Comparing Graph Embedding Methods

Jingyi Xu, Leo Torres, Tina Eliassi-Rad

August 27, 2021

1 Abstract

Graph embedding methods, aka network embedding algorithms, map a given graph structure into a low-dimensional space while preserving properties of the graph. Low-dimensional embeddings can be used efficiently in downstream tasks, such as graph reconstruction, node classification, link prediction, etc. We consider the problem determining how similar two network embedding algorithms are, as there is no clear-cut definition. We provide an empirical study on the relationships in ranking pairwise edge distances by different network embedding algorithms. Further, we show that by adjusting the hyper-parameter values of one network embedding method, we can further get the ranking determined by the embedding method to be closer to that of another network embedding method. Many works in Computer Science have been comparing the performances of downstream tasks of different graph embedding methods, mainly on social networks [12]. In Network Science, we are also concerned with other types of networks. Thus, we aim to fill in the gap between Computer Science and Network Science to study the performances of network embedding methods under a variety of networks.

We compare and contrast six network-embedding methods from three broad categories: matrix-factorization methods — Graph Factorization (GF) [1], High-Order Proximity Preserved Embedding (HOPE) [16], and Gemometric Laplacian Eigenmap Embedding (GLEE) [19]; random walk approaches — Deepwalk [17] and Node2vec [7]; deep architecture approach — Structural Deep Network Embedding (SDNE) [21]. In addition, we select Principal Component Analysis (PCA) [22], a widely-used dimensionality reduction method, for comparison. Our main contributions are as follows: (1) We present an approach measuring similarities between network embedding methods and provide a systematic comparison of network embedding methods based on their primary design choices, such as input graph representation, objective functions, optimization method and so on, please see Table 1. We demonstrate that matrix factorization methods correlate with deep neural network embedding methods. Random walk approaches correlate within themselves. (2) We show that we can manipulate how correlated SDNE to other embedding methods, by adjusting the hyper-parameters of SDNE. (3) We determine which graph embedding methods give best graph reconstruction performance under networks with different network properties. [1]

2 Introduction

In recent years, due to the success of representational learning, many graph embedding methods have been designed, and there are several survey papers trying to classify the popular graph embedding methods [4, 13, 8, 3]. Instead of classifying network embedding methods into different categories, we attempted to compare different graph embedding methods in a systematic way, based on their varying design choices. In Section 3, we present background on seven graph embedding methods we selected for comparison, and list the different design choices.

Meanwhile, in contrast to other comparison studies [15, 6, 5, 9], which focus on comparing graph embedding methods based on downstream tasks' accuracy, we study how close two graph embedding methods are correlated. For example, [15] compare graph embedding methods based on downstream link prediction and community clustering tasks. We use pair-wise Spearman Ranking Correlation to measure similarities.

[1] github repository of this work: `https://github.com/JoyXu123/Graph_Embedding_Research`

In Section 5, we perform the Spearman Ranking Correlation experiments on a series of real-world datasets, followed by a discussion of these results. To further show how to get two unrelated network embedding methods closer by tuning the hyper-parameters of one network embedding methods, we tune the hyper-parameters of Structural Deep Network Embedding (SDNE) in Section 6.

In addition, we perform a comparison of network embedding methods on one down-stream task: graph reconstruction. Many works in Computer Science have been comparing the performances of downstream tasks of different network embedding methods, mainly on social networks [12], which has high clustering coefficient, and the degree sequence follows a heterogeneous degree distribution. In order to complement this area of work, in Section 7, we created a series of synthesized datasets with various graph properties. We then perform graph reconstruction experiments on these datasets.

We study the following research questions: Given a kind of graph, which graph embedding algorithm gives the best accuracy in the downstream task: graph reconstruction? Is there a consensus among different embedding algorithms? How are the different embedding algorithms related to each other? We then study a variety of graph embedding algorithms across different networks, and address the following questions: (1) What are the different design choices employed by different embedding methods? (2) Given graph properties, which graph embedding method gives the best graph reconstruction accuracy? (3) Which set of graph embedding methods are more correlated? (4) How are the graph embedding methods correlated?

The contributions of our paper are as follows:

1. We constructed the first empirical study on comparing the inherent similarities between network embedding algorithms based on Spearman Ranking Correlation. Our empirical study of network embedding algorithms demonstrate the following:
   (a) Across dataset, GLEE, HOPE, SDNE and PCA are positively correlated on ranking edges.
   (b) DeepWalk and Node2vec has a high Spearman Ranking Correlation Score across many datasets.
2. We tuned hyper-parameters of SDNE to show the similarities of SDNE with other graph embedding methods. Our results demonstrated that:
   (a) SDNE and PCA are getting less correlated when hyper-parameters of SDNE set to preserve more of the first-order proximity of the graph.
   (b) The correlation between SDNE and other methods are subject to the changes in SDNE parameters.
   (c) SDNE and PCA are especially highly correlated (above 0.8) with the hyper-parameter α, which governs the importance of 1st-order proximity, sets to 0, and the hyper-parameter β, which governs the importance of 2nd-order proximity, sets to 1.
3. We analyze the graph reconstruction performance of graph embedding methods under different network properties. Our study demonstrates the following:
   (a) PCA, GLEE and SDNE consistently outperform the other graph embedding methods.
   (b) Random walk approaches (i.e Node2vec, DeepWalk) have a better performance on denser graphs than on sparse graphs.
   (c) Overall, SDNE has the best precision@k curve across many graphs.
   (d) GLEE records especially high accuracy when reconstructing the top 10% percent of the edges.

The rest of the paper is organized as follows: In Section 3, we provide information on the selected graph embedding algorithms in its widely-accepted categories. We encapsulate the primary design choices of these graph embedding methods. Section 4 gives a short summary of related work in this field. These are followed by our experiments and discussion in Sections 5, 6, and 7 respectively. We conclude our paper in Section 8.

3 Background

We define a graph as $G = \{V, E\}$. V is the set of nodes. Let $V = \{v_1, ..., v_i, ..., v_n\}$, where N is the number of nodes. E is the edge set, where $e_{ij} = (v_i, v_j) \in E$ denotes an edge from node i to node j. The adjacency matrix is denoted as A. If the input network is undirected, A is a symmetric matrix. If the input network is directed, $A_{ij} = 1$ represents an edge from node i to node j. S denotes a similarity matrix, in which each entry S_{ij} measures the similarity value between node v_i and node v_j, based on the similarity metric being used. The network embedding literature [13] generally divides network embedding algorithms into three categories: matrix factorization, random walk approaches, and deep architecture approaches. Table 1 provides a summary of the embedding methods used in this work.

Algorithm Name	**Input graph representation**	**Similarity radius around a node**	**Objective function**	**Optimization method**	**Similarity measure in embedded space**	**Time Complexity**
PCA[22]	Adjacency matrix	Entire graph	Minimize reconstruction error	Eigen decomposition	Decoder Matrix	O($\|V\|^3 + d^2\|V\|$)
Graph Factorization [1]	Adjacency matrix	1-hop neighborhood	Minimize regularized sum of squared errors	Gradient descent (GD)	Dot product	O($\|E\|d$)
SDNE [21]	Adjacency matrix	1, 2-hop neighborhood	Minimize regularized reconstruction error on local & global structures	Stochastic gradient descent (SGD)	Decoder Matrix	O($\|V\|\|E\|$)
Node2vec [7] and DeepWalk [17]	Random walks	Variable: depends on hyperparams	Maximize conditional log likelihood of a node's neighborhood	SGD with negative sampling	Dot product	O($\|V\|d$)
HOPE [16]	Similarity matrix	Entire graph	Minimize reconstruction error	SVD	Dot Product	O($\|E\|d^2$)
GLEE [19]	Laplacian matrix	Entire graph	Minimize reconstruction error	SVD	Reverse Dot Product	O($\|V\|^2 d$)

Table 1: Comparing various popular graph embedding methods based on their design choices.

3.1 Matrix Factorization Based Embedding

Some of the matrix factorization embedding approaches utilize loss function to update the matrix value, for example the graph factorization method [1].

Graph Factorization (GF) minimizes loss function $f(A, Z, \lambda) = \sum(A_{ij} - Z_i \cdot Z_j)^2 + \lambda \cdot \sum(Z_i)^2$, where A is the adjacency matrix, λ is the regularization parameter, and Z is the embedding matrix. Graph Factorization uses stochastic gradient descent to update each embedding node Z_i, $Z_i = Z_i - (A_{ij} - Z_i \cdot Z_j) \cdot Z_j - \lambda \cdot Z_i$.

High-Order Proximity Preserved Embedding (HOPE) [16], on the other hand, embeds an underlying network based on the transitivity of the network. Transitivity is a characteristic of undirected and directed graphs [18]. "Asymmetric Transitivity depicts the correlation among directed edges, that is, if there is a directed path from u to v, there is likely a directed edge from u to v." [16]. HOPE utilizes matrix factorization on the similarity matrix S. Throughout this paper, the similarity matrix employed by HOPE is Katz matrix. Katz index is a weighted summation over the path st between two vertexes. The weight of a path is a exponential function of its length.

$$S^{Katz} = \sum_{l=1}^{\infty} \beta \cdot A^l = \beta \cdot A \cdot S^{Katz} + \beta \cdot A$$

, where l represents the length of path. β here is the decay parameter of Katz index, and it determines how fast the weight of path decays when the length of path grows. After some transformation,

$$S^{Katz} = (I - \beta \cdot A)^{-1} \cdot \beta \cdot A$$

Then HOPE performs SVD on S. $U, \Sigma^2, V^T = SVD(S)$, in which U is a matrix of size $n \times (d/2)$, V^T is a size of $(d/2) \times n$. In the last step Z is formed by concatenating $U \times \Sigma$ with $V \times \Sigma$. Then Z is of size $n \times d$.

Geometric Laplacian Eigenmap Embedding (GLEE) [19] minimizes the loss function $||L - Z \times Z^T||^2$, where L is the Laplacian matrix and Z is the embedding.

3.2 Random Walk Approaches

Due to the popularity of Skip-gram model in natural language processing (NLP)–word2vec[14], a string of approaches is inspired by sampling paths of nodes from a random source node. Two nodes that frequently exist in surroundings of each other should be closer in embedding space as well. **DeepWalk** [17] uses the uniformly truncated random walk from the source node, whereas **Node2vec** [7] incorporates two more parameters p and q to induce biased random walks. Let α be the search bias. Consider a random walk that just traversed edge (t, v) and now resides at node v. The new transition probability of exploring the next node x from v is shown below:

$$\alpha_{pq}(t,x) \begin{cases} 1/p & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ 1/q & \text{if } d_{tx} = 2 \end{cases}$$

, where d_{tx} represents the length of path from t to x. DeepWalk can be thought of as one special case of Node2vec, where p=1 and q=1.

3.3 Deep Architecture Approaches

Structural Deep Network Embedding (SDNE) [21] utilizes encoder-decoder structure. Let X be the original input matrix, and Y be the embedded matrix. A is the adjacency matrix, $\hat{X}$ is the output matrix. The goal of SDNE is to minimize the loss function below.

$$L_{mix} = L_{2nd} + \alpha \times L_{1st} + v \times L_{reg}$$

where α is a hyper-parameter governs how much to penalize embeddings that fail to ensure first-order proximity. L_{1st} is the loss function due to first-order proximity. L_{2nd} is the loss function due to second-order proximity. L_{reg} is an L2-norm regularization term to prevent overfitting.

$$L_{1st} = \sum_{i,j=1}^{n} A_{ij} ||y_i - y_j||_2^2$$

where $A_{ij} = 1$ if there exists an edge between node i and node j; $A_{ij} = 0$ otherwise. Hence in essence, the hyper-parameter α determines how much to penalize if the embeddings fail to embed two nodes sharing an edge closer.

$$L_{2nd} = ||(\hat{X} - X) \odot B||_F^2 = \sum_{i}^{n} ||(\hat{x_i} - x_i) \odot b_i||_2^2$$

where $\odot$ means Hadamard product, $b_i = \{b_{i,j}\}_{j=1}^{n}$. If $A_{i,j} = 0, b_{i,j}$ = 1, else $b_{i,j} = \beta > 0$. This way, SDNE can penalize more if embedding gets reconstruction of edges wrong than if embedding gets reconstruction of non-edges wrong. β is a hyper-parameter governess the penalization if the second-order proximity is not preserved. The larger the β value, the more penalization on wrong reconstruction of edges.

3.4 Dimensionality Reduction Methods

Principal Component Analysis (PCA) [22], is a common dimensionality reduction method. It computes the principal components, most of the time only keeps a few most important principal components and uses them to reconstruct the data later on. We find PCA directly fitting for a graph embedding method, when we input the adjacency matrix into the PCA.

4 Related Work

Graph embedding methods transform graphs to a vector or a set of vectors, which preserve the properties of graphs and can be efficiently used for downstream tasks, such as graph reconstruction, link prediction, node classification, etc. Normally, the new vectors/embeddings are of much lower dimensionality than the original representation of the graph, i.e. the adjacency matrix. There are many comprehensive survey papers on network embedding. [3, 8]

However, there are a limited number of works [15, 6, 5, 9] that compare and evaluate various graph embedding methods due to some inherent difficulties in comparing different graph embedding methods. First, these methods feature various design decisions. For example, random walk approaches use dot product in the embedded space, while GLEE uses reverse dot product in the embedded space. Second, different graph embedding methods require different levels of parameter tuning. Deep-Architecture methods, such as SDNE, require extensive parameter tuning. On the other hand, Matrix-Factorization methods require almost zero tuning. Third, different downstream tasks were included in the original paper of the graph embedding methods. Fourth, the performance of the graph embedding methods highly depends on the type of graphs used as input.

Out of the literature comparing different graph embedding methods, [6] did an extensive work on evaluating most popular graph embedding methods based on how sensitive are accuracy of link prediction to domains of the graph (Social, Economic, Technological and Biological), the graph density, and embedding dimensions. [5] evaluates overfit and underfit of community detection algorithms, including network embedding methods, on 577 real networks. The authors show that no algorithm is always worse at the link prediction task than every other algorithm and superiority of algorithm is context specific. [15] evaluates 10 network embedding models solely on two machine learning tasks: community clustering task and link prediction task. [11] introduces unsupervised evaluation of multiple network Node Ranks using Link Prediction. [10] set out to understand what types of equivalences structural embeddings capture. They conduected both extrinsic and intrinsic comparisons of network embedding methods. For intrinsic comparison, they compare the ranking of equivalence score between two nodes by each network embedding method, with the ranking of the ground-truth equivalence score. The equivalence score they used are structural equivalence, automorphic equivalence, and regular equivalence. The ground-truth equivalence score/ranking was calculated by the well-known algorithm in each category. They discovered that none of the embedding methods are optimized to capture these three sociological equivalences. For extrinsic comparison, they ran a node clustering task with equivalence-specific node labels.

What differentiates our work from previous literature is that we are not only interetesed in compairng accuracy of different graph embedding methods, but also interested in measuring the inherent similarities between different graph embedding methods to see what properties of the graph are preserved by these methods. We employ "no ground truth". We input the same undirected unweighted graphs to seven selected methods and rank a set of edges based on the link scores predicted by the graph embedding.

5 Spearman Ranking Correlation Experiments

5.1 Methodology

We are interested in analyzing the relationships between different network-embedding methods. In particular, we (1) determine the correlations among the edge ranking induced by different methods, and (2) provide some insights on how design choices affect the correlations among the edge ranking. (3) locate clusters of methods that behave similarly.

In particular, we approach this problem from the ranking of edges and node-pairs by network embedding methods. Each network embedding method embeds G into a unique vector space respectively. In the first type of experiments, we randomly select a set of 1000 edges. We rank the set of edges based on the distance between the two nodes of the edge in the specific vector space. The smaller the distance between the nodes, the higher the ranking of the edge. By considering the problem from the perspective of ranking rather than considering raw distance between each pair of nodes, we are able to compare network embedding methods across very different ranges. Given a reference network, and m network embedding methods, we produce m rankings of the selected edges. We then compare the m rankings to one another in

order to determine similarities between the various methods. In the second type of experiments, we run the same experiment but this time we randomly select a set of 1000 node-pairs, which do not necessarily form an edge in the original network.

To determine the correlation between two rankings, we compute the Spearman ranking correlation. Spearman rank correlation measures the strength and direction of monotonic association between two rankings. It ranges from -1 to +1, where +1 indicates that the two ranks are exactly the same, and 0 indicates no association between ranks, whereas -1 indicates that two ranks are the same but in reversed order.

5.2 Datasets

We conducted our Spearman Ranking experiments on a variety of real-world undirected unweighted network datasets in different domains. The datasets are as follows. **Hamster** contains friendships and family links between users of the website hamsterster.com. **ego-facebook** is the network data collected from survey participants using Facebook app. **as-733** is an Internet Autonomous Systems Subgraphs. **ca-AstroPh** is the collaboration network on Astro Physics field. **ca-CondMat** is the collaboration network on Condense Matter field.

Network Name	# of Nodes	# of Edges	Directed	Weighted
Hamster	1.8K	12.5K	No	No
ego-Facebook	4K	88K	No	No
AS-733	6K	13K	No	No
ca-AstroPh	18K	198K	No	No
ca-CondMat	23K	93K	No	No

5.3 Experiment Details

We performed the experiments as the way indicated in Methodology Section, one on the sets of 1000 edges and the other on sets of 1000 node-pairs. The average pairwise Spearman Ranking correlation scores are presented in the heatmaps in Figures 1 to 5.

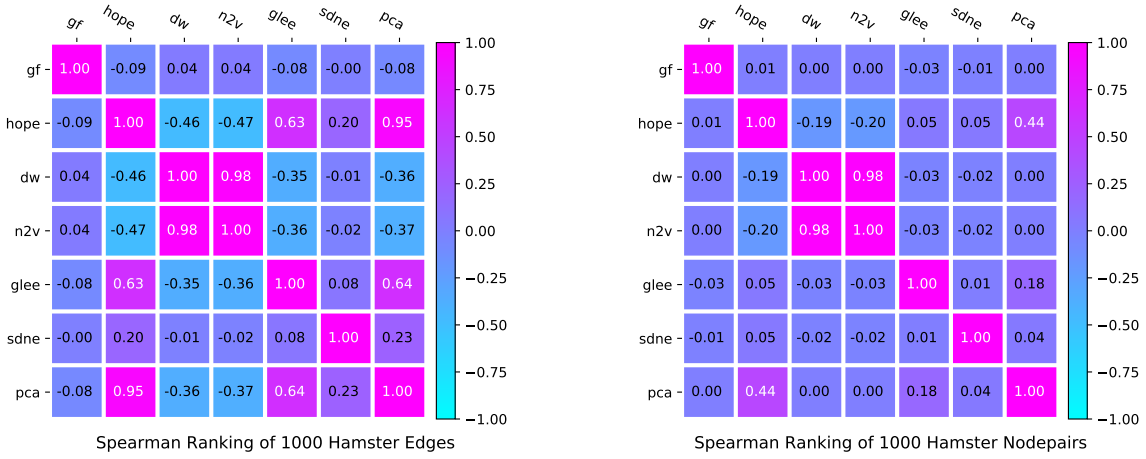

Figure 1: Pair-wise Spearman Ranking Correlation between 7 Graph Embedding methods on Hamster Network HOPE uses Katz matrix, SDNE $\alpha = 0.00001, \beta = 5$, Node2vec $p = 1, q = 3$. All graph embedding methods use 32 dimensions.

Observation & Discussion of the Results

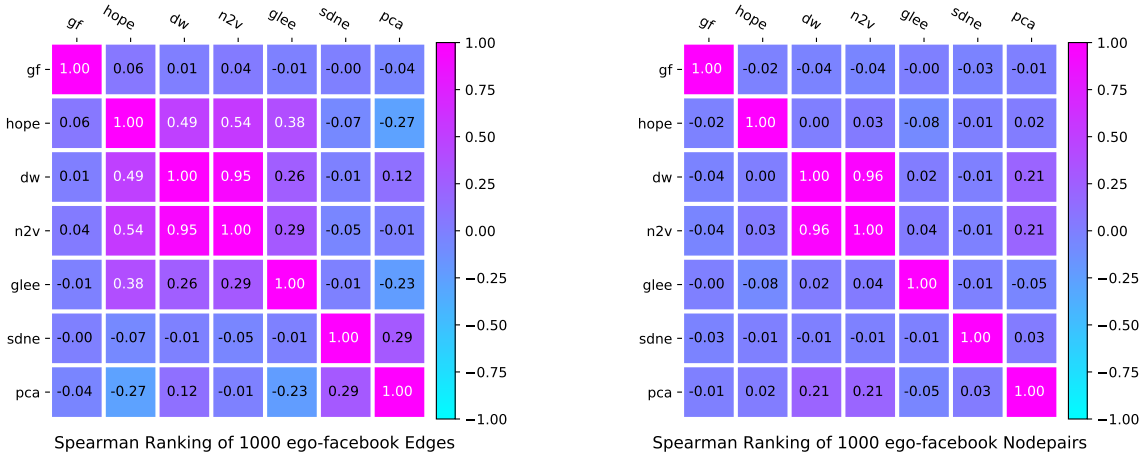

Figure 2: Pair-wise Spearman Ranking Correlation between 7 Graph Embedding methods on ego-facebook Network HOPE uses Katz matrix, SDNE $\alpha = 0.00001, \beta = 5$, Node2vec $p = 1, q = 3$. All graph embedding methods use 32 dimensions.

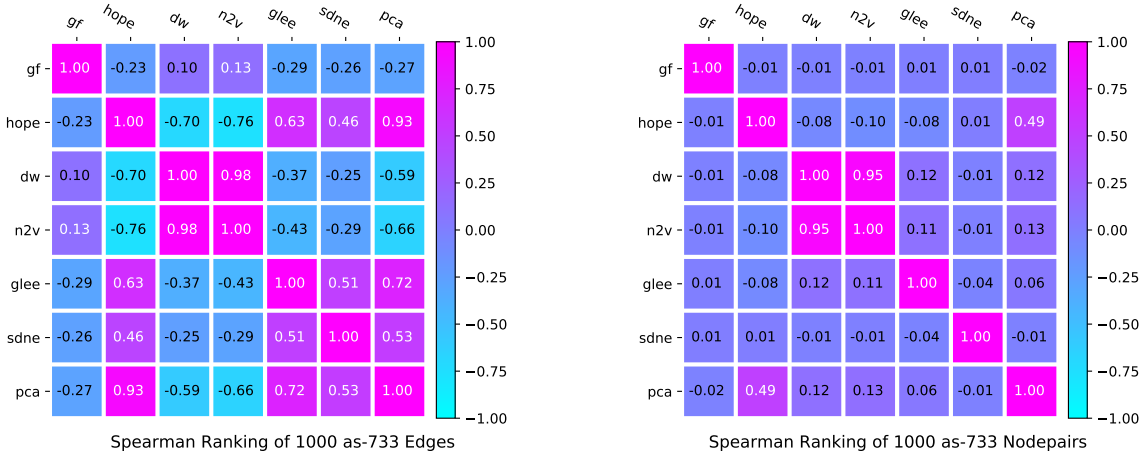

Figure 3: Pair-wise Spearman Ranking Correlation between 7 Graph Embedding methods on as-733 Network HOPE uses Katz matrix, SDNE $\alpha = 0.00001, \beta = 5$, Node2vec $p = 1, q = 3$. All graph embedding methods use 32 dimensions.

1. The correlation on node-pairs are mostly random (aka the Sparman Ranking Correlation Score is around 0), and can be ignored. This is somewhat expected, as majority of the node-pairs are non-edges. This indicates that the seven graph embedding methods do not have a consensus in determining the ranking of non-edges.

2. Random walk approaches Deepwalk and Node2vec are highly correlated. This came in as no surprise, as Node2vec develops based on Deepwalk by introducing two more parameters p and q to induce biased random walk. In comparison, DeepWalk has unbiased random walk. Deepwalk can be thought of as one case of Node2vec where p=1, q=1.

3. Across networks, HOPE, GLEE, PCA and SDNE are positively correlated. This result is somewhat of a surprise as these methods have different input matrices. HOPE uses Katz matrix; Glee uses Laplacian matrix; PCA and SDNE use adjacency matrix. Despite the different input matrix they use for the same network G, these four methods are overall always correlated on edge rankings. PCA, GLEE and HOPE all use matrix factorization. As shown in the next section, when SDNE sets its parameters $\alpha = 0, \beta = 1$, it aims to preserve the same graph properties as PCA.

4. Across networks, random walk approaches are negatively correlated with the sub-group of embed-

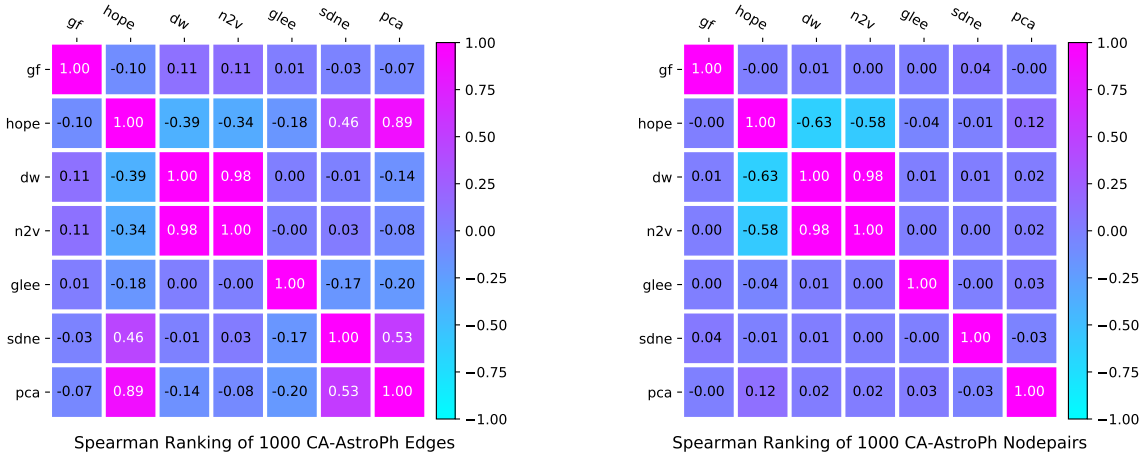

Figure 4: Pair-wise Spearman Ranking Correlation between 7 Graph Embedding methods on ca-AstroPh Network HOPE uses Katz matrix, SDNE $\alpha = 0.00001, \beta = 5$, Node2vec $p = 1, q = 3$. All graph embedding methods use 32 dimensions.

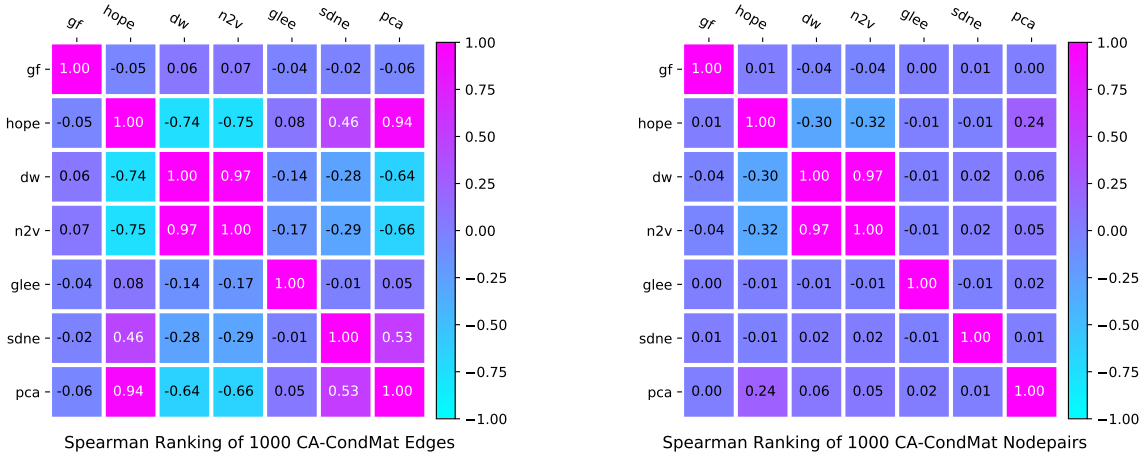

Figure 5: Pair-wise Spearman Ranking Correlation between 7 Graph Embedding methods on ca-CondMat Network HOPE uses Katz matrix, SDNE $\alpha = 0.00001, \beta = 5$, Node2vec $p = 1, q = 3$. All graph embedding methods use 32 dimensions.

ding methods SDNE, PCA, HOPE and GLEE.

6 Hyper-parameter Tuning of SDNE

From the previous set of experiments, we observe that SDNE, PCA, HOPE and GLee are quite correlated, despite that SDNE is a much more complicated deep neural network architecture. We also know that SDNE uses two parameters α and β to control the importance of maintaining 1st-order and 2nd-order proximity respectively. If α is large, the loss function of SDNE penalizes more when the two nodes sharing one edge are embedded far from each other. On the other hand, when β is large, the loss function of SDNE penalizes much more when an edge is not reconstructed than when a non-edge is not reconstructed. Based on this feature of SDNE, we can easily tell that SDNE's performance is most close to PCA, when $\alpha = 0, \beta = 1$, when reconstruction of edges and non-edge have the equal importance. We performed a variation of SDNE parameters and compared the Spearman Ranking correlation scores between SDNE and other graph embedding methods on various datasets.

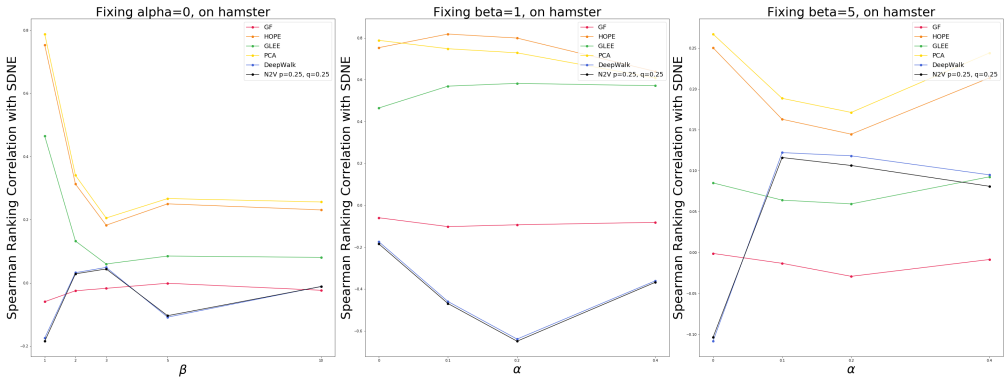

Figure 6: Average Spearman Ranking Correlation of other methods with SDNE over 3 sets of 1000 edges on Hamster Dataset, by varying the α, β parameters of SDNE. α governs the penalization of first-order proximity. The larger the α, the higher the loss function when two nodes sharing an edge are embedded further from each other. β governs the second-order proximity. The higher the β, the larger the penalization if the edges of the adjacency matrix are not reconstructed. All graph embedding methods use 32 dimensions.

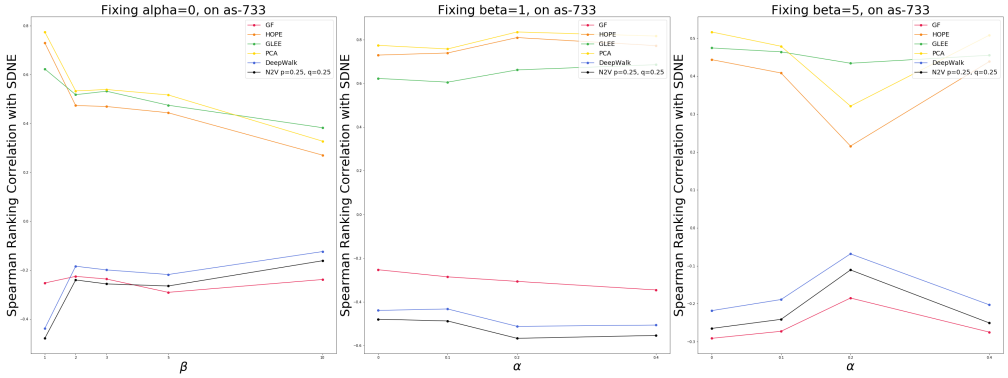

Figure 7: Average Spearman Ranking Correlation of other methods with SDNE over 3 sets of 1000 edges on Ego-Facebook Dataset, by varying the α, β parameters of SDNE. α governs the penalization of first-order proximity. The larger the α, the higher the loss function when two nodes sharing an edge are embedded further from each other. β governs the second-order proximity. The higher the β, the larger the penalization if the edges of the adjacency matrix are not reconstructed. All graph embedding methods use 32 dimensions.

Observation of Experiments From the second graph of Figure 6 to Figure 8, we can see that when we increase alpha value, the correlation between SDNE and PCA indeed decreases. This is expected. However, the correlation between HOPE and SDNE increases when we increase alpha value. One explanation of this is that HOPE embeds the nodes based on the amount of paths between two nodes, not based on the amount of common neighbors between nodes. Increasing the α value of SDNE, SDNE at least guarantees that nodes within paths of length one are embedded closer.

We further confirm our hypothesis: SDNE and PCA are closest when $\alpha = 0, \beta = 1$ as seen across different datasets. As we can see in the first graph of Figures 6 to 8, the Spearman Ranking Correlation between SDNE and PCA is highest when $\alpha = 0, \beta = 1$. As the beta values increase, SDNE penalizes much more if the reconstructed matrix get edges wrong than if the reconstructed matrix gets the non-edges wrong. It further differs from PCA, which treats edges and non-edges equally. Confirmed by the second and third graph of the figures, as α increases, SDNE places higher focus on first-order proximity, and the correlation between SDNE and PCA decreases.

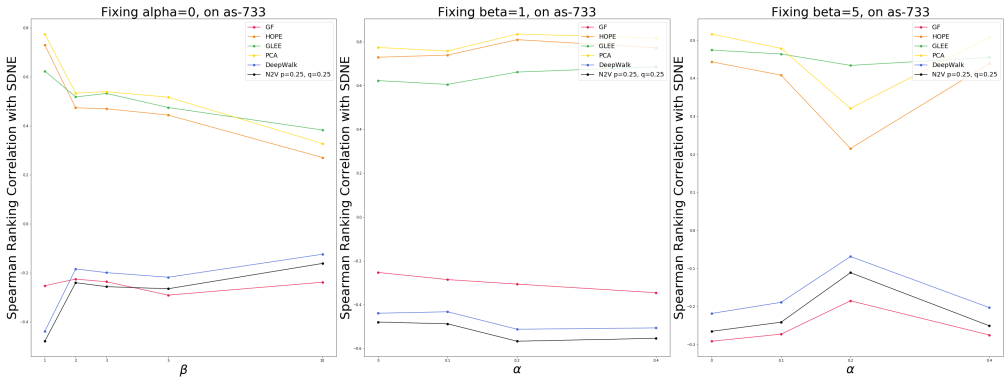

Figure 8: Average Spearman Ranking Correlation of other methods with SDNE over 3 sets of 1000 edges on as-733 Dataset, by varying the α, β parameters of SDNE. α governs the penalization of first-order proximity. The larger the α, the higher the loss function when two nodes sharing an edge are embedded further from each other. β governs the second-order proximity. The higher the β, the larger the penalization if the edges of the adjacency matrix are not reconstructed. All graph embedding methods use 32 dimensions.

7 Graph Reconstruction Task

Graph embedding algorithms preserve graphs in a low dimensional space. Graph reconstruction task, a downstream task, recovers the original graphs from the low dimensional embeddings. The accuracy of a graph reconstruction task can be used to evaluate the performance of graph embedding algorithms. The past work has compared these on real-world networks. [12] shows that majority of real-world benchmark networks used in the Computer Science domain have high clustering coefficients and high node degree heterogeneity. However, there are many other types of networks. We generated synthesized networks from a wide range of network models; ensured these synthesized networks with various range of average clustering coefficient and degree exponent γ when fitting to power-law distribution. We then tested the performance of various graph embedding algorithms on different types of networks. We synthesized 8 major types of networks with different properties. The eight networks are listed below: Erdős-Rényi (ER) network, Barabási-Albert (BA) network, Watts-Strogatz (WS) network, Configuration model with power-law degree sequence (CM), Stochastic Kronecker (KR) graphs, Hyperbolic (HG) graphs [2], Block Two-level Erdős-Rényi (BTER) and Lancichinetti-Fortunato-Radicchi (LFR) graphs. The γ value reported in the table 2 was calculated by fitting the network to a power-law degree distribution using tail-estimation method [20]. γ value falls between 2 and 3 indicates a heterogeneous degree distribution; while the larger the gamma value, the more homogeneous the degree distribution is.

Observation and Explanation of Experiments Figure 9 records graph reconstruction results of different network embedding algorithms. We calculate the pairwise distance between every pair of nodes based on their positions in the embedded space. We sort distances from the smallest to the largest. Precision@k represents the percentage of actual number of edges in the original graph over k top ranked node-pairs reconstructed by an embedding algorithm. The observations are following:

1. PCA, GLEE, and SDNE consistently outperform the other graph embedding methods. The explanation is that PCA, GLEE and SDNE's objective function is directly related to the down-stream task of graph reconstruction. As shown in Table 1, PCA's objective is to reduce reconstruction error of adjacency matrix; GLEE's objective is to reduce reconstruction error of laplacian matrix; SDNE's objective is to reduce reconstruction error on local and global structures. In comparison, random walk approaches maximize conditional log likelihood of a node's neighborhood. This objective is not directly linked to the reconstruction.

2. DeepWalk and Node2vec perform better on dense graphs than on sparse graphs.

   (a) This result can be observed by comparing our results on ER-1 and ER-2, or by comparing BA-1 and BA-2.

Network Name Name	Instances Properties Properties	Average Clustering Coefficient	Gamma γ	No. Edges	Avg Deg
1. ER	1.1 Sparse ER with N=1000, p=0.005	0.004	19.6	2,500	5
	1.2 Dense ER with N=1000, p=0.2	0.200	36.3	100,000	200
2. BA	2.1 N=1000, m=5	0.040	2.85	5,000	10
	2.2 N=1000, m=200	0.419	59.27	200,000	400
3. WS	3.1 N=1000, k=10, p=0.9 (homogeneous degree distribution with low number of triangles)	0.049	26.35	5,000	10
	3.2 N=1000, k=50, p=0.1 (homogeneous degree distribution with high number of triangles)	0.542	89.53	25,000	50
4. CM	4.1 power-law degree sequence with $\gamma = 2.7$	0.007	2.72	1000	2
5. KR	5.1 N=$2^{10} = 1024$, initiator matrix=[0.9,0.1; 0.1, 0.9]	0.003	9.93	1000	2
6. HG	6.1 N=1000, avg_k=10, $\gamma = 2$, T=0	0.743	2.26	5,000	10
	6.2 N=1000, avg_k=10, $\gamma = 11$, T=1	0.011	16.7	5,000	10
	6.3 N=1000, avg_k=50, $\gamma = 2$, T=0	0.844	2.49	25,000	50
	6.4 N=1000, avg_k=50, $\gamma = 11$, T=2	0.056	24.64	25,000	50
7. BTER	7.1 N=1000, avg_k=10	0.755	6.53	5,000	8
	7.2 N=1000, avg_k=50	0.803	8.21	25,000	50
8. LFR	8.1 N=1000, avg_k=10, ($\gamma = 2.25$, mixing parameter $\mu = 0.1$, community size distribution exponent $\beta = 2$)	0.330	2.75	5,000	10
	8.2 N=1000, avg_k=10 ($\gamma = 2.25$, mixing parameter $\mu = 0.5$, community size distribution exponent $\beta = 2$)	0.209	2.68	5,000	10

Table 2: Parameters Values in Synthesized Networks. Data sets used in this work (all undirected, unweighted). γ was calculated by fitting the synthesized network to a power-law degree distribution using tail-estimation method.

   (b) This phenomenon can be explained: with higher density of edges, random walk approaches explore more of the local neighborhood than the greater area, hence greater accuracy in reconstructing edges.

3. Controlling for the edge density on the same type of graph, DeepWalk and Node2vec perform better on graphs with high modularity than graphs with low modularity.

   (a) This result can be observed by comparing HG-2 and HG-3.

4. GLEE performs consistently well across all graph types when reconstructing the top 10% of edges.

   (a) GLEE has almost 100% precision@K at top 10% of edges.

   (b) This can be explained by the fact that the objective function of GLEE is directly linked to matrix reconstruction.

5. HOPE's graph reconstruction performance is all over the place.

   (a) We attribute this randomness in performance to HOPE's reliance on number of discounted paths between node pairs. The number of dicounted paths between node pairs in some graph may not correlate with a higher probability that a node pair is linked. Hence HOPE's embeddings are not well-suited for the task of graph reconstruction.

8 Conclusion

We systematically compared the different types of graph embedding methods based on their design choice. We compared the similarities between graph embedding methods based on Spearman Ranking Correlation in terms of ranking edges and ranking node-pairs. We discovered that SDNE PCA, HOPE and GLEE are positively correlated, despite having a much more complicated neural network architecture. The correlation between SDNE and PCA is high. This is because SDNE and PCA both tried to reconstruct the adjacency matrix as accurately as possible. PCA is simply one set of SDNE parameters, where the second-order proximity is emphasized. We further confirmed our hypothesis by varying the hyper-parameters of SDNE to show the Spearman Ranking Correlation with all other methods. We demonstrated that by keeping the hyper-parameters of SDNE $\alpha = 0, \beta = 1$, SDNE is closest to PCA. When α value increases, while keeping β value the same - in other words, placing more emphasis on first order proximity - the Spearman Ranking

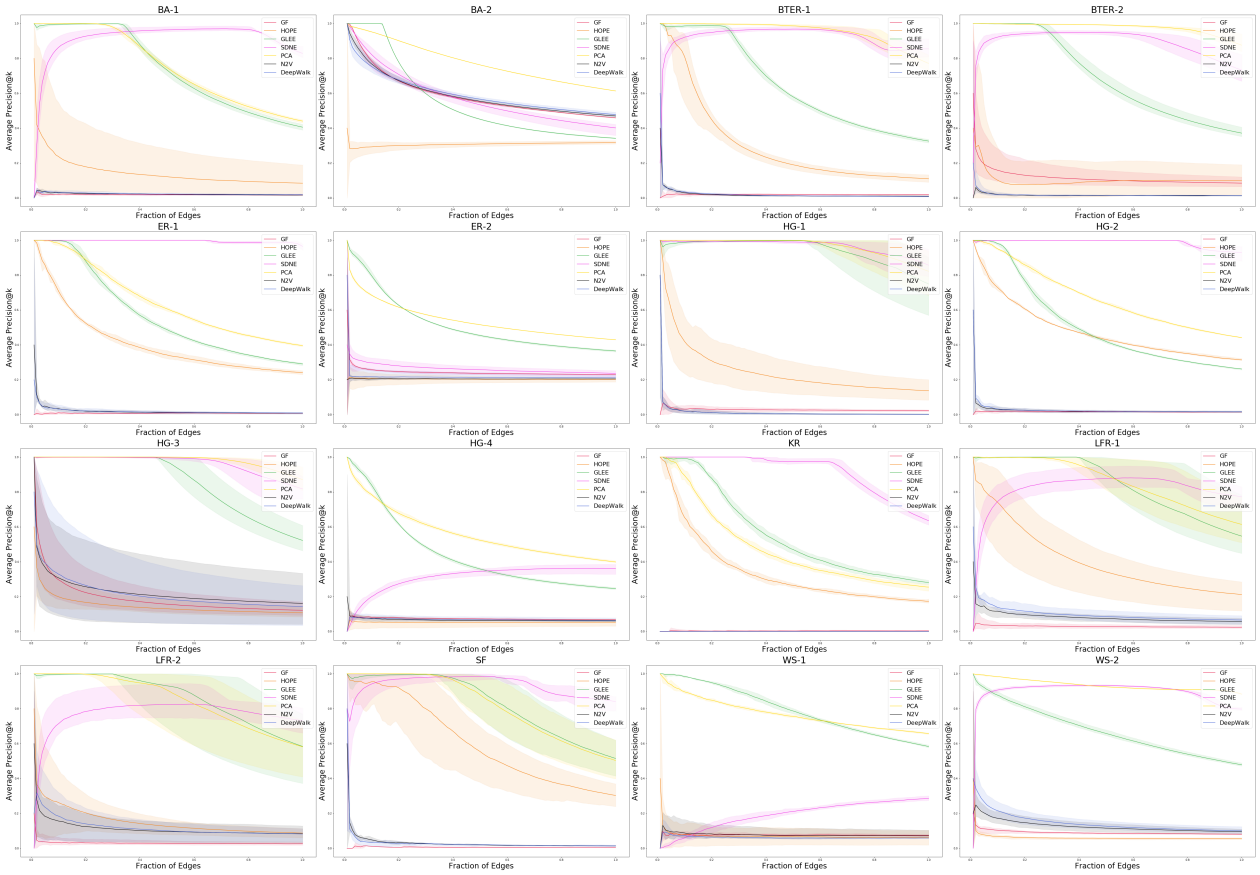

Figure 9: Graph Reconstruction Accuracy on Synthesized Datasets. Each plot shows the average precision@k results over 5 runs for each graph embedding method on a given type of graph. The shaded area represents the quantile from 2.5% to 97.5% for each method. Since the number of edges in each graph is different, we display the fraction of edges on the x-axis (instead of showing k values). All graph embedding methods use 32 dimensions.

Correlation between HOPE and SDNE increases. Increasing β value while keeping $\alpha = 0$, the correlation between SDNE and random walk approaches decreases.

We reported the different graph embedding methods' performance of graph reconstruction task on specially varied synthesized datasets. We showed that SDNE has the best precision@k curve across many datasets. Overall, the top four best performing methods in graph reconstruction tasks are SDNE, PCA, GLEE and HOPE. Random walk approaches perform better in denser graphs than in sparse graphs.

In this paper, we explored the similarities and differences between different network embedding methods and confirmed our hypothesis empirically. We hope this paper can serve as a guide to choose a specific graph embedding method for other researchers.

A Parameters Used in Graph Embedding Methods

References

[1] AHMED, A., SHERVASHIDZE, N., NARAYANAMURTHY, S., JOSIFOVSKI, V., AND SMOLA, A. J. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web* (2013), ACM, pp. 37–48.

[2] ALDECOA, R., ORSINI, C., AND KRIOUKOV, D. Hyperbolic graph generator. *Computer Physics Communications 196* (2015), 492–496.

Graph Embedding Methods	Values of Parameters
GF	
HOPE	uses Katz input matrix
GLEE	
PCA	
SDNE	$\alpha = 0.00001$, $\beta = 5$, regularization_par = 0.00001
Deepwalk	
Node2vec	p=1, q=3

Table 3: Parameters of Graph Embedding Methods in Spearman Ranking Correlation Experiment

Graph Embedding Methods	Values of Parameters
GF	
HOPE	uses Katz input matrix
GLEE	
PCA	
SDNE	β = [1, 2, 3, 5, 10] α = [0, 0.1, 0.2, 0.4], regularization_par = 0.00001
Deepwalk	
Node2vec	p=1, q=3

Table 4: Parameters of Graph Embedding Methods in SDNE Comparison Experiment

[3] CAI, H., ZHENG, V. W., AND CHANG, K. C.-C. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering 30*, 9 (2018), 1616–1637.

[4] CHAMI, I., ABU-EL-HAIJA, S., PEROZZI, B., RÉ, C., AND MURPHY, K. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675* (2020).

[5] GHASEMIAN, A., HOSSEINMARDI, H., AND CLAUSET, A. Evaluating overfit and underfit in models of network community structure. *IEEE Trans. Knowledge and Data Engineering (TKDE)* (2019). In press.

[6] GOYAL, P., HUANG, D., GOSWAMI, A., CHHETRI, S. R., CANEDO, A., AND FERRARA, E. Benchmarks for graph embedding evaluation. *arXiv preprint arXiv:1908.06543* (2019).

[7] GROVER, A., AND LESKOVEC, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (2016), ACM, pp. 855–864.

[8] HAMILTON, W. L., YING, R., AND LESKOVEC, J. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[9] JIN, J., HEIMAN, M., JIN, D., AND KOUTRA, D. Understanding and evaluating structural node embeddings.

[10] JIN, J., HEIMANN, M., JIN, D., AND KOUTRA, D. Towards understanding and evaluating structural node embeddings. *arXiv preprint arXiv:2101.05730* (2021).

[11] KRASANAKIS, E., PAPADOPOULOS, S., AND KOMPATSIARIS, Y. Linkauc: Unsupervised evaluation of multiple network node ranks using link prediction. In *International Conference on Complex Networks and Their Applications* (2019), Springer, pp. 3–14.

[12] LAROCK, T., SAKHAROV, T., BHADRA, S., AND ELIASSI-RAD, T. Understanding the limitations of network online learning. *Applied Network Science 5*, 1 (2020), 1–25.

[13] LI, B., AND PI, D. Network representation learning: a systematic literature review. *Neural Computing and Applications* (2020), 1–33.

[14] MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. S., AND DEAN, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[15] OLUIGBO, I., HADDAD, M., AND SEBA, H. Evaluating network embedding models for machine learning tasks. In *International Conference on Complex Networks and Their Applications* (2019), Springer, pp. 915–927.

[16] OU, M., CUI, P., PEI, J., ZHANG, Z., AND ZHU, W. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (2016), ACM, pp. 1105–1114.

[17] PEROZZI, B., AL-RFOU, R., AND SKIENA, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), ACM, pp. 701–710.

[18] SNIJDERS, T. A., PATTISON, P. E., ROBINS, G. L., AND HANDCOCK, M. S. New specifications for exponential random graph models. *Sociological methodology 36*, 1 (2006), 99–153.

[19] TORRES, L., CHAN, K. S., AND ELIASSI-RAD, T. Geometric laplacian eigenmap embedding. *CoRR abs/1905.09763* (2019).

[20] VOITALOV, I., VAN DER HOORN, P., VAN DER HOFSTAD, R., AND KRIOUKOV, D. Scale-free networks well done. *Physical Review Research 1*, 3 (2019), 033034.

[21] WANG, D., CUI, P., AND ZHU, W. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (2016), ACM, pp. 1225–1234.

[22] WOLD, S., ESBENSEN, K., AND GELADI, P. Principal component analysis. *Chemometrics and intelligent laboratory systems 2*, 1-3 (1987), 37–52.