i.e., vector constraints) in addition to KL divergence to maintain consistency.

Additionally, we investigated the effect of hyperparameters, γ for JT and λ for DT, by testing three values, 0.01, 0.05, and 0.1 (Fig. 6). Note that we could not test values larger than 0.1 since the optimization process of DT became unstable and collapsed. This time, we applied the four transformations to the 5-Gaussian dataset at the same time ($t = 1$) for comparison. Fig. 6 shows that JT was more robust to distortion, such as the smoothing effect, regardless of λ (see the overlap of the red and purple clusters) while DT was sensitive to its hyperparameter λ; for example, the smoothing effect between the red and purple clusters became more severe as we increase λ.

We also investigated the effect of hyperparameter k on the final embeddings with the 5-Gaussian dataset in two aspects, LCE and computational time. The result shows that LCE becomes high both when k is very small (2) and very large ($|nodes|/2$), see the inset. This is because small k will lead to an insufficient number of edges in kNN graphs for modeling the relationship between nodes effectively, whereas a large k might make GFD becomes ineffective for measuring node similarities. As for computational time, it increases with the increase of k. Therefore, we recommend that users set k to 3, 4, or 5 to achieve a better trade-off between fidelity and efficiency.



Experiment 1 showed that compared with DT, JT was more consistent and faithful for the tested dataset in terms of both qualitative and quantitative evaluation. Compared with OT and ET, JT was more consistent and had comparable fidelity. These findings suggest that JT achieves all three design goals mentioned in Sect. 4.1, and solved the conflict between G2 and G3 elegantly.

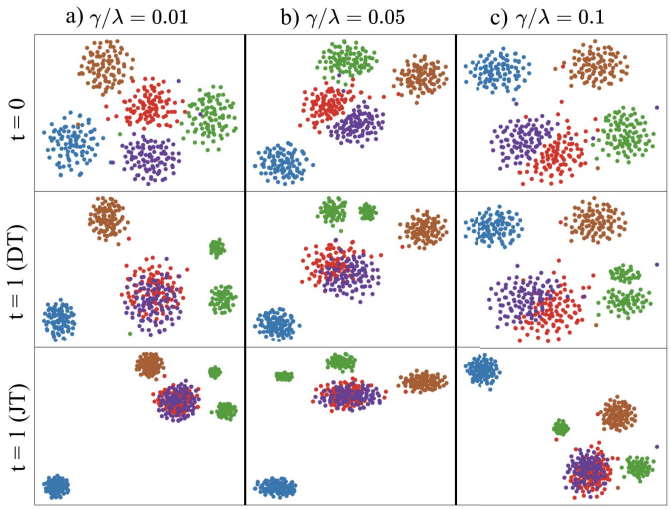


Fig. 6. Comparison of the effect of hyperparameters in Dynamic and Joint t-SNE. We used the result from Dynamic t-SNE as the projection of the first time frame for ease of comparison. For Dynamic t-SNE, as λ increases from 0.01 to 0.1, the effect of long-range interference at $t = 0$ becomes more obvious as the red and purple clusters gets closer, which is a future change. The same is true for smoothing effect at $t = 1$. Joint t-SNE is robust to the change of γ; see the red and purple clusters fully overlap regardless of γ.

5.2 Joint t-SNE on the MNIST Dataset

The goal of the second experiment was to assess the consistency of projections generated on a real-world dataset with synthetic transformations.

Datasets and Transformation We used the MNIST dataset [24] which consisted of the 60,000 black-and-white images of hand-written digits. As the initial dataset at $t = 0$, we collected the first 100 images of five digits (i.e., [0, 4]) from the MNIST dataset. In contrast to the Gaussian datasets in the previous experiment, mutating the images by adding or subtracting a constant to each pixel was not meaningful. Instead, we applied two replacement transformations at $t = 1$ with different levels of resulting similarity. First, we replaced the images of digit 0 with those of digit 9; we used the first 100 images of digit 9. We hypothesized that the existing cluster of digit 4 and the replaced cluster of digit 9 should become closer since the two digits were similar. Second, we replaced the images for digit 1 with those of digit 3; we used 100 images of digit 3 that were different from the images of digit 3 at $t = 1$. We hypothesized the two clusters should overlap completely at $t = 1$ since they depicted the same digit, 3.

Results and Discussion We show the projection results of ET, DT, and JT at $t = 0$ and $t = 1$ in Fig. 7. First, ET produced the projections that supported our hypotheses; the clusters for digits 0 and 4 (the blue and brown clusters) became closer as digit 0 was replaced with digit 9 at $t = 1$. Furthermore, the clusters for digits 1 and 3 (the green and purple clusters) were distant at $t = 0$ but overlapped at $t = 1$ after digit 1 (the green cluster) was replaced with digit 3. However, ET altered even an untouched cluster, the red cluster for digit 2, possibly giving the user misunderstanding that its local structure also changed.

The results from DT suffered from long-range interference and smoothing effects that we mentioned. At $t = 0$, the clusters for digits 0 and 4 (the blue and brown clusters) were close to each other even before the transformation took place. Because they would become similar at $t = 1$, DT placed them nearby to minimize the length of the locus of points. The projection was distorted early due to the changes happening

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114765, IEEE Transactions on Visualization and Computer Graphics

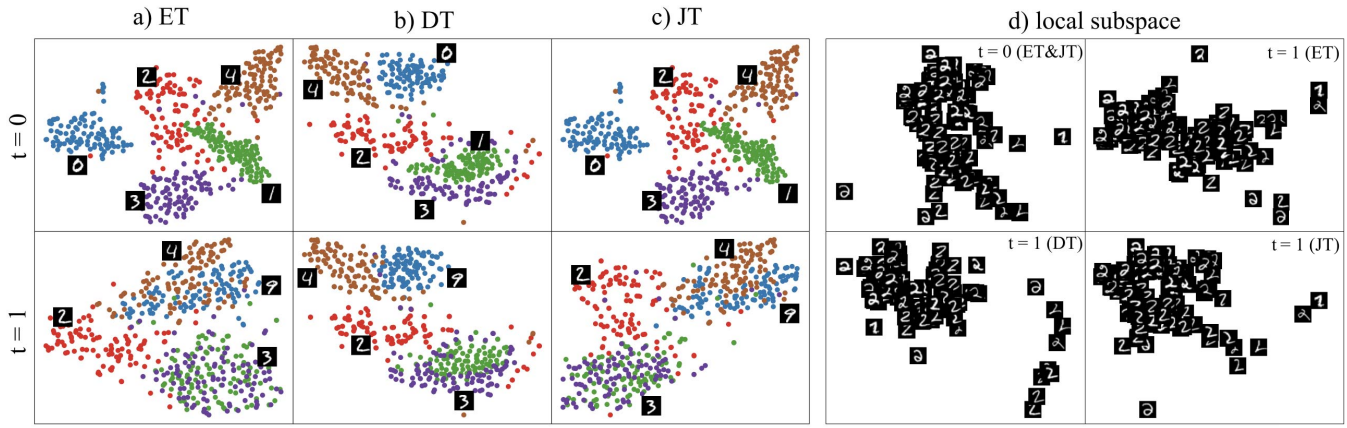


Fig. 7. Comparison of the MNIST dataset projection of three t-SNE techniques. Note that since the effect of long-range interference of Dynamic t-SNE was too serious in this case, we did not use its result for $t = 0$ as the initial projection for other methods as we did in other cases. a) Projecting each frame separately in Equal-initialization t-SNE could faithfully reveal the underlying structure. But when used for comparison tasks, the results can be misleading; for example, people can think that the red cluster for digit 2 changed and the purple cluster of digit 3 moved to the green cluster of digit 1, but the ground truth is the opposite. b) In Dynamic t-SNE, the changes between two frames are subtle even though there were substantial changes. These projections are too consistent, sacrificing fidelity. c) Joint t-SNE successfully detected the changes in topology while preserving the local structure of the red cluster for digit 2 where no change was made. d) Comparison of local subspace shows that Dynamic t-SNE created an artifact of digit 2 on the right side, whereas other methods did not.

in the future, i.e., long-range interference. The two clusters for digit 3 at $t = 1$ exhibited the smoothing effect; they should fully overlap each other but were separated because the points were forced to move minimally, maintaining their previous positions at $t = 0$. Due to these problems, DT generated very consistent but unfaithful projections.

In contrast, JT successfully displayed changes with a consistent layout. The two changes we made were clearly visible at $t = 1$; two clusters for digit 3 (the green and purple clusters) fully overlapped while the clusters for digits 4 and 9 (the brown and blue clusters) partially overlapped. Note that JT also preserved the local structure of the cluster of digit 2 (the red cluster) where no change was made, allowing the user to perform comparison tasks using that cluster as visual landmarks.

Experiment 2 showed that JT could capture the similarity of local structures between two frames. For similar structures (e.g., the red cluster of digit 2), JT placed the structures consistently in the projection space. Even for the structures that changed abruptly (e.g., the overlapped clusters of digit 3), it could project them faithfully. Note that maintaining the consistency can lower local quality metrics (Table 2) compared with unbound techniques such OT, but we believe the benefit of having consistent projections for comparison tasks can pay the cost.

5.3 Joint t-SNE on the VGG Dataset

As a real-world use case, we tested the four techniques on the VGG dataset [38]. We investigated how each layer in the VGG network transforms intermediate representations of images by projecting and comparing the activation. Note that in contrast to previous experiments, it is very difficult to have ground-truth interpretation on the transformation happening on a layer due to the high dimensionality and complexity of the dataset and network.

Dataset As input images, we collected 700 images of ten classes (70 images for each class) from ImageNet [11]. The ten classes consisted of eight animal species and two unrelated classes, which were cats (tiger cat and tabby cat), dogs (giant schnauzer and standard schnauzer), sharks (great white shark and tiger shark), fish (goldfinch and brambling), beach wagon, and military uniform.

The VGG-16 network has 23 layers where the first layer represents a raw input image, and the last layer corresponds to output class probabilities. Given the input images, the activation recorded on each layer forms a single high-dimensional dataset whose rows represent input images and columns represent the activation at each neuron. The datasets had different numbers of columns ranging from 1,000 to 3,497,984. For datasets that had more than 4,096 dimensions, we used random projection [25] to reduce the dimensionality to 4,096.

Results and Discussion Fig. 8 shows the evolution of activation on the last four layers, excluding the final softmax prediction layer in the VGG-16 network, projected by four different techniques. Overall, the separation between classes became clear at the first fully connected layer (*fc1*). In OT and ET, the positions of clusters changed drastically; for example, ET placed the clusters of birds (the red and pink clusters) on the left part of the projection at *block5-conv*, but the clusters moved to the lower part at *fc1* and finally to the lower-right part at *fc2*. Although it is known that the absolute position of the cluster is not important for t-SNE [43], the moving cluster makes the comparison between layers more challenging.

DT manifested several problems. First, the two classes of birds (the red and pink clusters) were largely separated at the *fc2* layer while other techniques placed them nearby. This phenomenon can be seen as an example of the smoothing effect; because the two clusters were distant previously (i.e., at *block5-conv*), placing the two clusters closer faithfully would cause a substantial loss. A similar problem happened for the two classes of dogs (the green and light green clusters) at *fc1*. A small subcluster of tiger cats (the light blue cluster) in a lower part remained separated, possibly causing misunderstanding that this subcluster is significantly different from other clusters for cats. Finally, we found that DT did not work on the full dataset due to a scalability issue while other techniques did since it tried to load all datasets in memory to optimize them together.

In contrast, JT generated projections that are locally similar to OT or ET projections but with consistent global structures. For example, two clusters for sharks (the purple and light purple clusters) became separated at *fc1*, which was visible in all three techniques. However, OT and ET flipped the two clusters vertically between *fc1* and *fc2*, which did not happen in Joint t-SNE projections.

6 General Discussion

Benefits of Joint t-SNE Through a series of experiments, we could identify the benefits of Joint t-SNE as follows: 1) Joint t-SNE provides visual consistency between projections. The vector constraints we added guide the optimization process to place invariant structures to consistent locations. Note that keeping projections consistent with the vector constraints naturally induces a loss in KL divergence (Table 2), but the loss was small compared to its benefits and can be controlled by a hyperparameter λ. 2) Joint t-SNE is more faithful than Dynamic t-SNE. Even the high-dimensional topology changes abruptly, the GFD-based edge similarity we proposed captures the changes and allows the optimization process to faithfully project the topology. 3) Joint t-SNE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2021.3114765, IEEE Transactions on Visualization and Computer Graphics

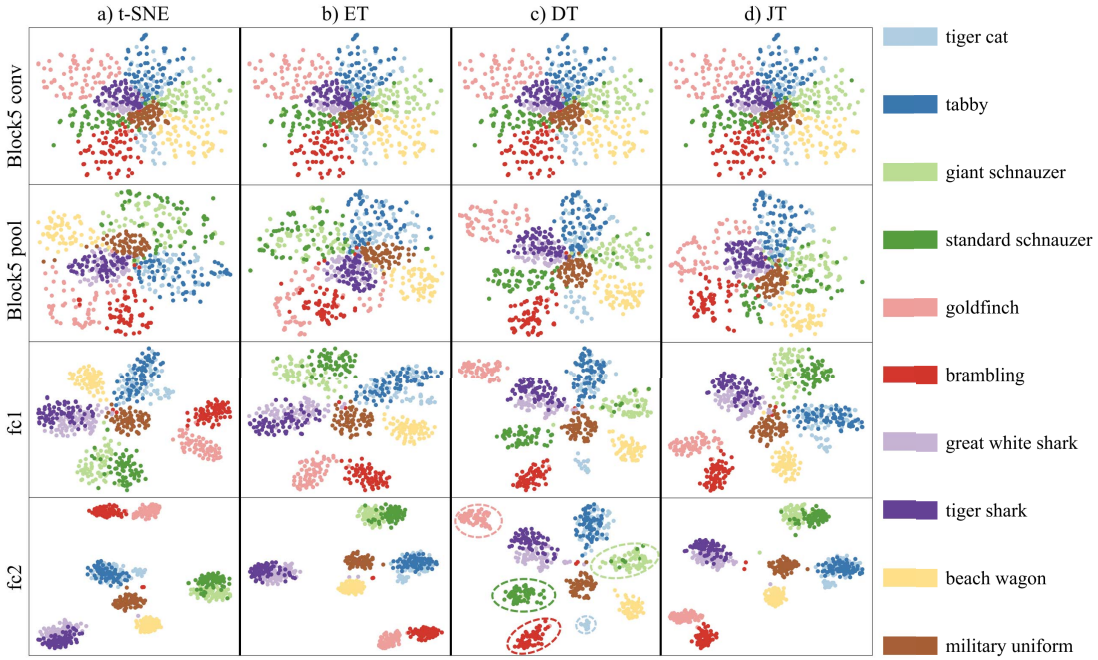


Fig. 8. Comparison of the VGG dataset projections of four t-SNE techniques. a) and b) t-SNE and Equal-initialization t-SNE produced faithful but inconsistent projections. See the red and pink clusters moving around the center. c) Dynamic t-SNE produced projections that are too rigid to reflect abrupt changes in topology. For example, the green and light green clusters remain separated at the end, failing to escape from their initial positions. d) Joint t-SNE generated more faithful projections that are robust to such abrupt changes. See the points from a pair of classes (two classes with the same hue) gather as in the t-SNE and Equal-initialization t-SNE projections while providing visual consistency.

is more flexible than Dynamic t-SNE since generating a projection at t only depends on the previous one $t-1$ and does not change the previous projection. It is more suitable for dynamic datasets where new time frames are added incrementally and causes a low computational burden since only two adjacent projections reside in memory at a time.

Analysis of Algorithmic Complexity Our implementation of Joint t-SNE mainly consists of three major parts: graphlet construction, graphlet-based similarity computation, and Joint t-SNE optimization. For a dataset with n nodes and k neighbors of each node are considered, suppose the number of graphlets is m, the time complexities of three parts are $O(km)$, $O(n)$, and $O(n^2)$, respectively. In total, the time complexity of Joint t-SNE is $O(n^2+km)$. When the graph is dense, m might be much larger than n^2, and the complexity is $O(n^2)$ for sparse graphs. We report the actual running time of the three methods in the supplementary material.

Extending to t-SNE Variants and Other Projection Techniques Since vector constraints do not change the inner workings of a projection technique, they can be integrated into accelerated t-SNE algorithms (e.g., Barnes-Hut t-SNE [40]), variants that use t-SNE internally (e.g., Hierarchical SNE [31] or progressive t-SNE algorithms [19, 32]), or even other gradient-based projection algorithms.

One of the most promising extensions is the use of Barnes-Hut t-SNE [40] that can improve the time complexity of Joint t-SNE from $O(n^2)$ to $O(nlogn)$. Barnes-Hut t-SNE internally builds kD trees to identify the k_{BH} neighbors of each point. Note that k, the number of neighbors used for computing feature vectors in our work, and k_{BH}, the number of neighbors for the approximation in Barnes-Hut t-SNE, have different meanings and scales. For k_{BH}, the original paper [40] suggested a number $\lfloor 3 \cdot perplexity \rfloor$ which is about 50 times larger than the minimum value we suggest for k (i.e., $k=3$) assuming $perplexity$ is set to 50 as in common settings. One benefit of having smaller k than k_{BH} is that we can reuse the kNN graph built for the Barnes-Hut approximation to reduce the overhead of introducing vector constraints.

Although Joint t-SNE is based on t-SNE, our concepts, measuring graph similarities and applying vector constraints, can be easily adapted to other projection techniques, such as Multidimensional Scaling (MDS) [9] or Uniform Manifold Approximation and Projection (UMAP) [27]. For MDS, we can directly add the vector constraints to the loss function; see the supplementary materials for an example. UMAP constructs a weighted directed kNN graph $\bar{G}$ and converts it to symmetric adjacency matrix B. In this case, for GFD computation, we need to sample edges with probabilities proportional to their weights in B in the random walk as in a previous framework [37].

Future Work The computation pipeline of Joint t-SNE can be further improved by using approximation [40] or exploiting loop-based parallelism [10] although we used a single-threaded version in the experiments. We can also consider an estimation framework [37] to speed up the graphlet counting process. We leave accelerating Joint t-SNE as future work. Another interesting future work would be to integrate vector constraints in hierarchical projection techniques, such as Hierarchical SNE [31], to speed up the computation and ensure the coherency between the projections at different levels.

7 Conclusion

We present Joint t-SNE to generate comparable projections of multiple high-dimensional datasets. Based on the GFD feature vectors capturing the local topological structures of points, we introduce a novel loss term to t-SNE, vector constraints, to guide the optimization process to preserve edge vectors between data points depending on their similarity between time frames. Through a series of experiments, we found that Joint t-SNE can generate consistent projections compared with the previous techniques while keeping projection fidelity. Over the years, multidimensional projections have been proven to be of great utility for giving an overview of a single high-dimensional data. Our study further extends their utility to a new dimension, enabling them to be used for multiple high-dimensional datasets such as multivariate time-oriented data even with changing numbers of dimensions.

Acknowledgments

This work is supported by the grants of the NSFC (61772315, 61861136012), the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No.VRLAB2020C08), and the CAS grant (GJHZ1862).

REFERENCES

[1] A. B. Alencar, K. Börner, F. V. Paulovich, and M. C. F. de Oliveira. Time-aware visualization of document collections. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pp. 997–1004, 2012. doi: 10.1145/2245276.2245469

[2] M. Ali, M. W. Jones, X. Xie, and M. Williams. Timecluster: dimension reduction applied to temporal data for visual analytics. *The Visual Computer*, 35(6):1013–1026, 2019. doi: 10.1007/s00371-019-01673-y

[3] J. Bernard, N. Wilhelm, M. Scherer, T. May, and T. Schreck. Timeseriespaths: Projection-based explorative analysis of multivariate time series data. 2012.

[4] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan. Guise: Uniform sampling of graphlets for large graph analysis. In *2012 IEEE 12th International Conference on Data Mining*, pp. 91–100. IEEE, 2012. doi: 10.1109/ICDM.2012.87

[5] A. Boggust, B. Carter, and A. Satyanarayan. Embedding comparator: Visualizing differences in global structure and local neighborhoods via small multiples. *arXiv preprint arXiv:1912.04853*, 2019.

[6] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 8–pp. IEEE, 2005. doi: 10.1109/ICDM.2005.132

[7] H. Bunke, P. J. Dickinson, M. Kraetzl, and W. D. Wallis. *A graph-theoretic approach to enterprise network dynamics*, vol. 24. Springer Science & Business Media, 2007. doi: 10.1007/978-0-8176-4519-9

[8] X. Chen, D. Xie, L. Wang, Q. Zhao, Z.-H. You, and H. Liu. Bnpmda: bipartite network projection for mirna–disease association prediction. *Bioinformatics*, 34(18):3178–3186, 2018. doi: 10.1093/bioinformatics/bty333

[9] M. A. A. Cox and T. F. Cox. *Multidimensional Scaling*, pp. 315–347. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-33037-0_14

[10] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. doi: 10.1109/CVPR.2009.5206848

[12] A. J. Enright and C. A. Ouzounis. Biolayout—an automatic graph layout algorithm for similarity visualization. *Bioinformatics*, 17(9):853–854, 2001. doi: 10.1093/bioinformatics/17.9.853

[13] T. Fujiwara, J.-K. Chou, S. Shilpika, P. Xu, L. Ren, and K.-L. Ma. An incremental dimensionality reduction method for visualizing streaming multidimensional data. *IEEE transactions on visualization and computer graphics*, 26(1):418–428, 2019. doi: 10.1109/TVCG.2019.2934433

[14] X. Gao, D. Hu, M. Gogol, and H. Li. Clustermap: compare multiple single cell rna-seq datasets across different experimental conditions. *Bioinformatics*, 35(17):3038–3045, 2019. doi: 10.1093/bioinformatics/btz024

[15] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011. doi: 10.1177/1473871611416549

[16] W. Hayes, K. Sun, and N. Pržulj. Graphlet-based measures are suitable for biological network comparison. *Bioinformatics*, 29(4):483–491, 2013. doi: 10.1093/bioinformatics/bts729

[17] A. Hidaka and T. Kurita. 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. The ISCIE Symposium on Stochastic Systems Theory and Its Applications, 2017. doi: 10.5687/SSS.2017.160

[18] D. Jäckle, F. Fischer, T. Schreck, and D. A. Keim. Temporal mds plots for analysis of multivariate data. *IEEE transactions on visualization and computer graphics*, 22(1):141–150, 2015. doi: 10.1109/TVCG.2015.2467553

[19] J. Jo, J. Seo, and J.-D. Fekete. Panene: A progressive algorithm for indexing and querying approximate k-nearest neighbors. *IEEE transactions on visualization and computer graphics*, 2018. doi: 10.1109/TVCG.2018.2869149

[20] D. Kobak and P. Berens. The art of using t-sne for single-cell transcriptomics. *Nature communications*, 10(1):1–14, 2019. doi: 10.1038/s41467-019-13056-x

[21] R. Kondor, N. Shervashidze, and K. M. Borgwardt. The graphlet spectrum. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 529–536, 2009. doi: 10.1145/1553374.1553443

[22] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, vol. 2002, pp. 315–22, 2002.

[23] O.-H. Kwon, T. Crnovrsanin, and K.-L. Ma. What would a graph look like in this layout? a machine learning approach to large graph visualization. *IEEE transactions on visualization and computer graphics*, 24(1):478–488, 2017. doi: 10.1109/TVCG.2017.2743858

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] P. Li, T. J. Hastie, and K. W. Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 287–296, 2006. doi: 10.1145/1150402.1150436

[26] L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[27] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[28] T. Milenković, V. Memišević, A. K. Ganesan, and N. Pržulj. Systems-level cancer gene identification from protein interaction network topology applied to melanogenesis-related functional genomics data. *Journal of the Royal Society Interface*, 7(44):423–437, 2010. doi: 10.1098/rsif.2009.0192

[29] T. Milenković and N. Pržulj. Uncovering biological network function via graphlet degree signatures. *Cancer informatics*, 6:CIN–S680, 2008.

[30] F. V. Paulovich, M. C. F. Oliveira, and R. Minghim. 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. doi: 10.1109/SIBGRAPI.2007.21

[31] N. Pezzotti, T. Höllt, B. Lelieveldt, E. Eisemann, and A. Vilanova. Hierarchical stochastic neighbor embedding. *Computer Graphics Forum*, 35(3):21–30, 2016. doi: 10.1111/cgf.12878

[32] N. Pezzotti, B. P. Lelieveldt, L. van der Maaten, T. Höllt, E. Eisemann, and A. Vilanova. Approximated and user steerable tsne for progressive visual analytics. *IEEE transactions on visualization and computer graphics*, 23(7):1739–1752, 2016. doi: 10.1109/TVCG.2016.2570755

[33] N. Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007. doi: 10.1093/bioinformatics/btl301

[34] N. Pržulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004. doi: 10.1093/bioinformatics/bth436

[35] P. E. Rauber, A. X. Falcão, and A. C. Telea. Visualizing time-dependent data using dynamic t-sne. In *EuroVis (Short Papers)*, pp. 73–77, 2016.

[36] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1-3):125–141, 2008. doi: 10.1007/s11263-007-0075-7

[37] R. A. Rossi, R. Zhou, and N. K. Ahmed. Estimation of graphlet counts in massive networks. *IEEE transactions on neural networks and learning systems*, 30(1):44–57, 2018. doi: 10.1109/TNNLS.2018.2826529

[38] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[39] I. Takigawa and H. Mamitsuka. Graph mining: procedure, application to drug discovery and recent advances. *Drug discovery today*, 18(1-2):50–57, 2013. doi: 10.1016/j.drudis.2012.07.016

[40] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[41] Y. Wang, Y. Wang, Y. Sun, L. Zhu, K. Lu, C.-W. Fu, M. Sedlmair, O. Deussen, and B. Chen. Revisiting stress majorization as a unified framework for interactive constrained graph visualization. *IEEE transactions on visualization and computer graphics*, 24(1):489–499, 2017. doi: 10.1109/TVCG.2017.2745919

[42] Y. Wang, Y. Wang, H. Zhang, Y. Sun, C.-W. Fu, M. Sedlmair, B. Chen, and O. Deussen. Structure-aware fisheye views for efficient large graph exploration. *IEEE transactions on visualization and computer graphics*, 25(1):566–575, 2018. doi: 10.1109/TVCG.2018.2864911

[43] M. Wattenberg, F. Viégas, and I. Johnson. How to use t-sne effectively. *Distill*, 2016. doi: 10.23915/distill.00002

[44] L. Zhang, Y. Han, Y. Yang, M. Song, S. Yan, and Q. Tian. Discovering discriminative graphlets for aerial image categories recognition. *IEEE*

Transactions on Image Processing, 22(12):5071–5084, 2013. doi: 10.1109/TIP.2013.2278465