Contents lists available at ScienceDirect

Visual Informatics

journal homepage: www.elsevier.com/locate/visinf

Visualizing large-scale high-dimensional data via hierarchical embedding of KNN graphs

Haiyang Zhu[a], Minfeng Zhu[a], Yingchaojie Feng[a], Deng Cai[a], Yuanzhe Hu[a], Shilong Wu[b,*], Xiangyang Wu[c], Wei Chen[a]

[a] State Key Lab of CAD&CG, Zhejiang University, Hangzhou, Zhejiang, China
[b] University of California Santa Cruz, San Francisco, USA
[c] Hangzhou Dianzi University, Hangzhou, Zhejiang, China

ARTICLE INFO

Article history:
Received 3 June 2021
Accepted 9 June 2021
Available online 26 June 2021

Keywords:
High-dimensional data visualization
KNN graph
Graph visualization

ABSTRACT

Visualizing intrinsic structures of high-dimensional data is an essential task in data analysis. Over the past decades, a large number of methods have been proposed. Among all solutions, one promising way for enabling effective visual exploration is to construct a k-nearest neighbor (KNN) graph and visualize the graph in a low-dimensional space. Yet, state-of-the-art methods such as the LargeVis still suffer from two main problems when applied to large-scale data: (1) they may produce unappealing visualizations due to the non-convexity of the cost function; (2) visualizing the KNN graph is still time-consuming. In this work, we propose a novel visualization algorithm that leverages a multi-level representation to achieve a high-quality graph layout and employs a cluster-based approximation scheme to accelerate the KNN graph layout. Experiments on various large-scale datasets indicate that our approach achieves a speedup by a factor of five for KNN graph visualization compared to LargeVis and yields aesthetically pleasing visualization results.

1. Introduction

Visualizing high-dimensional data has proven to be essential for medical health (Guo et al., 2020), life science (Mahfouz et al., 2015), and social network (Kwon et al., 2018; Chen et al., 2018), etc. Generally, the problem denotes the conversion of a high-dimensional dataset into a low-dimensional dataset. Analysts can study the data distribution and generate hypotheses about the input dataset.

Over the past decades, a large number of visualization methods for high-dimensional data have been proposed. t-SNE (Maaten and Hinton, 2008) is one of the most successful non-linear methods to visualize high-dimensional data with intrinsic nonlinear structures. However, the computational complexity of t-SNE scales quadratically with the number of data points. BH-SNE (Van Der Maaten, 2014) constructs a k-nearest neighbor (KNN) graph, and visualizes the graph in $O(N \log N)$ via quadtree. Recently, LargeVis (Tang et al., 2016) constructs an approximate KNN graph and then projects the graph using the negative sampling technique (Mikolov et al., 2013), which reduces the time complexity of KNN graph layout to $O(N)$. However, there remain two problems in visualizing large-scale high-dimensional data. First, starting from a random initialization, LargeVis is likely to produce unappealing visualizations due to the non-convex objective function, particularly for large-scale data. A better initialization is able to overcome the non-convex nature of LargeVis and generates an aesthetically pleasing visualization. Second, the computational complexity of existing algorithms is still not low. While LargeVis (Tang et al., 2016) greatly accelerates the KNN graph layout procedure, it is still time-consuming for large-scale data.

We base our solution for high-dimensional data visualization on the KNN graph and make several improvements against the issues mentioned above, including a multi-level representation to achieve aesthetically pleasing results and a gradient approximation scheme to speed up the visualization of the constructed KNN graph. In particular, we propose an efficient multi-level representation which constructs a series of hierarchical graphs with decreasing sizes in linear time and does not require additional weight computation for new graphs. The multi-level representation captures the global structure of the KNN graph. In this way, the sequence of generated graphs is recursively visualized by assigning positions from coarse to fine graphs, which may easily lead to an appropriate layout initialization and thereby

* Corresponding author.
E-mail addresses: hnsyzhy@zju.edu.cn (H. Zhu), minfeng_zhu@zju.edu.cn (M. Zhu), fycj@zju.edu.cn (Y. Feng), dengcai@cad.zju.edu.cn (D. Cai), cadhyz@zju.edu.cn (Y. Hu), swu97@ucsc.edu (S. Wu), wuxy@hdu.edu.cn (X. Wu), chenwei@cad.zju.edu.cn (W. Chen).

https://doi.org/10.1016/j.visinf.2021.06.002
2468-502X/

yields high-quality visualization. In addition, unlike previous approaches that visualize multi-level graphs separately, we treat a subgraph of the finest graph as a group. Then, the multi-level representation favors the effective reduction of the gradient approximation, that is, gradients of data points in a group (inherited from the multi-level structure) can be represented by the gradient of one representative. In this way, we assign the gradient from fine to coarse graphs and greatly simplify the gradient computation. Experimental results on various large-scale datasets demonstrate that the proposed graph layout algorithm for high-dimensional data visualization is five times faster than that of LargeVis with better visualization results on large-scale datasets.

In summary, the major contributions are as follows:

- a high-dimension data visualization scheme with a multi-level graph layout that is capable of producing aesthetically pleasing visualization.
- a gradient approximation scheme that accelerates the KNN graph layout procedure.

2. Related work

As a fundamental means for high-dimensional data analysis, visualization has been widely studied and applied (Sorzano et al., 2014; Ma and Maciejewski, 2020; Ma et al., 2018; Han et al., 2021). Existing successful visualization approaches come from the machine learning community and are usually classified into two categories: linear and nonlinear methods.

Linear methods project high-dimensional data based on a linear transformation. The distances among data points in high-dimensional space will be preserved in the low-dimensional space. Principal component analysis (PCA) (Jolliffe, 1986) is the most popular and widely used method, which preserves dimensions with the greatest variance. Likewise, the aim of Sammon mapping (Sammon, 1969) is to minimize the distance error between high- and low-dimensional data.

Nonlinear methods employ nonlinear distance metrics or local structures to capture manifold structures in the high-dimensional space. The goal of multidimensional scaling (MDS) (Kruskal, 1964) is to preserve pairwise distances between two spaces. Isomap (Tenenbaum et al., 2000) estimates the geodesic distance instead of Euclidean distance to minimize the pairwise distance error. Other nonlinear methods attempt to preserve the local structure: nearby points in high-dimensional space remain nearby in the low dimension space. The basic idea of locally linear embedding (LLE) (Saul and Roweis, 2003) is reconstructing data points with the linear combination of neighbors in high-dimensional space and minimizing the reconstruction error in low-dimensional space. Both Laplacian Eigenmap (Belkin and Niyogi, 2003) and locality preserving projections (LPP) (He and Niyogi, 2004) attempt to minimize the distance between nearby points.

Stochastic neighbor embedding-based approaches use probability rather than distance to measure the similarity among the data points. These approaches aim to minimize the Kullback–Leibler distance between two probability distributions in high-dimensional space and low-dimensional space. t-SNE (Maaten and Hinton, 2008; Han et al., 2019) shows its significant advantages in generating low-dimensional embedding. However, it is time-consuming to visualize large-scale datasets with $O(\mathrm{N}^2)$ computational complexity. To solve this issue, Maaten proposes Barnes–Hut-SNE (BH-SNE) (Van Der Maaten, 2014), which employs a KNN graph to estimate the large probability and visualize the graph in low-dimensional space in $O(\mathrm{N}\log\mathrm{N})$ by quadtrees. Largevis (Tang et al., 2016) utilizes an efficient algorithm to construct the KNN graph and speeds up the graph visualization by leveraging the negative sampling (Mikolov et al., 2013) and the edge sampling (Tang et al., 2015) techniques. LargeVis significantly reduces the computational complexity to linear. UMAP (McInnes et al., 2018) employs a suitable initialization for KNN graph visualization by spectral embedding. FIt-SNE (Linderman et al., 2019) approximates the gradient of each data point by fast Fourier transform, which reduces the computational complexity to linear. AtSNE (Fu et al., 2019) and t-SNE-CUDA (Chan et al., 2018) focus on accelerating t-SNE on the GPU platform. Although many methods have been developed to speed up t-SNE, they do not change the fact that one data point is repositioned at a time. We propose a cluster-based approximation scheme to treat a subgraph as a unit during high-dimensional data visualization based on a multi-level representation of the KNN graph.

The multi-level concept has been widely used to create a good visualization result. The multi-level representation is created by graph clustering (Meyerhenke et al., 2017), graph matching (Veldhuizen, 2007), or Monte Carlo process (HSNE) (Pezzotti et al., 2016). These methods have two limits when applied to dimensionality reduction: (1) the multi-level representation generation suffers from high computation cost; (2) the vertex or edge weights of coarse graphs need to be carefully defined and computed. Therefore, we propose a linear multi-level scheme that does not require additional weight computation.

3. Our approach

3.1. Preliminaries

Generally, the problem denotes the conversion of a high-dimensional dataset $X = \{x_1, x_2, \ldots, x_N\}, x_n \in \mathcal{R}^D$ into a low-dimensional dataset $Y = \{y_1, y_2, \ldots, y_N\}, y_n \in \mathcal{R}^2$ or $\mathcal{R}^3$. Then the low-dimensional representations are learned by minimizing the Kullback–Leibler divergences between the probability distributions of the high-dimensional data P and the low-dimensional data Q.

$$KL(P \parallel Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{1}$$

$$= \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \tag{2}$$

Note that the first part of this equation is a constant. Therefore, minimizing the Kullback–Leibler divergence is equal to maximize the following objective function:

$$\min KL(P \parallel Q) \Leftrightarrow \max \sum_i \sum_j \quad p_{ij} \log q_{ij} \tag{3}$$

The KNN graph is employed to approximate the probabilities between data points in the high-dimensional space. Since constructing exact KNN graphs is time-consuming, LargeVis proposes an efficient approximate KNN graph construction method. Mathematically, the probability distribution P is defined as follows:

$$p_{i|j} = \begin{cases} \dfrac{\exp\left(-d\left(x_i,x_j\right)^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-d\left(x_i,x_k\right)^2/2\sigma_i^2\right)} & \text{if } j \in NN_k\,(x_i) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} \tag{5}$$

where $d(x_i, x_j)$ is Euclidean distance between x_i and x_j, σ_i is the variance of Gaussian distribution on x_i, $NN_k(x_i)$ denotes the k-nearest neighbors of x_i, and N is the data size.

A heavy-tailed distribution is employed to measure the probability distribution Q in low-dimensional space:

$$q_{ij} = \frac{(1 + d(y_i, y_j)^2)^{-1}}{\sum_{k\neq l}(1 + d(y_i, y_j)^2)^{-1}} \tag{6}$$

Inspired by the negative sampling techniques (Mikolov et al., 2013), to compute the gradient of y_i, LargeVis randomly selects one connected point (positive sample) y_j in the KNN graph and M disconnected points (negative samples) $y_{j_k}, k = 1, 2, \ldots, M$. Thus, the objective function is reformulated as follows:

$$\max \sum_{i,j} p_{ij}[\log q_{ij} + \sum_{k=1}^{M} \gamma \log(1 - q_{ij_k})] \tag{7}$$

where γ is the parameter for balancing positive and negative weights. The gradient is given by:

$$dy_i = -\frac{2p_{ij}(y_i - y_j)}{1 + d(y_i, y_j)^2} + \sum_{k=1}^{M} \frac{2\gamma p_{ij}(y_i - y_{j_k})}{d(y_i, y_{j_k})^2(1 + d(y_i, y_{j_k})^2)} \tag{8}$$

LargeVis optimizes the objective function with the edge sampling method (Tang et al., 2015) by stochastic gradient descent. The edge sampling method randomly samples the positive sample (y_j) based on the probability p_{ij} to avoid the excessive gradient. The negative samples y_{j_k} are sampled by the negative sampling technique (Mikolov et al., 2013) according to the degree $d_{j_k} = \sum_l p_{j_k,l}$.

However, LargeVis suffers from two main drawbacks. First, LargeVis may produce unappealing visualizations for large-scale data due to the non-convexity of the cost function. LargeVis starts from an initial random configuration of data points and takes much time to converge when merging groups with the same label (see Fig. 3(a)). It is practically possible to generate aesthetically pleasing results if existing approaches can find an appropriate initialization. Second, it is time-consuming to compute the gradient for each data point to update the positions of all data points. Given two groups of data points (e.g., G_1 and G_2) with the same labels, we need to move at least $min(|G_1|, |G_2|)$ steps to merge two groups by updating the position of a data point at a time.

3.2. The key idea

Our solutions for accelerating LargeVis include two parts: a multi-level graph visualization scheme, and a gradient approximation scheme.

We first discuss how to generate aesthetically pleasing visualization with a better initialization. It is possible to find an initial layout of the global structure of the KNN graph first and then refine the layout later. We generate a multi-level representation to capture the global structure, e.g., a series of graphs from the KNN graph. Then, we iteratively refine the layout of the KNN graph from coarse to fine scales, where the initialization of a finer graph is directly derived from the final layout of a coarser graph. Second, we propose a gradient approximation scheme by treating a subgraph of the finest graph as a group and moving the group of similar data points together. We forward the gradient of one representative in the finest graph to the group in a coarse graph instead of computing the gradients of all data points. In this way, we can accelerate the KNN graph visualization procedure significantly (see Fig. 1).

3.3. Multi-level graph visualization

The multi-level concept has been widely used in graph visualization (Gajer and Kobourov, 2000; Hu, 2005). We propose an efficient multi-level scheme that constructs multi-level graphs in linear time without additional weight computation of new graphs. Our scheme includes two steps: graph coarsening and hierarchical refinement. The graph coarsening step generates a multi-level representation to capture the global structure of the KNN graph constructed by EFANNA (Fu and Cai, 2016). In the hierarchical refinement step, we progressively refine the layout of the generated graphs from coarse to fine.

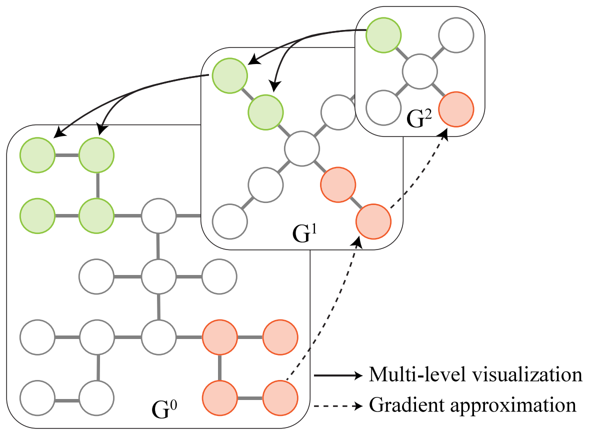

Fig. 1. An example of multi-level representation. For multi-level graph visualization, the initialization of a finer graph is derived from a coarser graph. We compute the gradient of one representative of G^0 and forward the gradient to the corresponding vertex in G^1 or G^2.

Graph Coarsening. Given a KNN graph $G^0 = (V^0, E^0)$ with a set of vertexes $V^0 = \{v_1^0, v_2^0, \ldots, v_N^0\}$ and a set of edges $E^0 \subset V^0 \times V^0$. In this case, each vertex represents a low-dimensional data point: $v_i^0 = \{y_i\}$. Each edge is a connection between two vertexes. We denote $e = (v_i^0, v_j^0) \in E^0$ if x_j belongs to the k-nearest neighbors of x_i: $x_j \in NN_K(x_i)$. The goal of graph coarsening is to generate a **multi-level representation**: a series of coarse graphs $G^0, G^1, G^2, \ldots, G^L$ with decreasing sizes, where G^L is the coarsest graph.

Given a graph $G^l = (V^l, E^l)$, we partition the vertex set of V^l into disjoint subsets and generate a coarser graph G^{l+1}. Each subset is collapsed into a new vertex in the graph G^{l+1}. For instance, if v_i^l and v_j^l are assigned into v_k^{l+1}, then $v_k^{l+1} = v_i^l \bigcup v_j^l$. In practice, we randomly select a vertex v_i^l and assign v_i^l and its k_{ml}-nearest neighbors into a new vertex of G^{l+1}. We employ the k_m-NN graph ($k_m \leq K$) instead of the KNN graph in graph coarsening because a vertex is more likely to share similar properties with its nearby neighbors instead of distant neighbors. We repeat the above process until all vertexes have been assigned. At last, we add edge (v_i^l, v_j^l) of E^l into E^{l+1}, if v_i^l and v_j^l are assigned to two vertexes of G^{l+1}. We stop graph coarsening when $|V^{l+1}| > \rho|V^l|, \rho = 0.8$. We choose $\rho = 0.8$ for two reasons. First, if ρ is very close to 1, the number of vertexes in G^{l+1} and G^l may be close. This can significantly increase the complexity of the multi-level algorithm. Second, if ρ is close to 0, the multi-level representation cannot capture high-level information. Based on the above consideration, we choose ρ to be 0.8 to keep the balance between computational efficiency and global structure extraction.

Hierarchical Refinement. For the coarsest graph G^L, we visualize the graph with random initialization. The optimal layout of the coarsest graph can be found at a low cost. Once the layout of a coarse graph G^l is generated, the initialization of a finer graph G^{l-1} is derived from G^l. The initial position of a vertex v_i^{l-1} of G^{l-1} is set to be the position of vertex v_j^l of G^l, if v_i^{l-1} is assigned to v_j^l in the graph coarsening step. Then, we recursively refine the layout from coarse to fine scales until the finest graph G^0 is done.

3.4. Cluster-based gradient computation

For the visualization of a graph G^l, the gradient of a group of data points $v = \{y_1, \ldots, y_n\}$ is computed as the average gradient of $y_1, \ldots, y_n$: $dv = (1/n)\sum_i dy_i$, where y_i is the low-dimensional

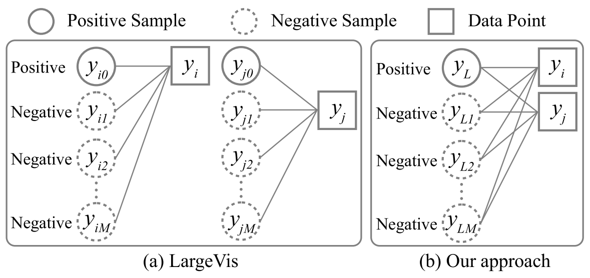

Fig. 2. An example of sharing positive and negative samples between two data points (y_i and y_j) in the same group. A solid circle represents a positive sample and a dashed circle encodes a negative sample. (a) LargeVis generates different positive and negative samples for y_i and y_j. (b) Alternatively, our approach shares the negative and positive samples between y_i and y_j.

position of the high-dimensional data x_i and dy_i is the gradient of y_i. However, it is time-consuming to sum over all gradients in $O(n\mathrm{M})$. Our basic idea is treating a subgraph of the finest graph as a group and sharing the gradient of one representative from fine to coarse scales if all gradients of the group are close to each other.

Given a vertex v which consists of a group of low-dimensional data points $\{y_1, y_2, \ldots, y_n\}$, we make the following assumptions to approximate the gradient. We first assume that data points in a group are similar to each other according to multi-level representation, meaning that they have a similar probability to other data points:

$$\mathrm{A1}: \quad p_{i,k} \approx p_{j,k}, \forall y_i, y_j \in v. \tag{9}$$

Then, as shown in Fig. 2(b), y_i and y_j can share positive and negative samples according to the sampling strategy. In addition, we assume that data points in a group are close to each other:

$$\mathrm{A2}: \quad y_i = y_j, \forall y_i, y_j \in v. \tag{10}$$

We share the positive y_L and negative samples $y_{L_k}, k = 1, 2, \ldots, \mathrm{M}$ in the gradient computation (A1). Then, according to the assumption (A2), we have: $d(y_i, y_L) = d(y_j, y_L)$, $d(y_i, y_{L_k}) = d(y_j, y_{L_k})$.

Therefore, y_i and y_j have similar gradients:

$$\begin{aligned} dy_i &= \frac{-2p_{iL}(y_i - y_L)}{1 + d(y_i, y_L)^2} + \sum_{k=1}^{M} \frac{2\gamma p_{iL}(y_i - y_{L_k})}{d(y_i, y_{L_k})^2(1 + d(y_i, y_{L_k})^2)} \\ &\approx \frac{-2p_{jL}(y_j - y_L)}{1 + d(y_j, y_L)^2} + \sum_{k=1}^{M} \frac{2\gamma p_{jL}(y_j - y_{L_k})}{d(y_j, y_{L_k})^2(1 + d(y_j, y_{L_k})^2)} \\ &= dy_j, \ \forall y_i, y_j \in v. \end{aligned} \tag{11}$$

The gradient of the vertex v can be regarded as the gradient of a random data point y_j in the group: $dv = \frac{1}{n}\sum_i dy_i \approx dy_j, \forall y_j \in v$. We are able to reduce the computation complexity from $O(n\mathrm{M})$ to $O(\mathrm{M})$. Computing the gradient once and assigning the gradient to other data points in the same group accelerates the optimization process significantly.

3.5. The computational complexity

The computational complexity of graph visualization includes graph coarsening and hierarchical refinement.

For graph coarsening, the worst case of creating G^{l+1} is accessing all vertexes and edges, yielding a computational complexity of $O(|V^l| + |E^l|)$. Let us assume that $|V^l| \leq 0.8|V^{l-1}|$ and $|E^l| \leq 0.8|E^{l-1}|$ for all $l = 1, \ldots, L$, then computational complexity of graph coarsening is $(|E^0| + |V^0|)(1 + \frac{4}{5} + \cdots + (\frac{4}{5})^{L-1}) \leq 5(|E^0| + |V^0|) \leq 5(\mathrm{K} + 1)\mathrm{N}$, where K is the size of the KNN graph. The computational complexity of creating all coarse graphs is $O(\mathrm{KN})$.

For hierarchical refinement, the gradient computation consists of $\mathrm{M} + 1$ distances and takes $O(\mathrm{M})$ time, where M is the number of negative samples. The number of iterations is usually proportional to the number of vertexes. For instance, the iteration number of visualizing G^l is $\mathrm{T}|V^l|$. The total computational complexity of hierarchical refinement is $\mathrm{T}|V^0|(1 + \frac{4}{5} + \cdots + (\frac{4}{5})^L)\mathrm{M} \leq 5\mathrm{T}|V^0|\mathrm{M} = 5\mathrm{TNM}$.

Therefore, the computational complexity of graph visualization is $O(\mathrm{KN+TNM})$, which is linear with the number of data points N. In our experiments, we set the number of iterations to be 500N.

4. Experiments

This section presents quantitative and qualitative results. All experiments are conducted on a single desktop PC with Intel Xeon E3 1245 CPU, 32GB memory, and Ubuntu 18.04 installed.

Datasets. The statistics for datasets are summarized in Table 1 (see Appendix for detail). Each image in the **MNIST** and Fashion-MNIST (**FMNIST**) datasets is considered as a 784-dimensional vector. For **CIFAR10** and **SVHN** datasets, we train powerful convolutional neural networks to extract features as a learned representation. We employ a text classification model, fastText (Joulin et al., 2016), to construct a 100-dimensional vector for each item in the **AG**'s News, **DBpedia**, Yahoo **Answers** and Amazon **Reviews** datasets. The **Reviews** dataset contains 3,000,000 reviews as well as the number of stars the user has given (from 1 to 5). The Common **Crawl** dataset consists of 2,000,000 pre-trained word vectors (Mikolov et al., 2018). We employ K-means to generate ten classes based on the high-dimensional vectors. We colorize data points according to their classes to facilitate the observation of the data distribution after dimensionality reduction.

Methods and Parameter Settings. We compare visualization results with that of BH-SNE (Van Der Maaten, 2014), LargeVis (Tang et al., 2016), UMAP (McInnes et al., 2018), and FIt-SNE (Linderman et al., 2019). For all methods, we construct a 100-NN graph as the input of graph visualization. For our method, we use the pre-set parameters of EFANNA for constructing the 100-NN graph. After a preliminary evaluation, the total number of iterations is 500N, and the parameter k_{ml} in the multi-level approach is set to be 3. For both approaches, the perplexity, γ, and the number of negative samples M are set to be 50, 7, and 5, respectively. We employ an early exaggeration technique used in t-SNE to find a better visualization. We set γ to be 1 for the finest graph and set γ to be 7 other graphs. Parameter sensitivity could be found in Appendix. We use the pre-set parameters for other methods.

Evaluation Metric. Following LargeVis (Tang et al., 2016), we adopt the k-NN classifier accuracy as the quality metric. We compute the nearest neighborhood classification accuracy based on 2D visualization result.

Table 1
Summary of datasets.

Dataset	Size	Dimension	Class number
MNIST	70,000	784	10
FMNIST	70,000	784	10
CIFAR10	60,000	1,024	10
AG	120,000	100	4
DBpedia	560,000	100	14
SVHN	630,420	256	10
Answers	1,400,000	100	10
Crawl	2,000,000	300	10
Reviews	3,000,000	100	5

Table 2
Performance comparison. We show the running time (in seconds) of KNN graph construction (GC), graph visualization (GV) and the entire layout process (Total), respectively. Unfilled items indicate the incapability of the corresponding algorithm caused by computational cost.

Dataset	Stage	MNIST	FMNIST	CIFAR10	AG	DBpedia	SVHN	Answers	Crawl	Reviews
BH-SNE	GC	3215	786	2950	**23**	**381**	122735	**1494**	(-)	(-)
UMAP	GC	**119**	**112**	**117**	111	632	**1139**	1777	(-)	(-)
FIt-SNE	GC	636	511	687	121	911	3118	3214	11975	6108
LargeVis	GC	624	475	790	137	1475	5954	6647	19292	12121
Ours	GC	311	291	337	217	1193	2993	2976	**9056**	**5621**
BH-SNE	GV	921	1008	839	1767	14508	17519	40982	(-)	(-)
UMAP	GV	178	183	153	302	13331	3797	13720	(-)	(-)
FIt-SNE	GV	**74**	**79**	**68**	**116**	1421	2045	6166	8595	13802
LargeVis	GV	2505	2524	2279	3155	11375	12558	29159	42306	66723
Ours	GV	103	115	96	169	**1101**	**1839**	**3213**	**6491**	**7148**
BH-SNE	Total	4135	1793	3789	1790	14889	140253	42476	(-)	(-)
UMAP	Total	**297**	**295**	**270**	413	13963	4936	15497	(-)	(-)
FIt-SNE	Total	710	590	755	**237**	2333	5165	9379	20570	19910
LargeVis	Total	3129	3000	3069	3292	12850	18512	35806	61598	78845
Ours	Total	414	406	433	386	**2294**	**4832**	**6189**	**15547**	**12769**

Table 3
Visualization quality. We report the accuracy of the *10*-NN classifier on 2D visualization results.

Method	MNIST	FMNIST	CIFAR10	AG	DBpedia	SVHN	Answers	Crawl	Reviews
BH-SNE	**97.20**	**82.51**	**96.73**	99.34	99.23	97.08	49.45	(-)	(-)
UMAP	96.26	76.32	96.39	**99.54**	**99.96**	**97.17**	76.64	(-)	(-)
FIt-SNE	93.15	76.92	94.95	99.45	95.28	96.29	49.26	33.80	26.06
LargeVis	96.71	80.49	96.63	99.52	99.95	97.00	77.70	**69.00**	**60.44**
Ours	96.67	79.13	96.53	99.49	**99.96**	97.03	**77.83**	67.99	59.83

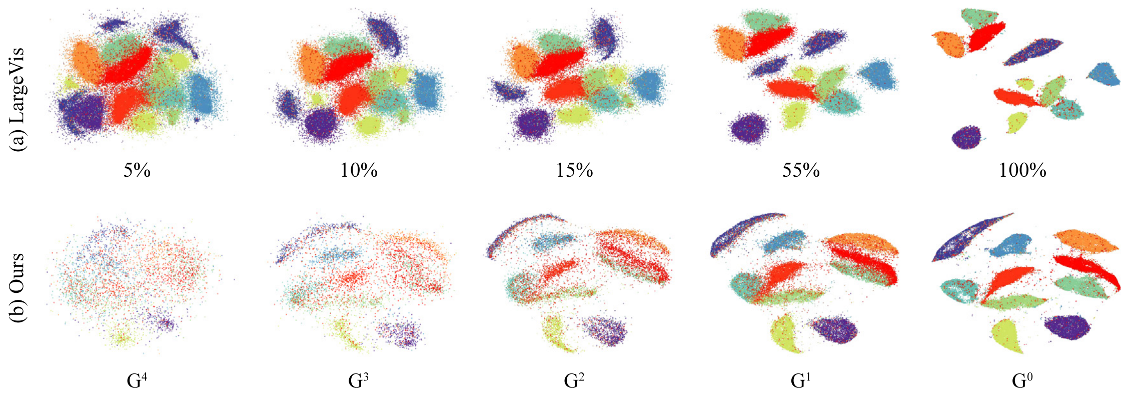

Fig. 3. The optimization process of LargeVis and our method on the MNIST dataset. LargeVis costs much time in merging groups of the same category. Our method generates a better initialization in the early stages.

4.1. Running time

Table 2 shows the running time on different datasets. We report the CPU time of KNN graph construction (GC), graph visualization (GV) and the total process (Total), respectively.

For the GC stage, our method is about twice as fast as LargeVis on all datasets except AG and DBpedia, thanks to the usage of EFANNA. The UMAP is the fastest method in KNN graph construction and similarity computation. However, UMAP consumes more memory than ours. For instance, UMAP requires 32 GB on the Answers dataset (18 GB for ours). For our method, we find that the GC stage sometimes takes more time than the graph visualization part. A future direction is to design a more efficient graph construction algorithm. The GV stage is the performance bottleneck of LargeVis and BH-SNE. Since we employ a gradient approximation scheme to reduce the total iteration number, our method achieves at least five times acceleration compared to LargeVis. Though FIt-SNE is the fastest method on small datasets, it suffers from high computational costs on large datasets, such as Answers. For the total time, our method achieves at least three times acceleration over LargeVis. On small dataset such as MNIST, LargeVis takes about 3129 s totally while our method takes only 414 s. For large-scale datasets such as Reviews, our method is five times faster than LargeVis. UMAP is slightly faster than ours on small datasets. However, our method achieves two times acceleration compared to UMAP on the Answers dataset. In summary, with the linear time complexity of graph coarsening and hierarchical refinement, the proposed method is more efficient than other methods when applied to large datasets.

4.2. Visualization quality

Table 3 evaluates the visualization quality by 10-NN classifier. BH-SNE with default learning rate is not stable when visualizing the approximate KNN graph. The performance of LargeVis and our method is stable on all datasets due to the edge sampling method. For the FMNIST dataset, the accuracy of our method is a litter lower. We conjecture the reason is that the multi-level approach cannot capture a very accurate structure of the KNN graph for this difficult dataset. Besides, our method achieves higher accuracy than UMAP on the FMNIST (2.8%) and Answers (1.2%) datasets. The visualization of FIt-SNE is arguable of lower

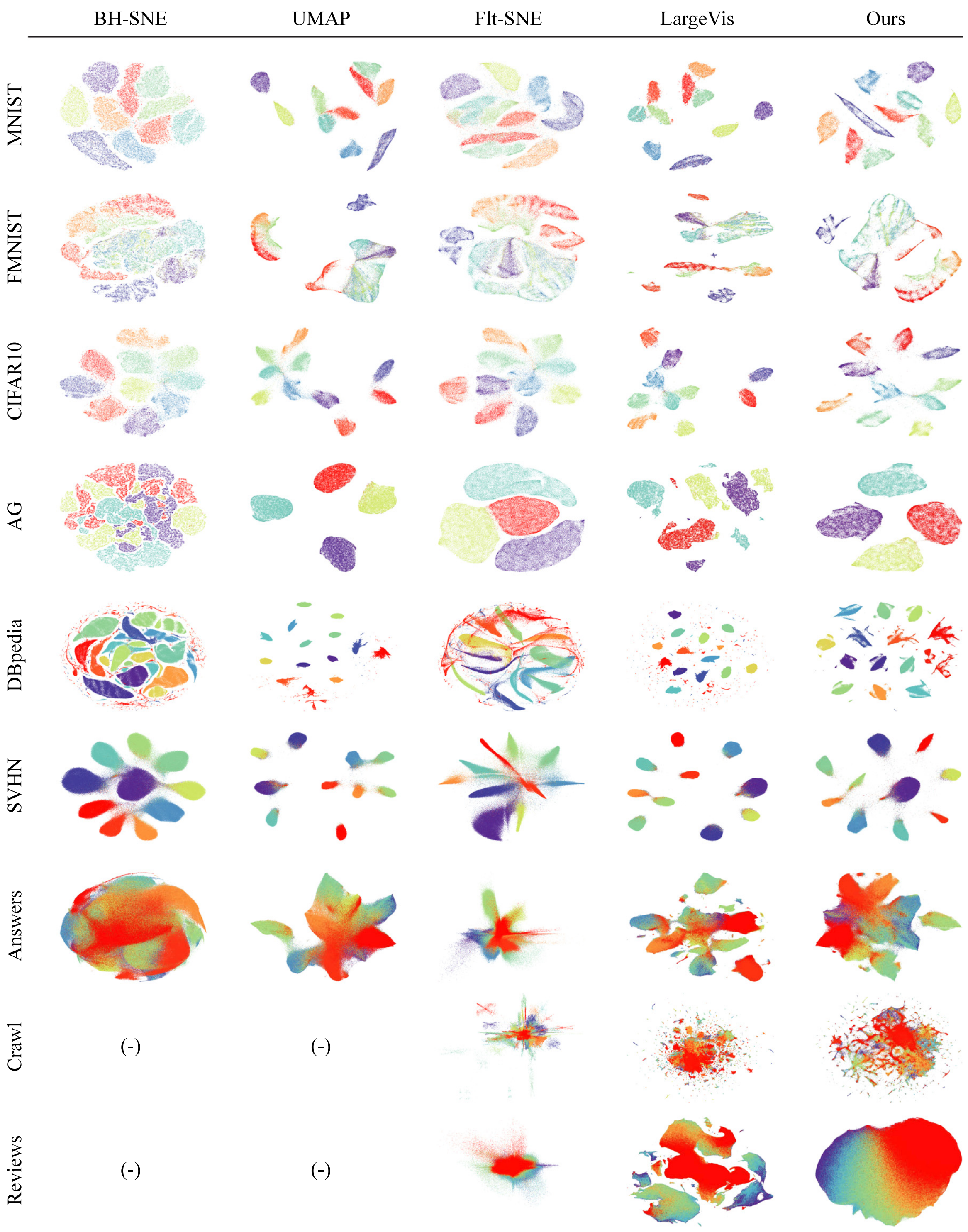

Fig. 4. A comparison of several visualization algorithms. The corresponding label or the category learned by K-means based on high-dimensional vectors is encoded in colors.

quality than other methods. The 10-NN accuracy gap between BH-SNE, LargeVis, and our method is very small, indicating that our method achieves a comparable visualization quality.

4.3. Visualization results

Fig. 3 shows the process of graph visualization on the MNIST dataset. For LargeVis, the intermediate result of the graph visualization process is visualized in a two-dimensional space. The number indicates the process of graph visualization. In Fig. 3, LargeVis generates groups at a very early stage and converges slowly when merging groups with the same label. We also show the visualization result of multi-level graphs (from G^4 to G^0) of our method. It is noticeable that our method leads to a better initialization and achieves a comparable result.

Fig. 4 compares the obtained visualization results, where rows indicate datasets and columns indicate visualization methods. The color of each dot represents the corresponding label. The visualization results mainly differ in cluster size and neighborhood preservation. For datasets with clear category information, the visualization results are meaningful and similar data points are projected closely. For instance, ten categories of data points are well separated on CIFAR10 and SVHN datasets. The word embedding of the Crawl dataset is learned with unsupervised learning. Thus, there is no obvious cluster structure in the visualization result. All methods produce comparable results on small datasets. However, when dealing with large datasets, methods with random initialization tend to converge to local minima and fail to persevere nearest neighbors. FIt-SNE produces visually inferior results on the Answers and Reviews dataset. For the AG and DBpedia datasets, ours generates a better visualization than BH-SNE and LargeVis. The data points of the same categories are projected closely, thanks to a good initialization and iterative refinement. Another interesting observation is that LargeVis falls into the local optima on the Reviews dataset. Instead, our results seem to be more reasonable than LargeVis. The visualization of the Reviews dataset is shown continuously from red (one star) to purple (five stars). In summary, our approach produces the same or better visualization results with more discriminative clusters.

5. Conclusions

In this paper, we present an efficient high-dimensional data visualization algorithm for large-scale data. In the future, we seek to implement a GPU version by exploiting parallelism. We plan to approximate the probability in high-dimensional space with smaller graphs (e.g., 10-NN graphs). In addition, a more effective multi-level representation is needed to achieve a better visualization quality.

Ethical approval

The study does not involve human subjects. All data used in the study are taken from public databases that were published in the past.

Author contributions

Haiyang Zhu: Conceptualization. Minfeng Zhu: Writing- Original draft. Yingchaojie Feng: Writing- Reviewing and Editing. Deng Cai: Supervision. Yuanzhe Hu: Software. Shilong Wu: Validation. Xiangyang Wu: Supervision. Wei Chen: Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This paper is supported by National Natural Science Foundation of China (61972122, 61772456).

Appendix

A.1. Datasets

1. **MNIST:** The MNIST[1] dataset includes 70,000 gray scale handwriting digital images. Each image contains 28 × 28 = 784 pixels and belongs to one of ten numbers from 0 to 9. Each image is considered as a 784-dimensional data point.
2. **FMNIST:** The fashion-MNIST[2] is a dataset consisting of 70,000 gray-scale images with labels from ten fashion product categories. It shares the same image size as the MNIST dataset.
3. **CIFAR10:** The CIFAR10[3] dataset includes 60,000 color images with 32 × 32 pixels. All images are labeled with ten classes. There are 6000 images in each class. We extract features using a powerful convolutional neural network, which consists of seven modules, each of which contains a 3 × 3 convolution layer followed by batch normalization and a ReLU non-linearity. We add max-pooling over 2 × 2 patches after every two convolution layers and after the seventh convolution layer. The last layer output with 1024 neurons is fed into a softmax classifier for ten categories. We extract the output of the last layer as a 1024-dimensional feature vector for each image.
4. **AG:** The AG's news corpus[4] includes 496,835 categorized news from more than 2000 sources. We use four largest categories with 30,000 articles of each category. Each article is represented by a 100-dimensional vector trained by a text classification model (fastText) (Joulin et al., 2016), which achieves 92.4% accuracy on the test dataset.
5. **DBPedia:** The DBPedia[5] is a public data infrastructure for structured information from Wikipedia. The DBPedia dataset consists of 560,000 Wikipedia articles with 14 classes. We employ the fastText to construct a 100-dimensional vector for each article. The classification performance on the test dataset is 98.6%.
6. **SVHN:** The street view house numbers (SVHN) dataset[6] is a real-world house number image dataset obtained from Google Street View images. We use all 630,420 color images in character level format where digits are resized to a resolution of 32 × 32 pixels. The SVHN dataset is much harder than MNIST. Therefore, We employ a pre-trained deep neural network to extract better representations for visualization. The neural network architecture contains seven convolution layers, each of which is a 3 × 3 convolution with batch normalization, ReLU activation, and dropout of 0.3. We add max-pooling over 2 × 2 patches after every two convolution layers and after the seventh convolution layer. The network then has a final linear layer followed by a softmax for classification. We extract 256-dimensional output of the last convolution layer as a learned representation.
7. **Answers:** We employ the text classification dataset from Yahoo! Answers Comprehensive Questions and Answers version 1.0.[7] The Yahoo Answers dataset consists of ten classes. Each class includes 140,000 questions, including question title, question content, and best answer. The feature vector of each question is also extracted by fastText whose test accuracy is 72.4%.
8. **Crawl:** The Crawl dataset consists of 2,000,000 word vectors trained on Common Crawl using unsupervised fastText. Each word is represented by a 300-dimensional vector. The distance between two vectors is an effective measurement for semantic similarity. We employ K-means to generate ten classes based on the high-dimensional vectors.

[1] http://yann.lecun.com/exdb/mnist/.

[2] https://github.com/zalandoresearch/fashion-mnist.

[3] https://www.cs.toronto.edu/~kriz/cifar.html.

[4] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html.

[5] https://wiki.dbpedia.org/develop/datasets.

[6] http://ufldl.stanford.edu/housenumbers/.

[7] https://webscope.sandbox.yahoo.com/.

Table 4
The 10-NN classifier accuracy with respect to the size of the k_{ml}-nearest neighbors in the multi-level approach.

k_{ml}	1	3	10	50	100
MNIST	96.55	**96.58**	96.44	96.56	96.28
FMNIST	79.47	79.10	79.10	79.14	**79.30**
CIFAR10	96.58	**96.59**	96.55	96.34	96.37
AG	99.46	**99.54**	99.47	99.45	99.51
DBPedia	99.95	99.95	99.95	**99.96**	**99.96**
SVHN	97.03	**97.10**	97.06	97.03	**97.10**
Answers	**77.87**	77.86	77.86	77.70	77.26
Crawl	68.04	68.07	**68.22**	68.07	68.19
Reviews	59.88	59.88	**59.89**	59.68	59.63

Table 5
The 10-NN classifier accuracy with respect to the number of negative samples M.

M	1	3	5	7	10
MNIST	96.32	96.40	96.62	96.59	**96.76**
FMNIST	76.17	78.50	79.22	79.57	**80.01**
CIFAR10	96.46	96.44	**96.47**	96.46	96.07
AG	99.44	99.49	**99.53**	99.49	99.52
DBPedia	**99.96**	**99.96**	99.95	99.94	99.94
SVHN	**97.18**	97.11	97.05	97.05	96.96
Answers	77.42	77.73	**77.82**	77.48	77.64
Crawl	65.50	67.56	67.90	68.03	**68.30**
Reviews	59.82	59.83	59.82	**59.93**	59.87

Table 6
The 10-NN classifier accuracy with respect to the iteration number.

TN	100N	300N	500N	1000N	1500N
MNIST	96.28	96.38	**96.61**	96.44	96.50
FMNIST	77.25	79.05	79.25	**79.73**	79.43
CIFAR10	96.02	96.46	96.50	**96.67**	96.46
AG	99.48	**99.56**	99.54	99.54	99.48
DBPedia	99.91	**99.95**	**99.95**	99.94	**99.95**
SVHN	96.95	97.05	**97.10**	97.05	97.06
Answers	76.39	77.50	**77.84**	77.68	77.56
Crawl	63.71	67.06	67.92	**68.58**	64.78
Reviews	59.77	59.90	59.87	**60.04**	59.95

9. **Reviews:** The Amazon reviews dataset, which is obtained from the Stanford Network Analysis Project (Leskovec and Krevl, 2014), contains 3,000,000 reviews as well as the number of stars the user has given (from 1 to 5). The fastText model is employed to represent each review with a 100-dimensional vector. The test accuracy of the model is 60.3% on the Amazon reviews dataset.

A.2. Parameter sensitivity

We report the 10-NN classifier accuracy with respect to the parameters in our approach.

Table 4 shows the 10-NN classifier accuracy with respect to the k_{ml}-nearest neighbors used to construct the multi-level representation. We find that the accuracy keeps stable for all k_{ml} on almost all dataset, and small k_{ml} achieves a little higher accuracy. It is more reasonable to merge a vertex with its close neighbors instead of distant neighbors. Therefore, we employ 3-nearest neighbors to capture global structure in our multi-level approach.

Table 5 shows the 10-NN classifier accuracy with respect to the number of negative samples M. The accuracy keeps stable on almost all dataset when M becomes large enough (e.g., 5). We choose $M = 5$ to keep the balance between computational efficiency and visualization quality.

Table 6 lists the 10-NN classifier accuracy with respect to the iteration number. When the iteration number increase, the accuracy converges very fast. The accuracy becomes very stable when the iteration number is large than 500N.

Table 7
The 10-NN classifier accuracy with respect to the parameter γ.

γ	0.1	1	5	7	10
MNIST	95.53	96.46	96.52	**96.69**	96.54
FMNIST	73.66	77.22	**79.13**	79.01	79.46
CIFAR10	95.36	96.36	96.39	**96.49**	96.48
AG	**99.51**	99.46	99.49	99.46	99.47
DBPedia	99.94	**99.95**	99.94	**99.95**	99.94
SVHN	**97.09**	97.08	97.04	97.05	97.07
Answers	72.92	77.05	77.66	77.73	**77.86**
Crawl	54.26	65.38	67.64	**68.01**	67.97
Reviews	59.85	59.82	59.84	59.86	**59.90**

Table 7 reports the 10-NN classifier accuracy with varied γ. We can see that the accuracy drops when γ is too small. Therefore, we set the parameter γ to be 7.

References

Belkin, M., Niyogi, P., 2003. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. 15 (6), 1373–1396.

Chan, D.M., Rao, R., Huang, F., Canny, J.F., 2018. T-SNE-CUDA: GPU-accelerated t-SNE and its applications to modern data. In: 30th International Symposium on Computer Architecture and High Performance Computing. IEEE, pp. 330–338.

Chen, W., Guo, F., Han, D., Pan, J., Nie, X., Xia, J., Zhang, X., 2018. Structure-based suggestive exploration: a new approach for effective exploration of large networks. IEEE Trans. Vis. Comput. Graphics 25 (1), 555–565.

Fu, C., Cai, D., 2016. Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph. ArXiv Preprint arXiv:1609.07228.

Fu, C., Zhang, Y., Cai, D., Ren, X., 2019. Atsne: Efficient and robust visualization on GPU through hierarchical optimization. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, pp. 176–186.

Gajer, P., Kobourov, S.G., 2000. GRIP: Graph drawing with intelligent placement. In: International Symposium on Graph Drawing. Springer, pp. 222–228.

Guo, R., Fujiwara, T., Li, Y., Lima, K.M., Sen, S., Tran, N.K., Ma, K.-L., 2020. Comparative visual analytics for assessing medical records with sequence embedding. Vis. Inform. 4 (2), 72–85.

Han, D., Pan, J., Guo, F., Luo, X., Wu, Y., Zheng, W., Chen, W., 2019. RankBrushers: interactive analysis of temporal ranking ensembles. J. Visual. 22 (6), 1241–1255.

Han, D., Pan, J., Zhao, X., Chen, W., 2021. Netv.js: A web-based library for high-efficiency visualization of large-scale graphs and networks. Vis. Inform. 5 (1), 61–66.

He, X., Niyogi, P., 2004. Locality preserving projections. In: Advances in Neural Information Processing Systems. pp. 153–160.

Hu, Y., 2005. Efficient, high-quality force-directed graph drawing. Math. J. 10 (1), 37–71.

Jolliffe, I.T., 1986. Principal component analysis and factor analysis. In: Principal Component Analysis. Springer New York, New York, NY, pp. 115–128. http://dx.doi.org/10.1007/978-1-4757-1904-8_7.

Joulin, A., Grave, E., Bojanowski, P., Mikolov, T., 2016. Bag of tricks for efficient text classification. ArXiv Preprint arXiv:1607.01759.

Kruskal, J.B., 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29 (1), 1–27.

Kwon, O., Crnovrsanin, T., Ma, K.L., 2018. What would a graph look like in this layout? A machine learning approach to large graph visualization. IEEE Trans. Vis. Comput. Graphics 24 (1), 478–488. http://dx.doi.org/10.1109/TVCG.2017.2743858.

Leskovec, J., Krevl, A., 2014. SNAP datasets: Stanford large network dataset collection. http://snap.stanford.edu/data.

Linderman, G.C., Rachh, M., Hoskins, J.G., Steinerberger, S., Kluger, Y., 2019. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. Nat. Methods 16 (3), 243.

Ma, Y., Maciejewski, R., 2020. Visual analysis of class separations with locally linear segments. IEEE Trans. Vis. Comput. Graphics 27 (1), 241–253.

Ma, Y., Tung, A.K., Wang, W., Gao, X., Pan, Z., Chen, W., 2018. Scatternet: A deep subjective similarity model for visual analysis of scatterplots. IEEE Trans. Vis. Comput. Graphics 26 (3), 1562–1576.

Maaten, L.v.d., Hinton, G., 2008. Visualizing data using t-SNE. J. Mach. Learn. Res. 9 (Nov), 2579–2605.

Mahfouz, A., van de Giessen, M., van der Maaten, L., Huisman, S., Reinders, M., Hawrylycz, M.J., Lelieveldt, B.P., 2015. Visualizing the spatial gene expression organization in the brain through non-linear similarity embeddings. Methods 73, 79–89.

McInnes, L., Healy, J., Melville, J., 2018. UMAP: Uniform manifold approximation and projection for dimension reduction. ArXiv E-Prints, arXiv:1802.03426.

Meyerhenke, H., Nöllenburg, M., Schulz, C., 2017. Drawing large graphs by multilevel maxent-stress optimization. IEEE Trans. Vis. Comput. Graphics 24 (5), 1814–1827.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A., 2018. Advances in Pre-Training Distributed Word Representations. In: Proceedings of the International Conference on Language Resources and Evaluation.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. pp. 3111–3119.

Pezzotti, N., Höllt, T., Lelieveldt, B., Eisemann, E., Vilanova, A., 2016. Hierarchical stochastic neighbor embedding. Comput. Graph. Forum 35 (3), 21–30.

Sammon, J.W., 1969. A nonlinear mapping for data structure analysis. IEEE Trans. Comput. C-18 (5), 401–409. http://dx.doi.org/10.1109/T-C.1969.222678.

Saul, L.K., Roweis, S.T., 2003. Think globally, fit locally: unsupervised learning of low dimensional manifolds. J. Mach. Learn. Res. 4 (Jun), 119–155.

Sorzano, C.O.S., Vargas, J., Montano, A.P., 2014. A survey of dimensionality reduction techniques. ArXiv Preprint arXiv:1403.2877.

Tang, J., Liu, J., Zhang, M., Mei, Q., 2016. Visualizing large-scale and high-dimensional data. In: Proceedings of the 25th International Conference on World Wide Web, 2016, pp. 287–297.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q., 2015. Line: Large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, pp. 1067–1077.

Tenenbaum, J.B., De Silva, V., Langford, J.C., 2000. A global geometric framework for nonlinear dimensionality reduction. Science 290 (5500), 2319–2323.

Van Der Maaten, L., 2014. Accelerating t-SNE using tree-based algorithms. J. Mach. Learn. Res. 15 (1), 3221–3245.

Veldhuizen, T.L., 2007. Dynamic multilevel graph visualization. ArXiv Preprint arXiv:0712.1549.