Canonical Representation of Biological Networks Using Graph Convolution

Mengzhen Li
mxl994@case.edu
Case Western Reserve University
Cleveland, Ohio, USA

Mustafa Coşkun
coskunmustafa@ankara.edu.tr
Ankara University
Turkey

Mehmet Koyutürk
mxk331@case.edu
Case Western Reserve University
Cleveland, Ohio, USA

ABSTRACT

Graph machine learning algorithms are being commonly applied to a broad range of prediction tasks in systems biology. These algorithms present many design choices depending on the specific application and available data, making it difficult to choose from different options. An important design criterion in this regard is the definition of "topological similarity" between two nodes in a network, which is used to design convolution matrices for graph convolution or loss functions to evaluate node embeddings. Many measures of topological similarity exist in network science literature (e.g., random walk based proximity, shared neighborhood) and recent comparative studies show that the choice of topological similarity can have a significant effect on the performance and reliability of graph machine learning models.

We propose GraphCan, a framework for computing *canonical representations* for biological networks using a similarity-based Graph Convolutional Network (GCN). GraphCan integrates multiple node similarity measures to compute canonical node embeddings for a given network. The resulting embeddings can be utilized directly for downstream machine learning tasks. We comprehensively evaluate GraphCan in the context of various link prediction tasks in systems biology. Our results show that GraphCan consistently delivers improved prediction accuracy over algorithms that directly use the adjacency matrix of the input network, and the integration of multiple similarity measurements improves the robustness of the framework. The implementation of GraphCan can be found in https://github.com/Meng-zhen-Li/Similarity-based-GCN.git.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Applied computing** → *Bioinformatics*; • **Theory of computation** → *Graph algorithms analysis.*

KEYWORDS

Similarity Matrices, Graph Convolutional Networks, Network Embedding

BCB '23, September 3–6, 2023, Houston, TX, USA

ACM ISBN 979-8-4007-0126-9/23/09…$15.00
https://doi.org/10.1145/3584371.3612963

ACM Reference Format:
Mengzhen Li, Mustafa Coşkun, and Mehmet Koyutürk. 2023. Canonical Representation of Biological Networks Using Graph Convolution. In *14th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '23), September 3–6, 2023, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3584371.3612963

1 INTRODUCTION

Mining knowledge from large-scale biological networks has become very popular in a broad range of applications in systems biology and molecular medicine. Biological systems are composed of interacting cellular components, e.g. genes, proteins, or metabolites. The associations among the cellular components are modeled as biological networks[12], including protein-protein interaction networks[23, 25], drug-drug interaction networks[27], etc.

Network embedding techniques have been useful in applying sophisticated machine learning techniques to prediction tasks in network biology[21, 24]. Network embeddings provide low-dimensional representations of the nodes in the network to extract features that represent the topological characteristics of the network. The representations in the embedding space reflects the similarities of the nodes in the network topology[6].

Network embedding is an efficient approach for omic data analysis. Multiple tasks can be done by network embedding, e.g. link prediction, node classification, etc. When applied to biological networks, network embedding techniques are shown to be effective in many biomedical prediction tasks. Mapping biological networks into a low-dimensional space enables effective application of machine learning methods in the downstream tasks. Applications of network embedding are becoming ubiquitous in biological data analysis, including identification of cell types[18], prioritization of candidate disease genes[2], and prediction of disease-gene associations [10].

Graph convolutional networks(GCN)[15] have become ubiquitous in many machine learning tasks. It is also shown effective in computing network embeddings. GCN takes a feature matrix and the adjacency matrix of the graph as input, and outputs a low-dimensional representation of each node.

Any network embedding or graph convolution based machine learning technique takes as input the raw input network and performs learning on this raw network. An important drawback of the raw input networks is the skewness of the degree distribution [14], as well as the distribution of network density across different localities of the network [5]. In graph analysis tasks, this causes imbalance in the distribution of flow across the network, creating or enhancing bias in the resulting models [8]. A solution that is commonly applied to address these issues is to use an alternate

graph representations to represent the topological information of input network(s), e.g. similarity graphs for graph convolution [5].

In this article, we propose GraphCan, which learns canonical graph representations for biological networks, in which the topological relationships between the nodes are preserved while edge weights are distributed uniformly across the network. Instead of defining canonical graphs based on a single measure of topological similarity, it can be useful to integrate multiple measures of similarity that capture different aspects of network topology. The proposed framework (i) integrates a multitude of topological similarity measures, (ii) computes low-dimensional embeddings representing these similarity measures using a Graph Convolutional Network (GCN), (iii) computes consensus embeddings from these embeddings using canonical correlation. The computed canonical representation can be utilized directly for downstream machine learning tasks.

We comprehensively evaluate GraphCan in the context of various link prediction tasks in systems biology. Our results show that GraphCan outperforms other network embedding methods that directly use the adjacency matrix of the input network, and is robust against missing edges from the input graph. The integration of multiple similarity measures increases the robustness of the model.

2 BACKGROUND

2.1 Network Embedding

Network embedding aims to learn a low-dimensional representation of nodes in networks[22]. Given a graph $G = (V, E)$, a node embedding is a function $f : V \rightarrow \mathbb{R}^d$ that maps each node $v \in V$ to a vector in $\mathbb{R}^d$ where $d \ll |V|$. The representation is learned such that the proximity in the embedding space reflects the proximity/similarity in the network. Nodes that are similar in the network are close to each other in the embedding space. There are different types of approaches to define the proximity of the nodes, and thus different mapping functions from network topology to embedding space.

2.2 Graph Convolutional Networks and Graph Auto-Encoders

Graph Convolutional Network(GCN) is a scalable approach for semi-supervised learning on graph-structured data that is based on an efficient variant of convolutional neural networks which operate directly on graphs[15]. Graph structures are encoded as a neural network model $f(X, A)$, where $X \in \mathbb{R}^{n \times k}$ is a matrix of node feature vectors(k is the number of features) and $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix of the graph. If no node features are available, X is an identity matrix of the same size as A.

Graph Autoencoder(GAE) is a framework for unsupervised learning on graph-structured data based on the variational auto-encoder (VAE)[16]. Given an unweighted and undirected graph $G = (V, E)$, GAE learns an embedding $Z \in \mathbb{R}^{n \times d}$ using a graph-based neural network with a two-layer GCN encoder and an inner-product decoder. The inner product of the embedding Z reconstructs the adjacency matrix: $\hat{A} = \sigma(ZZ^T)$ where σ is the sigmoid activation function. The learned GAE embedding can be applied to graph learning tasks like link prediction.

3 METHODS

We propose GraphCan, a graph convolutional network (GCN) based framework to compute a canonical representation for an input network, by integrating multiple measures of topological similarity between pairs of nodes.

3.1 Workflow of GraphCan

Figure 1 shows the workflow of the proposed framework, which can be roughly divided into three parts: (i) the GCN-based encoder, (ii) the inner-product decoder, and (iii) computation of consensus embedding. The encoder consists of two GCN layers, and a fully-connected neural network layer. The decoder computes the inner products of the encoded embeddings to reconstruct similarity matrices that capture node similarities based on different measures of topological similarity. Once separate embeddings representing each similarity measure are computed, a consensus embedding (Section 3.5) is computed to integrate the learned embeddings.

The input to GraphCan is the adjacency matrix $A \in \mathbb{R}^{n \times n}$ of the input network $G = (V, E)$, a dimensionality parameter d, and a set of topological similarity measures. An example list of the similarity measures that are used in our experiments are shown in Figure 1. Using the input similarity measures, we compute "true" similarity matrices $S_1, S_2, \ldots, S_k \in \mathbb{R}^{n \times n}$ from A, each representing a different similarity measure. Using these k similarity measures, GraphCan learns d-dimensional embeddings for the nodes in V, such that proximity in the embedding space is associated with similarity according to the multiple similarity measures that are specified in the input.

To compute individual embeddings, GraphCan uses the featureless version of GCNs. The two GCN hidden layers generate an embedding $Z \in \mathbb{R}^{n \times d}$, and the separate embeddings $Z_1, Z_2, \ldots, Z_k \in \mathbb{R}^{n \times d}$ for respective similarity matrices $S_1, S_2, \ldots, S_k$ are computed with a fully-connected hidden layer. The inner product decoder uses the inner product $Z_i Z_i^T$ of as the reconstruction of the similarity matrices. The loss function for the neural network is the mean square error between these inner products and similarity matrices, thus the separate embeddings are optimized to minimize the difference between the "true" and reconstructed similarity matrices. The consensus embedding of the separate embeddings is the learned canonical representation $Z_{GraphCan} \in \mathbb{R}^{n \times d}$.

3.2 Topological Similarity Measures

Node similarity measures aim to quantify the similarity between a pair of nodes in a network in terms of their topological properties and/or location in the network[20]. There is a multitude of topological similarity measures in the literature, used for tasks ranging from link prediction and denoising to community detection and network visualization. Each topological measure captures a different aspect of the topological relationship between a pair of nodes, including the overlap between neighbors, distance in the network, multiplicity of the paths connecting the two nodes, or the association between their proximity profiles [9].

Our objective here is to generate a comprehensive view of the pairwise relationships between the nodes of a network - one that can be reliably used for downstream machine learning applications.

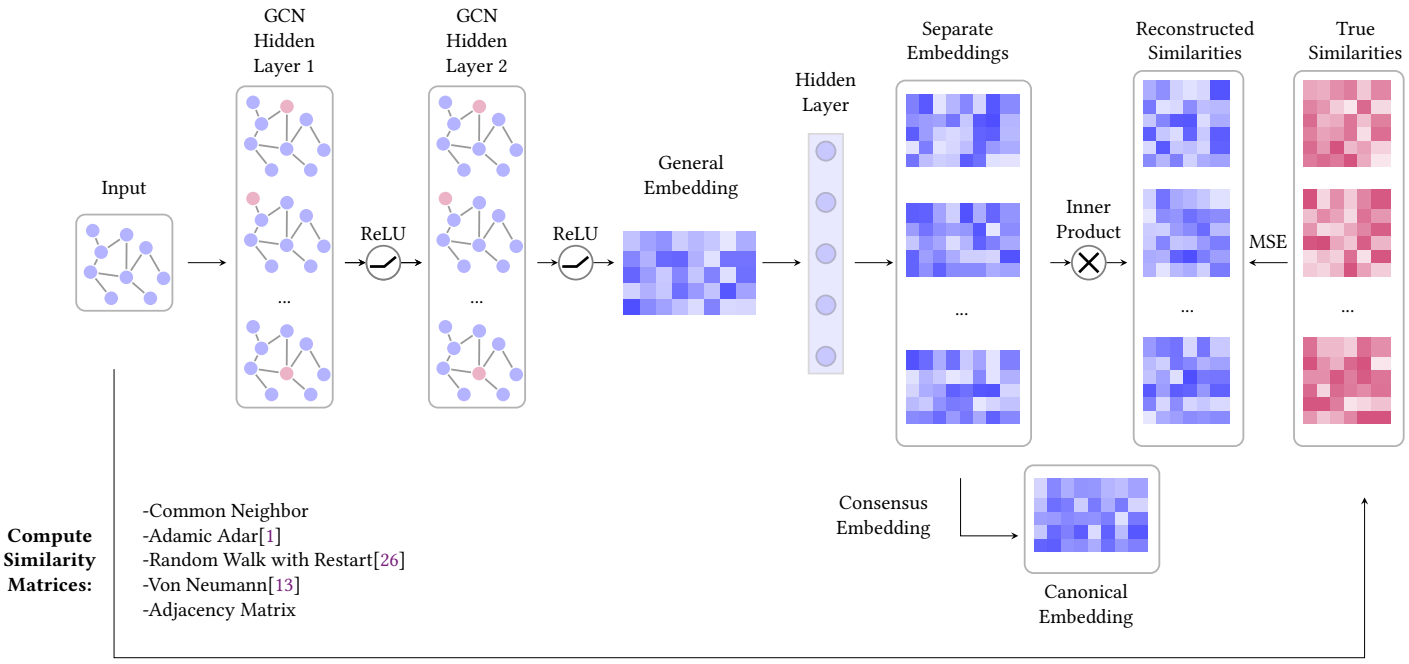

Figure 1: Illustration of GraphCan. The input is the adjacency matrix of the network. The general embedding is generated using a two-layer GCN, and the separate embeddings are generated by a hidden neural network layer. The reconstructed similarity matrices are the inner products of the separate embeddings. The final output embedding is the consensus embedding of the optimized separate embeddings. The loss function is the Mean Square Error(MSE) betweeen true similarities and reconstructed similarities.

For this purpose, we propose to integrate multiple similarity measures, thereby enabling multiple similarity measures to complement each other to extract reliable relationships between pairs of nodes. While GraphCan can be trained with any set of similarity measures, for proof-of-concept we here utilize a small set of measures that span both direct neighbor based (local) and propagation-based (global) approaches:

- **Common Neighbor** measures the overlap between the neighbors of two nodes:

$$CN(x, y) = |N(x) \cup N(y)| \tag{1}$$

  where $N(x)$ and $N(y)$ respectively denote the sets of neighbors of x and y.
- **Adamic Adar**[1] is based on the premise that common neighbors with many neighbors are less informative on the relationship between two nodes as compared to common neighbors with a small number of neighbors. It refines the notion of common neighbor by assigning more weight to less-connected common neighbors:

$$AA(x, y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{log|N(u)|} \tag{2}$$

- **Random Walk Based Proximity** is a network propagation method that considers a random particle that transmits to neighbors with the probability that is proportional to their edge weights[26], and there is a probability for the particle to restart at each stage. Starting from a node x, the probability distribution of the particle being at each node at time t can be computed as:

$$RWR_{t+1}^{(x)} = (\alpha W)RWR_t + (1 - \alpha)RWR_0^{(x)} \tag{3}$$

  where $W \in \mathbb{R}^{|V| \times |V|}$ denotes the matrix of transition probabilities of a random walk between each pair of nodes, obtained by colum-normalizing A. $RWR_0^{(x)}$ is the restart vector with all entries zero except for $RWR_0^{(x)}(x) = 1$ and α denotes the probability of restart. The random-walk based proximity between nodes x and y is computed as $(RWR^{(x)}(y) + RWR^{(y)}(x))/2$.
- **Von Neumann Proximity** is also a network propagation method. For a given maximum path length t, it can be computed by the following expression:

$$A_s = A \oslash \sqrt{D_r \odot D_c} \tag{4}$$

$$VN = \sum_{i=1}^{t} \alpha^i A_s^i \tag{5}$$

  where $D_r \odot D_c$ indicates the element-wise dot product of the row and column degrees in matrix form($D_r(u, v) = |N(u)|$, $D_c(u, v) = |N(v)|$), and $\oslash$ is the element-wise divide operation. A is the adjacency matrix and α is the diffusion

factor[13]. The Von Neumann proximity between nodes x and y is defined as $VN(x, y) = VN(y, x)$.

- The **Adjacency Matrix** can also be considered as a similarity measure by itself (i.e., two nodes are considere similar if they are connected in the network). An identity matrix is added to the adjacency matrix when used as a similarity matrix.

For a given input network, we compute five similarity matrices using these five similarity measures. We then use the similarity matrices as the "labels" for the graph convolutional network, i.e., we train the GCN to learn embeddings that can reconstruct these similarity matrices.

3.3 Similarity-Based Graph Convolutional Network

For generality, let k denote the number of similarity matrices that are utilized by GraphCan. In this paper, we have $k = 5$ (we also use $k = 1$ in Experimental Results for comparison purposes).

The graph convolutional network of GraphCan can be considered as a variational graph autoencoder with one input and multiple outputs. The encoder of GraphCan consists of a two-layer GCN:

$$GCN(X, A) = \text{ReLU}(\tilde{A}\text{ReLU}(\tilde{A}XW_0)W_1) \tag{6}$$

where X denotes feature matrix for the nodes, which is an identity matrix of the same size as A since our networks are featureless. $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix, where D is the degree matrix of A. The output of the encoder is a general embedding $Z \in \mathbb{R}^{n\times d}$ where d is the size of the second GCN layer. W_0 and W_1 are the weights of the two GCN layers.

The decoder consists of two parts, a hidden layer and an inner-product decoder. The hidden layer is a fully-connected neural network layer:

$$Z_s = ZW_2 \tag{7}$$

where $W_2 \in \mathbb{R}^{d\times kd}$ denotes the weight matrix of the hidden layer. Then the output $Z_s \in \mathbb{R}^{n\times kd}$ is splitted to k blocks, $Z_1, Z_2, \ldots, Z_k \in \mathbb{R}^{n\times d}$, each of which denote the separate embeddings to reconstruct the k similarity matrices. The inner product decoder computes the inner products of the separate embeddings, which are also the reconstructed similarities for $1 \le i \le k$:

$$\hat{S}_i = Z_i Z_i^T. \tag{8}$$

3.4 Loss Function and Optimization

To optimize the separate embeddings, the model minimizes the Mean Squared Error (MSE) between each reconstructed similarity and the true similarity:

$$L = \sum_{i=1}^{k} \sum_{u,v\in V} (S_i(u, v) - \hat{S}_i(u, v))^2 \tag{9}$$

Because of the sizes of the weights, optimizing all the weights together is expensive and time-consuming, especially when there are many similarity measures. If we split W_2 into multiple parts and optimize separately, the optimization will be faster and distributed(parallel) learning can be applied(computing the gradients of all blocks at the same time and compute the average of the gradients for W_0 and W_1. Therefore, the loss of each pair of S_i and $\hat{S}_i$ is computed and optimized separately:

$$\min_{W_0,W_1,W_{2i}} \sum_{u,v\in V} (S_i(u, v) - \hat{S}_i(u, v))^2 \tag{10}$$

where $W_{2i} \in \mathbb{R}^{d\times d}$ is computed by dividing W_2 (Equation7) into k blocks. In each iteration, we compute the gradients for W_0, W_1, and W_{2i} for each i so that we get k gradients for W_0 and W_1, and one gradient for each W_{2i}. After we compute all gradients for all i, the gradients of W_{2i} can be concatenated to construct the gradient of W_2, and the gradients of W_0 and W_1 are computed as the average of the k gradients of W_0 and W_1.

3.5 Consensus Embedding

Once a separate embedding for each similarity matrix is computed, we compute a *consensus embedding* from these separate embeddings to obtain a *canonical embedding* for the input network. For this purpose, we use dimensionality reduction on $Z_i \in \mathbb{R}^{n\times d}$ to compute a d-dimensional node embedding Z_c for G. In the context of network integration, Generalized Canonical Correlation Analysis (GCCA) is shown to outperform other dimensionality reduction methods in computing consensus embeddings[17], since it can accurately map shared dimensions in different embedding spaces. The embeddings we aim to integrate here represent similarity measures that are expected to have many common dimensions, thus GraphCan also computes consensus embedding using GCCA:

$$Z_{GraphCan} = GCCA(Z_1, Z_2, \ldots, Z_k) \tag{11}$$

where $GCCA$ represents the generalized canonical correlation analysis, which is shown to be effective in representing the integration of network versions. The consensus embedding $Z_{GraphCan}$ is the canonical representation of graph G and can be used to perform downstream tasks.

4 EXPERIMENTAL RESULTS

4.1 Experimental Data and Benchmark

We apply GraphCan to biological link prediction tasks to assess its ability to obtain meaningful representations of biological network. For this purpose, we use the benchmarking framework provided by BioNEV[28]. BioNEV provides four datasets representing three different biomedical link prediction problems (Table 1). On each of these networks, for a given network embedding, BioNEV uses this network embedding to train a binary classifier (using logistic regression) for each pair of node representations, which in turn is used for link prediction. The statistics of each network is shown on Table 1, including data types and multiple topological features. The average clustering coefficient of bipartite graphs(DDAs) are computed as the average square clustering coefficient[19].

In our experiments, we split the input networks into training and testing sets, where the training set contains 80% of the edges from the original graph, and 20% of the edges are used for testing. The negative train sets and testing sets contain the same numbers of random node pairs without edges as the training sets. We repeat the train and test splits 5 times for each experiment and report the mean and variance of performance figures across these runs.

Dataset	Network Type	#Nodes	#Edges	Density	Average Degree	Average Clustering Coefficient
CTD-DDA [7]	drug-disease association	12,765	92,813	0.11%	7.27	0.0597
DrugBank-DDI [27]	drug-drug interaction	2,191	242,027	10.08%	110.46	0.5453
NDFRT-DDA [3]	drug-disease association	13,545	56,515	0.06%	4.17	0.1951
STRING-PPI [25]	protein-protein interaction	15,131	359,776	0.31%	23.78	0.4279

Table 1: Statics of the biological networks used in the experiments.

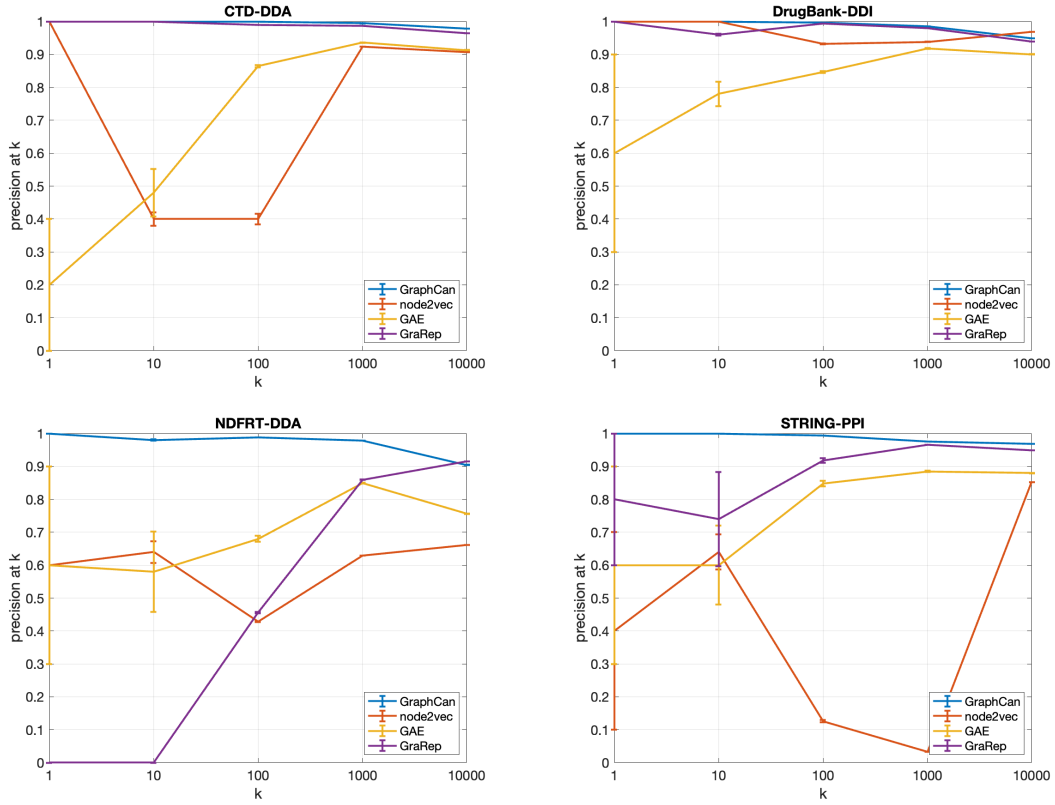

Figure 2: Early precision of GraphCan in link prediction, compared to three state-of-the-art node embedding methods. Each panel shows a different dataset. The x-axis shows the number of predictions (k) from 1 to 10000, and y-axis shows the the precision when the top k predictions are considered as positive predictions. The curves and error show the the mean and variance across 5 different runs. The number of dimensions of embeddings is fixed to 32 for all methods.

4.2 Baseline Node Embedding Methods

We compare the GraphCan model with three existing node embedding methods of different categories:

- **Node2vec** [11] is a random-walk based node embedding method. It runs random walk starting from the nodes in the network, and use the random walk paths to preserve the structural proximity of the network.
- **GAE** [16] is a neural-network based embedding method. It learns node embeddings using a two-layer GCN and a inner-product decoder. Here, we use the featureless version of GAE.
- **GraRep** [4] is a matrix-factorization based embedding method. It constructs the high-order proximity matrix based on transition probability by a random walk with specific length. The model is optimized by matrix factorization techniques(Singular Value Decomposition).

Default parameters of baseline methods are used in all experiments.

4.3 Early Precision Performance

We first compare the performance of algorithms in terms of their early precision in link prediction. In link prediction tasks, the focus is on the ranking of pairs and thus the accuracy higher-ranking

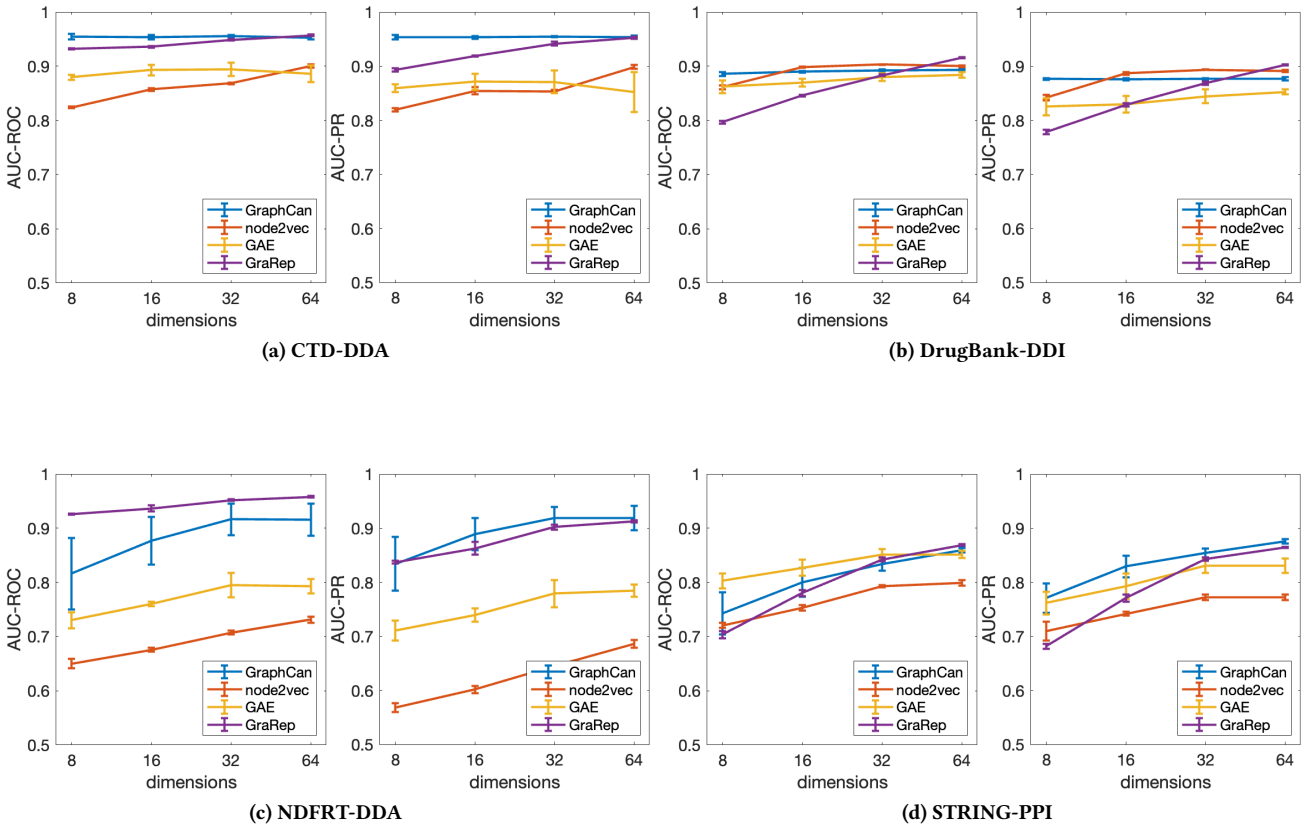

Figure 3: Link prediction performance of GraphCan compared with three state-of-the-art node embedding methods, as a function of number of embedding dmensions. For each dataset, area under ROC curve (AUROC) is shown on the left, area under Precision-Recall curve (AUPR) is shown on the right. The curves and error bars respectively show the average and standard deviation of performance figures across 5 runs.

node pairs is of particular importance. Precision at k (or early precision) is a measurement in recommendation system indicating the percentage of true positives among the top k ranked links:

$$Precision@k = \frac{\#\text{ true positives in top-k predictions}}{k} \tag{12}$$

In link prediction, precision at k shows the proportion of true edges (in the test split) between the top-k node pairs predicted by the model.

Figure 2 shows the precision at k in link prediction of GraphCan and baseline methods. As seen in the figure, GraphCan clearly outperforms all baseline models on all datasets in terms of early precision performance. The baseline methods tend to achieve higher precision as k gets larges, but the highest ranking node pairs, but they cannot reach the precision of GraphCan in most cases. Therefore, the high-ranking links predicted by GraphCan are more likely to be accurate than the baseline methods. Note also that the early precision performance of graph auto-encoder (GAE) and GraRep has high variance for smaller value k, while the variance for GraphCan is lower, indicating that GraphCan provides more robust performance for top predictions.

4.4 Effect of the Number of Dimensions

To provide a comprehensive view on the performance of GraphCan, we also consider area under ROC curve (AUROC) and area under precision-recall curve (AUPR) as performance criteria. Using these performance criteria, we assess the link prediction performance of GraphCan as a function of the number of embedding dimensions and compare against baseline methods. The results of these analyses are shown in Figure 3. As seen in the figure, GraphCan outperforms the baseline methods in most cases.

For CTD-DDA, the overall performance of GraphCan is best among all embedding methods for all embedding dimensions, with more performance difference for lower number of dimensions. For Drugbank-DDI, GraphCan performs best for lower number of dimensions according to both performance criteria, but GraRep and

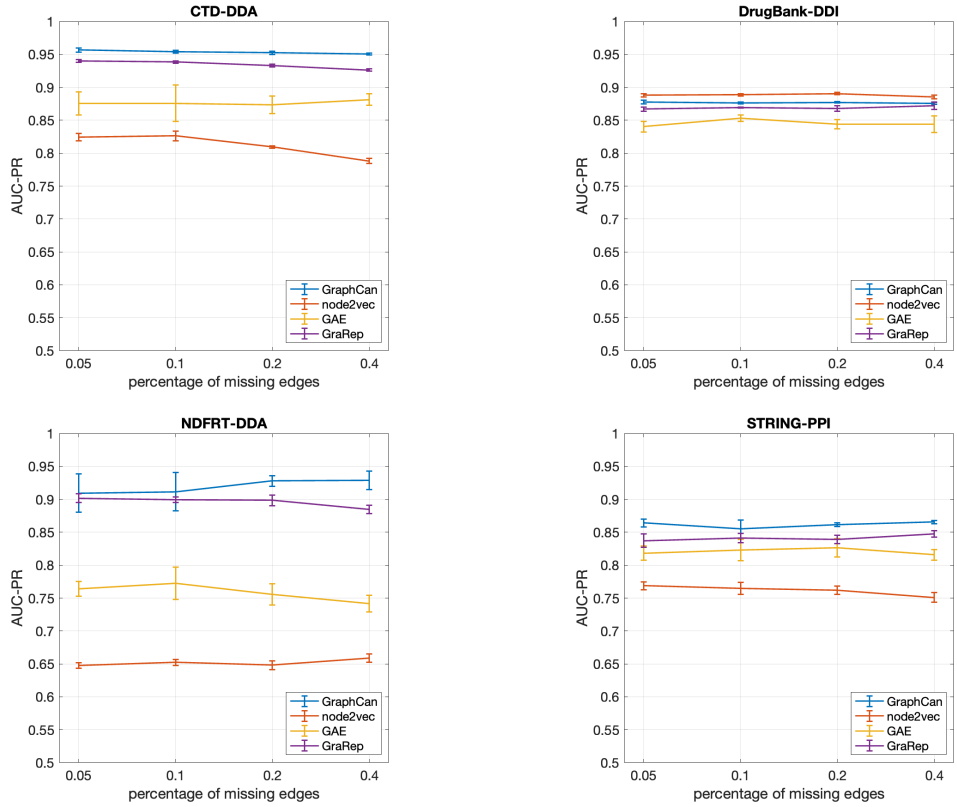

Figure 4: Robustness of GraphCan to data sparsity. x-axes show the fraction of edges that are removed from the network, y-axes show the corresponding link prediction AUPR of each method on the resulting incomplete network. The curves show the average of results among 5 runs, and the error bars show the standard deviation. The number of dimensions is 32 for all experiments shown here.

node2vec catch up with increasing number of dimensions. Unlike GraphCan, the performance of these two methodss is particularly dependent on the number of dimensions. For NDFRT-DDA, GraRep performs best according to AUROC, while GraphCan performs best according to AUPR. The performance of GraphCan is quite variable on this dataset and it is more dependent on the number of embedding dimensions. For STRING-PPI, we also observe that GraphCan delivers the best AUPR while as good as other methods according to AUROC. In general, GraphCan is highly competitive among network embedding methods in link prediction, and its performance is particularly pronounced when AUPR is considered.

4.5 Link Prediction with Incomplete Graphs

Many biological network datasets are largely incomplete and the sparsity of the network can degrade the performance of machine learning algorithms that utilize these networks. One of the motivations of our proposed framework is to alleviate the effect of data sparsity. To assess robustness of GraphCan against missing edges in network datasets, we perform missing data analysis. For this purpose, we remove different fractions of edges from the input networks, and assess the link prediction performance of all methods on the incomplete graphs. Figure 4 shows the results of this analysis. We observe that GraphCan does not have an obvious drop in performance as the percentage of missing edges goes up from 5% to 40%, and has higher AUC-PRs than the baseline methods in most datasets. For the two drug-disease association prediction problems(CTD-DDA and NDFRT-DDA), the performance of baseline methods starts to decrease as the missing percentage increases from 0.1 to 0.4. For DrugBank-DDI dataset, all methods including GraphCan are robust against missing edges, and the reason might be the graph is relatively denser than all other graphs. For STRING-PPI, the performance of node2vec and GAE decreases a little as the missing percentage becomes 0.4. Therefore, GraphCan is robust against missing edges and performs a good prediction even with incomplete graphs.

4.6 Effect of Integrating Multiple Similarity Measures

Without integrating multiple similarity measures, GraphCan can be considered as a variational graph autoencoder that learns an embedding whose inner product is a similarity matrix of the input network(consensus embedding is not needed in this case). To assess the value added by integrating multiple node similarity measures,

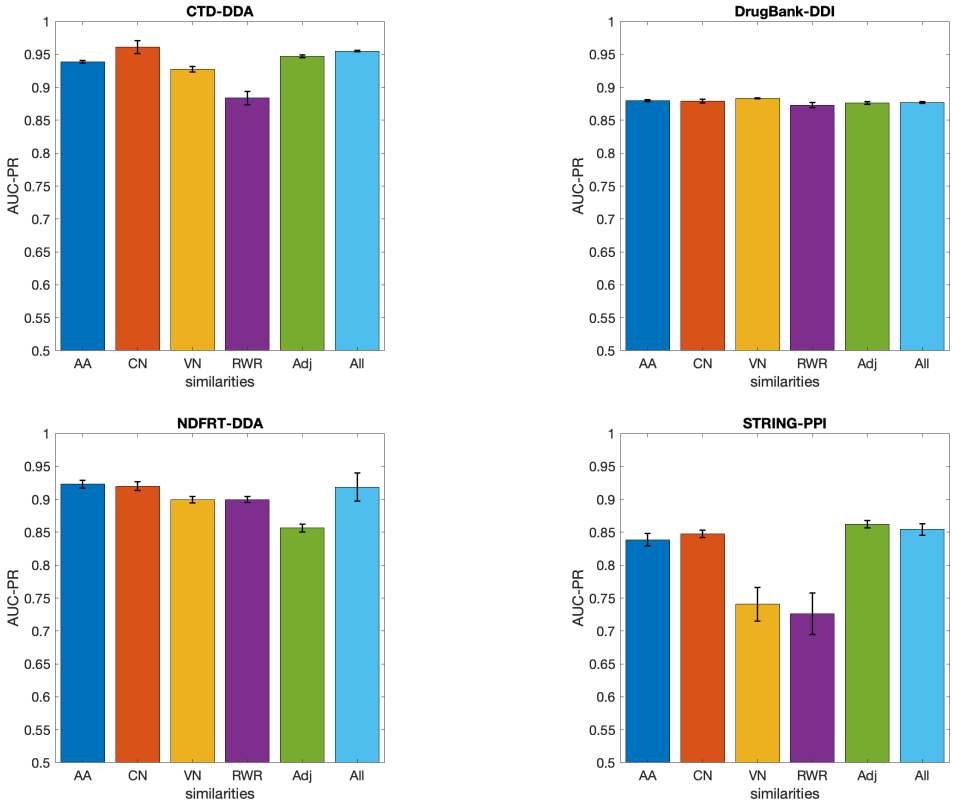

Figure 5: Integrating multiple similarity measures using GraphCan vs. using only one similarity measure. In each subfigure (dataset), the colored bars show the AUPR of link prediction performance of the embeddings computed using a single similarity measure. Different colors of bars represent the average performance of GraphCan using different similarity measures, and the light blue bars are the results using all the similarity measures. The error bars show the standard deviations among 5 runs. The number of dimensions is fixed to 32.

Datasets	Average	Max	Min	Best Similarity	GraphCan
CTD-DDA	0.9315	0.9610	0.8836	CN	0.9550
DrugBank-DDI	0.8783	0.8830	0.8730	VN	0.8768
NDFRT-DDA	0.8996	0.9230	0.8564	AA	0.9186
STRING-PPI	0.8030	0.8622	0.7264	Adj	0.8544

Table 2: The performance of GraphCan using all similarity measures compared to using a single similarity measure. The first three columns are the results when using only one similarity meausure. The average, maximum, and minimum AUC-PRs among experiments with different similarities are shown. The "best similarity" is the similarity that gives the maximum AUPR. The last column is the AUPR of GraphCan when all similarities are used.

we compare the link prediction performance of the canonical embedding provided by GraphCan to the embeddings obtained by using a single topological similarity measure. Figure 5 shows the link prediction results when using only one similarity measure compared with using all of them in GraphCan. As seen in the figure, an individual similarity measure may have quite variable performance depending on the dataset. For example, RWR performs worst on the CTD-DDA network, while the raw adjacency matrix delivers the poorest peformance on the NDFRT-DDA network. The reason for this might be different topological features, e.g. density or neighborhood structures, might affect the learning process. However, we observe in figure 5 that when using all similarity measures, the link prediction performance can be as good as that of the best similarity measure on each dataset. In other words, the framework provided

by GraphCan effectively extracts and utilizes information from different similarity measures that complement each other.

Table 2 compares the AUPR of the model using single similarities with GraphCan. From the table, the link prediction performance of each similarity measure varies a lot, and the best performance can be given by different similarity measures. However, while using all similarity measures, GraphCan performs as good as the best when using one similarity measure, despite the lower performance for some of the similarity measures. Therefore, by utilizing multiple similarity measures in the model, GraphCan extracts complementary information from each similarity measure and thus delivers more robust performance compared to using a single similarity measure.

5 CONCLUSION

In this article, we propose GraphCan, a similarity-based graph convolutional network(GCN) for learning canonical representation of biological networks. The learned representation integrates embeddings representing multiple similarity measures. We systematically test GraphCan on biological link prediction tasks, and the experimental results show that GraphCan outperforms other baseline methods in terms of ranking and predicting new links. Our noise analysis shows that GraphCan is robust against incomplete graphs with missing edges. Furthermore, we show that by integrating multiple similarity measures, GraphCan gains a better view of the network topology, and achieves better outcomes than using single similarity measures. A potential limitation of the paper is that we avoid single nodes when we split training graphs and testing sets to make sure that all nodes exist in embeddings, thus the edge removal process is not uniformly at random. Directions of future research include applying GraphCan to other downstream tasks, e.g. node classification. Another potential application of GraphCan is to take node features into account, so that GraphCan can be used in a broader range of machine learning settings.

REFERENCES

[1] Lada A Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social networks* 25, 3 (2003), 211–230.

[2] Adrián Bazaga, Dan Leggate, and Hendrik Weisser. 2020. Genome-wide investigation of gene-cancer associations for the prediction of novel therapeutic targets in oncology. *Scientific reports* 10, 1 (2020), 1–10.

[3] Olivier Bodenreider. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research* 32, suppl_1 (2004), D267–D270.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 891–900.

[5] Mustafa Coşkun and Mehmet Koyutürk. 2021. Node similarity-based graph convolution for link prediction in biological networks. *Bioinformatics* 37, 23 (2021), 4501–4508.

[6] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering* 31, 5 (2018), 833–852.

[7] Allan Peter Davis, Cynthia J Grondin, Robin J Johnson, Daniela Sciaky, Roy McMorran, Jolene Wiegers, Thomas C Wiegers, and Carolyn J Mattingly. 2019. The comparative toxicogenomics database: update 2019. *Nucleic acids research* 47, D1 (2019), D948–D954.

[8] Sinan Erten, Gurkan Bebek, Rob M Ewing, and Mehmet Koyutürk. 2011. DA DA: degree-aware algorithms for network-based disease gene prioritization. *BioData mining* 4, 1 (2011), 1–20.

[9] Sinan Erten, Gurkan Bebek, and Mehmet Koyutürk. 2011. Vavien: an algorithm for prioritizing candidate disease genes based on topological similarity of proteins in interaction networks. *Journal of computational biology* 18, 11 (2011), 1561–1574.

[10] Yuchong Gong, Yanqing Niu, Wen Zhang, and Xiaohong Li. 2019. A network embedding-based multiple information integration method for the MiRNA-disease association prediction. *BMC bioinformatics* 20, 1 (2019), 1–13.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[12] Pietro Hiram Guzzi and Swarup Roy. 2020. *Biological Network Analysis: Trends, Approaches, Graph Theory, and Algorithms*. Elsevier.

[13] Takahiko Ito, Masashi Shimbo, Taku Kudo, and Yuji Matsumoto. 2005. Application of kernels to link analysis. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 586–592.

[14] Jian Kang, Yan Zhu, Yinglong Xia, Jiebo Luo, and Hanghang Tong. 2022. Rawlsgcn: Towards rawlsian difference principle on graph convolutional network. In *Proceedings of the ACM Web Conference 2022*. 1214–1225.

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).

[17] Mengzhen Li, Mustafa Coşkun, and Mehmet Koyutürk. 2022. Consensus embedding for multiple networks: Computation and applications. *Network Science* 10, 2 (2022), 190–206.

[18] Xiangyu Li, Weizheng Chen, Yang Chen, Xuegong Zhang, Jin Gu, and Michael Q Zhang. 2017. Network embedding-based representation learning for single cell RNA-seq data. *Nucleic acids research* (2017).

[19] Pedro G Lind, Marta C Gonzalez, and Hans J Herrmann. 2005. Cycles and clustering in bipartite networks. *Physical review E* 72, 5 (2005), 056127.

[20] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. 2009. Similarity index based on local paths for link prediction of complex networks. *Physical Review E* 80, 4 (2009), 046122.

[21] Walter Nelson, Marinka Zitnik, Bo Wang, Jure Leskovec, Anna Goldenberg, and Roded Sharan. 2019. To embed or not: network embedding as a paradigm in computational biology. *Frontiers in genetics* 10 (2019), 381.

[22] Ryan A Rossi, Di Jin, Sungchul Kim, Nesreen K Ahmed, Danai Koutra, and John Boaz Lee. 2019. From community to role-based graph embeddings. *arXiv e-prints* (2019), arXiv–1908.

[23] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. 2006. BioGRID: a general repository for interaction datasets. *Nucleic acids research* 34, suppl_1 (2006), D535–D539.

[24] Chang Su, Jie Tong, Yongjun Zhu, Peng Cui, and Fei Wang. 2020. Network embedding in biomedical data science. *Briefings in bioinformatics* 21, 1 (2020), 182–197.

[25] Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. 2019. STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research* 47, D1 (2019), D607–D613.

[26] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*. IEEE, 613–622.

[27] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. 2018. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic acids research* 46, D1 (2018), D1074–D1082.

[28] Xiang Yue, Zhen Wang, Jingong Huang, Srinivasan Parthasarathy, Soheil Moosavinasab, Yungui Huang, Simon M Lin, Wen Zhang, Ping Zhang, and Huan Sun. 2020. Graph embedding on biomedical networks: methods, applications and evaluations. *Bioinformatics* 36, 4 (2020), 1241–1251.