Evaluating Representation Learning and Graph Layout Methods for Visualization

Edith Heiter, Bo Kang, Tijl De Bie, and Jefrey Lijffijt

Department of Electronics and Information Systems, IDLab, Ghent University, 9000 Ghent, Belgium
first.last@ugent.be

Abstract. Graphs and other structured data have come to the forefront in machine learning over the past few years due to the efficacy of novel representation learning methods boosting prediction performance in various tasks. Representation learning methods embed the nodes in a low-dimensional real-valued space, enabling the application of traditional machine learning methods on graphs. These representations have been widely premised to be also suited for graph visualization. However, no benchmarks or encompassing studies on this topic exist. We present an empirical study comparing several state-of-the-art representation learning methods with two recent graph layout algorithms, using readability and distance-based measures as well as link prediction performance. Generally, no method consistently outperformed the others across quality measures. The graph layout methods provided qualitatively superior layouts when compared to representation learning methods. Embedding graphs in a higher-dimensional space and applying t-Distributed Stochastic Neighbor Embedding for visualization improved the preservation of local neighborhoods, albeit at substantially higher computational cost. A longer version of this paper was recently published in the IEEE Computer Graphics and Applications journal (volume 42, issue 3, 2022). By presenting it at the workshop on mining and learning with graphs, we aim to reach the graph machine learning community in addition to the visualization and application-oriented audience of the journal.

1 Introduction

Visualization of data is useful both for understanding data and cross-checking models trained on that data. Consequently, it may lead to new insights, better models, the detection of outliers, etc. Visualization is frequently used in data science. However, it is not straightforward to visualize a *graph*. The vertices and edges typically do not have a real-valued representation, hence the primary problem in graph visualization is to find a good representation of the vertices in two-dimensional space.

Traditionally this area was known as *graph drawing* and methods that produce a representation of the nodes of a graph on a two-dimensional real-valued space were referred to as *graph-layout algorithms*. Early research considered problems such as to find planar embeddings (no edge crossings), later research also

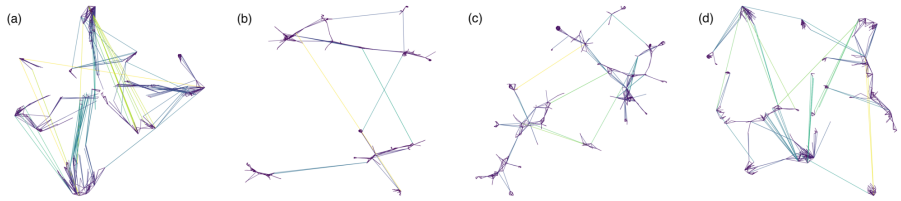


Fig. 1: Visualizations of the *netscience* collaboration graph using t-Stochastic Neighbor Embedding (t-SNE) and different embedding methods: (a) Arbitrary-order Proximity Preserved Network Embedding (AROPE), (b) Conditional Network Embedding (CNE), (c) DeepWalk, (d) Graph Auto-Encoder (GAE). Color corresponds to edge length: yellow (long) to dark blue (short).

embeddings of large graphs, leading also to the development of still popular force-directed approaches [3,18,6,**?**]. In contrast, *graph representation learning methods* (sometimes also referred to as *network embedding methods*) have not been developed specifically for visualization, but rather they aim to embed graphs into a low-dimensional space (depending on the method between 8 to 128 dimensions in general), to enable subsequent use of common machine learning techniques.

There exist many graph-layout algorithms, each with a different objective or optimization procedure. Someone who wants to inspect or analyze a graph visually is then already faced with a difficult choice of which method would be most appropriate for their goal. This difficulty is amplified by the introduction of dozens of representation learning methods for graphs, which have been premised to also produce good visualizations (see, e.g., [15,9,17,10]).

It is not clear which method to use, or even which aspects vary by choosing one method instead of another. For example, in Figure 1 we show visualizations from different representation learning methods. It is not obvious which visualization is better than another or how they differ.

Representation learning methods are often evaluated by coloring the vertices according to a class label, and a good visualization is then one where nodes from the same class are (visually) grouped together. But such evaluations have limited scope, because whether an algorithm does well depends strongly on what type of structure from the graph aligns with the node labels.

New graph layout methods are evaluated on more immediate quality measures, but are not compared to representation learning methods (see, e.g., [9,21]). It is an open question how to generally evaluate graph layouts and whether some currently used methods outperform others.

Contributions. In this paper, we perform an empirical analysis of graph visualizations by representation learning and graph layout algorithms. We investigate the differences in their visualizations visually and quantitatively using a diverse set of quality measures. Representation learning methods are often used to compute node embeddings in more than two dimensions. We thus evaluate native two-dimensional and also higher-dimensional node representations that we em-

bed into two dimensions using the well-known t-distributed stochastic neighbor embedding (t-SNE) method. We base our analysis on publicly available methods and provide the full source code, which can be easily extended to include other methods or datasets. Lastly, we give recommendations on how to apply t-SNE to retain important graph structure and how different methods could be improved to yield better visualizations.

2 Setup of the study

In this section we introduce our choice of quality measures, graphs, and evaluated methods. The definitions of the quality measures are provided in the supplement.

2.1 Quality measures

There exist a variety of scores to measure the quality of graph layouts. The graph representation learning community typically evaluates graph embeddings on tasks such as clustering, link prediction, graph reconstruction, or node classification. These are inspired by and suitable to evaluate embeddings as models of the data, either considering the performance of an embedding towards reconstruction or prediction. In contrast, recent methods for graph visualization ([9,10,21]) are mostly evaluated by means of readability, shape-based, or distance-based measures. Such measures inform more directly about the quality of a graph layout, as they are easy to interpret, but they may not capture to what extent we can do inference and prediction using a layout.

As the measures may be complementary to each other, we evaluate graph layouts on measures from each of these groups: *Crosslessness* [16,10] measures the proportion of edge crossings that are avoided in a graph layout with respect to all possible pairwise edge crossings. *Minimum angle* [16], measures how much the smallest angle between incident edges at each node deviate from the optimal angle that could be achieved when all edges are evenly spaced. *Edge length variation* [4] quantifies the variability of the edge lengths relative to their mean. *Gabriel graph similarity* [2] is a shape-based quality measure for layouts of large graphs. We use the GLAM (Graph Layout Aesthetic Metrics) toolbox [10] to compute these four readability measures. The distance-based measures that we evaluate are *neighborhood preservation* [9] and *stress* [21].

Finally, we use the EvalNE toolbox [13] to quantify the suitability of the layouts for *link prediction*. For link prediction, a random connected subset of 80% of the edges is used to compute the layout and the remaining 20% together with the same number of randomly sampled non-existing edges are used for testing. While some methods have built-in edge embeddings (AROPE, CNE, GAE), others only result in node embeddings. In the latter case, we use EvalNE to transform the node embeddings into edge embeddings using their element-wise average, Hadamard product, L_1, or L_2 distance. The measure reported is the AUC-ROC of a logistic classifier that predicts the test edge probabilities.

Table 1: Characteristics of graphs.

Name	$\|V\|$	$\|E\|$	Type	Source
karate club	34	78	social network	[1]
can96	96	429	artificial network	[1]
netscience	379	914	co-authorship graph	[1]
facebook	4 039	88 234	social network	[11]
powergrid	4 941	6 594	infrastructure graph	[1]
twitter	44 631	73 133	social network	[19]

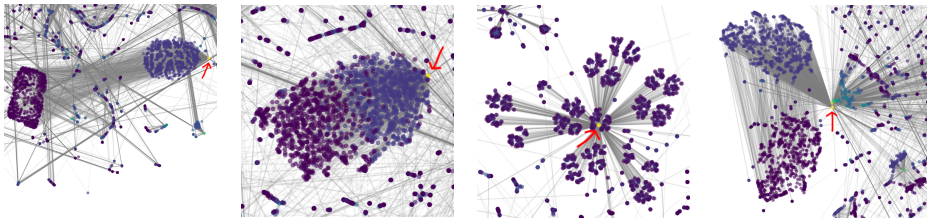
2.2 Datasets

We evaluate visualizations of the largest connected components of six undirected and unweighted graphs listed in Table 1. The *karate club* graph is a commonly used example and models the social interactions of the sports club members. *can96* is an artificial mesh structure, *netscience* a co-authorship graph from network scientists, and *powergrid* represents the electricity grid of the western United States. The *facebook* friendship graph is commonly evaluated on the link prediction task. The largest graph, a *twitter* feed graph, has been used in the experiments by Zellmann et al. [19] and provided to us by the authors.

2.3 Evaluated methods

In this study we compare graph visualizations from seven methods, with parameter settings and links to their source code in Table 11 of the Supplement. We followed the parameter recommendations of the original publications where possible. We selected four graph representation learning methods: Arbitrary-order Proximity Preserved Network Embedding (AROPE), Conditional Network Embedding (CNE), DeepWalk (DW), and Graph Auto-Encoders (GAE), which are state-of-the-art methods each using a distinct approach [14]. These methods are mostly used to compute low-dimensional node embeddings in more than two dimensions and the two-dimensional graph layouts for visualization are then obtained by t-SNE [12]. We denote their node embeddings by, e.g., AROPE128, indicating the dimensionality before applying t-SNE. We also evaluate native two-dimensional node representations for all graph representation learning methods denoted by e.g., AROPE2.

The representatives for graph layout methods are the Fruchterman-Reingold (FR) algorithm and DRGraph. We chose FR due to its popularity and evaluate its naive implementation as well as a recent more scalable grid-based optimization. DRGraph is a recently published method that is faster than existing layout techniques but produces visualizations with comparable readability and neighborhood preservation.



(a) AROPE128 (b) CNE16 (c) DW128 (d) GAE16

Fig. 2: Placement of low-degree nodes (dark) around a high-degree hub node (yellow, marked with red arrow) for the *twitter* graph. While nodes are clustered according to their degree in AROPE128 and GAE16, and to some extend in CNE16, leaf nodes are close to their parent in DW128.

Arbitrary-order Proximity Preserved Network Embedding [20] preserves higher-order proximities by factorizing a linear combination of powers of the adjacency matrix. It is efficient through the use of eigen-decomposition reweighting.

Conditional Network Embedding [7] is a probabilistic network embedding method based on Maximum Likelihood Estimation that can factor out prior knowledge about the graph structure using a Bayesian approach.

DeepWalk [15] captures the neighborhood structure of nodes using random walks. The node embeddings are updated using the Skip-gram model, maximizing the co-occurrence probability of nodes in the same neighborhood.

Graph Auto-Encoders [8] embed the graph into a latent space using a graph convolutional network encoder and an inner product decoder.

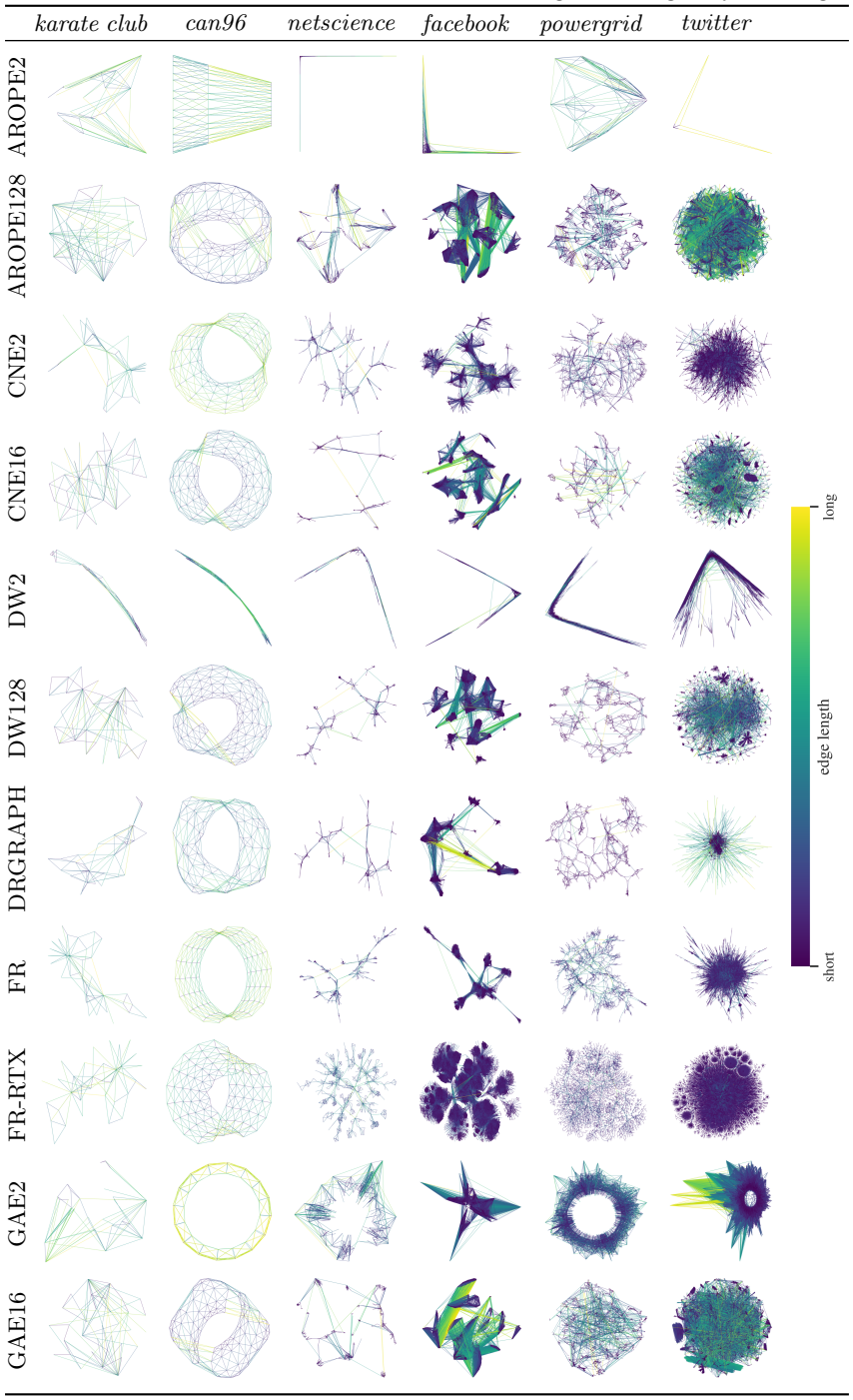
Fruchterman-Reingold [3] is a force-directed approach with attraction forces between connected nodes and repulsion between all nodes. Node positions are updated iteratively with a temperature cooling scheme that reduces the displacement of nodes over time. We compare the $\mathcal{O}(|V|^2)$ implementation of Fruchterman-Reingold by Hagberg et al. [5] (version 2.2) with a GPU-optimized implementation by Zellmann et al. [19], denoted as FR-RTX.

DRGraph [21] preserves the first-order neighbors from the graph in the low-dimensional representation by minimizing the difference between node similarities and layout proximities. It is based on the idea of tsNET [9] but improved the computational and memory complexity by introducing a sparse distance matrix, negative sampling, and a multi-level optimization process.

3 Empirical results

We first compare the graph layouts qualitatively based on the visualizations in Table 2 and then evaluate the quantitative measures. The code and the embeddings are available at https://github.com/aida-ugent/graph-vis-eval. The Sup-

Table 2: Visualizations based on node embeddings showing only the edges.



plement contains node-link diagrams (Table 12) and scores for all layouts (Tables 1 to 10).

3.1 Visual inspection of the embeddings

AROPE. The first embedding coordinate of graph layouts by AROPE2 is proportional to the eigenvector centrality of the nodes, making the embedding unlike the others. Central nodes may be identified on one side and leaf nodes on the other side of the embedding for *karate club*, *can96*, and *powergrid*. The nodes for the other three graphs are mostly aligned along two axes, thus hiding most of their connections. Embeddings by AROPE128 reveal more details of the graph structure. In the *twitter* graph layout in Figure 2a we observe a clustering according to node-degree.

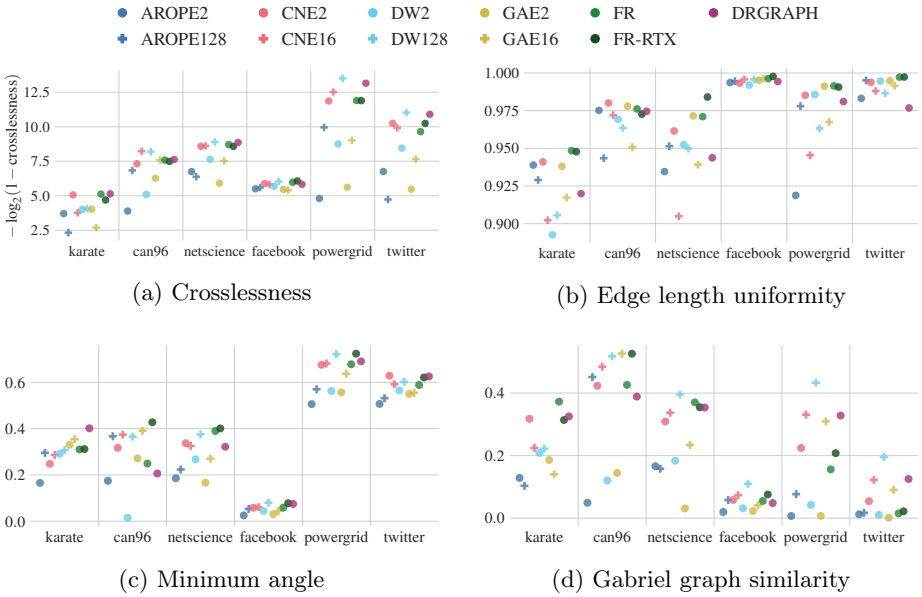
CNE. The native two-dimensional embeddings of CNE2 exhibit a proximity-based arrangement of the nodes where hub nodes are placed in the center of their connections. The embeddings by CNE16 are similar to AROPE128, DW128, and GAE16. For the *twitter* graph, we find the t-SNE embedding is more readable as it shows cluster structure. We see in Figure 2b that CNE16 also clusters nodes by degree.

DeepWalk. The embeddings by DW2 conceal the underlying shape of the graph as nodes are mostly arranged on a curved line. DW128 produces readable embeddings with a clear cluster structure. In Figure 2c we note that DW128 embeds low-degree nodes close to their connections resulting in a star shape around the hub node.

DRGraph. The graph layouts by DRGRAPH for *netscience* and *powergrid* are appealing and very similar to the layouts by DW128. The different communities of the *facebook* graph are well visible, but in the node-link diagram (Supplement, Table 13) we notice some long edges that have a leaf node on one end. Long edges also dominate the visualization of the *twitter* graph and make it difficult to observe any structure in the center of the layout. We do not know whether the parameter settings or the fact that DRGRAPH only preserves first-order graph distances cause the 'hairball' structure of this visualization.

FR. Both implementations result in graph layouts with similar global structure but different local node arrangement. For FR, we observe that clusters are more compact and nodes are pushed away from the center. For FR-RTX, nodes are distributed more evenly around a shared connection, which hinders the formation of visible clusters for *powergrid* but improves the readability of *twitter*. We assume this is the effect of approximating the repulsive forces in FR-RTX, causing only the closest nodes to repel each other.

GAE. The embeddings from GAE2 have a distinct circular shape due to the inner product decoder. High-degree node embeddings have large coordinates and low-degree nodes are placed near the origin. In this graph layout, we can easily identify the most central hub nodes, e.g., for *facebook* or *twitter*. The embeddings



(a) Crosslessness

(b) Edge length uniformity

(c) Minimum angle

(d) Gabriel graph similarity

Fig. 3: Average readability scores for each dataset, ordered by number of nodes. Crosslessness and edge length uniformity are scaled to better show the relative differences between methods.

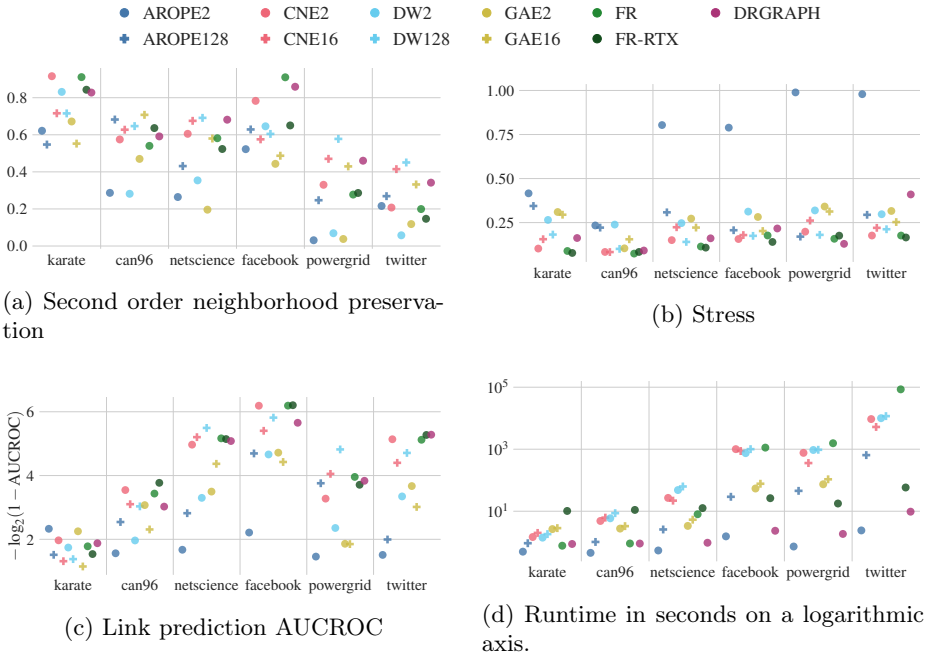
by GAE16 and AROPE128 have an similar local structure as shown in Figure 2d. In both embeddings, nodes from the same cluster are arranged by degree.

3.2 Readability measures

We show the scores for crosslessness, edge-length uniformity, minimum angle, and Gabriel graph similarity, averaged over four runs, in Figure 3. Averaging over all graph datasets, the graph layout methods outperform the representation learning methods (Supplement Table 10). DW128 achieves high scores for the layouts of larger graphs. The high minimum-angle scores of the layouts by FR-RTX and DW128 stem from the star-shaped arrangement of hub-nodes and their connections (see Figure 2c). The layouts by AROPE2, DW2, and GAE2 generally have small angles between incident edges as they optimize the dot-product similarity of connected nodes. Further, we observe that AROPE2, DW2, and GAE2 score poorly for the Gabriel shape measure. These methods optimize the node embeddings for dot-product similarity whereas the Gabriel graph similarity measure retrieves neighbors based on Euclidean distance.

3.3 Distance-based measures

We show the scores for second-order neighborhood preservation and stress in Figures 4a and 4b, and the scores of the first-order neighborhood preservation



(a) Second order neighborhood preservation

(b) Stress

(c) Link prediction AUCROC

(d) Runtime in seconds on a logarithmic axis.

Fig. 4: Average neighborhood preservation, stress, AUC-ROC, and runtime over four experiment repetitions. We scale the AUC values to better show the relative differences.

measure in the Supplement; they are highly similar to the Gabriel similarity measure. DW128 and DRGRAPH preserve the second-order neighborhood best. Considering all methods, the neighborhoods of the *facebook* graph are better preserved than the neighborhoods of *powergrid* or *twitter*. We presume that the community structure of the graph aligns well with the second-order neighborhoods. FR places these communities far apart and thus achieves the highest score. The differences in the stress measure are subtle. Averaged over all datasets, FR and FR-RTX result in layouts with lowest stress. The t-SNE based embeddings from AROPE128, DW128, and GAE16 are more distance faithful than their native two-dimensional node embeddings but the opposite holds for CNE2 and CNE16.

3.4 Link-prediction

Results are presented in Figure 4c. No single method achieves the highest link prediction score on more than two datasets. Averaging over all datasets, the graph layouts based on the Fruchterman-Reingold algorithm result in the highest link predict scores. While it is difficult to identify much structure in the hairball-

shaped *twitter* graph layouts by FR and DRGRAPH they score highly on the link-prediction task.

3.5 Runtime

The average runtime to embed the whole graphs using an Intel Xeon CPU E5-2620 v4 @ 2.10GHz with one GeForce GTX 1080Ti is depicted in Figure 4d. We ran the experiments for GAE on *twitter* on a different machine with 256GB RAM resulting in mean runtimes of 5020s for GAE2 and 4635s for GAE16.

We notice that AROPE2 is the fastest method and that the runtime increase for AROPE128 is mainly caused by t-SNE, which took about 22, 43, and 660 seconds to reduce the dimensionality from 128 to 2 for *facebook*, *powergrid*, and *twitter* respectively. FR-RTX has a slightly larger runtime than the other methods on the smaller graphs possibly due to a small startup cost for a user-interface. Notably, on the twitter dataset, FR-RTX runs in less than a minute, while FR takes almost 24 hours. CNE16 has lower runtime than CNE2 on the larger graphs as the optimization of the 16-dimensional node representations stabilizes earlier than the two-dimensional representations.

4 Discussion

In this study we have shown that visualizations by graph layout methods scored higher on the chosen quality measures than the native two-dimensional node embeddings by representation learning methods. The combination of DeepWalk with t-SNE resulted in visualizations with the best local neighborhood preservation and highest scores in Gabriel graph similarity but does not scale to larger graphs. We believe that there is great potential in comparing a wider range of scalable graph layout and representation learning methods on real-world graphs with millions of nodes.

The standard definitions of Gabriel graph similarity, neighborhood preservation, and stress all assume that the graph-theoretical distances are reflected by the Euclidean distances in the low-dimensional embedding. Methods such as AROPE, DeepWalk, and GAE optimize the embedding based on dot-product similarity. From this perspective, it is not surprising that the graph layouts by AROPE2, DW2, and GAE2 score worse on these measures. Although other distances than Euclidean may be interpretable, it is not obvious to what extent that may be the case. In addition to the quality measures, the standard version of t-SNE also defines high and low-dimensional neighborhoods based on Euclidean distance. To retain the graph neighborhoods, we would have to adjust the similarity definition of t-SNE.

4.1 Recommendations for practical use

It is important to note that the choice of method is not universal and depends on the task for which the visualization is used. The designer of the visualization should ask the question which quality measure is most important to judge

the effectiveness of the visualization, for that task. For example, out-of-sample quality measures like link prediction performance may not be important in the context when a static graph is to be analyzed that does not have any missing edges. In other contexts, generalizability of the proximity of nodes, for which link prediction performance is a proxy, could be desirable.

No single winner emerged from the comparisons of graph representation learning with graph layout methods. Nonetheless, some general patterns did emerge:

Graph layout methods. We found that FR and FR-RTX resulted in readable layouts, score best on stress minimization, and achieved the highest link-prediction scores. The latter is especially surprising as the representation learning methods are focused on generalization performance, through use of higher-order information (paths, convolutions), supervised learning (use of negative edges), or both.

DRGRAPH is a fast method that preserves the local graph neighborhoods better than both versions of Fruchterman-Reingold. While the neighborhood preservation decreases for larger graphs, it may be still the best choice to use either of the computationally efficient graph layout methods instead of a representation learning method.

Graph representation learning methods. While the native two-dimensional embeddings of CNE2 perform well in terms of quality, it appears that the only reason to consider the AROPE2, DW2, and GAE2 embeddings is to better understand the method itself. They are outperformed, typically by a wide margin, by the t-SNE embeddings of higher-dimensional node representations. DW128 is arguably the best representation learning method included in the evaluation. Compared to DRGRAPH, it provides a more readable visualization for the difficult *twitter* graph, but is also 1000 times slower. For larger graphs, we expect the relative difference in runtime to grow further.

4.2 Future work

Improving the visual quality. We observed that the naive Fruchterman-Reingold algorithm results in embeddings where cluster structure is more pronounced compared to the grid optimized version where nodes are distributed evenly across the layout. Adjusting the repulsive forces to depend on node degrees as in ForceAtlas2 [6] could improve the perceived clusteredness of the layout, by embedding leaf nodes closer to their parent.

Combination of graph representation learning with t-SNE. The t-SNE visualizations of the large *twitter* graph (CNE16, DW128, and GAE16) are of high quality and suggest that preserving only the small distances of a low-dimensional node representation avoids the 'hairball' embedding. It appears this allows deriving an integrated method that could outperform existing methods. Besides, it would be worthwhile to improve the scalability of the DW128 combination.

Finally, we hope this evaluation inspires follow-up work to further connect the representation learning with the graph drawing community.

Acknowledgements This research was funded by the ERC under the EU's 7th Framework and H2020 Programmes (ERC Grant Agreement no. 615517 and 963924), the Flemish Government ("Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen"), and the FWO (project no. 11J2322N, G0F9816N, 3G042220).

References

1. Davis, T.A., Hu, Y.: The university of florida sparse matrix collection. ACM Transactions on Mathematical Software (TOMS) **38**(1), 1–25 (2011)
2. Eades, P., Hong, S.H., Nguyen, A., Klein, K.: Shape-Based Quality Metrics for Large Graph Visualization. JGAA **21**(1), 29–53 (2017)
3. Fruchterman, T., Reingold, E.: Graph drawing by force-directed placement. Software-Practice Exp. **21**(11), 1129–1164 (1991)
4. Hachul, S., Jünger, M.: Large-graph layout algorithms at work: An experimental study. JGAA **11**(2), 345–369 (2007)
5. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab. (2008)
6. Jacomy, M., Venturini, T., Heymann, S., Bastian, M.: ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. PLoS ONE **9**(6), e98679 (Jun 2014)
7. Kang, B., Lijffijt, J., De Bie, T.: Conditional network embeddings. In: Proc. of ICLR (2019)
8. Kipf, T.N., Welling, M.: Variational graph auto-encoders. NeurIPS Workshop on Bayesian Deep Learning (2016)
9. Kruiger, J.F., Rauber, P.E., Martins, R.M., Kerren, A., Kobourov, S., Telea, A.C.: Graph layouts by t-SNE. Computer Graphics Forum **36**(3), 283–294 (Jun 2017)
10. Kwon, O.H., Crnovrsanin, T., Ma, K.L.: What Would a Graph Look Like in This Layout? A Machine Learning Approach to Large Graph Visualization. IEEE TVCG **24**(1), 478–488 (2018)
11. Leskovec, J., McAuley, J.: Learning to discover social circles in ego networks. NeurIPS **25**, 539–547 (2012)
12. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. JMLR **9**(11) (2008)
13. Mara, A., Lijffijt, J., De Bie, T.: EvalNE: A framework for evaluating network embeddings on link prediction. In: Reproducibility in Machine Learning Workshop@ICLR (2019)
14. Mara, A.C., Lijffijt, J., De Bie, T.: Benchmarking network embedding models for link prediction: Are we making progress? In: Proc. of IEEE DSAA. pp. 138–147. IEEE (2020)
15. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proc. of ACM SIGKDD. pp. 701–710 (2014)
16. Purchase, H.C.: Metrics for graph drawing aesthetics. JVLC **13**(5), 501–516 (2002)
17. Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: Verse: Versatile graph embeddings from similarity measures. In: Proc. of WWW Conference. pp. 539–548 (2018)

18. Walshaw, C.: A multilevel algorithm for force-directed graph drawing. In: International Symposium on Graph Drawing. pp. 171–182. Springer (2000)
19. Zellmann, S., Weier, M., Wald, I.: Accelerating force-directed graph drawing with RT cores. In: Proc. of IEEE VIS. pp. 96–100 (2020)
20. Zhang, Z., Cui, P., Wang, X., Pei, J., Yao, X., Zhu, W.: Arbitrary-order proximity preserved network embedding. In: Proc. of ACM SIGKDD. pp. 2778–2786 (2018)
21. Zhu, M., Chen, W., Hu, Y., Hou, Y., Liu, L., Zhang, K.: Drgraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. IEEE TVCG **27**(2), 1666–1676 (2020)

Evaluating Representation Learning and Graph Layout Methods for Visualization (Supplemental Material)

Edith Heiter, Bo Kang, Tijl De Bie, and Jefrey Lijffijt

Department of Electronics and Information Systems, IDLab, Ghent University, 9000 Ghent, Belgium
first.last@ugent.be

Evaluation measures

We denote an undirected, unweighted, and connected graph $G = (V, E)$ by the set of nodes V with $|V| = n$ and the set of edges $E = \{(i, j)\} \subseteq V \times V$. We refer to the graph-theoretical shortest-path distance between nodes i and j by d_{ij}. With $N(i, h) = \{j \in V | \mathrm{d}_{ij} \leq h, i \neq j\}$, we denote set of nodes in the h-order neighborhood of node i. With the following readability measures we evaluate two-dimensional graph layouts $Y \in \mathbb{R}^{n \times 2}$, where the embedding of a node i is denoted as $y_i \in \mathbb{R}^2$ and the edges are drawn using straight lines.

Crosslessness [6] measures the proportion of edge crossings that are avoided in a graph layout with respect to all possible pairwise edge crossings. We adopt the definition by Kwon et al. [4]

$$\text{crosslessness} = \begin{cases} 1 - \frac{c}{c_{\max}} & c_{\max} > 0 \\ 1 & \text{otherwise} \end{cases},$$

where c is the number of crossings and $c_{\max}$ the upper bound on the number of crossings. $c_{\max}$ is defined as $\frac{|E|(|E|-1)}{2} - \frac{1}{2}\sum_{i \in V} |N(i,1)|(|N(i,1)|-1)$, where $|N(i,1)|$ is the degree of node i.

Minimum angle [6] measures how much the smallest angle between incident edges at each node deviates from the optimal angle that could be achieved when all edges are evenly spaced.

$$\text{minimum angle} = 1 - \frac{1}{|V|} \sum_{i \in V} \frac{|\theta(i) - \theta_{\min}(i)|}{\theta(i)},$$

where $\theta(i) = \frac{360}{|N(i)|}$ is the optimal angle at node i and $\theta_{\min}(i)$ is the minimum angle between any two edges incident to node i.

Edge length variation [2] measures the variability of the edge lengths relative to their mean. It is defined as the coefficient of variance

$$l_{cv} = \frac{l_\sigma}{l_\mu} = \sqrt{\frac{\sum_{e \in E}(l_e - l_\mu)^2}{|E| \cdot l_\mu^2}}.$$

To be in line with the other readability measures where higher values are better, we define for graphs with $|E| > 1$,

$$\text{edge length uniformity} = 1 - \frac{l_{cv}}{\sqrt{|E|-1}} \ ,$$

where the coefficient of variance is normalized by its upper bound.

Gabriel graph similarity [1] is a shape-based quality measure for layouts of large graphs. It is based on the intuition that a representation of the global graph structure can be accurate despite a high number of edge crossings. It is defined as the mean Jaccard similarity

$$\frac{1}{|V|} \sum_{i \in V} \frac{|N_G(i,1) \cap N_S(i,1)|}{|N_G(i,1) \cup N_S(i,1)|} \ ,$$

between the first-order neighborhood in the original graph N_G and the neighborhood in the Gabriel graph N_S. The Gabriel graph over the set of nodes V has an edge connecting two nodes i and j if a disk with y_i and y_j on its boundary and diameter $\|y_i - y_j\|$ does not contain any other node.

Neighborhood preservation [3] is defined as the Jaccard similarity

$$\frac{1}{|V|} \sum_{i=1}^{N} \frac{|N_G(i,h) \cap N_Y(y_i,k_i)|}{|N_G(i,h) \cup N_Y(y_i,k_i)|} \ ,$$

where $N_G(i,h)$ is the graph neighborhood of node i with distance at most h and $N_Y(y_i,k_i)$ are the $k_i = |N_G(i,h)|$ nearest neighbors based on Euclidean distances in the embedding. We evaluate the neighborhood preservation for $h \in \{1,2\}$.

Stress measures how accurate a graph layout represents the actual graph-theoretic node distances. We define the stress similar to Zhu et al. [7] as

$$\frac{2}{|V|(|V|-1)} \sum_{i<j \in V} \left(\frac{\alpha \|y_i - y_j\| - \mathrm{d}_{ij}}{\mathrm{d}_{ij}} \right)^2 \ ,$$

where $\alpha \geq 0$ scales the embedding to minimize the stress as follows:

$$\begin{aligned}
\arg\min_{\alpha}(\text{stress}) &= \arg\min_{\alpha} \sum_{i<j \in V} \frac{1}{\mathrm{d}_{ij}^2} \left(\alpha \|y_i - y_j\| - \mathrm{d}_{ij}\right)^2 \\
&= \arg\min_{\alpha} \sum_{i<j \in V} \left(\frac{\alpha \|y_i - y_j\| - \mathrm{d}_{ij}}{\mathrm{d}_{ij}} \right)^2 \\
&= \arg\min_{\alpha} \sum_{i<j \in V} \left(\frac{\alpha^2 \|y_i - y_j\|^2}{\mathrm{d}_{ij}^2} - \frac{2\alpha \|y_i - y_j\|}{\mathrm{d}_{ij}} + 1 \right)
\end{aligned}$$

Taking the derivative with respect to α and equating to zero yields:

$$\alpha = \frac{\sum_{i<j \in V} \frac{\|y_i - y_j\|}{\mathrm{d}_{ij}}}{\sum_{i<j \in V} \frac{\|y_i - y_j\|^2}{\mathrm{d}_{ij}^2}} \ .$$

Evaluation Results

Table 1: First-order neighborhood preservation

	karate	can96	netscience	facebook	powergrid	twitter
AROPE2	.215	.133	.242	.131	.008	.014
AROPE128	.128	.678	.317	.325	.102	.026
CNE2	.412	.521	.470	.279	.289	.070
CNE16	.277	.615	.514	.321	.389	.168
DW2	.310	.166	.285	.183	.048	.012
DW128	.261	.668	.640	.417	.506	.277
DRGRAPH	.439	.509	.569	.258	.409	.174
FR	.383	.392	.453	.285	.151	.019
FR-RTX	.377	.641	.532	.307	.250	.024
GAE2	.283	.264	.145	.124	.010	.002
GAE16	.150	.698	.403	.230	.371	.130

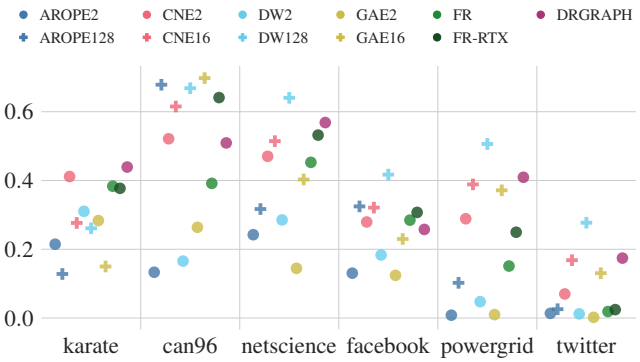


Fig. 1: First-order neighborhood preservation

Table 2: Crosslessness

	karate	can96	netscience	facebook	powergrid	twitter
AROPE2	.9232	.9327	.9907	.9779	.9642	.9907
AROPE128	.7999	.9912	.9880	.9794	.9990	.9623
CNE2	.9701	.9937	.9974	.9831	.9997	.9992
CNE16	.9267	.9967	.9974	.9823	.9998	.9990
DW2	.9383	.9709	.9950	.9806	.9977	.9971
DW128	.9405	.9966	.9979	.9847	.9999	.9995
DRGRAPH	.9716	.9949	.9979	.9824	.9999	.9995
FR	.9713	.9948	.9976	.9841	.9997	.9988
FR-RTX	.9613	.9944	.9974	.9851	.9997	.9992
GAE2	.9386	.9870	.9834	.9772	.9796	.9775
GAE16	.8449	.9948	.9946	.9764	.9981	.9951

Table 3: Edge length uniformity

	karate	can96	netscience	facebook	powergrid	twitter
AROPE2	.939	.975	.935	.994	.919	.983
AROPE128	.929	.943	.951	.995	.978	.995
CNE2	.941	.980	.962	.993	.985	.994
CNE16	.902	.972	.905	.996	.945	.988
DW2	.893	.969	.952	.992	.986	.995
DW128	.906	.964	.950	.996	.963	.987
DRGRAPH	.920	.975	.944	.994	.981	.977
FR	.948	.976	.971	.996	.991	.997
FR-RTX	.948	.973	.984	.998	.991	.997
GAE2	.938	.978	.972	.995	.991	.995
GAE16	.917	.951	.939	.996	.967	.992

Table 4: Minimum angle

	karate	can96	netscience	facebook	powergrid	twitter
AROPE2	.166	.175	.186	.025	.506	.507
AROPE128	.295	.367	.224	.053	.570	.532
CNE2	.248	.317	.337	.058	.676	.629
CNE16	.286	.374	.325	.061	.682	.592
DW2	.291	.015	.267	.044	.562	.565
DW128	.308	.365	.375	.080	.722	.602
DRGRAPH	.401	.206	.322	.075	.691	.626
FR	.310	.249	.390	.059	.678	.588
FR-RTX	.312	.427	.401	.078	.724	.621
GAE2	.330	.271	.167	.031	.557	.550
GAE16	.354	.391	.270	.044	.637	.555

Table 5: Gabriel shape

	karate	can96	netscience	facebook	powergrid	twitter
AROPE2	.129	.049	.166	.020	.007	.012
AROPE128	.103	.451	.158	.058	.077	.017
CNE2	.318	.423	.309	.058	.225	.054
CNE16	.225	.484	.337	.073	.331	.122
DW2	.209	.120	.184	.031	.042	.010
DW128	.222	.518	.395	.110	.433	.195
DRGRAPH	.325	.389	.354	.048	.328	.125
FR	.373	.426	.370	.055	.156	.015
FR-RTX	.314	.526	.355	.076	.208	.022
GAE2	.186	.144	.031	.023	.007	.001
GAE16	.140	.526	.234	.041	.309	.090

Table 6: Second-order neighborhood preservation

	karate	can96	netscience	facebook	powergrid	twitter
AROPE2	.622	.287	.264	.522	.031	.216
AROPE128	.547	.682	.431	.629	.247	.269
CNE2	.916	.575	.605	.783	.330	.208
CNE16	.716	.627	.675	.575	.470	.415
DW2	.831	.282	.354	.646	.069	.058
DW128	.715	.647	.691	.605	.578	.451
DRGRAPH	.828	.591	.681	.859	.460	.342
FR	.911	.540	.582	.910	.278	.199
FR-RTX	.843	.636	.523	.650	.286	.147
GAE2	.672	.470	.196	.443	.038	.118
GAE16	.552	.708	.580	.487	.429	.332

Table 7: Stress

	karate	can96	netscience	facebook	powergrid	twitter
AROPE2	.416	.234	.804	.789	.989	.979
AROPE128	.344	.221	.308	.207	.170	.294
CNE2	.103	.083	.150	.157	.198	.177
CNE16	.156	.083	.224	.179	.261	.220
DW2	.265	.239	.247	.312	.319	.297
DW128	.182	.101	.140	.175	.180	.212
DRGRAPH	.162	.092	.161	.217	.129	.410
FR	.089	.074	.114	.177	.158	.176
FR-RTX	.078	.083	.108	.140	.176	.166
GAE2	.310	.104	.273	.282	.341	.316
GAE16	.295	.155	.222	.202	.313	.253

Table 8: Link prediction AUC-ROC

	karate	can96	netscience	facebook	powergrid	twitter
AROPE2	.8011	.6613	.6863	.7843	.6361	.6492
AROPE128	.6500	.8284	.8583	.9613	.9261	.7490
CNE2	.7445	.9143	.9680	.9863	.8968	.9716
CNE16	.5978	.8834	.9729	.9764	.9397	.9526
DW2	.7011	.7438	.8985	.9605	.8046	.9016
DW128	.6156	.8779	.9778	.9823	.9646	.9618
DRGRAPH	.7278	.8772	.9705	.9801	.9300	.9743
FR	.7089	.9075	.9721	.9863	.9356	.9713
FR-RTX	.6556	.9268	.9717	.9865	.9236	.9741
GAE2	.7900	.8811	.9114	.9620	.7246	.9214
GAE16	.5489	.7983	.9516	.9535	.7227	.8763

Table 9: Runtime in seconds with logarithmic color scaling.

	karate	can96	netscience	facebook	powergrid	twitter
AROPE2	0.5	0.5	0.5	1.5	0.7	2.4
AROPE128	0.9	1.0	2.6	29.2	45.4	645.2
CNE2	1.5	4.9	26.9	1006.3	763.2	9382.6
CNE16	2.0	6.2	21.9	874.3	357.8	5225.0
DW2	1.4	5.9	48.2	747.2	944.3	10023.1
DW128	1.8	8.6	62.8	1004.4	959.6	11681.6
DRGRAPH	0.9	0.9	1.0	2.3	1.9	9.7
FR	0.8	0.9	8.0	1125.6	1559.9	85345.0
FR-RTX	10.2	11.0	12.6	26.3	17.7	58.2
GAE2	2.7	2.8	3.4	54.1	74.1	5019.7
GAE16	2.9	3.3	5.3	77.0	107.0	4635.3

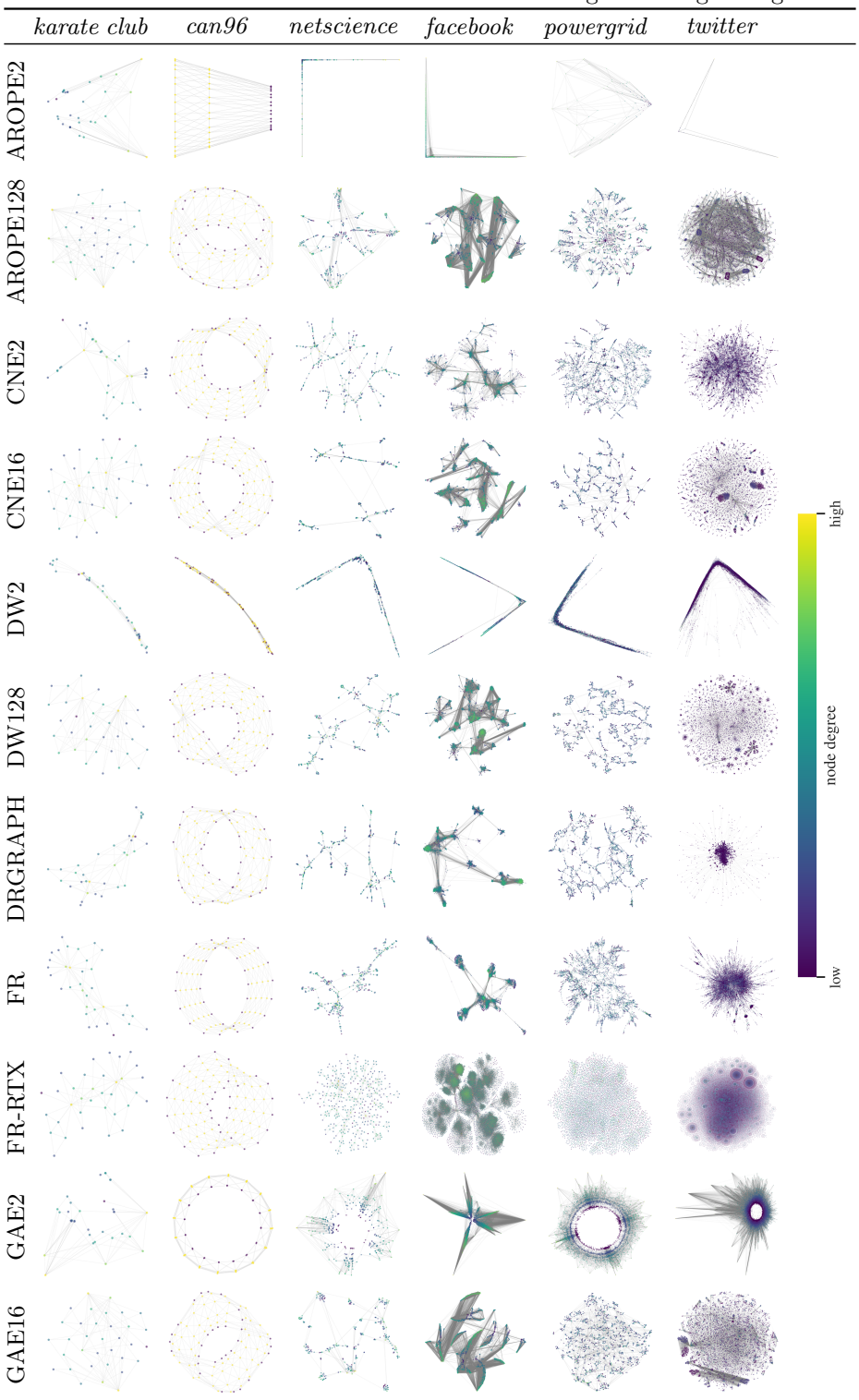
Table 10: Mean ranks of all methods on all quality measures. For each quality measure and network dataset we ranked all methods and averaged these ranks over the datasets.

	crosslessness	edgelengthuniformity	minimumangle	shapegabriel	1np	2np	stress	AUCROC
AROPE2	9.7	8.0	10.7	10.2	10.0	9.2	10.8	9.3
AROPE128	9.2	6.7	7.5	7.7	6.2	6.7	7.7	8.2
CNE2	4.7	4.8	5.5	5.0	4.8	4.5	3.5	4.0
CNE16	4.5	8.3	5.0	3.7	4.2	4.3	5.0	5.0
DW2	7.8	7.3	8.5	8.8	8.7	8.2	8.8	8.3
DW128	2.0	8.0	3.5	2.2	2.5	3.3	4.5	4.3
DRGRAPH	2.5	7.5	4.0	4.3	3.5	3.5	5.7	4.3
FR	3.8	2.0	5.2	5.0	6.2	5.2	2.2	3.3
FR-RTX	4.3	2.3	2.0	3.8	4.7	5.5	2.0	3.5
GAE2	9.3	3.5	8.3	9.8	9.7	9.8	8.8	6.3
GAE16	8.2	7.5	5.8	5.5	5.7	5.8	7.0	9.3

Table 11: Evaluated methods with parameter settings. The values in bold either deviate from the default settings or did not have given default values. The abbreviations for graph representation learning methods include the dimensionality of the node representations. For AROPE, we use only 10 instead of 128 embedding dimensions for the smaller networks *karate club*, *can96*, and *netscience* before applying t-SNE.

	Method	Type	Parameters	Abbrv.
Graph representation learning	Arbitrary-order Proximity Preserved Network Embedding https://github.com/ZW-ZHANG/AROPE	matrix factorization	order 3, weights [1,0.1,0.01]	AROPE2, AROPE128
	Conditional Network Embedding https://bitbucket.org/ghentdatascience/cne	probabilistic	**learning rate 0.05, ftol e^{-3}, epochs 1000**	CNE2, CNE16
	DeepWalk https://github.com/phanein/deepwalk	random walks, Skip-gram	number-walks 80, walk-length 40, window-size 10, **workers 4**	DW2, DW128
	Graph Auto-Encoders https://github.com/tkipf/gae	auto-encoder	default parameters with features=0	GAE2, GAE16
Graph layout	Fruchterman-Reingold https://networkx.org/documentation/networkx-2.2/reference/generated/networkx.drawing.layout.spring_layout.html	spring-electrical, force-directed	**iterations 1000**	FR
	Fruchterman-Reingold with ray-tracing acceleration https://github.com/owl-project/owl-graph-drawing	force-directed, spring-electrical	bench true, mode rtx, **epochs 10000**	FR-RTX
	DRGraph https://github.com/ZJUVAG/DRGraph	distance based dimensionality reduction	neg 5, samples 400, gamma 0.1, mode 1, A 2, B 1	DRGRAPH

Table 12: Node-link visualization with node coloring according to degree.



References

1. Eades, P., Hong, S.H., Nguyen, A., Klein, K.: Shape-Based Quality Metrics for Large Graph Visualization. JGAA **21**(1), 29–53 (2017)
2. Hachul, S., Jünger, M.: Large-graph layout algorithms at work: An experimental study. JGAA **11**(2), 345–369 (2007)
3. Kruiger, J.F., Rauber, P.E., Martins, R.M., Kerren, A., Kobourov, S., Telea, A.C.: Graph layouts by t-SNE. Computer Graphics Forum **36**(3), 283–294 (Jun 2017)
4. Kwon, O.H., Crnovrsanin, T., Ma, K.L.: What Would a Graph Look Like in This Layout? A Machine Learning Approach to Large Graph Visualization. IEEE TVCG **24**(1), 478–488 (2018)
5. Mara, A., Lijffijt, J., De Bie, T.: EvalNE: A framework for evaluating network embeddings on link prediction. In: Reproducibility in Machine Learning Workshop@ICLR (2019)
6. Purchase, H.C.: Metrics for graph drawing aesthetics. JVLC **13**(5), 501–516 (2002)
7. Zhu, M., Chen, W., Hu, Y., Hou, Y., Liu, L., Zhang, K.: Drgraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. IEEE TVCG **27**(2), 1666–1676 (2020)