EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Hertie Institute for AI and Brain Health, Data Science for Vision Research, Philipp Berens
Graduate School of Neural Information Processing
University of Tübingen

Graph Layouts and Contrastive Learning using Neighbour Embedding Algorithms

Alica Leonie Guzmán
supervised by Dr. Dmitry Kobak

Abstract. Graphs are an ubiquitous form of data, used to describe for instance relationships in a social network or citations between papers. To obtain a reasonably good embedding of a graph in two dimensions, many graph layout methods have evolved. One big branch of those methods are force-directed methods. In this work we present Graph t-SNE as a new force-directed dimensionality reduction method that enables graph embeddings of comparable quality to existing methods. Therefore, we used an open source library for t-SNE for all our implementations and show embeddings of five distinct benchmark datasets which can be obtained without the need to develop a novel algorithm. We investigated the framework with Laplacian Eigenmaps initialization, with altering the exaggeration value and varying the degree of freedom for the low-dimensional similarity t-kernel. We also optimized embeddings based on additionally available node feature information. Finally, we combined feature and edge information aiming to improve our graph layouts. Therefore, we introduce Contrastive Graph t-SNE, which has similarities to a parametric implementation of t-SNE. Positive pairs correspond to features of connected node pairs and negative pairs to two distinct node features. For the datasets investigated here, this method does not improve the accuracies of the layouts, however the method remains promising for datasets where node features encode more useful neighbourhood information.

Duration: 05-12-22 - 06-03-23

Contents

1 Introduction

Graph data is a useful and structured representation of data describing relationships of any kind. Among many others, graphs can describe social networks, traffic networks and knowledge networks.

Data visualization is a useful tool for exploration and one aim when processing graph data is to enable a visualization in two dimensions. When finding a representation of relationships the most obvious 2D-representation is a node-link diagram: 'nodes' referring to the 2D-positions and 'links' corresponding to the edges or connections between them.
One can now easily think of the arrangement of three nodes with two certain links in two dimensions, and already get an idea of the upcoming problem: the representation is ambiguous since the nodes can be placed in very different configurations. So how do we find a good 2D-visualization?

We can uniquely represent a graph by a matrix, the so called adjacency matrix (Bondy and Murty 1976). Using this as starting point, many different techniques aim finding a good representation of a graph in two dimension, which is also called graph layout or graph embedding.
With a growing number of graph problems since the mid-20th century, there has been a lot of

research on how to find those 2D-visualizations of especially large graphs (Hu and Shi 2015). A short overview of these methods will be given in the theory part of this report. Many of them are so called force-directed methods which optimize the graph embedding by assigning forces to the nodes and finding an equilibrium where the overall sum of forces is minimized.
One can separate the force-directed methods in two categories: distance-preserving and neighbourhood-preserving methods (Kruiger et al. 2017). While distance-preserving methods aim to keep the distances between all pairs of points from the high-dimensional data in the low-dimensional representation, neighbourhood-preserving methods aim to keep the closest neighbours, having the shortest distance in the high-dimensional space. Because distance-preserving methods require to compute all distances in the graph, they get infeasible for a high node number.

Two neighbourhood-preserving methods have recently been shown to reach high performances in graph embeddings: tsNET (Kruiger et al. 2017) and t-FDP (Zhong et al. 2023). These two methods are related to t-SNE (Maaten and Hinton 2008), which is a dimensionality reduction method to preprocess and visualize data. Since these methods develop their own algorithm, an easy alternative could be to directly obtain graph layouts by t-SNE.

To further develop this idea, in this work, we used t-SNE under the name Graph t-SNE to obtain graph layouts of five distinct benchmark datasets. Apart from that, we also obtained embeddings using t-SNE of node feature data.
For evaluation we investigated two metrics of our embeddings: the neighbourhood-preservation metric NN recall and the kNN accuracy.
Subsequently, we investigated the framework to improve the obtained embeddings. We initialized t-SNE with Laplacian Eigenmaps (Belkin and Niyogi 2003), we altered the late exaggeration value (Böhm et al. 2022) in the t-SNE optimization and varied the degree of freedom for the low-dimensional similarity t-kernel (Kobak et al. 2020). All of those are implemented in the open source library `openTSNE` (Poličar et al. 2019), which we used for all our experiments. Although we did not find any considerably enhanced embeddings, we obtained embeddings of two graphs having similarities to those obtained with tsNET and t-FDP from Kruiger et al. 2017 and Zhong et al. 2023, respectively.
To use additional node feature information we developed Contrastive Graph t-SNE, a framework based on Contrastive Neighbour Embedding (Damrich et al. 2022) since Graph Contrastive Learning and Graph Neural Networks have recently shown to perform well in learning graph representations (Pei et al. 2020; Wang et al. 2022; Xie et al. 2023; Zhang et al. 2022). Although our contrastive approach did not enhance the graph layouts for the datasets investigated here, it still remains promising for further datasets, where node features encode more useful neighbourhood information.

2 Background

In the following chapter I will briefly explain the theoretical background and give computational details of my work.

2.1 t-SNE

In 2008, van der Maaten and Hinton introduced t-distributed Stochastic Neighbour Embedding (t-SNE) (Maaten and Hinton 2008), a Dimensionality Reduction (DR) technique to visualize high-dimenional data in a two dimensional map. It is a variation of Stochastic Neighbour Embedding (SNE) (Hinton and Roweis 2002), taking symmetric SNE and introducing a t-distributed kernel for low-dimensional similarities. Both methods make use of so called similarities, also called affinities, in both high and low dimensions. These give a measure of how similar two data points are. In contrast to linear DR techniques such as Classical Multidimensional Scaling (MDS) (Torgerson 1952) or Principal Component Analysis (PCA) (Hotelling 1933), t-SNE belongs to the family of nonlinear DR techniques, which enables to hold similar points close together in the low-dimensional representation and thus is neighbourhood-preserving. Next to t-SNE, nowadays also the nonlinear DR method UMAP (McInnes et al. 2018) is widely used. This is very differently motivated (c.f. Böhm n.d.), but it has been shown that it can be seen as negative sampling applied to the t-SNE loss function (Damrich et al. 2022).

In this work, we will use t-SNE for graph layouts. This is further described in Section 3.1. Here, I will first introduce the general concept of t-SNE for dimensionality reduction.

For the high-dimensional data points x_i with $i = 1, \dots, N$, t-SNE measures similarities by symmetrized, normalized affinities $p_\text{ij} = \frac{p_\text{i|j}+p_\text{j|i}}{2N}$ in the high-dimensional space. These high-dimensional affinities are conditional probabilities that follow a Gaussian distribution centered around data point x_i with variance σ_i (Maaten and Hinton 2008), such that:

$$p_\text{j|i} = \frac{\exp(-\frac{||x_\text{i}-x_\text{j}||}{2\sigma_\text{i}^2})}{\sum_\text{k\neq i}\exp(-\frac{||x_\text{i}-x_\text{k}||}{2\sigma_\text{i}^2})}. \tag{2.1}$$

All $p_\text{i|i}$ are set to zero.
The variance σ_i is found by producing a distribution $p_\text{j|i}$ which yields a fixed user-specified perplexity Perp $= 2^H$ with $H = -\sum_\text{j} p_\text{j|i}\log_2(p_\text{j|i})$ the entropy of the distribution.
One can see that the distance is inverse proportional to the similarity, such that the lower the distance between two data points, the more similar they are and thus, the higher the similarity/affinity.
To ease computations, a common trick is to approximate these high-dimensional conditional probabilities by a uniform distribution assigning $\frac{1}{k}$ to the k nearest neighbours of each data point and 0 to all other points.

To find a low-dimensional representation y_i with $i = 1, \dots, N$, the low-dimensional counterparts $q_\text{ij} = q_\text{ji}$ are defined as

$$q_\text{ij} = \frac{(1+||y_\text{i}-y_\text{j}||^2)^{-1}}{\sum_\text{k\neq l}(1+||y_\text{k}-y_\text{l}||^2)^{-1}}, \tag{2.2}$$

using a student t-distribution with one degree of freedom. All $q_\text{i|i}$ are set to zero. This choice of probability distribution for the low-dimensional space avoids the 'crowding problem' and other optimization problems of the earlier version SNE (Hinton and Roweis 2002, Maaten and Hinton 2008). The crowding problem is described in the following: to prevent a far displacement of moderately-distant map points because of the conversion to lower dimensionality, SNE exerts a very small attractive force to those points. This results in a crowded embedding (Maaten and Hinton 2008).

To find the low-dimensional representation t-SNE minimizes the Kullback-Leibler divergence between the two probability distributions. It is given by

$$C_\text{KL} = \sum_\text{i}\sum_\text{j} p_\text{ij}\log(\frac{p_\text{ij}}{q_\text{ij}}). \tag{2.3}$$

The low-dimensional coordinates are optimized by minimizing this cost function by gradient descent. Up to a constant factor, with $w_\text{ij} = (1+||y_\text{i}-y_\text{j}||^2)^{-1}$ and $\frac{\partial w_\text{ij}}{\partial y_\text{i}} = -2w_\text{ij}^2(y_\text{i}-y_\text{j})$, the gradient can be written as

$$\frac{\partial C_{\mathrm{KL}}}{\partial y_{\mathrm{i}}} \sim \sum_{\mathrm{j}} p_{\mathrm{ij}} N w_{\mathrm{ij}} (y_{\mathrm{i}} - y_{\mathrm{j}}) - \frac{N}{\rho \sum_{\mathrm{k} \neq \mathrm{l}} w_{\mathrm{kl}}} \sum_{\mathrm{j}} w_{\mathrm{ij}}^2 (y_{\mathrm{i}} - y_{\mathrm{j}}), \tag{2.4}$$

where $\rho = 1$ is the exaggeration value (Böhm et al. 2022).[1]
The first term describes the attractive forces between the data points, the second term describes the repulsive forces.
Making use of uniform high-dimensional similarities, we obtain many $p_{\mathrm{ij}} = 0$, which facilitates computation, since many entries of the first sum in Equation (2.4) are equal to zero.
A standard optimization trick known as *early exaggeration* increases the attractive forces during the first iterations to allow similar points to come together more effectively. This is done by multiplying ρ to the attractive forces for a number of iterations when starting the optimization. The default early exaggeration uses $\rho = 12$ for the first 250 iterations, and we use default settings unless differently specified.

Instead of the t-distribution kernel with one degree of freedom $k(d) = \frac{1}{1+d^2}$, one can consider a general, scaled t-distribution kernel

$$k(d) = \frac{1}{(1 + \frac{d^2}{\alpha})^{\alpha}} \tag{2.5}$$

for $\alpha = \frac{\nu+1}{2} > 0$ and ν the degree of freedom (Yang et al. 2009, Kobak et al. 2020). This formulation allows negative degrees of freedom, which results in heavier tails than allowed for any t-distribution. For $\alpha \to \infty$ the kernel has a Gaussian limit, which corresponds to SNE (Hinton and Roweis 2002). For $\alpha = 1$ it is the Cauchy kernel, as implemented in t-SNE (Maaten and Hinton 2008).

2.2 Graph Layouts

Graphs are an ubiquitous form of data, which for instance can express the relations in a social network, the visits of connected websites or citations. For information visualization and rapid pattern detection in graphs, it is desirable to find a 2D-visualization of the graph, which is also called graph layout or graph embedding. One challenge is to find an automatic and good way of visualizing graphs with multiple nodes and edges. This requires to assign positions to the graph nodes. One layout type is a planar node-link diagram, which is a two-dimensional representation of all nodes and their connections. After introducing basic graph theory, I will give an overview of graph layout algorithms concerning this work.
An undirected graph contains a set of vertices or nodes V and a set of edges E (Hu and Shi 2015). Two nodes are connected whenever there is an edge between them and we can call them graph neighbours. A graph is associated and can be fully expressed by the adjacency matrix (Bondy and Murty 1976). If ν is the number of vertices and ε the number of edges, the adjacency is a $\nu \times \nu$-matrix, containing vertices as rows and vertices as columns.
Edges and nodes in a graph can either be weighted or unweighted (Noack 2009). Here, I will focus on the special case of an unweighted graph, with node weight equal to 1 and edge weight ω equal to 1 if nodes are connected and equal to 0 if not. The graph adjacency is given by the weights between all pair of nodes. For unweighted graphs the adjacency is binary.

[1]Even though here ρ is put in the denominator, in the implementation ρ is multiplied to the first sum representing the attractive forces.

2.2.1 Force-Directed Layouts

Force-directed layouts are layouts coming from energy models (since force is the negative gradient of energy) and they are based on node or edge repulsion and attraction (Noack 2009). To assign appropriate positions in a two-dimensional space, an energy function is minimized to an equilibrium state. This process of optimization is non-deterministic and can get stuck in a local minimum (Jacomy et al. 2014).
I will introduce here the two basic models the most popular models are based on: the spring electric model and the stress model.

Spring Electric Models The spring electric model was first introduced by Eades 1984 and later improved by Fruchterman and Reingold 1991. The idea is to look at edges as springs connecting two nodes. Starting from an initial layout the spring forces move the system to an equilibrium state. Generally, the attractive forces can be written as

$$F_{\mathrm{a}} = \omega_{\mathrm{ij}}||y_{\mathrm{i}} - y_{\mathrm{j}}||^{a} \tag{2.6}$$

for node y_{i} being connected to y_{j} ($i \leftrightarrow j$) and the repulsive forces as

$$F_{\mathrm{r}} = ||y_{\mathrm{i}} - y_{\mathrm{j}}||^{p} \tag{2.7}$$

for $i \neq j$. a and r are real constants and $a > r$ to avoid infinite distances (Noack 2009). Practically, it also holds $a \geq 0$ and $r \leq 0$. Fruchterman and Reingold 1991 used $a = 2$ and $r = -1$. To better preserve clusters in the graph layout, Noack 2007 introduced the LinLog model with $a = 0$ and $r = -1$. In between those two models lies the nowadays also widely used algorithm ForceAtlas2 (Jacomy et al. 2014) with $a = 1$ and $r = -1$.
If $r < 0$, the computation of shortest-path distances is required for all pair of nodes. Methods based on these distances, which first need to be computed, are computationally expensive and even computationally intractable for medium-large sized datasets (Noack 2009). To prevent computational infeasibility, there are several methods to approximate the repulsive forces, e.g., by a Barnes-Hut (Barnes and Hut 1986) optimization in ForceAtlas2 (Jacomy et al. 2014). Others can be reviewed in (Noack 2009, Hu and Shi 2015).

Stress Models In comparison to the spring electrical models, the stress models assume that springs connect every pair of vertices in the graph (Hu and Shi 2015). Here, the ideal spring length d_{ij} matches the predefined edge lengths and the models are thus distance-preserving. Generally, the cost function or energy is given by

$$E = \sum_{\mathrm{i} \neq \mathrm{j}} \omega_{\mathrm{ij}}(||y_{\mathrm{i}} - y_{\mathrm{j}}|| - d_{\mathrm{ij}})^{2}. \tag{2.8}$$

For $\omega_{\mathrm{ij}} = \frac{1}{d_{\mathrm{ij}}}$ the embedding is known as Sammon Mapping, a special case of metric Multidimensional Scaling (MDS) (cf. Ghojogh et al. 2020).
Note: stress models require a full-distance matrix and thus, a computation of all shortest-path distances, which is computationally expensive or even intractable.

Not Force-Directed Methods Another related group of methods, which are not force-directed, are based on Classical Multidimensional Scaling (Torgerson 1952) and known under the term Strain Models. These are linear and try to fit the inner product of positions and have a global

optimum which can be calculated by spectral decomposition (Hu and Shi 2015, Brandes and Pich 2006). These models also require to compute the full distance matrix which is computationally expensive or even intractable. Pivot MDS (Brandes and Pich 2006) is a fast method to approximate classical MDS based on Landmark MDS (Silva and Tenenbaum 2002), which operates only on a distance matrix of a small set of landmarks from the original nodes. In extension to Landmark MDS, Pivot MDS also uses the distance information to non-landmarks by operating on a distance matrix from all nodes to the landmarks (Brandes and Pich 2006).
To enable capturing the non-linearity of the manifold, Isomap (Tenenbaum et al. 2000) is a nonlinear special case of classical MDS using an approximation of geodesic instead of euclidean distances for the construction of the distance matrix (Ghojogh et al. 2020).

2.2.2 Related Work

One aim of this work is to use t-SNE for graph embeddings. As revised so far, force-directed layouts are graph embeddings based on attractive and repulsive forces that act on the graph nodes in the embedding. The t-SNE cost function gradient also contains attractive and repulsive forces that can act on the nodes by giving an affinity matrix as input which can be derived by the graph adjacency. How exactly the affinity matrix is derived, will later be described in Section 3.1.
In this section I will introduce related work that has been used to generate graph layouts in two dimensions, namely tsNET (Kruiger et al. 2017) (coming from 't-SNE' and 'network') and t-distributed Force Directed Placement (t-FDP) (Zhong et al. 2023).

tsNET The tsNET algorithm makes use of the t-SNE loss function from Equation (2.3). The first step is to compute all graph-theoretic shortest path distances $||x_\mathrm{i} - x_\mathrm{j}||$ for $i, j = 1, \ldots, N$. These are fed into the loss function

$$C_{t\mathrm{sNET}} = \lambda_\mathrm{KL} C_\mathrm{KL}(||x_\mathrm{i} - x_\mathrm{j}||) + \frac{\lambda_\mathrm{C}}{2N} \sum_\mathrm{i} ||y_\mathrm{i}||^2 - \frac{\lambda_\mathrm{r}}{2N^2} \sum_{\mathrm{i} \neq \mathrm{j}} \log\left(||y_\mathrm{i} - y_\mathrm{j}|| + \epsilon_\mathrm{r}\right) \tag{2.9}$$

with λ_KL, λ_C and λ_r the weights of the different terms, and $\epsilon_\mathrm{r} = \frac{1}{20}$ a small regularization constant (Kruiger et al. 2017). While the first term corresponds to the t-SNE loss function from Equation (2.3), the second describes an additional compression for faster optimization and the third term an additional repulsion to prevent cluttering and dispersion. Same as in t-SNE, the cost function is minimized using gradient descent. The opimization is done in three steps: the first step is random initialization, the second step optimization with $(\lambda_\mathrm{KL}, \lambda_\mathrm{C}, \lambda_\mathrm{r}) = (1, 1.2, 0)$ until node displacement is sufficiently small and the third step is optimization with $(\lambda_\mathrm{KL}, \lambda_\mathrm{C}, \lambda_\mathrm{r}) = (1, 0.01, 0.6)$ for pulling nodes apart. This procedure follows a similar logic to the early exaggeration in t-SNE but is differently implemented: in the second optimization step, the compression and hence attraction is increased, and in the third step, the repulsion is enhanced.
The authors also introduce a refined version tsNET* which differs from tsNET by initialization with Pivot MDS and using the parameter setting $(\lambda_\mathrm{KL}, \lambda_\mathrm{C}, \lambda_\mathrm{r}) = (1, 0.1, 0)$ in the second stage. It is on average around 15% faster than tsNET and often improves the graph layouts.
Same as t-SNE, the method depends only on the single parameter perplexity. Is is deterministic depending on the initialization. The method is tested on 20 different graphs and compared to several other layout methods by a normalized stress metric and a neighbourhood-preservation metric. While the method does not perform significantly better in the stress metric, the neighbourhood preservation mostly is outperforming other methods.

To modify the performance of tsNET and make it especially working on large graphs, Zhu et al. 2021 introduced DRGraph, which makes use of a sparse distance matrix, uses negative sampling to approximate the gradient and uses a multi-level approach to additionally speed-up the optimization. DRGraph approaches Pivot MDS in runtime performance and outperforms all compared graph drawing methods in memory complexity for large graphs. The layout quality of tsNET and DRGraph are comparable. While both perform very similar in the neighbourhood-preservation metric, DRGraph reaches slightly better stress quality (Zhu et al. 2021).

t-FDP t-FDP is a method coming from the spring electrical model using a heavy-tailed t-force $F_{\mathrm{t}}(d) = \frac{d}{(1+d^2)^{\varphi}}$, where $\varphi \geq 1$ is the degree of freedom and $d = ||y_{\mathrm{i}} - y_{\mathrm{j}}||$ the distance between two nodes in the layout.
The final applied force to node i is given by

$$\begin{aligned} F(i) = & \sum_{\mathrm{i}\neq\mathrm{j}} \frac{||y_{\mathrm{i}} - y_{\mathrm{j}}||}{(1 + ||y_{\mathrm{i}} - y_{\mathrm{j}}||^2)^{\gamma}} \frac{y_{\mathrm{i}} - y_{\mathrm{j}}}{||y_{\mathrm{i}} - y_{\mathrm{j}}||} + \\ & \alpha \sum_{\mathrm{i}\leftrightarrow\mathrm{j}} \left(||y_{\mathrm{i}} - y_{\mathrm{j}}|| + \frac{\beta ||y_{\mathrm{i}} - y_{\mathrm{j}}||}{1 + ||y_{\mathrm{i}} - y_{\mathrm{j}}||^2} \right) \frac{y_{\mathrm{i}} - y_{\mathrm{j}}}{||y_{\mathrm{i}} - y_{\mathrm{j}}||} \end{aligned} \tag{2.10}$$

with α the weight for the attractive force, β the weight for the attractive t-force and γ the exponent for the repulsive t-force component (Zhong et al. 2023). The authors empirically estimated the parameters $(\alpha, \beta, \gamma) = (0.1, 8, 2)$ working well on the tested graphs. The first term of the equation corresponds to the repulsive and the latter term to the attractive forces. In comparison to the earlier spring model, this model has an additional attractive short-range force and replaces the repulsive force by the short-range repulsive t-force. In comaprison to tsNET the attractive t-force only acts on actual edges and is not weighted by the distance and is combined with long-range spring forces.
To speed up optimization the authors propose to use interpolation-based Fast Fourier Transform (ibFFT) to approximate the repulsive forces which was initially designed to speed up t-SNE. Using this implementation the authors achieve a run-time performance that is better than those of most comparable methods that can handle big graphs such as DRGraph (Zhu et al. 2021).
The embeddings of 50 different graphs with the sped-up algorithm are compared to several other graph layout methods by using relative errors between two layouts of four different metrics, including the same neighbourhood preservation metric as in the tsNET paper and a comparable normalized stress metric. All of the graph layout methods are initialized with Pivot MDS.
Next to t-FDP, only few algorithms are able to handle all graphs. For smaller graphs t-FDP performs slightly worse than the stress based models for the stress metric and thus distance-preservation, but similar to Fruchtermann and Rheingolds model and better than tsNET. In neighbourhood preservation t-FDP performs similarly to tsNET for less than 500 nodes, but shows clear outperformance for higher number of nodes.

2.2.3 Laplacian Eigenmaps

Different than in the related work, we initialize t-SNE with random initialization or Laplacian Eigenmaps (LE). Pivot MDS requires the knowledge of a full distance matrix for part of the nodes, while we only use the graph adjacency containing only first neighbours.
I will now briefly introduce the concept of LE, firstly implemented for data representation and DR by Belkin and Niyogi 2003.

The laplacian of a graph is a symmetric, positive-semidefinite matrix given by $\mathbf{L} = \mathbf{D} - \mathbf{A}$ with $\mathbf{A}$ being the adjacency matrix and $\mathbf{D}$ being the degree matrix. The degree matrix is a diagonal $\nu \times \nu$ matrix having only non-zero values on the diagonal which correspond to the degree of each node. This node degree is given by the number of edges connecting the node to other nodes.
LE leverages the spectral decomposition of the laplacian matrix. With given adjacency matrix the algorithm works as follows.

For each connected component the eigenvectors $\mathbf{f}$ and eigenvalues λ for the generalized eigenvalue problem

$$\mathbf{Lf} = \lambda \mathbf{Df} \tag{2.11}$$

are computed. The eigenvectors corresponding to the second-smallest till the $m + 1$-smallest eigenvalues are used for a m-dimensional embedding. The eigenvector to the smallest eigenvalue is not used because this eigenvalue is 0 and the vector is the constant vector which would collapse all nodes onto the same constant subspace of $m - 1$ dimensions. For a two-dimensional embedding this would mean that all nodes are collapsed to a constant real number.

The justification for obtaining a meaningful embedding with this method is the following:
In a binary adjacency matrix, each pair of connected nodes have edge weight $\omega = 1$ and each not connected pair $\omega = 0$. For a good embedding one can thus minimize

$$\sum_{\mathrm{ij}} (y_\mathrm{i} - y_\mathrm{j})^2 \omega_\mathrm{ij} = \sum_{\mathrm{ij}} (y_\mathrm{i}^2 + y_\mathrm{j}^2 - 2 y_\mathrm{i} y_\mathrm{j}) \omega_\mathrm{ij} = \sum_{\mathrm{i}} y_\mathrm{i}^2 d_\mathrm{i} = \sum_{\mathrm{j}} y_\mathrm{j}^2 d_\mathrm{j} - 2 \sum_{\mathrm{i,j}} y_\mathrm{i} y_\mathrm{j} \omega_\mathrm{ij} = 2\mathbf{y}^\mathrm{T} \mathbf{L} \mathbf{y} \tag{2.12}$$

with d_i being the degree of node i (Belkin and Niyogi 2003). One can thus minimize the right hand side subject to $\mathbf{y}^\mathrm{T}\mathbf{D}\mathbf{y} = \mathbf{1}$ to remove an arbitrary scaling factor. The vector that is minimizing the function is given by the solution of Equation (2.11).

For improvement in consistency (Luxburg et al. 2008) one can formulate a normalized graph laplacian. In the implementation we used, this is done by default. The eigenvalue problem from Equation (2.11) can be reformulated as

$$\mathbf{L}_\mathrm{norm} \mathbf{g} = \lambda \mathbf{D} \mathbf{g} \tag{2.13}$$

with $\mathbf{L}_\mathrm{norm} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2}$ and $\mathbf{g} = \mathbf{D}^{1/2} \mathbf{f}$ (Böhm et al. 2022).

It has been shown, that the attractive term of the gradient of t-SNE reduces to the LE gradient (left side of Equation (2.12)) without the quadratic constraint (Böhm et al. 2022). The latter corresponds to the repulsive forces of other force-directed layout methods. Because in t-SNE affinities are probabilities that sum to one, the degree matrix is approximately the identity matrix and one can expect that the leading eigenvectors of Equation (2.11) and Equation (2.13) are very close. For exaggeration approximating infinity t-SNE indeed produces embeddings very similar to LE (Böhm et al. 2022).

2.3 Contrastive Neighbour Embeddings

In recent years, deep learning in graphs has become increasingly popular (Liu et al. 2021). Self-supervised techniques, such as graph contrastive learning, are of particular interest, since they do not require labeled data. The idea of contrastive learning is to obtain a representation from a neural network, which is trained on the data, where similar points (positive samples) are mapped closely together while dissimilar points (negative samples) lie far apart. Contrastive learning is widely used in image recognition, where a positive sample often is represented by two augmentations of the same sample image. In Graph Contrastive Learning (GCL) this idea to learn representations of data is transferred to graph data (You et al. 2020). Among many others, popular augmentations in GCL to obtain positive samples for learning are node dropping, edge perturbation and masking (Wang et al. 2022). For augmentation based approaches, negative samples are given by two different original graphs.

However, augmentation based contrastive methods often fail regarding performance in so called heterophilic graphs. These are graphs with provided node labels, where two nodes with the same label are less likely to be connected or close together. In contrast, homophilic graphs are those graphs where two nodes of the same label are more likely to be connected or close together.
To better handle heterophilic graphs augmentation-free approaches have recently been introduced, for instance Local-GCL (Zhang et al. 2022) and Augmentation-Free GCL (AF-GCL) (Wang et al. 2022). Both methods outperform previous ones on heterophilic graphs.
Also Graph Neural Networks (GNN) have recently become more popular in graph learning (Xie et al. 2023). Most of them are also based on the homophily assumption, causing rising interest on novel approaches for heterophilic graphs (Pei et al. 2020; Zheng et al. 2022).

Instead of non-parametric t-SNE, as introduced in Section 2.1, one can also implement a parametric version of t-SNE. Instead of optimizing the embedding directly, the idea of parametric t-SNE is to input data samples to a neural network and train it such that it outputs appropriate 2D-coordinates which preserve the neighbourhood from the high-dimensional space in the embedding. This is done by using the InfoNCE loss. InfoNCE tries to predict the position of the data sample in an tuple containing m noise samples (noise contrastive samples) and the data sample (Damrich et al. 2022).

A parametric version of t-SNE can also be seen as contrastive learning method with two-dimensional output representation.
To implement this, we use a framework called Contrastive Neighbour Embedding (CNE) (Damrich et al. 2022). Originally, CNE was used for neighbour embeddings of image data. To train the network, different loss functions related to the neighbour embedding methods t-SNE, UMAP and NCVis were implemented to reveal their internal relationships.

For graphs, we do not have high-dimensional data points x_i from Section 2.1 to input into t-SNE, but sometimes node features are provided with the graph data. These features contain additional information for the nodes which might enable to enhance the graph layout.
To follow this idea we input the node feature vectors as data in the multilayered neural network of CNE. The output layer has as many neurons as we have nodes of the graph. These outputs represent the 2D-embedded coordinates of the features. The network trains its weights subject to a loss functions which contains edge information. This means, that we construct the positive samples, such that two feature vectors are attracted whenever their nodes are connected. Whenever two nodes are not connected, their features represent a negative sample. This is way of constructing positive and negative samples is similar to that in Local-GCL (Zhang et al. 2022).

2.4 Evaluation Metrics

To evaluate our embeddings, we use two metrics: nearest-neighbour (NN) recall and kNN accuracy.

NN recall The NN recall describes how well the neighbourhood in the high-dimensional space is preserved in the low-dimensional space. It is computed as follows:
For each node i the number of its graph neighbours nN_{i} and their indices $\{N_{\mathrm{x}}\}_{\mathrm{i}}$ are computed. This can be easily done by summing each row of the adjacency matrix, containing a 1 at every connected node and a 0 at every not connected node. Then, for each node, the indices $\{N_{\mathrm{y}}\}_{\mathrm{i}}$ of exact the same number of euclidean nearest neighbours is computed in the 2D-embedding using `sklearn.neighbors` and its function `NearestNeighbors`. One can now obtain the NN recall for each node by

$$\text{NN recall}_{\mathrm{i}} = \frac{|\{N_{\mathrm{x}}\}_{\mathrm{i}} \cap \{N_{\mathrm{y}}\}_{\mathrm{i}}|}{nN_{\mathrm{i}}}. \tag{2.14}$$

The NN recall is finally obtained by taking the mean over all nodes.

***k*NN accuracy** The kNN accuracy is a metric that reflects how well node labels are predicted by a classifier that chooses a label based on the labels of its k nearest neighbours.
It can only be used when the dataset provides labeled nodes. It is computed as follows:
The set of nodes is divided into a testing and a training set. Then, the `kNeighborsClassifier` from `sklearn.neighbors` with $k = 10$ is trained on the training set and the respective labels. The trained classifier was used to obtain predictions for the test set. Finally, the kNN accuracy can be computed by the mean of the boolean vector comparing predictions with test labels.

2.5 Data

We tested our implementation on five distinct benchmark datasets:

- can_96 and dwt_1005: Two structural datasets containing 96 nodes and 768 edges, and 1005 nodes and 8621 edges, respectively. Both are obtained from the SuiteSparse Matrix Collection (formerly: Florida Sparse Matrix Collection), publicly available on `https://sparse.tamu.edu/` (Davis and Hu 2011).[2]

- Amazon Computer and Amazon Photo: Two real world datasets, which are segments of an Amazon Co-Purchase dataset from 2015 (McAuley et al. 2015). Amazon Computer has $3,752$ nodes and $491,722$ edges. Amazon Photo has $7,650$ nodes and $238,163$ edges. Nodes indicate different items and edges indicate which items are frequently brought together. Both datasets are provided with numeric class labels which represent different subcategories of the products. While Amazon Computer has 10 classes, Amazon Photo has 8. Moreover, both datasets contain binary features which are bag-of-words encoded product reviews. Amazon Computer has 767 node specific features and Amazon Photo has 745 node specific features. The datasets were obtained from the deep graph library (`dgl`) (Wang et al. 2019) in python. Both graphs are not fully connected.

- Chameleon: A real word dataset containing $2,277$ nodes and $31,421$ edges. The data was collected from English Wikipedia in 2018 and describes page-on-page networks on Chameleons, publicly available on `https://github.com/benedekrozemberczki/datasets#wikipedia-article-networks` (Rozemberczki et al. 2021). Nodes

[2]For dwt_1005 c.f. `https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/dwt/dwt_1005.html`

indicate different articles and edges the mutual links between them. Each node has $3,132$ binary features, where 1 indicates the presence and 0 the absence of informative nouns in the article. Moreover, the dataset contains target values describing the average monthly traffic between October 2017 and November 2018 for each article (node). We used labels that were obtained by classifying the target values in 5 subcategories (cf. Pei et al. 2020). It is a fully connected graph.
Different to the Amazon datasets, the Chameleon dataset has way lower homophily, indicating that similar nodes are less likely to be connected together.

2.6 Implementation

All our implementation are in `python` 3.10.0. For t-SNE we use the open source library `openTSNE` 0.6.2 (Poličar et al. 2019). If not stated differently we used default settings: early exaggeration with a value of $\rho_1 = 12$ for the first 250 iterations and then optimize other 500 iterations with $\rho_2 = 1$, a learning rate of $\frac{n}{\rho}$ where n is the number of nodes and ρ the actual exaggeration value, and a momentum of 0.8. For most of the experiments the degree of freedom for the t-distributed kernel is set to $\alpha = \nu = 1$.
For initialization we used `openTSNE` settings 'random' for random initialization and the function `SpectralEmbedding` from `sklearn.manifold` 1.1.3 (Pedregosa et al. 2011) for LE initialization. We partially added random Gaussian noise with mean 0 and standard deviation 10^{-6} because of numerical issues.
For storing and processing graph data and finding the connected components, we used `networkx` 2.8.4 (Hagberg et al. 2008).
For evaluating the embeddings, we used `sklearn.neighbours`. For the kNN accuracy the number of neighbours was set to $k = 10$ and the size of the test set was set to 500 nodes. The set for training was obtained by the getting the set difference between all nodes and the test set.
For the implementation of the Contrastive Graph t- SNE we used a framework which has been previously developed for contrastive learning on image data and is called Contrastive Neighbour Embedding (CNE) (Damrich et al. 2022), available under `https://github.com/berenslab/contrastive-ne`. If not differently specified, the negative sampling loss and a batch size of $b = 128$ was used. The network was trained for 100 epochs.
For plotting and animations we used `matplotlib` 3.6.2 and its collections `pyplot` and `animation`. For the small structural datasets edges are plotted. For the other datasets edges are plotted with transparency whenever we depict a node-link diagram. Elsewise edges are not plotted.

3 Results

The idea of this work is to use t-SNE (Maaten and Hinton 2008) for generating graph layouts. Since t-SNE is a dimensionality reduction method it suits the purpose of reducing the high-dimensional space of the node connections given by the graph adjacency or node features, to a two-dimensional embedding: a node-link-diagram or node clustering, respectively.

3.1 Graph t-SNE

In this first section we will show the resulting 2D-graph-layouts using the neighbourhood information given by the graph adjacency in t-SNE. In this case the resulting layout is a node-link diagram of the graph.

3.1.1 Standard Implementation

In standard t-SNE we have high-dimensional data points x_i for $i = 1, \ldots, N$ that we want to embed by finding the respective lower-dimensional coordinates y_i. For graph data we do not have these high-dimensional data points x_i. The high-dimensional affinities, which give a measure of how similar two data points are in the high-dimensional space, can thus not be obtained from Equation (2.1). To construct the affinity matrix we therefore follow an approach which is similar to the uniform approximation discussed in Section 2.1:

1. The graph adjacency can be seen similar to a kNN-graph. For unweighted and undirected graphs, which we will investigate here, the graph adjacency is a binary matrix containing N rows and N columns representing each all N nodes. If two nodes are connected we will call them graph neighbours and the adjacency will have an entry of 1. The graph adjacency thus can be seen as graph neighbourhood matrix. To obtain uniformly distributed high-dimensional conditional probabilities we therefore normalize each row, such that the graph neighbours of each node i are uniformly distributed around their node i.

2. Then, following the standard procedure, we symmetrize and normalize the matrix to obtain the high-dimensional similarities.

With random initialization and standard settings as discussed in Section 2.6, we now can use t-SNE with our precomputed affinities to get two-dimensional graph embeddings. The computed embeddings are depicted in Figure 1 and Figure 2. Without any further fine tuning, we already reach structural embeddings with NN recalls of 0.867 for can_96, 0.794 for dwt_1005, 0.421 for Amazon Computer and 0.472 for Amazon Photo, and kNN accuracies of 0.892 for Amazon Computer and 0.920 for Amazon Photo. The recalls decrease for increasing node number. Higher node number implies higher starting dimension which impedes the neighbourhood preservation. In Figure 2 we can observe clusters emerging from the same classes. Not all classes form a separate cluster, for instance the purple class of Amazon Computer and the red, light blue and dark blue classes of Amazon Photo separate in several little clusters.

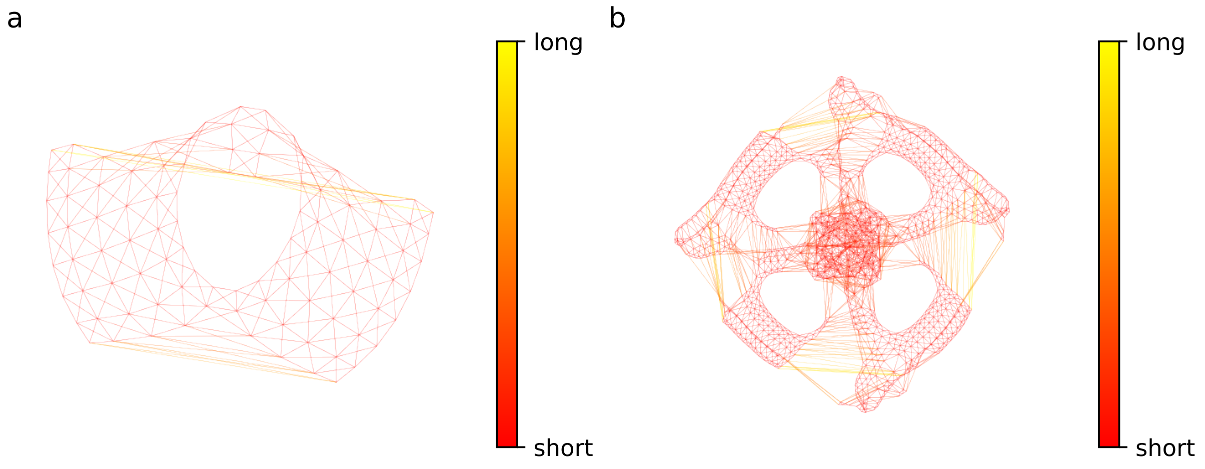

Figure 1: Graph t-SNE with random initialization of (a) can_96 (NN recall= 0.867) and (b) dwt_1005 (NN recall= 0.794). Colourcoding indicates the length of the edges.

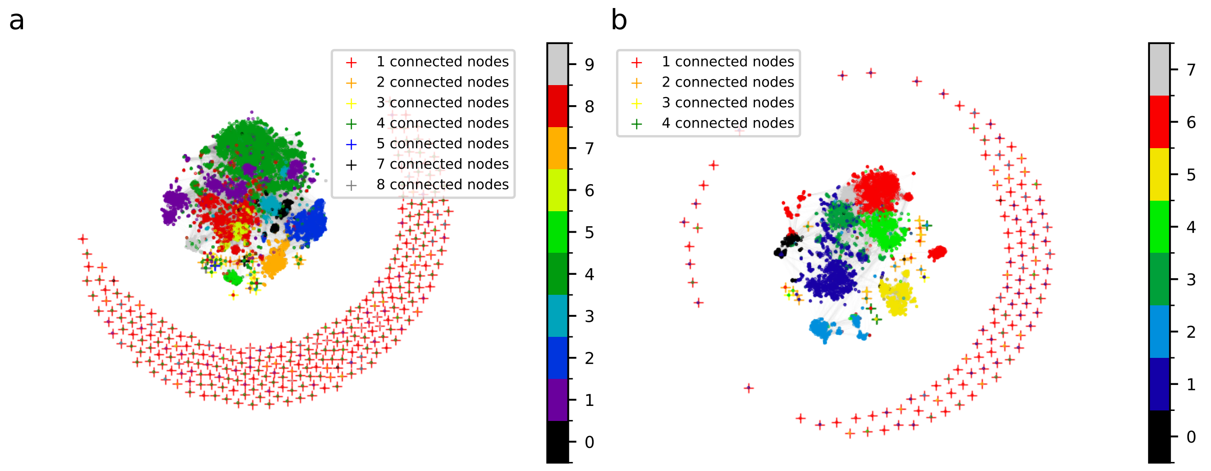

Figure 2: Graph t-SNE and random initialization of (a) Amazon Computer (NN recall= 0.421, kNN accuracy= 0.892) and (b) Amazon Photo (NN recall= 0.472, kNN accuracy= 0.920). Coloured by classes. Coloured crosses indicate nodes from differently sized connected components of the graph.

Disconnected Components. In the layouts of the Amazon datasets in Figure 2, one can observe that many points move to the periphery. These points are mainly repelled. This let us hypothesizing that those nodes are poorly connected. To check this, we investigated the number of connections in the graph adjacency and then computed all connected components of the graph. A connected component is a subset of nodes that are connected which each other but not with other nodes of the graph. In the Amazon graph, this would mean, that items are frequently brought together but never together with items which are not in that subset. The points in the periphery, marked with red crosses, are those nodes, that do not have any graph neighbours and thus do not feel attraction to any other node. These nodes represent items that are only brought alone. Some other outliers can be identified by nodes of different smaller connected components. These are indicated by coloured crosses in Figure 2.
The Graph t-SNE after removing all small connected components is shown in Figure 3. Slightly higher kNN accuracies of 0.930 for both Amazon datasets layouts are reached and less outliers appear in the embeddings.

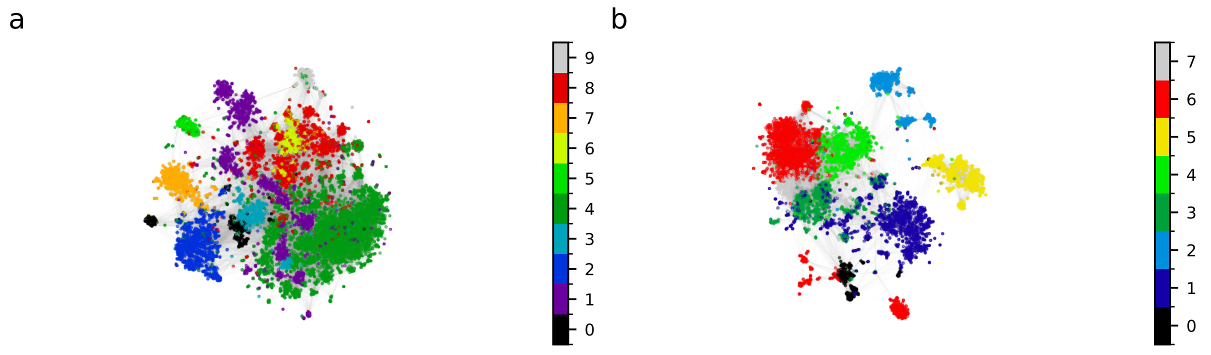

Figure 3: Graph t-SNE with random initialization of largest fully connected component of (a) Amazon Computer (NN recall= 0.421, kNN accuracy= 0.930) and (b) Amazon Photo (NN recall= 0.470, kNN accuracy= 0.930). Coloured by classes.

3.1.2 Laplacian Eigenmap initialization

To improve the graph layouts we would like to choose a more meaningful initialization. Since we do not want to compute other graph distances which are required for methods like MDS or Pivot MDS, we initialize with LE, which is also pre-implemented in `openTSNE`. When optimizing t-SNE with the pre-implemented LE initialization, two problems emerged. First, the embeddings are not deterministic. Second, the embeddings for the smaller graphs get stuck in local minima such that some nodes are overlapping. To prevent these issues, we implemented LE initialization by using `SpectralEmbedding` from `sklearn.manifold`, setting a random state and adding a slightly bit of Gaussian noise. The noisy laplacian eigenmaps are then passed to the t-SNE optimization as initial coordinates. Based on our report of the issues, the upcoming version of `openTSNE` 0.7.0[3] provides a deterministic spectral initialization and includes adding noise by default.

The LEs of the structural graph data are depicted in Figure 4. The LE of can_96 has a NN recall of 0.356, dwt_1005 has a NN recall of 0.382. One can see the typical weak repulsion resulting in over-attracted embeddings. As discussed in Section 2.2.3, we can indeed reproduce matching embeddings with t-SNE when setting the exaggeration $\rho = 12$ throughout the optimization, shown in Figure 8 and Figure 9d. For the Amazon datasets one can expect that LEs just separate

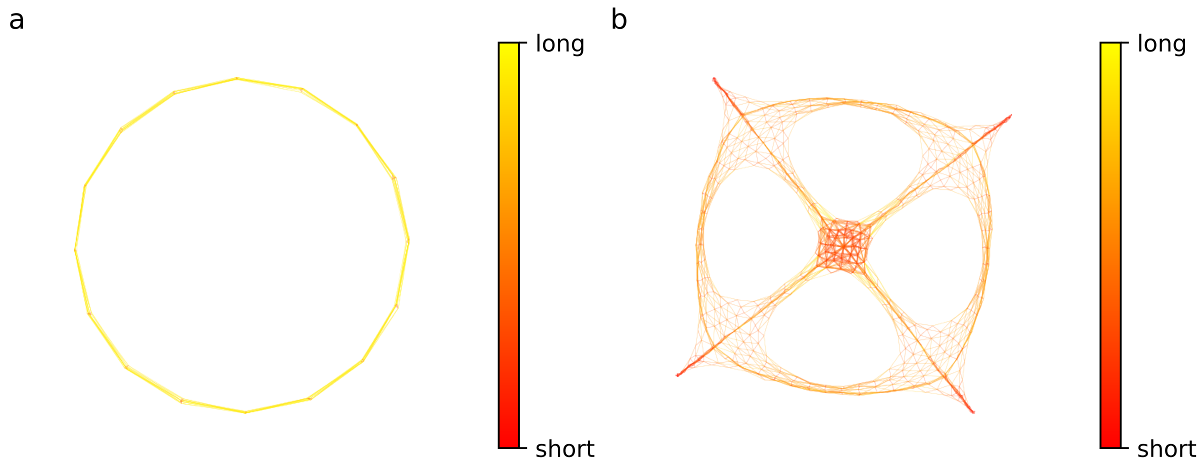

Figure 4: Laplacian Eigenmaps of (a) can_96 (NN recall= 0.356) and (b) dwt_1005 (NN recall= 0.382). Colourcoding indicates the length of the edges.

the connected components, which is indeed the case and is depicted in Figure 5. While all unconnected nodes share the same location, the number of separately visible nodes of connected components match the number of connected components of a certain size (indicated in the legend). This behaviour of Laplacian Eigenmaps to separate only the connected components and loose local information of each connected component is commonly known (Zhang et al. n.d.). For further graph embeddings we removed the small connected components and stay with the subgraph of $13,381$ nodes for Amazon Computer and $7,487$ for Amazon Photo. The LEs of both graphs are depicted in Figure 6. Again, the embeddings underlie strong attraction. Generally, we observe that the LE initializations for t-SNE do not enhance the graph layouts considerably. We do find improvements of the layouts of can_96 and dwt_1005 reaching NN recalls of 0.899 and 0.807, respectively. The t-SNE graph layouts of these two datasets with

[3]see `https://mobile.twitter.com/pavlinpolicar/status/1625882997244284929`

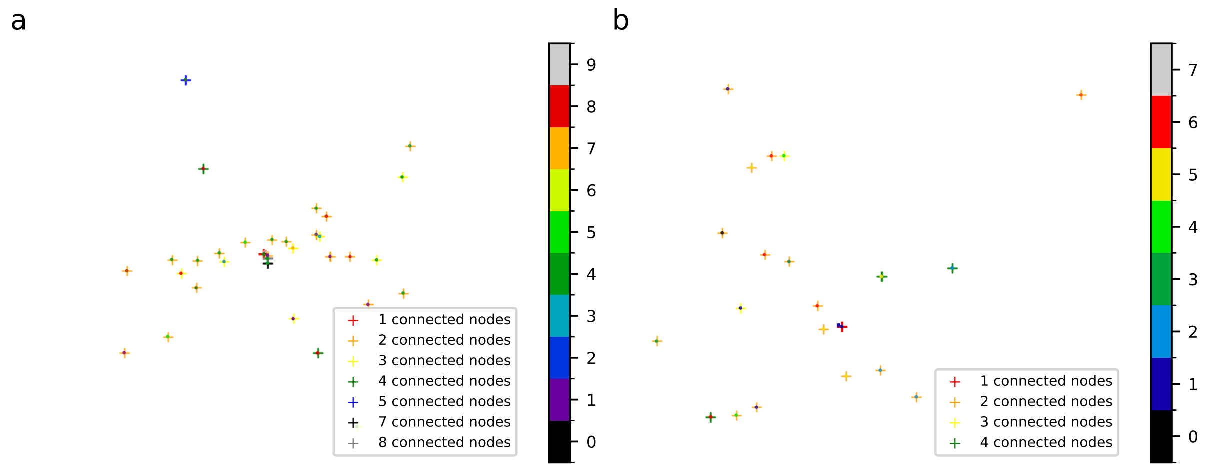

Figure 5: Laplacian Eigenmaps of (a) Amazon Computer and (b) Amazon Photo. Coloured crosses indicate nodes from differently sized connected components. Nodes coloured by classes.

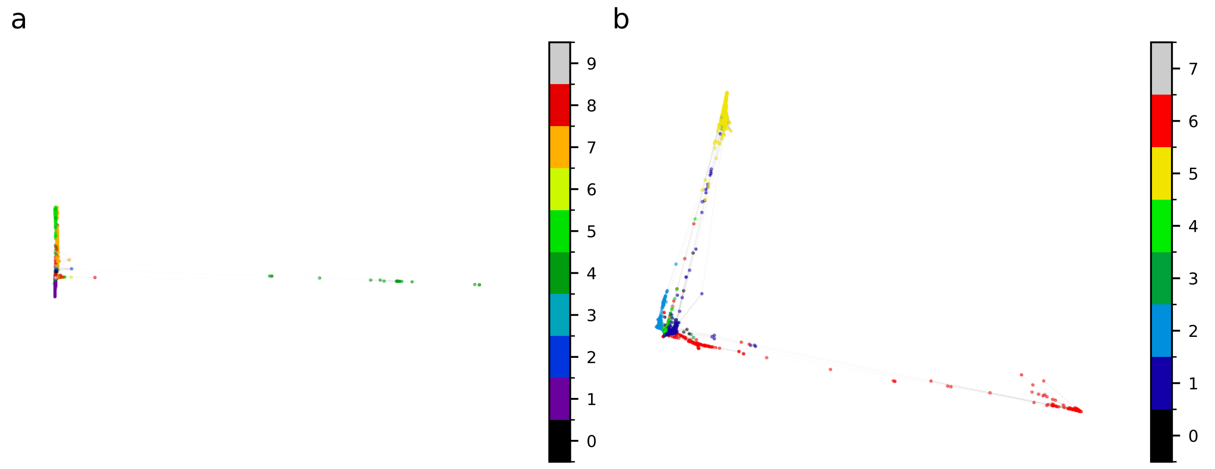

Figure 6: Laplacian Eigenmaps of the largest fully connected components of (a) Amazon Computer (NN recall$= 0.067$, kNN accuracy$= 0.0.782$) and (b) Amazon Photo (NN recall$= 0.162$, kNN accuracy$= 0.830$). Nodes coloured by classes.

LE initialization are depicted in Figure 7. We can observe, that the upper edge of the circle in can_96 unfolds a slightly bit. Also, the protrusions in the outer part of dwt_1005 unfold slightly more.
Since spectral initialization does not enhance the layouts considerably, in the following, we will show our results for random initialization.

Annealed Exaggeration. By generating animations of the t-SNE optimization we found that it might be beneficial to perform annealing when switching the exaggeration value to smooth the inflation after 250 iterations. Therefore, we performed early exaggeration for the first 125 iterations and then performed annealing for other 125 iterations by decreasing the exaggeration and increasing the learning rate parameter for each iteration step-wise till an exaggeration of 1 is reached. By doing so, we observed jumping of nodes in the animation instead of the expected smooth inflation after 125 iterations. To investigate this further, we compared the optimization of 125 iterations to 125 optimizations with each 1 iteration which lead to inconsistent outcomes. Further investigations revealed a bug in `openTSNE` 0.6.2 which is fixed in the upcoming version

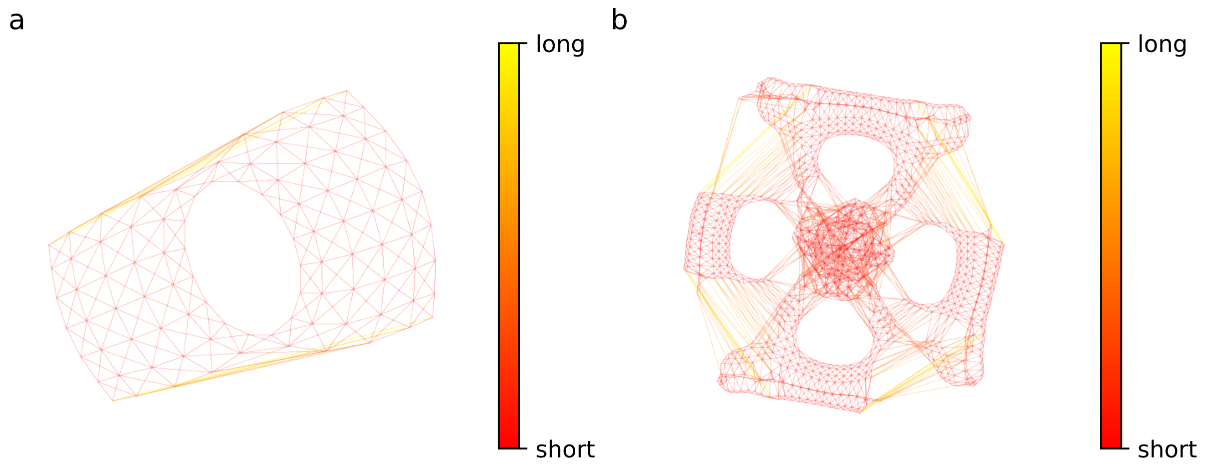

Figure 7: Graph t- SNEs with spectral initialization of (a) can_96 (NN recall= 0.899) and (b) dwt_1005 (NN recall= 0.807). Colourcoding indicates edge length.

`openTSNE` 0.7.0.
Because of this finding, we did not perform any further annealing for t-SNE optimizations.

3.1.3 Late Exaggeration

To investigate the attraction-repulsion spectrum we tested different values for the exaggeration ρ_2 from Equation (2.4) for the last 500 iterations.
With increasing exaggeration we expect that nodes cluster more tightly together since the attraction is increased. This is demonstrated for the smaller structural datasets in Figure 8 and Figure 9. The embeddings of the Amazon datasets are shown in the Appendix A. Even though the NN recalls and kNN accuracies decrease for increasing exaggeration (c.f. Figure 8 and Figure 9), one can observe visually more pleasant embeddings, as e.g. can_96 for $\rho_2 = 2$ from Figure 8b. Here, the circle unfolds completely. Figure 8b is similar to the embedding of can_96 obtained with tsNET* by Kruiger et al. 2017. In Figure 9c with $\rho_2 = 4$, one can also observe a similar embedding of dwt_1005 as produced by tsNET*.
As discussed in Section 2.2.3, one can also observe in Figure 8 and Figure 9 that increasing exaggeration results in layouts similar to those obtained by LE from Figure 4.

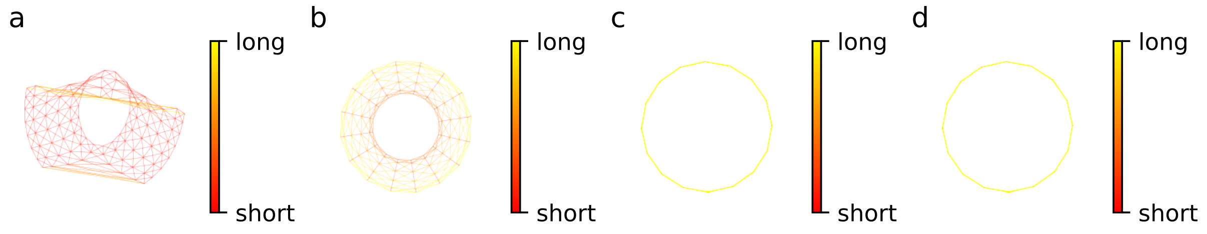

Figure 8: Graph t- SNEs with random initialization of can_96 and varying later exaggeration value ρ_2: (a) $\rho_2 = 1$ (default, NN recall $= 0.867$), (b) $\rho_2 = 2$ (NN recall $= 0.775$), (c) $\rho_2 = 4$ (NN recall $= 0.358$), (d) $\rho_2 = 12$ (NN recall $= 0.358$). Colourcoding indicates edge length.

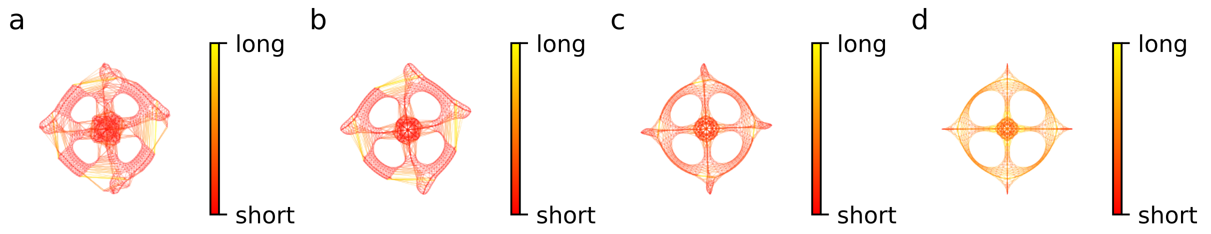

Figure 9: Graph t- SNEs with random initialization of dwt_1005 and varying late exaggeration value ρ_2: (a) $\rho_2 = 1$ (default, NN recall $= 0.794$), (b) $\rho_2 = 2$ (NN recall $= 0.736$), (c) $\rho_2 = 4$ (NN recall $= 0.563$), (d) $\rho_2 = 12$ (NN recall $= 0.403$). Colourcoding indicates edge length.

3.1.4 Degree of Freedom

To investigate different levels of cluster structure preservation one can tune the degree of freedom of the t-distributed kernel of the low-dimensional similarities (Kobak et al. 2020) in t-SNE. Therefore, we investigated the four different degrees $\alpha = 0.5$, $\alpha = 1$, which corresponds to the Cauchy kernel and is the default setting in `openTSNE`, $\alpha = 10$ and $\alpha = 100$. Given the limit behaviour of the t-distribution approaching the Gaussian distribution for $\alpha \to \infty$, one expect an SNE-typical embedding for $\alpha = 100$. One can indeed observe, a shift from global structure preservation to finer graph layouts with increasing degree of freedom in Figure 10 and Figure 11. High degrees of freedom for dwt_1005, depicted in Figure 11c and Figure 11d, lead to similar embeddings to that produced with t-FDP by Zhong et al. 2023.
However, NN recalls reach highest values for default settings $\rho_2 = 1$ and $\alpha = 1$. The embeddings of the Amazon datasets are shown in Appendix A.

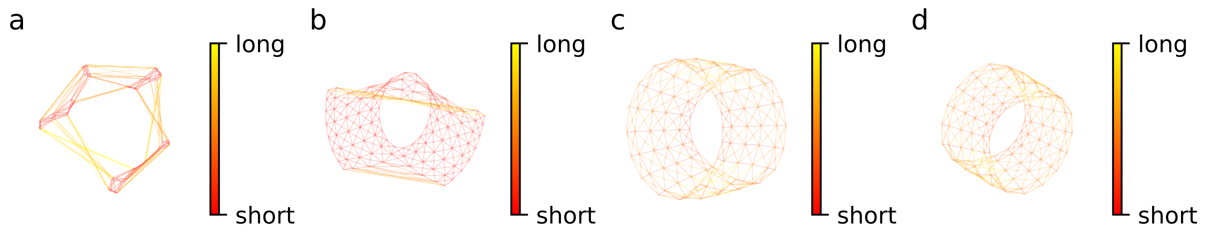

Figure 10: Graph t- SNEs with random initialization of can_96 and varying degree of freedom α: (a) $\alpha = 0.5$ (NN recall $= 0.677$), (b) $\alpha = 1$ (default, NN recall $= 0.867$), (c) $\alpha = 10$ (NN recall $= 0.768$), (d) $\alpha = 100$ (NN recall $= 0.775$).

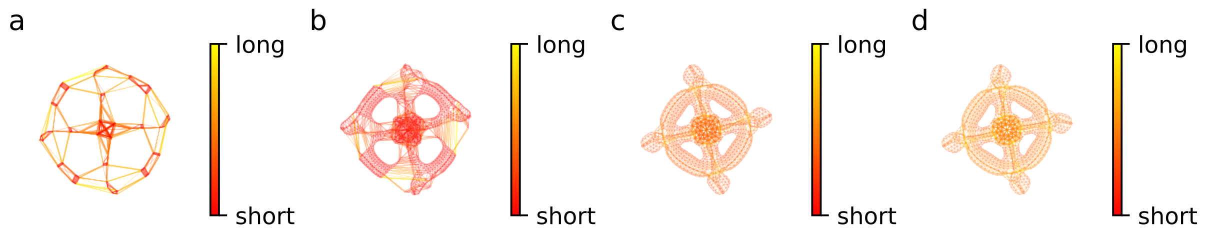

Figure 11: Graph t- SNEs with random initialization of dwt_1005 and varying degree of freedom α: (a) $\alpha = 0.5$ (NN recall $= 0.665$), (b) $\alpha = 1$ (default, NN recall $= 0.794$), (c) $\alpha = 10$ (NN recall $= 0.732$), (d) $\alpha = 100$ (NN recall $= 0.719$).

3.2 Feature t-SNE

In this section we will show embeddings resulting from the use of node features of our graph data.
Since now, we use node features for embedding graphs with t-SNE, and we thus have high-dimensional data-points x_{i}, we can directly pass the node feature matrix containing N rows (nodes) and F features (binary node feature vectors) to `openTSNE`. By default the similarities are computed by a Gaussian mixture kernel.
The embeddings of these additional feature information are shown in Figure 12. The kNN accuracies of the embeddings are 0.786 for Amazon Computer and 0.834 for Amazon Photo. As indicated by the kNN accuracies, one can observe that clusters do partially separate, but less pronounced than in the graph neighbourhood embeddings. Also, more clusters within a class emerge than when using the edge information.

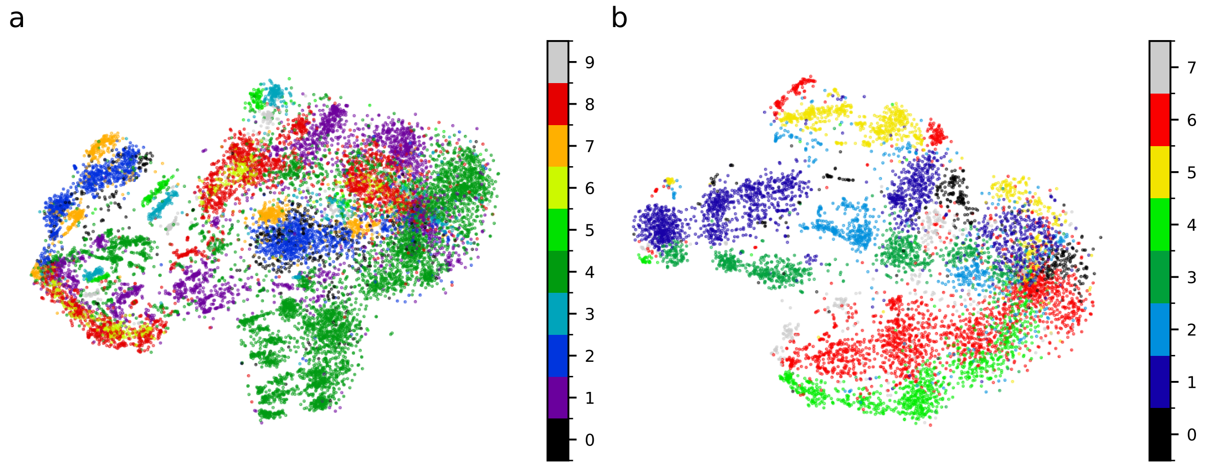

Figure 12: Feature t-SNE with random initialization of the largest fully connected components of (a) Amazon Computer (kNN accuracy $= 0.786$) and (b) Amazon Photo (kNN accuracy $= 0.834$). Nodes coloured by classes.

Aiming to improve the graph layouts, we tried to merge these two sources of information: the node connections and the node features. To do so, we first computed a kNN matrix with k neighbours in the features space with `kneighbours_graph` from `sklearn.neighbours`. To now connect the information, we performed an xOR operation on the graph adjacency and the kNN feature adjacency for different number of neighbours k in the feature space. Afterwards the new 'merged' adjacency is preprocessed as before (c.f Section 3.1). Then, we optimized the embeddings with t-SNE using default settings and random initialization on the largest connected components of Amazon Computer and Amazon Photo and computed the kNN accuracies for different k. The trend is depicted in Figure 13.
Following the overall trend, we conclude that the more feature neighbours we include, the less accuracies we reach in the embeddings. This can be an indicator that the additional feature space does not improve the node embedding. However, the accuracy is the only metric investigated here. Example embeddings for $k = 20$ are depicted in the Appendix A (Figure 20).

3.3 Contrastive Graph t- SNE

Another possibility to merge the two sources of information - node connections and node feature information - is a contrastive learning approach. As outlined in Section 2.3, we use the

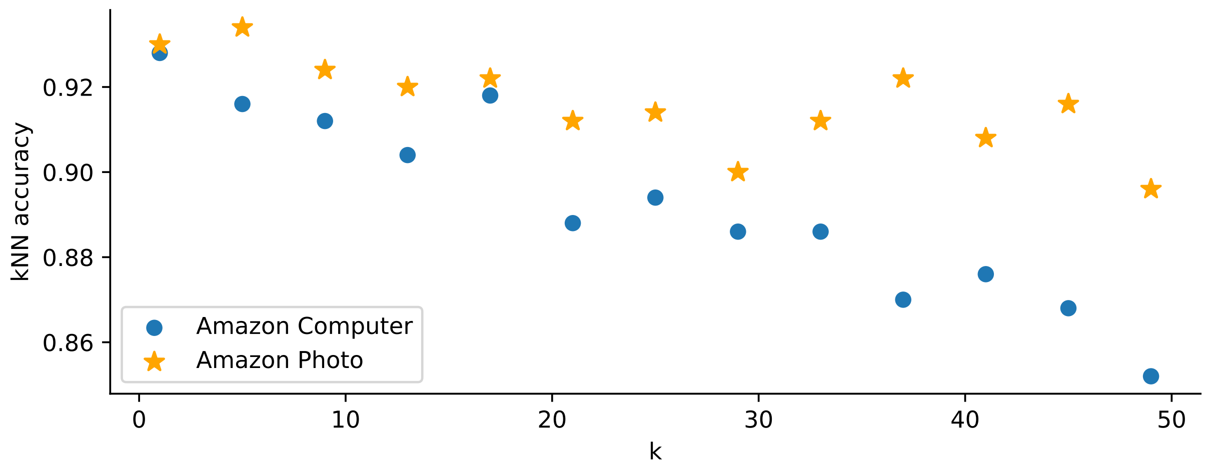

Figure 13: kNN accuracy of t-SNE with random initialization of largest connected components of Amazon datasets with precomputed combined graph adjacency against number of feature neighbours k. Up to k connections are added to the graph adjacency whenever the k nearest feature neighbours do not match the graph neighbours.

Contrastive Neighbour Embedding (CNE) framework, which was developed for contrastive neighbour embeddings of image data. In our approach we treat connected nodes as positive samples and unequal nodes as negative samples.
We investigate the embeddings of both Amazon graphs and the Chameleon Wikipedia graph with negative sampling loss and a batch size of $b = 1024$ for Amazon Computer and $b = 128$ else, since a high batch size conflicts comparably small graphs. We set the loss function to negative sampling, since we had issues with setting InfoNCE. Nonetheless, InfoNCE is desirable since it can be used as loss function of a parametric implementation of t-SNE, as implemented by Damrich et al. 2022.
The CNEs of the largest connected components of the Amazon datasets are shown in Figure 14. Amazon Computer reaches a kNN accuracy of 0.842 and Amazon Photo one of 0.910, both slightly worse than the Graph t-SNE emeddings from Figure 3. In this case the contrastive approach does not surpass the Graph t-SNE embeddings. Additional feature information does not improve the embedding in both cases, adding feature information through the input adjacency and via a parametric composition.

3.3.1 Chameleon Dataset

Different from the Amazon datasets, the Wikipedia network Chameleon has lower homophily degree, which means that nodes with the same label are less likely to be connected. Computing the Graph t- SNE indeed results below-chance performance with a kNN accuracy of 0.176. Computing the feature t- SNE results in Figure 15a with a kNN accuracy of 0.500. Some clusters, each with homogeneous class labels, separate well in the periphery. However, around the center of the embedding we can neither observe cluster separation nor class homogeneity. To compute the accuracy in the high-dimensional feature space the simple logistic regression model of `sklearn.linear_model` is trained on a training subset ($90\,\%$) of the feature matrix of the Chameleon dataset. With a test set size of $10\,\%$ the predictive accuracy reaches 0.631, which is similar to the accuracies reached by the discussed augmentation-free GCL methods, which also perform well on heterophilic graphs such as Chameleon, reaching accuracies around 0.650 (c.f. Zhang et al. 2022, Wang et al. 2022).

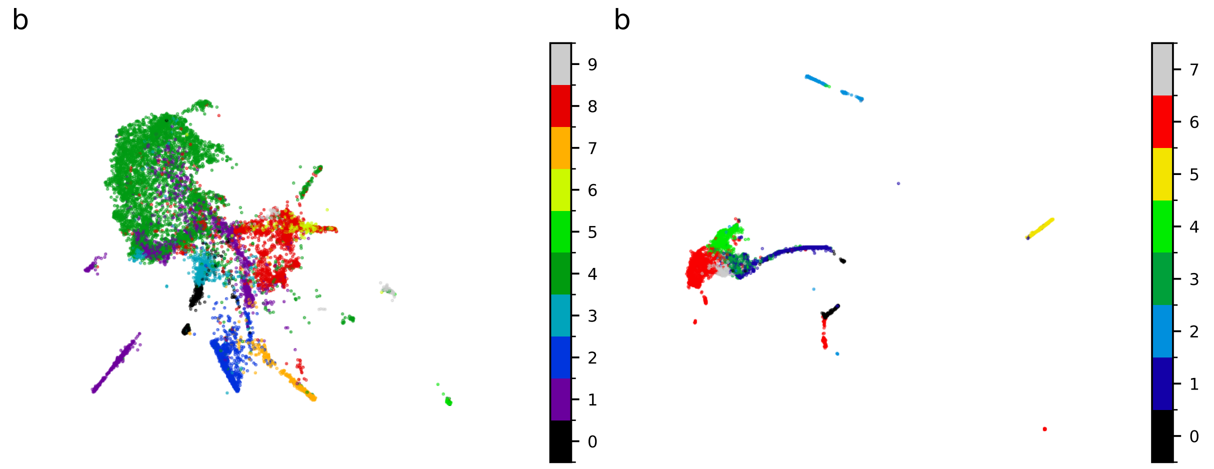

Figure 14: CNE with negative sampling loss of largest fully connected components of (a) Amazon Computer ($b = 1024$, kNN accuracy $= 0.842$) and (b) Amazon Photo ($b = 128$, kNN accuracy $= 0.910$). Nodes are coloured by classes.

The Contrastive Neighbour Embedding of Chameleon is depicted in Figure 15b. The kNN accuracy of 0.224 reflects the irregular structure in the embedding. The many little clusters in the periphery of Figure 15a cannot be observed in Figure 15b. Even though we see several other clusters, they do not match the provided labels. Even if the kNN accuracy is shifted slightly upwards from the Graph t-SNE of Chameleon, the result is not comparable to the other discussed contrastive approaches (Wang et al. 2022, Zhang et al. 2022) and will be further discussed in Section 4.

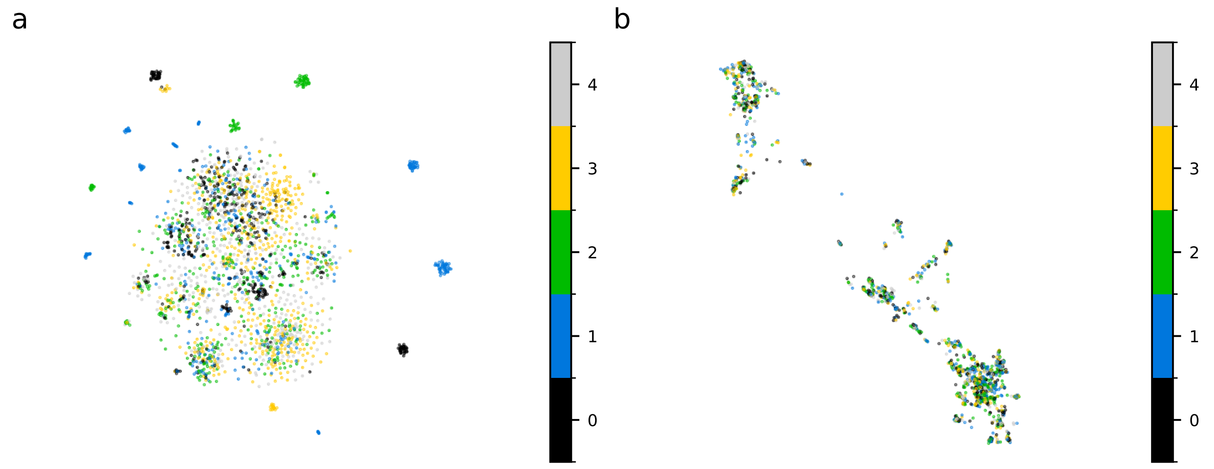

Figure 15: (a) Feature t-SNE with random initialization of the Chameleon Dataset (kNN accuracy $=$ 0.500). (b) CNE of Chameleon Dataset (kNN accuracy $=$ 0.224). Nodes are coloured by classes.

4 Discussion

In this work we created different embeddings of five distinct benchmark graph datasets with t-SNE (Maaten and Hinton 2008) using the implementation in the open source library `openTSNE` (Poličar et al. 2019).

First, we obtained embeddings by using the edge information from the adjacency matrix to compute an affinity matrix containing uniformly distributed affinities for each node (row). We call this method Graph t-SNE.
Second, we obtained embeddings by using node feature information provided in the Amazon datasets. These node features can be directly input to t-SNE as high-dimensional data points. We call the embeddings obtained with feature information Feature t-SNE.
Third, we tested two approaches to combine those two sources of information, aiming to improve the graph layouts. In the first approach, we added connections to the graph adjacency for the k nearest feature neighbours whenever the nodes were not connected. We varied the number of neighbours k from 1 to 50. The second approach was to obtain embeddings with CNE (Damrich et al. 2022) using negative sampling loss. This can be seen as a parametric implementation of t-SNE using features as input, but constructing positive pairs by node connections. We call this method Contrastive Graph t-SNE.
We evaluated our embeddings by two metrics: the NN recall and the kNN accuracy (for $k = 10$). The NN recall is a neighbourhood preservation metric, and the kNN accuracy tells us how well class labels can be predicted by their k nearest neighbours.

The results of our graph t-SNE with default setting already showed reasonably well layouts with high neighbourhood preservation for the small structural graph datasets (NN recall of 0.867 for can_96 and 0.794 for dwt_1005) and high accuracies (kNN accuracy of 0.892 for Amazon Computer and 0.920 for Amazon Photo) for the larger, labeled Amazon datasets. The high accuracies are also reflected visually in the graph layouts (Figure 2). The obtained clusters are mostly class homogeneous and some classes split in several clusters. Since larger datasets have higher original dimensionality, the neighbourhood preservation is more difficult, which results in lower NN recalls.
The Amazon datasets are known to have high homophily degrees: 0.777 for Amazon Computer and 0.827 for Amazon Photo (Zhang et al. 2022). Since the homophily degree gives the probability that two connected nodes share the same label, this means, that similarly labeled nodes are very likely to be connected. This is reflected by the high accuracies.

Our investigation of the Graph t-SNE framework includes testing Laplacian Eigenmaps initialization, trying to anneal exaggeration, increasing the late exaggeration values and varying the degree of freedom of the low-dimensional similarity kernel.
The initialization generally has high impact on embeddings resulting from t-SNE (Kobak and Linderman 2021). LE initialization slightly increased the investigated metrics, but did not result in considerably different embeddings. Further initialization techniques remain to be tested.
Annealing the exaggeration when switching it after 250 iterations revealed a bug in `openTSNE` 0.6.2. and is now correctly implemented in `openTSNE` 0.7.0. and can be tested in future work.
Setting the late exaggeration value for can_96 to $\rho_2 = 2$ unfolded the graph layout such that we obtained a planar circle (Figure 8b). A similar embedding is obtained by tsNET* from Kruiger et al. 2017 which performs an optimization comparable to early exaggeration. These layouts are visually more pleasing than the one obtained with $\rho_2 = 1$ (Figure 8a) and raises two topics for further research: first, investigating the exact relationship between tsNET and t-SNE with early exaggeration and second, finding further metrics to evaluate our embeddings. The recall obtained with $\rho_2 = 1$ is higher than that obtained with $\rho_2 = 2$. This suggests that future work is needed for computing and finding other metrics for evaluating the graph layouts. Neighbourhood preservation is a useful metric to evaluate neighbourhood preservation methods like t-SNE. However, it might be crucial to investigate also distance-preservation metrics like the normalized-stress or even find metrics that capture how simple or pleasing the embedding

looks like, e.g., uniform edge length or the crosslessness of edges.
Altering the degree of freedom result in embeddings of dwt_1005 similar to that obtained with t-FDP by Zhong et al. 2023. Again, it would be interesting to investigate the exact relationship between those two methods. T-FDP uses three parameters to construct the forces acting on the two-dimensional nodes, which the authors obtained empirically. Changing the configuration of these parameters (Equation (2.10)) results in different forces. The behaviour of those and how they relate mathematically to increasing the degree of freedom of the t-kernel for the low-dimensional similarities need to be investigated in future work.

When computing the Feature t-SNE, we reach kNN accuracies of 0.786 and 0.834 for the embeddings of the largest connected components of Amazon Computer and Amazon Photo, respectively. This indicates that the features also provide information for the class separation of the bought items (nodes). However, the kNN accuracies of the embeddings based on the feature information are lower than the ones based on the edge information (Figure 12). This is also reflected by the clusters separation being less pronounced.
We aimed to enhance the cluster separation in the embeddings of the Graph t-SNE by using this additional feature information.

When combining both feature and edge information by adding the k nearest feature neighbours to the graph adjacency whenever there was no connection between the two nodes beforehand, we observed that increasing k resulted in a lower accuracy (Figure 13). Again, our work is limited by the kNN accuracy, being the only metric computed to evaluate different embeddings.
In the second approach we performed Contrastive Graph t-SNE. The embeddings obtained yielded slightly lower kNN accuracies than those obtained by Graph t-SNE (Figure 14). Because of implementation issues and the time shortage at the end of the project, we used negative sampling loss which is more similar to a parametric version of UMAP. Nevertheless, in the future this should be revised to obtain embeddings with InfoNCE for better comparison to Graph and Feature t-SNE.
Apart from that, future work should concern how our Contrastive Graph t-SNE relates to other graph learning approaches, like those by Wang et al. 2022, Zhang et al. 2022, Pei et al. 2020. These approaches claim to work well on heterophilic graphs, which has been a great challenge in graph learning. When claiming that Contrastive Graph t-SNE has a tight relationship with other augmentation-free graph learning approaches, one would expect similar accuracies. However, applying Contrastive Graph t-SNE to the heterophilic graph dataset Chameleon does not considerably improve the kNN accuracy from the below-chance level one obtained with Graph t-SNE. Therefore, it remains to be identified where these differences lie.
On a different note, it has been found that high-frequency information is preferred for heterophilic graphs (Wang et al. 2022). One could also test tuning Graph t-SNE for heterophilic graphs by high-frequency initialization, such as taking eigenvectors corresponding to larger eigenvalues of the graph laplacian as initialization embedding.

These two approaches to combine both sources of information – feature and edge information – did not improve the neighbourhood embeddings or the cluster separation in the graph layouts for the here investigated Amazon datasets. However, if feature information can improve the embedding might depend on the dataset. For instance, citation networks could be promising, where nodes represent different papers and edges represent their connections. Features could encode informative nouns in the abstract, so that the topic similarity is encoded. Future research should investigate these methods to merge those two types of information on different labeled real life datasets.

From our work two main future direction arise. First, one could obtain embeddings of several benchmark datasets with tsNET (Kruiger et al. 2017) and t-FDP (Zhong et al. 2023) for reproduction and systematic comparison to our Graph t-SNE in terms of embedding quality and optimization time. In comparison to these methods, our Graph t-SNE allows to directly use the adjacency information without computing any further shortest path distances of the graph. Moreover, it is easy to implement using `openTSNE`.
Second, one also could investigate the Contrastive Graph t-SNE with different loss functions on different benchmark datasets and address the question how this framework relates to other augmentation-free graph contrastive learning methods, as Local-GCL from Zhang et al. 2022 and AF-GCL from Wang et al. 2022, and Graph Neural Networks to learn graph representations (c.f. Xie et al. 2023, Zheng et al. 2022). Here, the attention should focus especially on the question why our graph contrastive learning does not work comparably well on the heterophilic Chameleon dataset as these methods, reaching around $40\,\%$ higher accuracies (c.f. Zhang et al. 2022, Pei et al. 2020, Wang et al. 2022). It needs to be investigated if these methods operate differently on the feature space to achieve these high accuracies. These may arise from the intrinsically higher accuracy of the feature space in comparison to the intrinsic accuracy of the edge information since heterophilic graphs are less likely to have a connection between two nodes of the same label.

5 References

Barnes, J. H. and Piet Hut (1986). “A hierarchical O(N log N) force-calculation algorithm.” In: *Nature* 324, pp. 446–449.

Belkin, Mikhail and Partha Niyogi (2003). “Laplacian Eigenmaps for Dimensionality Reduction and Data Representation.” In: *Neural Computation* 15.6, pp. 1373–1396.

Böhm, Jan Niklas et al. (2022). “Attraction-Repulsion Spectrum in Neighbor Embeddings.” In: *Journal of Machine Learning Research* 23.95, pp. 1–32.

Böhm, Niklas (n.d.). *Dimensionality Reduction with Neighborhood Embeddings*. `https://jnboehm.com/boehm-mscthesis.pdf`. Accessed: 2023-03-07.

Bondy, J. A. and U. S. R. Murty (1976). *Graph Theory with Applications*. New York: Elsevier.

Brandes, Ulrik and Christian Pich (2006). “Eigensolver Methods for Progressive Multidimensional Scaling of Large Data.” In: *International Symposium Graph Drawing and Network Visualization*.

Damrich, Sebastian et al. (2022). *Contrastive learning unifies t-SNE and UMAP*.

Davis, Timothy A. and Yifan Hu (2011). “The university of Florida sparse matrix collection.” In: *ACM Trans. Math. Softw.* 38, 1:1–1:25.

Eades, P. (1984). “A heuristic for graph drawing.” In: *Congressus Numerantium* 42, pp. 149–160.

Fruchterman, Thomas M. J. and Edward M. Reingold (1991). “Graph drawing by force-directed placement.” In: *Software: Practice and Experience* 21.11, pp. 1129–1164.

Ghojogh, Benyamin et al. (2020). *Multidimensional Scaling, Sammon Mapping, and Isomap: Tutorial and Survey*.

Hagberg, Aric A. et al. (2008). “Exploring Network Structure, Dynamics, and Function using NetworkX.” In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux et al. Pasadena, CA USA, pp. 11–15.

Hinton, Geoffrey E. and Sam T. Roweis (2002). “Stochastic Neighbor Embedding.” In: *NIPS*.

Hotelling, Harold (1933). “Analysis of a complex of statistical variables into principal components.” In: *Journal of Educational Psychology* 24, pp. 498–520.

Hu, Yifan and Lei Shi (2015). “Visualizing large graphs.” In: *WIREs Computational Statistics* 7.2, pp. 115–136.

Jacomy, Mathieu et al. (2014). “ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software.” In: *PloS one* 9, e98679.

Kobak, Dmitry and George Linderman (2021). “Initialization is critical for preserving global data structure in both t-SNE and UMAP.” In: *Nature Biotechnology* 39, pp. 1–2.

Kobak, Dmitry et al. (2020). “Heavy-Tailed Kernels Reveal a Finer Cluster Structure in t-SNE Visualisations.” In: *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, pp. 124–139.

Kruiger, J. F. et al. (2017). “Graph Layouts by t-SNE.” In: *Computer Graphics Forum* 36.3, pp. 283–294.

Liu, Yixin et al. (2021). “Graph Self-Supervised Learning: A Survey.” In: *CoRR* abs/2103.00111.

Luxburg, Ulrike von et al. (2008). “Consistency of spectral clustering.” In: *The Annals of Statistics* 36.2.

Maaten, Laurens van der and Geoffrey E. Hinton (2008). “Visualizing Data using t-SNE.” In: *Journal of Machine Learning Research* 9, pp. 2579–2605.

McAuley, Julian J. et al. (2015). “Image-based Recommendations on Styles and Substitutes.” In: *CoRR* abs/1506.04757.

McInnes, Leland et al. (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*.

Noack, Andreas (2007). “Energy Models for Graph Clustering.” In: *J. Graph Algorithms Appl.* 11, pp. 453–480.

— (2009). “Modularity clustering is force-directed layout.” In: *Phys. Rev. E* 79 (2), p. 026102.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pei, Hongbin et al. (2020). “Geom-GCN: Geometric Graph Convolutional Networks.” In: *CoRR* abs/2002.05287.

Poličar, Pavlin G. et al. (2019). “openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding.” In: *bioRxiv*.

Rozemberczki, Benedek et al. (2021). “Multi-Scale Attributed Node Embedding.” In: *Journal of Complex Networks* 9.2.

Silva, Vin de and Joshua B. Tenenbaum (2002). “Global Versus Local Methods in Nonlinear Dimensionality Reduction.” In: *NIPS*.

Tenenbaum, Joshua B. et al. (2000). “A Global Geometric Framework for Nonlinear Dimensionality Reduction.” In: *Science* 290.5500, pp. 2319–2323.

Torgerson, Warren S. (1952). “Multidimensional scaling: I. Theory and method.” In: *Psychometrika* 17, pp. 401–419.

Wang, Haonan et al. (2022). *Augmentation-Free Graph Contrastive Learning with Performance Guarantee*.

Wang, Minjie et al. (2019). “Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks.” In: *arXiv preprint arXiv:1909.01315*.

Xie, Yaochen et al. (2023). “Self-Supervised Learning of Graph Neural Networks: A Unified Review.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.2, pp. 2412–2429.

Yang, Zhirong et al. (2009). “Heavy-Tailed Symmetric Stochastic Neighbor Embedding.” In: *NIPS*.

You, Yuning et al. (2020). *Graph Contrastive Learning with Augmentations*.

Zhang, Hanlin et al. (n.d.). “Component preserving and adaptive Laplacian Eigenmaps for data reconstruction and dimensionality reduction.” In: *Machine Learning, Multi Agent and Cyber Physical Systems*, pp. 642–649.

Zhang, Hengrui et al. (2022). *Localized Contrastive Learning on Graphs*.

Zheng, Xin et al. (2022). *Graph Neural Networks for Graphs with Heterophily: A Survey*.

Zhong, Fahai et al. (2023). “Force-Directed Graph Layouts Revisited : A New Force Based on the T-Distribution.” In: *IEEE Transactions on Visualization and Computer Graphics*.

Zhu, Minfeng et al. (2021). “DRGraph: An Efficient Graph Layout Algorithm for Large-scale Graphs by Dimensionality Reduction.” In: *IEEE Transactions on Visualization and Computer Graphics* 27.2, pp. 1666–1676.

A Appendix

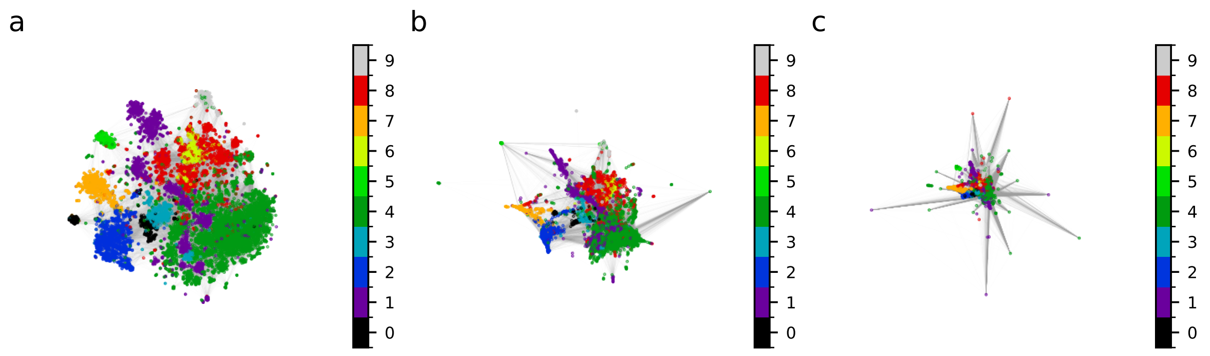

Figure 16: Graph t-SNE with random initialization and varying late exaggeration value of the largest connected component of Amazon Computer for (a) $\rho_2 = 1$ (NN recall $= 0.421$, kNN accuracy $= 0.930$), (b) $\rho_2 = 4$ (NN recall $= 0.189$, kNN accuracy $= 0.874$), and (c) $\rho_2 = 12$ (NN recall $= 0.118$, kNN accuracy $= 0.830$). Nodes are coloured by class labels.

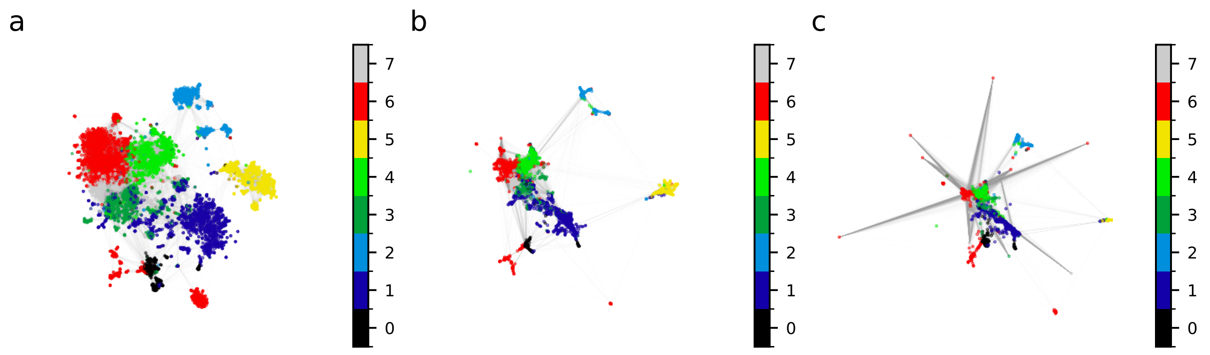

Figure 17: Graph t-SNE with random initialization and varying late exaggeration value of the largest connected component of Amazon Photo for (a) $\rho_2 = 1$ (NN recall $= 0.470$, kNN accuracy $= 0.930$), (b) $\rho_2 = 4$ (NN recall $= 0.271$, kNN accuracy $= 0.912$), and (c) $\rho_2 = 12$ (NN recall $= 0.190$, kNN accuracy $= 0.890$). Nodes are coloured by class labels.

The outliers in Figure 16c and Figure 17c for high exaggeration value ρ_2 emerged through convergence issues. Reducing the learning rate by a divisor of 10 lead to convergence. The reasoning was not further investigated.

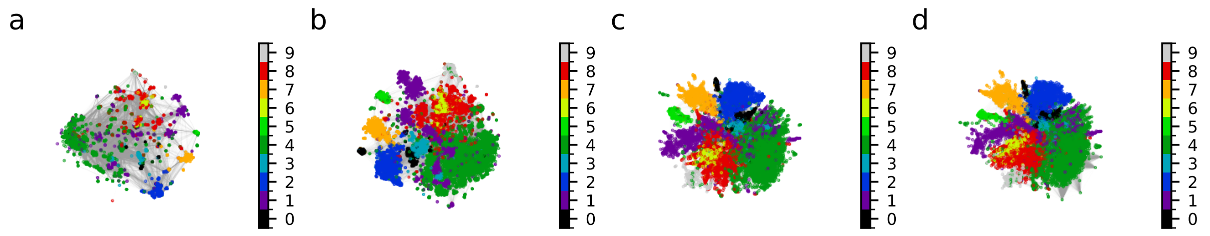

Figure 18: Graph t-SNE with random initialization and varying degree of freedom of low-dimensional similarity kernel of the largest connected component of Amazon Computer for (a) $\alpha = 0.5$ (NN recall $= 0.371$, kNN accuracy $= 0.930$), (b) $\alpha = 1$ (NN recall $= 0.421$, kNN accuracy $= 0.930$), (c) $\alpha = 10$ (NN recall $= 0.183$, kNN accuracy $= 0.842$), and (d) $\alpha = 100$ (NN recall $= 0.141$, kNN accuracy $= 0.812$). Nodes are coloured by class labels.

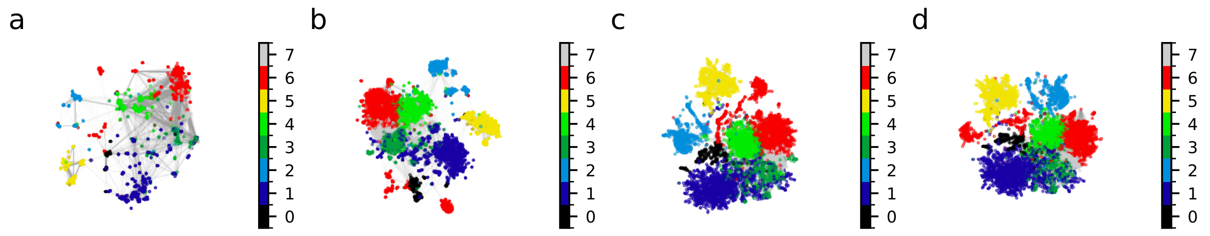

Figure 19: Graph t-SNE with random initialization and varying degree of freedom of low-dimensional similarity kernel of the largest connected component of Amazon Photo for (a) $\alpha = 0.5$ (NN recall $= 0.427$, kNN accuracy $= 0.926$), (b) $\alpha = 1$ (NN recall $= 0.470$, kNN accuracy $= 0.930$), (c) $\alpha = 10$ (NN recall $= 0.282$, kNN accuracy $= 0.908$), and (d) $\alpha = 100$ (NN recall $= 0.246$, kNN accuracy $= 0.888$). Nodes are coloured by class labels.

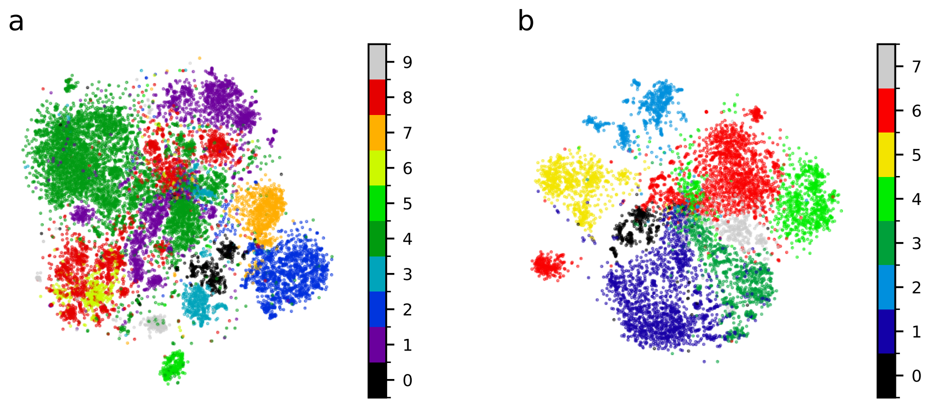

Figure 20: Graph t-SNE with random initialization with precomputed combined graph adjacency for $k = 20$ nearest feature neighbours of largest fully connected component of (a) Amazon Computer (kNN accuracy of 0.888) and (b) Amazon Photo (kNN accuracy of 0.912). Nodes are coloured by class labels.