NETWORK SCIENCE

Deep representation learning of protein-protein interaction networks for enhanced pattern discovery

Rui Yan[1], Md Tauhidul Islam[2]*, Lei Xing[1,2,3]*

Protein-protein interaction (PPI) networks, where nodes represent proteins and edges depict myriad interactions among them, are fundamental to understanding the dynamics within biological systems. Despite their pivotal role in modern biology, reliably discerning patterns from these intertwined networks remains a substantial challenge. The essence of the challenge lies in holistically characterizing the relationships of each node with others in the network and effectively using this information for accurate pattern discovery. In this work, we introduce a self-supervised network embedding framework termed discriminative network embedding (DNE). Unlike conventional methods that primarily focus on direct or limited-order node proximity, DNE characterizes a node both locally and globally by harnessing the contrast between representations from neighboring and distant nodes. Our experimental results demonstrate DNE's superior performance over existing techniques across various critical network analyses, including PPI inference and the identification of protein functional modules. DNE emerges as a robust strategy for node representation in PPI networks, offering promising avenues for diverse biomedical applications.

INTRODUCTION

Biological networks are instrumental in modeling complex biological systems by providing a detailed blueprint of the myriad interactions among genes, proteins, and other cellular components (*1*, *2*). These networks delineate entities as nodes and their interactions—spanning from physical connections to functional associations—as edges, laying the groundwork for unraveling the complexities of biological systems and processes (*3*–*5*). For instance, in protein-protein interaction (PPI) networks, the intricate web of connections contains crucial information for understanding cellular processes and disease mechanisms (*6*–*8*). However, deciphering these complex networks to gain biological insights poses a substantial challenge. Network embedding, a process where interconnected nodes within a network are mapped into vectors in a lower dimension while preserving certain network structure properties and node relationships, is a commonly used approach to discern patterns within biological networks (*9*–*11*). The accuracy of network embedding critically determines the success of downstream data analysis and applications.

The underlying structure of a biological network is widely recognized to be highly nonlinear because of complex and nonadditive interactions (*12*–*14*), and encompasses both local (i.e., immediate connections) and high-order (i.e., clustering) structures (*15*). Despite extensive efforts to develop network embedding methods capable of coping with such complexity, a practical solution remains elusive. Traditional network embeddings primarily aim to capture node proximity through methods such as matrix factorization (*12*, *16*–*18*) or shallow models (*12*, *19*, *20*). However, these methods often encounter limitations because of their reliance on low-rank approximations or oversimplified network structures, hindering their ability to fully capture the highly nonlinear patterns and resulting in suboptimal embeddings (*21*). Recently, deep learning–based techniques (*22*–*25*) have emerged to tackle the problem by leveraging multiple layers of nonlinear transformations to capture the complex network structure.

For instance, variational graph autoencoder (VGAE) (*22*) uses graph neural networks to enhance the expressiveness of node embeddings by aggregating information from their neighborhoods. While it preserves certain nonlinear structural aspects within node embeddings, the algorithm solely focuses on patterns within the local neighborhood of each node, thus limiting its capacity to understand the node interrelationships across the wider network. Efforts like Deep Graph Infomax (DGI) aim to mitigate these limitations by preserving global structural information via aligning node embeddings with a global graph summary (*23*). However, because the focus is on global structure, this approach may overlook fine-grained local details. Deep Graph Contrastive Representation Learning (GRACE) captures global information indirectly by learning embeddings that are invariant to graph corruptions introduced by data augmentation (*24*). The effectiveness of GRACE may depend on the quality of these data augmentations.

Here, we introduce a general graph representation learning framework that uses deep learning to preserve the nonlinear and multifaceted structure of networks in a lower-dimensional space for high-performance analyses of biological networks. Our proposed method, referred to as discriminative network embedding (DNE), characterizes each node through a nonlinear contrast between the representations of its direct neighbors and nodes that are farther away in the network. Figure 1 illustrates the framework of our proposed DNE method. The proposed approach allows a holistic perspective on the role of each node in the network: It highlights the immediate connections of a node, such as interactions between proteins in PPI networks, and also its community affiliations within the network, such as protein functional modules. We demonstrate that DNE substantially outperforms existing network embedding methods for various networks and multiple downstream tasks, including link prediction (i.e., prediction of PPIs) and node clustering (i.e., identification of functional modules). DNE also offers the flexibility to combine node features with network structures for improved performance. By uniquely incorporating protein sequence features from protein

[1]Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305, USA. [2]Department of Radiation Oncology, Stanford University, Stanford, CA 94305, USA. [3]Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA.
*Corresponding author. Email: tauhid@stanford.edu (M.T.I.); lei@stanford.edu (L.X.)

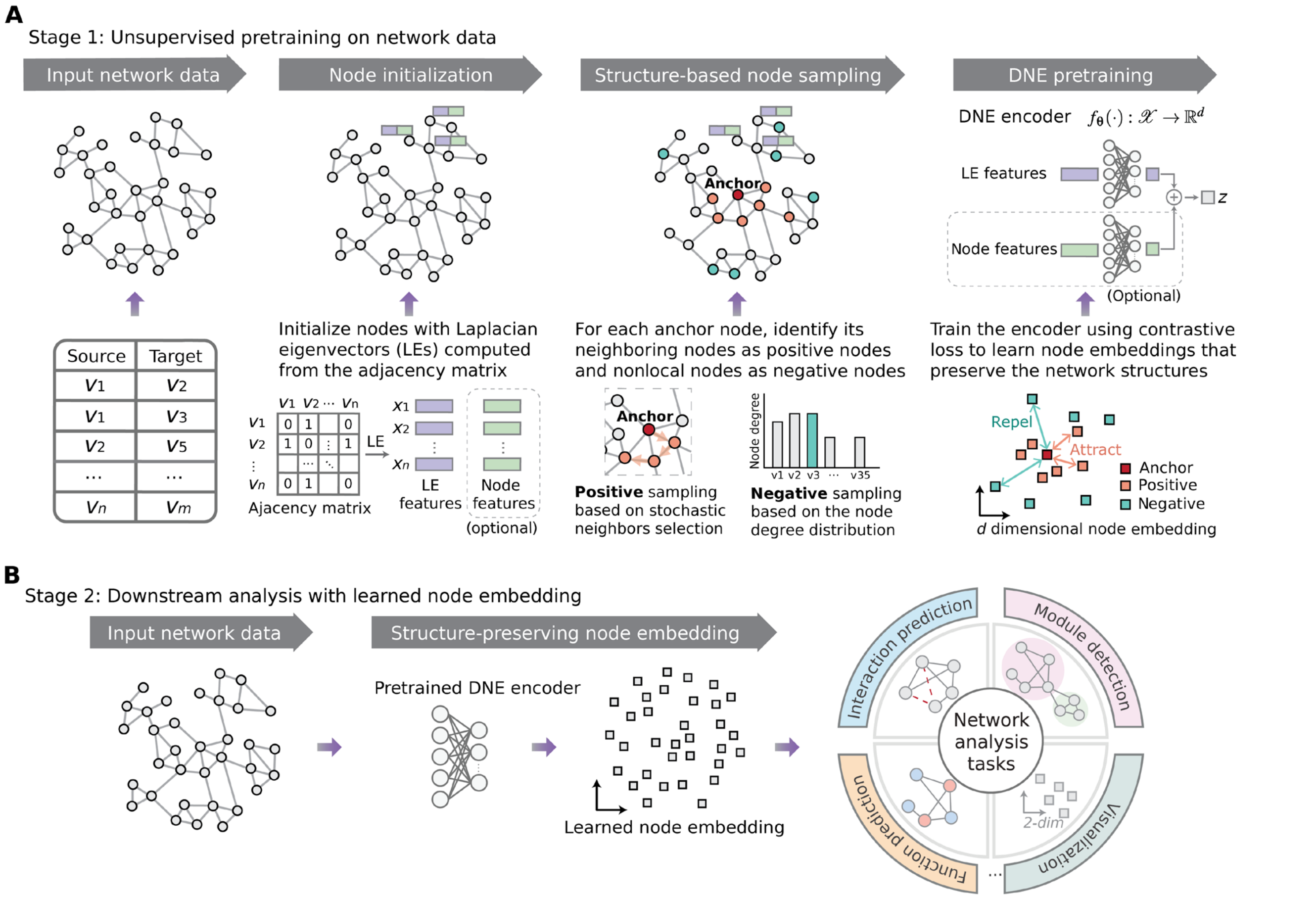
A
Stage 1: Unsupervised pretraining on network data
Input network data
Node initialization
Structure-based node sampling
DNE pretraining
Source
Target
V1 V2
V1 V3
V2 V5
… …
Vn Vm
Initialize nodes with Laplacian eigenvectors (LEs) computed from the adjacency matrix
V1 V2 ··· Vn
Ajacency matrix
LE
x1
x2
xn
LE features
Node features (optional)
For each anchor node, identify its neighboring nodes as positive nodes and nonlocal nodes as negative nodes
Anchor
Positive sampling based on stochastic neighbors selection
Node degree
v1 v2 v3 ··· v35
Negative sampling based on the node degree distribution
DNE encoder $f_\theta(\cdot) : \mathcal{X} \to \mathbb{R}^d$
LE features
Node features
z
(Optional)
Train the encoder using contrastive loss to learn node embeddings that preserve the network structures
Repel
Attract
Anchor
Positive
Negative
d dimensional node embedding
B
Stage 2: Downstream analysis with learned node embedding
Input network data
Structure-preserving node embedding
Pretrained DNE encoder
Learned node embedding
Interaction prediction
Module detection
Network analysis tasks
Function prediction
Visualization
2-dim
…

Fig. 1. Overview of DNE. (**A**) DNE comprises three main steps: (i) initializing nodes using Laplacian eigenvectors (LEs) of the network's adjacency matrix, optionally concatenated with node features when available; (ii) identifying node neighbors as positive nodes via stochastic neighbors selection and selecting nodes from other network regions as negative nodes, based on the distribution of node degrees; and (iii) embedding each node through a deep learning encoder, optimizing the encoder's parameters to ensure the node embeddings preserve discrimination between neighboring and nonlocal nodes. (**B**) Utilization of the pretrained encoder to generate node representations for versatile downstream analysis tasks.

language models, DNE substantially boosts downstream performance compared to traditional methods. Moreover, we demonstrate that DNE can be applied to various network types beyond PPI networks. The proposed method introduces a fresh paradigm for network analysis and promises to broadly advance biomedical data science.

RESULTS

DNE consistently outperforms existing network embedding methods in link prediction across PPIs

We first demonstrate the efficacy of DNE in link prediction, which predicts the likelihood of edge existence based on the known network structural information. For this purpose, we benchmark our method and conduct a comparative analysis with DNE and other leading algorithms for predicting protein interactions in PPI networks, using the following PPIs: (i) a plant interactome comprising 2774 proteins and 6205 PPIs, from the *Arabidopsis thaliana* interactome (*26*); (ii) a worm interactome with 2528 proteins and 3864 PPIs, based on *Caenorhabditis elegans* (*27*); (iii) a yeast interactome with 2674 proteins and 7075 PPIs from *Saccharomyces cerevisiae* (*28*); (iv) a human interactome consisting of 8272 proteins and 52,548 PPIs, derived from *HuRI* (*29*). For each of the four interactomes, a 20% subset of the edges is randomly selected for testing and subsequently removed to form a training network. We then conduct a fivefold cross-validation on the remaining data to obtain optimal performance. This process is repeated for 10 independent runs.

The performance comparison of DNE with 11 other network embedding methods [i.e., DGI (*23*), GRACE (*24*), Variational Graph Normalized Autoencoder (VGNAE) (*25*), VGAE (*22*), Node2Vec (*20*), GraRep (*16*), High-Order Proximity preserved Embedding (HOPE) (*17*), Large-scale Information Network Embedding (LINE) (*19*), Network Embedding as Matrix Factorization (NetMF) (*18*), Locally Linear Embedding (LLE) (*12*), and Singular Value Decomposition (SVD)] is presented in Fig. 2. In the task of predicting links on the *A. thaliana* dataset, DNE scores highest in both the area under the precision-recall curve (PR-AUC) and the area under the receiver operating characteristic curve (ROC-AUC) (Fig. 2, A and B). Notably, DNE achieves a mean ROC-AUC of 88.05% across 10 runs, representing approximately a 4% improvement over the results of the next best methods. Moreover, DNE consistently excels across all PPI networks studied (Fig. 2C and figs. S1 to S6), demonstrating its robustness and adaptability to various network characteristics, such

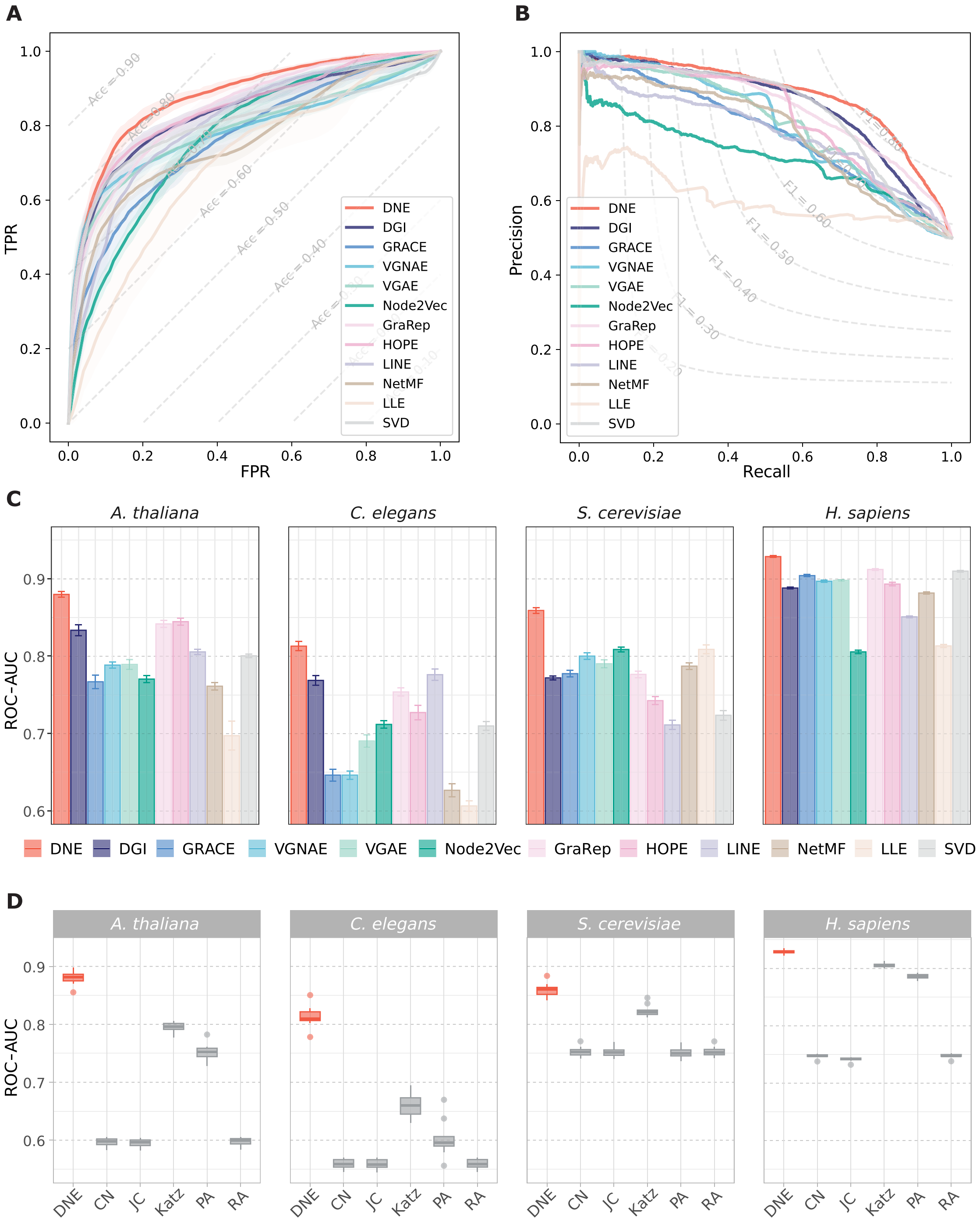

Fig. 2. Performance of different methods for link prediction across four PPI benchmarks. (**A**) ROC and (**B**) PR curves of DNE compared with 11 other network embedding methods for PPI prediction on the *A. thaliana* dataset. Dashed lines represent level curves for accuracy and F1 score in (A) and (B), respectively. (**C**) Comparison of DNE with network embedding methods in four PPI benchmarks, presenting mean and SDs of ROC-AUC scores from 10 independent runs. (**D**) Comparison of DNE with similarity-based link prediction methods in four PPI benchmarks, presenting ROC-AUC scores from 10 runs. The central line within the box denotes the mean, the box edges represent the first and third quartiles, and the whiskers extend to ±1.5 times the interquartile range.

as the degree and density in existing PPI networks (see table S1 for detailed network statistics). In contrast, the performance rankings of other methods fluctuate. The observed enhancement in performance and the consistency of its outcomes across various runs underscore DNE's efficacy in capturing the structural information of the provided PPI networks.

We further compare DNE with five heuristic-based link prediction methods (*30*), which use heuristic node similarity scores including Common Neighbors (CN), Jaccard Index (JC), Katz Index (Katz), Preferential Attachment (PA), and Resource Allocation Index (RA), for link prediction (Fig. 2D). It is observed that the performance of methods such as Common Neighbors, Jaccard Index, and Resource Allocation Index falls short of expectations. DNE, on the other hand, demonstrates a notable improvement, exceeding over 8% in ROC-AUC scores for the *A. thaliana* and *C. elegans* networks over these heuristic approaches. This performance gap highlights the limitations of solely depending on preexisting similarity metrics for predicting new interactions. For example, because of the complex behavior of biological networks, the existence of common neighbors does not necessarily indicate a linkage.

DNE effectively identifies functional modules in PPIs

Module detection in PPI and other biological networks is a crucial task aimed at identifying clusters of closely interconnected nodes, where each cluster signifies a group of proteins that share similar functions (*31*). We evaluate the performance of DNE in module identification using PPI data from *S. cerevisiae*. The *S. cerevisiae* offers an excellent testing ground for network clustering due to the extensive knowledge available about its protein complexes. For reference standards, we use IntAct protein complexes (*32*), Kyoto Encyclopedia of Genes and Genomes (KEGG) pathways (*33*), and GO Biological Processes (GOBP) (*34*). Figure 3 compares multiple embedding techniques on these three module detection benchmarks. Different network embedding techniques are used to represent proteins in a continuous vector space. Subsequently, hierarchical agglomerative clustering (*35*) is applied to these learned embeddings to identify functional modules in PPIs. For evaluation, we compute adjusted mutual information (AMI) score to assess the correspondence between the clusters identified by these methods and the annotated complexes contained in *S. cerevisiae*.

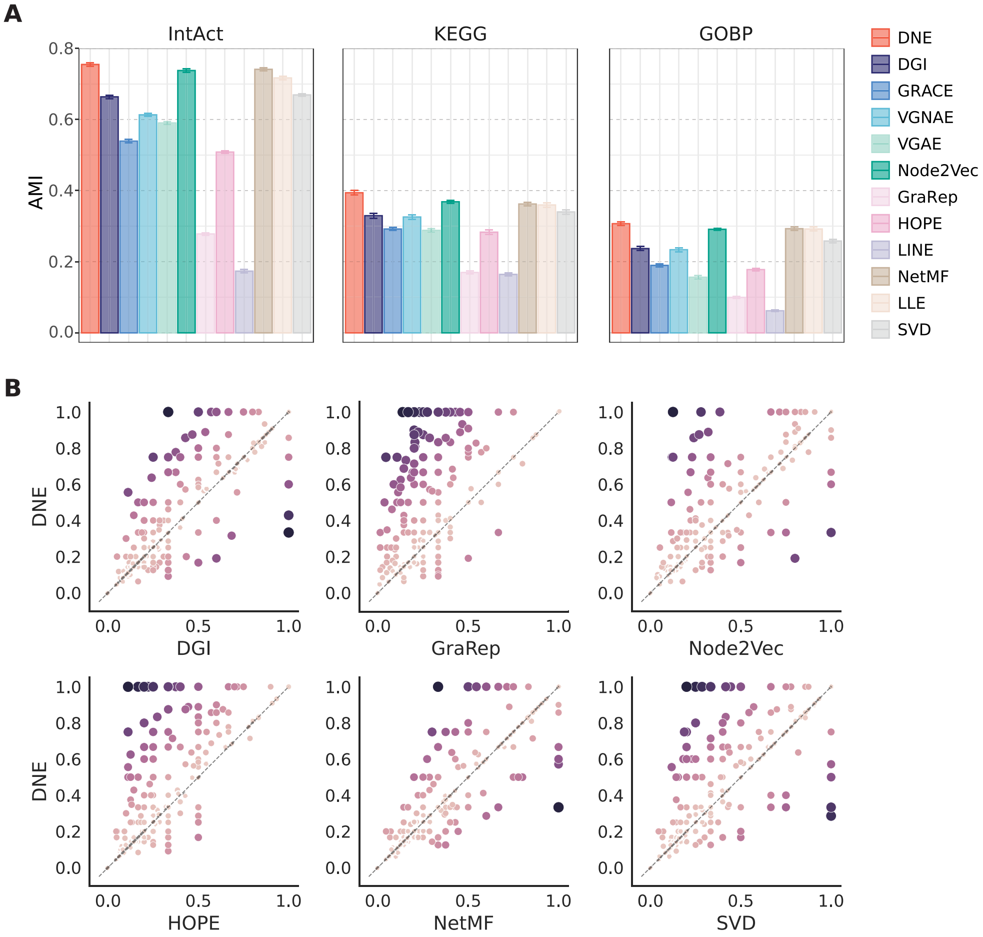

Fig. 3. Performance of different network embedding methods for module identification. (**A**) AMI scores computed from 10 independent runs by using annotated complexes from IntAct, KEGG, and GOBP as reference standards. Mean values are reported, and error bars represent the SDs of the scores. (**B**) Comparison of per-module Jaccard scores between DNE and six representative baselines. Each point represents a protein complex. The *x* axis and *y* axis represent the per-module overlap (Jaccard) scores obtained by the specified baseline method and DNE, respectively. A score of 0 indicates that no members in the complex were captured, and 1 indicates that all members in the complex were captured. The color and size of each point indicate the difference in Jaccard scores between DNE and other baseline methods for the corresponding complex.

DNE excels in predicting protein complexes, exhibiting a substantially higher mean AMI score than other methods in the benchmarks. Specifically, DNE achieves a 2% improvement in AMI compared to Node2Vec and NetMF and surpasses other baseline methods by a considerable margin, ranging from 10 to 50% (Fig. 3A). Moreover, we evaluate the degree of overlap between known protein complexes and their predicted modules using the Jaccard index. In addition, we compute the disparity in the Jaccard index between DNE and six representative baseline methods for each complex (Fig. 3B). Our observations reveal that DNE not only identifies more complexes but also achieves higher overlap scores. To better understand the performance of DNE in module identification, we scrutinize the Retromer complex (Fig. 4). Comprising genes PEP8, VPS35, VPS29, VPS17, and VPS5, the Retromer complex plays a pivotal role in vacuolar protein sorting (*36*). DNE successfully captures all members of this complex through its learned embeddings, whereas other methods merely capture a subset of the module or include spurious members. This analysis underscores DNE's ability to provide biologically meaningful and accurate embeddings for inferring protein functions.

DNE offers the flexibility of integrating protein features from protein language models

Unlike many network embedding methods that primarily focus on learning network structural information without considering node features, DNE provides the flexibility to incorporate these features into the embedding process. In this study, we enrich the *S. cerevisiae* network by retrieving its associated protein sequences from the *Saccharomyces* Genome Database (*37*) and converting them into protein features using a pretrained protein language model, ESM-2 (*38*). These features, which contain rich semantic information about proteins, are then integrated into the PPI network as node features (Fig. 5A). We chose this dataset for evaluation because it provides a complete set of protein sequences for each protein in its PPI network, whereas other benchmarks we considered do not offer complete protein sequences.

To evaluate the capability of DNE to integrate node features, we explored three distinct scenarios by using the following: (i) only protein features extracted from ESM-2; (ii) network structures alone; (iii) a combination of network structures and ESM-2 protein features. DNE demonstrates a substantial improvement, with over 20% rise in ROC-AUC for PPI prediction on the *S. cerevisiae* dataset, compared to using solely the protein features (Fig. 5B). Furthermore, DNE consistently outperforms other baseline methods (DGI, GRACE, and VGNAE) in scenarios both with and without node features. DNE also effectively integrates node features, demonstrating a 1.3% increase in ROC-AUC when including features compared to scenarios without them. These results highlight the effectiveness of DNE in improving protein representations by harmonizing protein sequence features with network structure information.

DISCUSSION

We have introduced a network embedding technique named DNE to learn meaningful and discriminative node embeddings from a given network. DNE characterizes each node in terms of the contrast between the representations from its immediate neighbors and farther nodes, in contrast to traditional methods (*12*, *16*–*18*, *22*) that focus primarily on limited-order proximity among nodes. By considering both the local connectivity pattern and interactions with the broader network, DNE allows for a more holistic understanding of node relationships within the network. Our evaluation of DNE on multiple PPI datasets demonstrates its enhanced capability over existing methods in accurately predicting PPIs and identifying functional modules. DNE also exhibits robustness against network perturbations and consistently outperforms other methods across different perturbation ratios (fig. S8). Moreover, DNE effectively captures biologically meaningful signals by reflecting the proximity between proteins in both PPI n-hop distances and Gene Ontology functional similarities via its embeddings (fig. S10).

While DNE has the capability to derive node embeddings solely from the structural information of networks, it also offers the flexibility

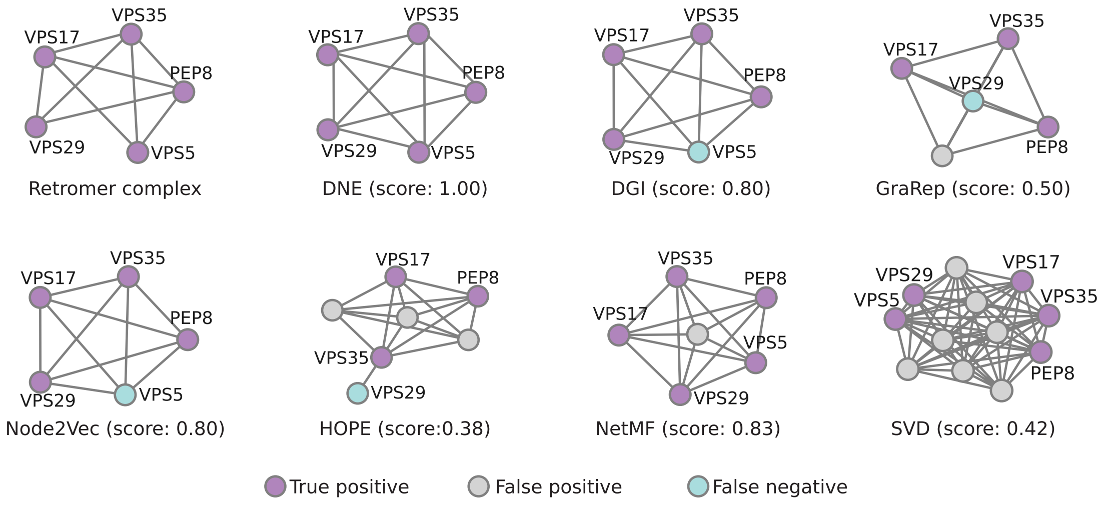

Fig. 4. Evaluation of the overlap between the predicted complex and the standard Retromer complex. The Retromer complex, as annotated by IntAct, serves as a benchmark to assess the performance of various methods in module identification. This standard complex consists of five members: PEP8, VPS35, VPS29, VPS17, and VPS5. The degree of overlap between the predicted complexes and the standard complex is measured using the Jaccard index. Purple indicates that the predicted member is part of the standard complex, gray indicates that the predicted member is not part of the standard complex, and green denotes that a member from the standard complex has not been captured by the prediction.

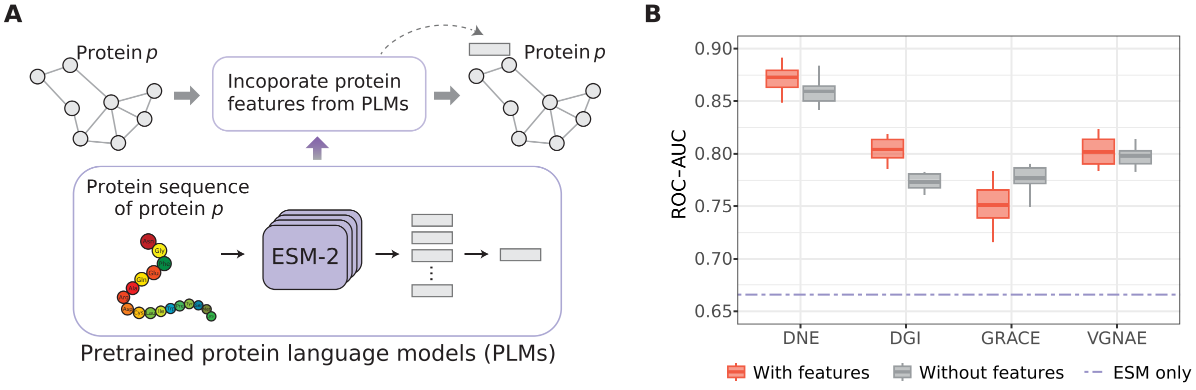

Fig. 5. Performance comparison of various methods in link prediction incorporating protein features. (**A**) Integrating protein features from PLMs as node features in PPI networks for network embedding learning. (**B**) ROC-AUC scores for DNE and other baseline methods on the *Saccharomyces cerevisiae* dataset, derived from 10 independent runs. The purple dashed line (ESM only) indicates scenarios using only protein features extracted from ESM-2. Gray boxes indicate cases considering only network structures, while red boxes depict cases incorporating both network structures and node features.

to incorporate node features into the embedding process when these features are available. In biological networks like PPIs, where each node represents a protein, node (or protein) features can be sourced from diverse sources such as amino acid sequences (*39*, *40*), the three-dimensional (3D) structure of proteins (*41*), and protein localization (*42*), providing additional information of proteins beyond their topological function within the network. Our method offers a remarkable approach for predicting PPIs by enhancing the network embedding through the integration of protein sequence features derived from pretrained protein language models. This integration substantially improves PPI prediction accuracy compared to existing methods that rely solely on sequence data.

Overall, DNE offers several advantages for network analysis. First, it generates a more discriminative embedding that captures not only the local connectivity patterns of each node but also distinguishes these patterns from those of other parts of the network. This enables a more accurate representation of each node's structural role and community membership, reducing the likelihood of overfitting to local network noise. Second, by incorporating data from immediate neighbors as well as other network segments, DNE provides a more holistic view of the entire network. Third, DNE can leverage both the structure of networks and node features to generate more enriched embeddings. In this work, these embeddings are used to infer protein interactions and identify functional modules. Further applications may include disease gene prediction (*43*), where the embeddings help identify proteins associated with disease mechanisms, and protein function prediction (*39*) to facilitate the annotation of proteins in newly sequenced genomes. It is worth noting that DNE's applicability extends beyond PPI networks and is applicable to various domains. Initial findings on diverse network types, such as citation networks (*44*), power grids (*45*), and internet service provider networks (*46*), suggest DNE's broader applicability (fig. S7). Therefore, our proposed method marks a notable advancement in network embedding and offers an urgently needed solution for high-performance network analysis.

While the proposed method shows promise for network analysis, future enhancements are feasible. First, the method currently prioritizes structural information over node features. While DNE can incorporate node features, they primarily serve to initialize the embeddings so that the final embeddings can reflect these node attributes. This process could be improved by considering the similarity among node features alongside node connections during the sampling of context nodes. Second, the proposed method uses multilayer perceptrons (MLPs) as the encoder. It could also be intriguing to investigate alternative network types for potential use as encoders, such as graph neural networks.

Biological networks such as PPIs serve as a backbone for advancing our understanding of complex biological systems. However, their inherent complexity often poses challenges in analysis and hinders downstream applications. In this study, we presented a self-supervised network embedding technique aimed at providing more discriminative low-dimensional embeddings of high-dimensional network data. The proposed technique uniquely captures the intrinsic characteristics of each node by leveraging insights from both its local environment and the broader network context. Extensive experimental studies across various biological networks demonstrate that this dual perspective offers a comprehensive and robust representation of the network, enabling reliable pattern discovery and accurate downstream network analysis. Thus, DNE promises to be a valuable tool for the fields of bioinformatics and systems biology.

MATERIALS AND METHODS

DNE is proposed to embed network nodes into low-dimensional representations to facilitate downstream biological analysis. The pretraining stage of DNE comprises three key steps (Fig. 1): (i) initializing nodes using node positional encoding obtained from eigenvalue decomposition of the network's adjacency matrix, optionally concatenated with node features when available; (ii) identifying node neighbors as positive context nodes via stochastic neighbor selection, based on random walks, and selecting nodes from other network regions as negative context nodes, based on the distribution of node degrees; (iii) embedding each node through MLP encoders, where the encoder's parameters are optimized to reduce distances between embeddings of the anchor node and its positive context nodes, while increasing those between the anchor node and its negative context pairs. The overall framework of DNE is shown in Fig. 1.

Preliminaries

Given an input network $G(V, E)$, where V represents nodes and E represents edges, DNE aims to learn its node representations, such that all nodes ($V = \{v_i \mid i = 1, 2, \ldots, |V|\}$) can be converted from high-dimensional space into a set of low-dimensional vectors, denoted as $\{\mathbf{z}_i \in \mathbb{R}^l \mid i = 1, 2, \ldots, |V|\}$, where l is the dimension of the output vectors and $l \ll |V|$. By learning effective node representations that capture the essential properties of the network, DNE aims to improve the speed and accuracy of graph analytics tasks, as opposed to directly performing such tasks in the complex high-dimensional graph domain.

Node input embedding construction

To construct node input embeddings x_i for nodes v_i ($i = 1, 2, \ldots, |V|$), eigenvalue decomposition is performed on the adjacency matrix to obtain node positional encodings. These encodings are then concatenated with node features (optional) to produce the input embeddings.

Positional encoding

Positional encoding involves eigenvalue decomposition on the network's adjacency matrix A to capture nodes' positions within the network's structure. The adjacency matrix A is a square matrix of size $|V| \times |V|$, with each element A_{ij} indicating the presence (or absence) of an edge between nodes v_i and v_j. Specifically, in an unweighted graph, if nodes v_i and v_j are connected, $A_{ij} = 1$ (for weighted graphs, $A_{ij} = w$, where w is the edge weight), otherwise, $A_{ij} = 0$. Specifically, DNE implements Laplacian eigenvectors (LEs) for node positional encodings.

Laplacian eigenvectors

LE-based positional encoding performs eigenvalue decomposition (*47*) of the normalized graph Laplacian matrix L, which is given by $L = D^{-1/2} A D^{-1/2}$. Here, D is the degree matrix, a diagonal matrix with the nodes' degrees on its diagonal, and A is the adjacency matrix of the graph

$$L = Q \Lambda Q^T$$

Here, Q represents the matrix of eigenvectors, and Λ is the diagonal matrix of eigenvalues. The k smallest nontrivial eigenvectors from L (excluding the trivial eigenvalue of zero) are then used as positional encoding. This method effectively captures the connectivity and relative distances between nodes in the graph G.

Node features

In addition to the adjacency matrix A, a network may also have an associated node feature matrix X, with dimensions $|V| \times |F|$, where $|F|$ represents the number of features of each node. These node features represent the specific characteristics of each node. For instance, in a PPI network, where nodes represent proteins and edges denote protein interactions, node features can be sourced from amino acid sequences (*39*, *40*), the 3D structure of proteins (*41*), and protein localization (*42*).

Positive and negative context nodes sampling

For each given node in a network, we treat it as the anchor node and initiate short random walks from this node to its neighbors, selecting nodes that co-occur on these walks as positive context nodes. In addition, an equal number of nodes are randomly sampled from the rest of the graph, following a probability distribution over the nodes where each node's probability of being sampled is proportional to its degree raised to a specific power (*20*). These nodes are considered as negative pairs.

Positive context nodes sampling via stochastic neighbors selection based on random walks

Random walks are used to sample a set of neighboring nodes for each node in the training graph $G(V, E)$. In random walks, we start at a chosen node and move to a neighboring node based on a probability distribution, known as the transition probability. This process is similar to a Markov chain, where the next state (or node) we move to depends only on the current state. Each step in the walk is determined by these transition probabilities, which dictate how likely it is to move from one node to its neighbor. A random walk rooted at node v_i (considered as the anchor node) can be represented by $\mathbf{W}_{v_i} = (r_0, r_1, \ldots, r_l)$ of length l over the graph from the source node $r_0 = v_i$. Specifically, the probability distribution of moving from one node to its neighbors in a random walk can be represented as follows

$$\mathbf{p}_{t+1}(i) = \sum_{j:(i,j) \in E} \frac{\mathbf{p}_t(j)}{\deg(j)}$$

where $\mathbf{p}_t(j)$ is the probability of being at node v_j at step t and $\deg(j)$ is the degree of node v_j (the number of edges connected to v_j). The transition probabilities can also be represented in matrix form as

$$\mathbf{p}_{t+1} = (AD^{-1})\mathbf{p}_t = (AD^{-1})^t \mathbf{p}_0$$

where $\mathbf{p}_0$ is the initial probability distribution across nodes, and $\mathbf{p}_t$ is the probability distribution after t steps. The matrix D is the diagonal degree matrix, with $D_{ii} = \deg(i)$ and zeros elsewhere. In our implementation, we initiate a specified number of random walks, denoted by γ, each with a length of l, from each node v_i. Specifically, we choose $l = 10$ and $= 100$ as the default settings. These random walks effectively explore the network, facilitating the sampling of nodes that are representative of the local neighborhood structure of the graph. An ablation study was conducted to evaluate the effects of positive node sampling parameters—walk length (l) and walk number (γ)—on link prediction, as shown in fig. S9. Overall, the model demonstrates robustness to variations in these parameters. On the basis of the analysis, increasing the walk length to a certain threshold captures more neighborhood nodes, but further increases can introduce noise by including distant or irrelevant nodes. Similarly, while increasing the walk number initially captures more nodes within the specified walk length, excessive increases may lead to redundancy without adding major improvements.

Negative context nodes sampling based on the distribution of node degrees

Negative context nodes are sampled using the distribution $P_v = \deg(v)^\beta$, where β is set to 0.75 on the basis of prior works (*48*). This distribution is further normalized such that the probabilities sum up to 1. Consequently, nodes with higher degrees are more likely to be sampled as negative context nodes, and vice versa.

Encoder training via contrastive loss

Encoder architecture

DNE leverages MLPs to generate structure-aware representations from node input embeddings. Given $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|V|})$, where $\mathbf{x}_i$ denotes the input embedding of i-th node, the DNE encoder learns a function $f_\theta(\cdot): \mathcal{X} \to \mathbb{R}^d$ that encodes $\mathbf{x}_i \in \mathcal{X}$ into a fixed-dimension vector representation $f_\theta(x_i) \in \mathbb{R}^d$. The encoder comprises two MLP blocks, each containing a fully connected layer, GeLU activation, batch normalization, and a dropout layer. The encoder serves as a nonlinear projection head to map nodes input into fixed-length vector representations

$$\mathrm{MLPBlock}(X) = \mathrm{DropOut}\left\{\mathrm{BN}\left[\mathrm{GeLU}(WX + \mathbf{b})\right]\right\}$$

$$f_\theta(X) = \mathrm{MLPBlock}(\mathrm{MLPBlock}X)$$

Note that when focusing only on network structures, we use a single MLP encoder. However, to integrate node features, we implement a dual encoder approach—one encoder for network structures and another for node features—with their outputs subsequently concatenated (see Fig. 1A).

Encoder training using contrastive loss

For a given set of node inputs $\{\mathbf{x}_i\}$, our encoder $f_\theta(\cdot): \mathcal{X} \rightarrow \mathbb{R}^d$ maps a node input $\mathbf{x}_i$ into a vector of dimension d such that nodes from the same random walks have similar embeddings and vice versa. Our contrastive loss minimizes the embedding distance between a pair of node inputs $(\mathbf{x}_i, \mathbf{x}_j)$ if they are from the same random walk but maximizes the distance otherwise

$$\begin{aligned}\mathcal{L}\left(\mathbf{x}_i, \mathbf{x}_j, \theta\right) = &\ \mathbf{1}\left[y_i = y_j\right] \|\left[f_\theta\left(\mathbf{x}_i\right) - f_\theta\left(\mathbf{x}_j\right)\right]\|_1^2 \\ &+ \mathbf{1}\left[y_i \neq y_j\right] \max\left\{0, \epsilon - \|\left[f_\theta\left(\mathbf{x}_i\right) - f_\theta\left(\mathbf{x}_j\right)\right]\|_1\right\}^2\end{aligned}$$

where $y_i = y_j$ if nodes v_i and v_j are from the same random walks; otherwise, $y_i \neq y_j$. $f_\theta\left(\mathbf{x}_i\right)$ and $f_\theta\left(\mathbf{x}_j\right)$ are outputs of the DNE encoder parameterized by θ applied to inputs $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively. The simplified loss can be written as

$$\begin{aligned}\mathcal{L}(\theta) = &\sum_{u \in V} \sum_{v \in N_R(u)} |f_\theta\left(\mathbf{x}_u\right) - f_\theta\left(\mathbf{x}_v\right)|^2 \\ &+ \sum_{i=1}^{k} \max\left[0, \epsilon - |f_\theta\left(\mathbf{x}_u\right) - f_\theta\left(\mathbf{x}_{n_i}\right)|\right]^2, n_i \sim P_v\end{aligned}$$

where V refers to the entire node set of the input network, $N_R(u)$ represents the sampled neighbors for the source node u using random walks, n_i denotes the negative context nodes drawn from the degree-based probability distribution $P_v = \deg(v)^\beta$, k represents the number of negative context nodes, and ϵ is a constant margin. Here, the first term encourages embeddings of each node u and its neighbors v to be similar by minimizing the squared L_1 norm between their embeddings. The second term ensures embeddings of u and a negative sample n_i are at least ϵ apart by penalizing embeddings that are too close (less than ϵ) with a squared penalty. The loss function $\mathcal{L}(\theta)$ is optimized through Adam. During optimization, the parameters θ of the encoder f_θ are adjusted to minimize $\mathcal{L}(\theta)$.

In the optimal scenario where $\mathcal{L}\left(\theta^*\right)$ is minimized, the embeddings of each node $u \in V$ can be expressed as a nonlinear combination of the features of its positive and negative context nodes, as shown by the equation

$$f_{\theta*}\left(\mathbf{x}_u\right) = \frac{\sum_{v \in N_R(u)} f_{\theta*}\left(\mathbf{x}_v\right)^2 - \sum_{i=1}^{k} f_{\theta*}\left(\mathbf{x}_{n_i}\right)^2 + k\epsilon}{2\left[\sum_{v \in N_R(u)} f_{\theta*}\left(\mathbf{x}_v\right)^2 - \sum_{i=1}^{k} f_{\theta*}\left(\mathbf{x}_{n_i}\right)^2\right]}$$

Here, the presence of positive context nodes contributes positively to the embedding of the anchor node, while its negative context nodes contribute negatively. The weights are determined by the relative importance of positive and negative context nodes, adjusted by $k\epsilon$ and the denominator.

Network datasets

Our study incorporates a collection of interactome networks from various organisms to evaluate the performance of our method compared to existing network embedding techniques for link prediction. These datasets include the following: (i) the *A. thaliana* interactome, featuring 2774 proteins and 6205 PPIs (*26*); (ii) the *C. elegans* interactome, consisting of 2528 proteins and 3864 PPIs (*27*); (iii) the *S. cerevisiae* yeast interactome, comprising 2674 proteins and 7075 PPIs (*28*); and (iv) an extensive human interactome from the HuRI (*29*) project, which includes 8272 proteins and 52,548 PPIs. These diverse datasets allow us to conduct a comprehensive evaluation across different species and showcase the adaptability of our method.

Benchmark construction for module detection

To assess the effectiveness of our method in identifying functional modules within PPIs, we used data from the *S. cerevisiae* yeast network. For constructing our benchmarks for module detection, we obtained annotations from several sources: IntAct protein complexes (*32*), the KEGG pathways (*33*), and GOBP (*34*). Modules were identified on the basis of the collection of genes annotated with a particular term from each source. Modules consisting of only one gene were excluded because of their lack of informational value.

Protein sequence features obtained using ESM-2

Our study further enhances the *S. cerevisiae* PPI by integrating protein sequences from the *Saccharomyces* Genome Database (http://sgd-archive.yeastgenome.org/sequence/S288C_reference/orf_protein/), alongside features derived from the pretrained protein language model, ESM-2 (*38*). We selected this dataset for evaluation because it provides complete protein sequences for each protein in its PPI network, unlike other benchmarks we considered. ESM-2 is a transformer-based language model trained on around 65 million unique sequences and learned representations of protein sequences that reflect their biological properties. In this study, we used the pretrained ESM-2 model (esm2_t36_3B_UR50D) to generate protein sequence features, each of length 2560. This was achieved by averaging the last layer outputs of ESM-2 for each amino acid in a protein sequence. The pretrained model is available for public access on HuggingFace.

Downstream prediction models

We developed downstream models for various tasks using node embeddings obtained from network embedding methods as inputs. For link prediction, which was considered as a binary classification task (presence or absence) based on link embeddings, we first constructed link embeddings using four operations: the Hadamard product, absolute difference, squared difference, and averaging of the node embeddings from the start and end nodes of each link. Subsequently, we used logistic regression (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html) as the classifier, with a fivefold cross-validation applied to achieve the best performance of each method under different operators. For module identification, we applied hierarchical agglomerative clustering (*35*) using the AgglomerativeClustering function from the scikit-learn library (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html) to cluster the nodes on the basis of their embeddings. We investigated the performance of each method across different linkage methods (single, average, and complete), distance metrics (Euclidean and cosine), and thresholds to obtain the best performance for each method.

Evaluation strategy

To evaluate the methods for link prediction, we randomly selected 20% of the edges E from the input network for testing to create the test edge set E_{test}, while the remaining edges formed the training network and its corresponding training edge set E_{train}. We then conducted a fivefold cross-validation on the remaining data to obtain optimal performance and repeated this process for 10 independent runs. To train our logistic regression classifier with both existing and nonexisting links, we constructed a dataset comprising positive and negative edge examples. Specifically, we selected 10% of edges from E_{train} as positive examples, and sampled an equal number of nonlinked node pairs from the network as negative examples. The performance of each network embedding method for link prediction was evaluated using the ROC-AUC and the PR-AUC, both widely used metrics for this task (*49*).

For module identification, we subsampled the module sets to ensure that each gene was assigned to a single cluster, aligning with the assumption behind standard clustering evaluation metrics like the AMI. Subsequently, we assessed our predicted cluster sets against the benchmark module sets using AMI as our primary metric. We report the highest AMI score for each method to ensure the optimal cluster set for each dataset across clustering parameters is used. This evaluation was repeated 10 times to account for score variations due to the cluster subsampling strategy. Moreover, to assess the similarity between the clusters identified by our methods and known IntAct complexes, we used the Jaccard score (ranges from 0 to 1) to quantitatively measure the set overlaps.

Supplementary Materials

This PDF file includes:
Sections S1 to S8
Figs. S1 to S11
Tables S1 and S2

REFERENCES AND NOTES

1. A.-L. Barabasi, Z. N. Oltvai, Network biology: Understanding the cell's functional organization. *Nat. Rev. Genet.* **5**, 101–113 (2004).
2. U. Alon, Biological networks: The tinkerer as an engineer. *Science* **301**, 1866–1867 (2003).
3. D. M. Camacho, K. M. Collins, R. K. Powers, J. C. Costello, J. J. Collins, Next-generation machine learning for biological networks. *Cell* **173**, 1581–1592 (2018).
4. A.-L. Barabási, N. Gulbahce, J. Loscalzo, Network medicine: A network-based approach to human disease. *Nat. Rev. Genet.* **12**, 56–68 (2011).
5. R. Bonneau, Learning biological networks: From modules to dynamics. *Nat. Chem. Biol.* **4**, 658–664 (2008).
6. L. Hakes, J. W. Pinney, D. L. Robertson, S. C. Lovell, Protein-protein interaction networks and biology–what's the connection? *Nat. Biotechnol.* **26**, 69–72 (2008).
7. S. Wang, R. Wu, J. Lu, Y. Jiang, T. Huang, Y.-D. Cai, Protein-protein interaction networks as miners of biological discovery. *Proteomics* **22**, e2100190 (2022).
8. T. Tang, X. Zhang, Y. Liu, H. Peng, B. Zheng, Y. Yin, X. Zeng, Machine learning on protein–protein interaction prediction: Models, challenges and trends. *Brief. Bioinform.* **24**, bbad076 (2023).
9. W. Nelson, M. Zitnik, B. Wang, J. Leskovec, A. Goldenberg, R. Sharan, To embed or not: Network embedding as a paradigm in computational biology. *Front. Genet.* **10**, 452819 (2019).
10. C. Su, J. Tong, Y. Zhu, P. Cui, F. Wang, Network embedding in biomedical data science. *Brief. Bioinform.* **21**, 182–197 (2020).
11. M. M. Li, K. Huang, M. Zitnik, Graph representation learning in biomedicine and healthcare. *Nat. Biomed. Eng.* **6**, 1353–1369 (2022).
12. S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**, 2323–2326 (2000).
13. D. Luo, F. Nie, H. Huang, C. H. Ding, Cauchy graph embedding, in *Proceedings of the 28th International Conference on Machine Learning* (2011), pp. 553–560.
14. D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 1225–1234.
15. P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding. *IEEE Trans. Knowl. Data Eng.* **31**, 833–852 (2019).
16. S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (2015), pp. 891–900.
17. M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, Asymmetric transitivity preserving graph embedding, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 1105–1114.
18. J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, J. Tang, Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec, in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (2018), pp. 459–467.
19. J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in *Proceedings of the 24th International Conference on World Wide Web* (2015), pp. 1067–1077.
20. A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 855–864.
21. W. Ju, Z. Fang, Y. Gu, Z. Liu, Q. Long, Z. Qiao, Y. Qin, J. Shen, F. Sun, Z. Xiao, J. Yang, J. Yuan, Y. Zhao, Y. Wang, X. Luo, M. Zhang, A comprehensive survey on deep graph representation learning. *Neural Netw.* **173**, 106207 (2024).
22. T. N. Kipf, M. Welling, Variational graph auto-encoders. arXiv:1611.07308 [stat.ML] (2016).
23. P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax, in *International Conference on Learning Representations* (2019).
24. Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Deep graph contrastive representation learning, in *ICML Workshop on Graph Representation Learning and Beyond* (2020).
25. S. J. Ahn, M. Kim, Variational graph normalized autoencoders, in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management* (2021), pp. 2827–2831.
26. Arabidopsis Interactome Mapping Consortium, M. Dreze, A.-R. Carvunis, B. Charloteaux, M. Galli, S. J. Pevzner, M. Tasan, Y.-Y. Ahn, P. Balumuri, A.-L. Barabási, V. Bautista, P. Braun, D. Byrdsong, H. Chen, J. D. Chesnut, M. E. Cusick, J. L. Dangl, C. de los Reyes, A. Dricot, M. Duarte, J. R. Ecker, C. Fan, L. Gai, F. Gebreab, G. Ghoshal, P. Gilles, B. J. Gutierrez, T. Hao, D. E. Hill, C. J. Kim, R. C. Kim, C. Lurin, A. MacWilliams, U. Matrubutham, T. Milenkovic, J. Mirchandani, D. Monachello, J. Moore, M. S. Mukhtar, E. Olivares, S. Patnaik, M. M. Poulin, N. Przulj, R. Quan, S. Rabello, G. Ramaswamy, P. Reichert, E. A. Rietman, T. Rolland, V. Romero, F. P. Roth, B. Santhanam, R. J. Schmitz, P. Shinn, W. Spooner, J. Stein, G. M. Swamilingiah, S. Tam, J. Vandenhaute, M. Vidal, S. Waaijers, D. Ware, E. M. Weiner, S. Wu, J. Yazaki, Evidence for network evolution in an *Arabidopsis* interactome map. *Science* **333**, 601–607 (2011).
27. N. Simonis, J.-F. Rual, A.-R. Carvunis, M. Tasan, I. Lemmens, T. Hirozane-Kishikawa, T. Hao, J. M. Sahalie, K. Venkatesan, F. Gebreab, S. Cevik, Empirically controlled mapping of the *Caenorhabditis elegans* protein-protein interactome network. *Nat. Methods* **6**, 47–54 (2009).
28. N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis, T. Punna, Global landscape of protein complexes in the yeast *Saccharomyces cerevisiae*. *Nature* **440**, 637–643 (2006).
29. K. Luck, D.-K. Kim, L. Lambourne, K. Spirohn, B. E. Begg, W. Bian, R. Brignall, T. Cafarelli, F. J. Campos-Laborie, B. Charloteaux, A reference map of the human binary protein interactome. *Nature* **580**, 402–408 (2020).
30. A. Kumar, S. S. Singh, K. Singh, B. Biswas, Link prediction techniques, applications, and performance: A survey. *Physica A Stat. Mech. Appl.* **553**, 124289 (2020).
31. S. Choobdar, M. E. Ahsen, J. Crawford, M. Tomasoni, T. Fang, D. Lamparter, J. Lin, B. Hescott, X. Hu, J. Mercer, Assessment of network module identification across complex diseases. *Nat. Methods* **16**, 843–852 (2019).
32. S. Orchard, M. Ammari, B. Aranda, L. Breuza, L. Briganti, F. Broackes-Carter, N. H. Campbell, G. Chavali, C. Chen, N. Del-Toro, The MIntAct project–IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Res.* **42**, D358–D363 (2014).
33. M. Kanehisa, S. Goto, KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* **28**, 27–30 (2000).
34. M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, Gene ontology: Tool for the unification of biology. *Nat. Genet.* **25**, 25–29 (2000).
35. D. Müllner, Modern hierarchical, agglomerative clustering algorithms. arXiv:1109.2378 [stat.ML] (2011).
36. M. N. Seaman, The retromer complex: From genesis to revelations. *Trends Biochem. Sci.* **46**, 608–620 (2021).

37. J. M. Cherry, C. Adler, C. Ball, S. A. Chervitz, S. S. Dwight, E. T. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, D. Botstein, SGD: *Saccharomyces* Genome Database. *Nucleic Acids Res.* **26**, 73–79 (1998).
38. Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
39. M. Kulmanov, F. J. Guzmán-Vega, P. Duek Roggli, L. Lane, S. T. Arold, R. Hoehndorf, Protein function prediction as approximate semantic entailment. *Nat. Mach. Intell.* **6**, 220–228 (2024).
40. M. Kulmanov, M. A. Khan, R. Hoehndorf, DeepGO: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics* **34**, 660–668 (2018).
41. Z. Gao, C. Jiang, J. Zhang, X. Jiang, L. Li, P. Zhao, H. Yang, Y. Huang, J. Li, Hierarchical graph learning for protein–protein interaction. *Nat. Commun.* **14**, 1093 (2023).
42. T. Hamp, B. Rost, Evolutionary profiles improve protein–protein interaction prediction from sequence. *Bioinformatics* **31**, 1945–1950 (2015).
43. S. K. Ata, M. Wu, Y. Fang, L. Ou-Yang, C. K. Kwoh, X.-L. Li, Recent advances in network-based methods for disease gene prediction. *Brief. Bioinform.* **22**, bbaa303 (2021).
44. A. K. McCallum, K. Nigam, J. Rennie, K. Seymore, Automating the construction of internet portals with machine learning. *Inf. Retr.* **3**, 127–163 (2000).
45. D. J. Watts, S. H. Strogatz, Collective dynamics of 'small-world' networks. *Nature* **393**, 440–442 (1998).
46. N. Spring, R. Mahajan, D. Wetherall, Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Comput. Commun. Rev.* **32**, 133–145 (2002).
47. M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **15**, 1373–1396 (2003).
48. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in *Advances in Neural Information Processing Systems* (2013).
49. Y. Yang, R. N. Lichtenwalter, N. V. Chawla, Evaluating link prediction methods. *Knowl. Inf. Syst.* **45**, 751–782 (2015).

Acknowledgments

Funding: The authors would like to thank the grant support from Stanford Human-Centered Intelligence (HAI) and National Institutes of Health (NIH) (1R01CA223667, 1R01CA176553, and 1R01CA227713). **Author contributions:** L.X. conceived the experiments. R.Y. conducted the experiments and analyzed the results. R.Y., M.T.I., and L.X. wrote the manuscript. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** Code and data supporting this study are available on Github (https://github.com/rui-yan/DNE) and Zenodo (10.5281/zenodo.13752181). All data needed to evaluate the conclusions of the paper are present in the paper and/or the Supplementary Materials.

Submitted 13 May 2024
Accepted 14 November 2024
Published 18 December 2024
10.1126/sciadv.adq4324