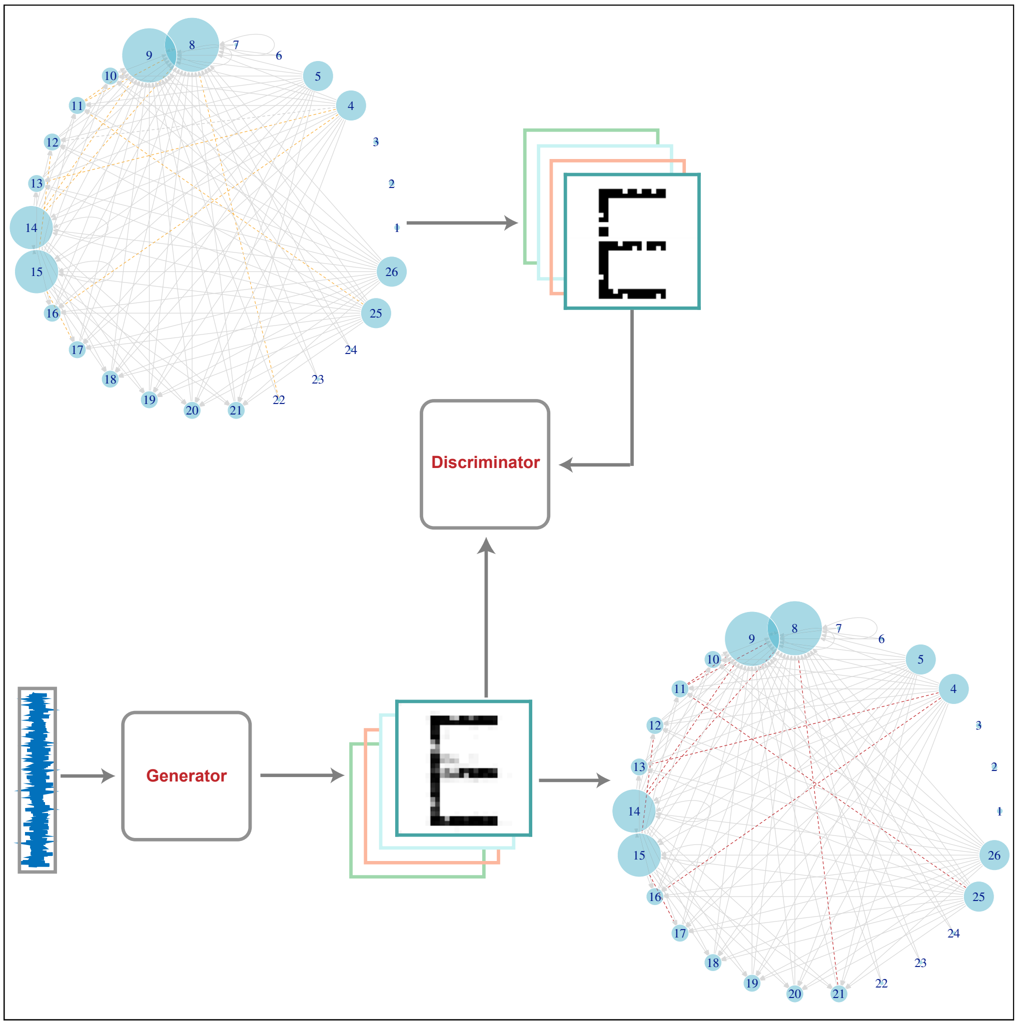
iScience

CellPress
OPEN ACCESS

Article

Link Prediction through Deep Generative Model



Xu-Wen Wang, Yize Chen, Yang-Yu Liu

yyl@channing.harvard.edu

HIGHLIGHTS

A novel link prediction method based on deep generative models is developed

This method works for general undirected or directed complex networks

Leveraging structural patterns at different scales, this method outperforms others

Wang et al., iScience *23*, 101626
October 23, 2020 © 2020 The Author(s).
https://doi.org/10.1016/j.isci.2020.101626

Article

Link Prediction through Deep Generative Model

Xu-Wen Wang,[1] Yize Chen,[2] and Yang-Yu Liu[1,3,*]

SUMMARY

Inferring missing links based on the currently observed network is known as link prediction, which has tremendous real-world applications in biomedicine, e-commerce, social media, and criminal intelligence. Numerous methods have been proposed to solve the link prediction problem. Yet, many of these methods are designed for undirected networks only and based on domain-specific heuristics. Here we developed a new link prediction method based on deep generative models, which does not rely on any domain-specific heuristic and works for general undirected or directed complex networks. Our key idea is to represent the adjacency matrix of a network as an image and then learn hierarchical feature representations of the image by training a deep generative model. Those features correspond to structural patterns in the network at different scales, from small subgraphs to mesoscopic communities. When applied to various real-world networks from different domains, our method shows overall superior performance against existing methods.

INTRODUCTION

Networks have become an invaluable tool for describing the architecture of various complex systems, be they technological, biological, or social in nature (Albert and Barabási, 2002; Newman, 2003; Boccaletti et al., 2006; Scholtes et al., 2014). Mathematically, any real-world network can be represented by a graph, G(V,E), where V={1, 2, ⋯,N} is the node set and E⊆V×V is the link set. A link, denoted as a node pair (i,j) with i,j∈V, represents certain interaction, association, or physical connection between nodes i and j, which could be either directed or undirected, weighted or unweighted. For many systems (especially biological systems), the discovery and validation of links require significant experimental efforts. Consequently, many real-world networks mapped so far are substantially incomplete (Von Mering et al., 2002; Han et al., 2005). For example, a recent estimate indicates that in human cells the explored protein-protein interactions cover less than 20% of all potential protein-protein interactions (Sahni et al., 2015). How to tease out the missing interactions based on the discovered ones? In network science and machine learning, this is commonly known as the link prediction problem (Liben-Nowell and Kleinberg, 2007; Clauset et al., 2008).

An accurate link prediction method will greatly reduce the experimental efforts required to establish the network's topology and/or accelerate mutually beneficial interactions that would have taken much longer to form serendipitously. Consequently, link prediction has many real-world applications (Hulovatyy et al., 2014; Martínez, Berzal and Cubero, 2016a). In biomedicine, link prediction can be used to infer protein-protein interactions or drug-target interactions (Zhang et al., 2005; Campillos et al., 2008; Luo et al., 2017). In e-commerce, it can help build better recommender systems, e.g., Amazon's "people who bought this also bought" feature (Linden et al., 2003). On social media, it can help build potential connections such as the "people you may know" feature on Facebook and LinkedIn (Blagus et al., 2012). In criminal intelligence analysis, link prediction can assist in identifying hidden co-participation in illicit activities (Berlusconi et al., 2016).

Numerous methods, such as similarity-based algorithms (Katz, 1953; Barabási and Albert, 1999; Friedman et al., 1999; Sarukkai, 2000; Guimerà and Sales-Pardo, 2009; Lü and Zhou, 2011; Perozzi et al., 2014; Chen et al., 2017; Kovács et al., 2019), maximum likelihood algorithms (Clauset et al., 2008; Guimerà and Sales-Pardo, 2009; Pan et al., 2016), probabilistic models (Heckerman et al., 2007; Chaney et al., 2015), and deep learning-based methods (Tavakoli et al., 2017; Chiu and Zhan, 2018), especially graph representation learning-based methods (Niepert et al., 2016, Ahmed and Kutzkov, no date; van den Berg et al., 2017; Hamilton et al., 2017; Schlichtkrull et al., 2017; Murphy et al., 2019; Srinivasan and Ribeiro, 2019) have been

[1]Channing Division of Network Medicine, Brigham and Women's Hospital and Harvard Medical School, Boston, MA 02115, USA

[2]Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195, USA

[3]Lead Contact

*Correspondence: yyl@channing.harvard.edu

https://doi.org/10.1016/j.isci.2020.101626

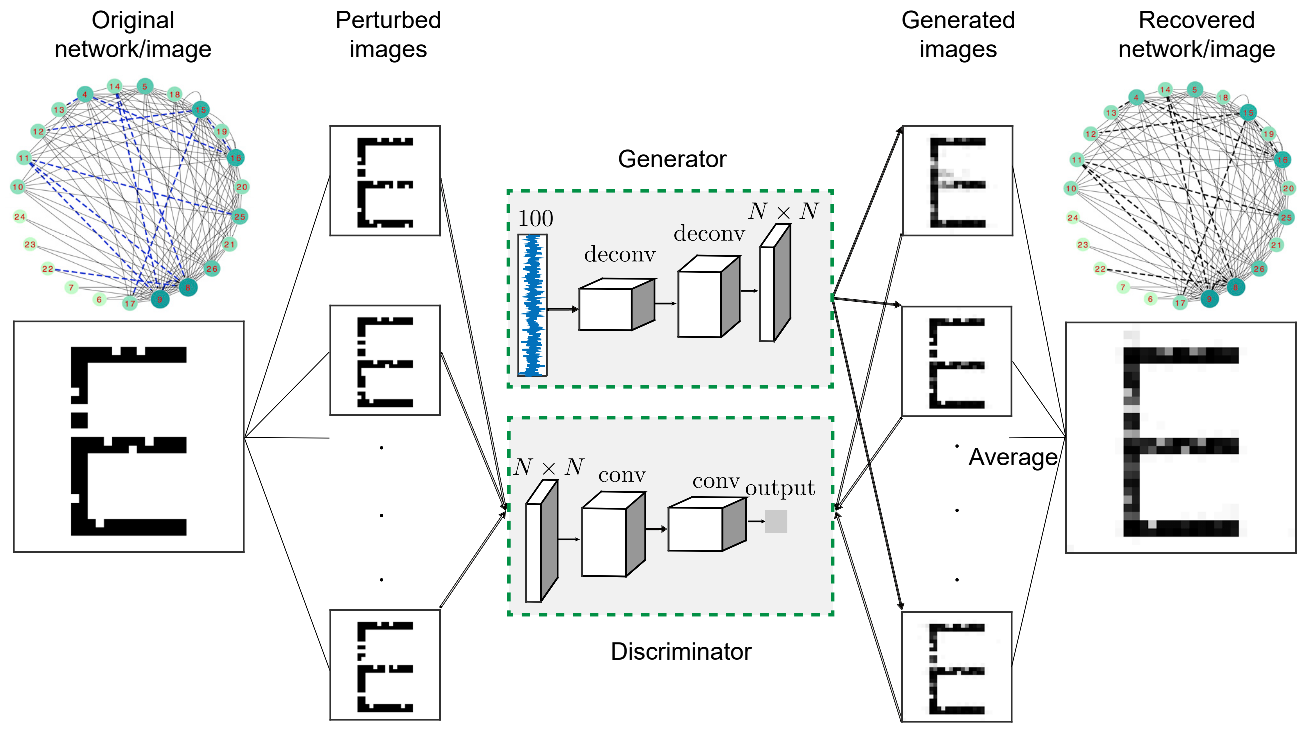


Figure 1. Demonstration of Our Link Prediction Method on a Directed Network

The adjacency matrix of this directed network (with 28 nodes and 118 links) looks like the binary image of letter E with 12 missing pixels. Note that 5 isolated nodes are not shown in the network presentation. We perturb the original network (image) by removing 5 links at random in M different ways to obtain a pool of perturbed networks (images) I_i ($i = 1, \ldots, M$) ($M = 5000$ for this example). This input dataset will be fed into the generative adversarial networks (GANs) that consist of two deep artificial neural networks: generator and discriminator. The generator takes the noise drawn from a uniform distribution (with 100 dimensions for this example) as input and produces fake images. The discriminator is a binary classifier that tells whether a given image is a real one from the input dataset or a fake one produced by the generator. Over the course of training iterations, the generator can produce convincing fake images P from the feedback offered by the discriminator. The pixel value P_{ij} in the fake grayscale image P can be used to calculate the existent probability of a link between a node pair: $\alpha_{ij} = 1 - P_{ij}$. The final existent probability is calculated by averaging α_{ij} over S ($S = 500$) generated fake networks.

developed to solve the link prediction problem (see Supplementary Information Section 1 for brief descriptions of those existing methods). Yet, many of these existing methods (such as similarity-based algorithms) are designed for undirected networks. Moreover, most of these methods are based on domain-specific heuristics (Sarukkai, 2000), and hence their performances differ greatly for networks from different domains.

A powerful link prediction method that does not rely on any domain-specific heuristic and works for general complex networks has been lacking (Martínez, Berzal and Cubero, 2016b). Here, we fill the gap by developing a link prediction method based on deep generative models (DGMs) (see Figure 1 for a schematic demonstration).

RESULTS

Key Idea

Our key idea is to treat the adjacency matrix of a network as the pixel matrix of a binary image. In other words, present (or absent) links will be treated as pixels of value (0 or 1), respectively. By perturbing the original input network (image) in many different ways through randomly removing a small fraction of present links, we obtain a pool of perturbed networks (images). Those perturbed images will be fed into a DGM to create fake images that look similar to the input ones (see Supplementary Information Section 3 for details of DGMs). Those fake images (networks) will be used to perform link prediction in the original image (network). For the DGM, here we leverage one of the most popular ones, Generative Adversarial Networks (GANs), that consist of two deep artificial neural networks (called *generator* and *discriminator*) contesting with each other in a game theory framework (Goodfellow et al., 2014; Arjovsky et al., 2017). The generator takes random noise from a known distribution as input and transforms them into fake images through a deconvolutional neural network. The discriminator is a binary classifier (based on a convolutional neural network), which determines whether a given image looks like a real one from the input dataset or like a fake one artificially created by the generator. Over the course of training iterations, the discriminator learns to tell real images from fake ones. At the same time, the generator uses feedback from the discriminator to learn how to produce convincing fake images to fool the discriminator so that it cannot distinguish from real ones (see Supplementary Information Section 3 for details). To better train the generator, one can use the Wasserstein distance to quantify the dissimilarity between fake and real images. During the training process, through minimizing the Wasserstein distance, the generator learns to assign link probabilities between each node pair (including both observed and unobserved links) to fool the discriminator so that it cannot distinguish real and fake images. Note that the assigned link probabilities to those observed links will be quite close to one, whereas the link probabilities assigned to those unobserved links will be close to zero but not exactly zero. (This process is also known as the smooth process [Yeh et al., 2017]). Hence the generated fake images are grayscale, even though all the input images fed to GAN are binary. If the probabilities assigned to missing links are much higher than that of nonexistent links, then the link prediction is much better than random guess (see Figure S1 for an intuitive explanation).

Demonstration Using Synthetic Networks

To demonstrate our DGM-based link prediction, let us consider a toy example: a small directed network of 28 nodes and 118 links (Figure 1): 106 solid links form the training set, whereas 12 dashed links form the probe set. Those nodes are labeled appropriately so that the adjacency matrix of this network looks like a binary image of letter **E** with 12 missing pixels, corresponding to 12 removed or "missing" links. Note that those probe links will never enter the learning process. First, we create M perturbed binary images by randomly removing a fraction q of pixels of value 0 (i.e., those present links) from the original image (network). Second, we use the M perturbed binary images as input to train GANs, which will eventually generate S fake grayscale images that look similar to the input ones. In this example, we choose M = 5,000, q = 0.1, and S = 500. The existent likelihood of the link between nodes i and j, denoted as α_{ij}, in the corresponding fake network is simply given by $\alpha_{ij} = 1-P_{ij}$, where P_{ij} is the rescaled pixel value (ranging from 0 to 1) in each fake grayscale image. Finally, we take the average value $\alpha_{ij} = 1-P_{ij}$ over all the S fake images to get the overall existent likelihood of the link (i,j). Note that in this toy example all the 12 missing links display higher α_{ij} than that of nonexistent links, so they are all successfully recovered. Figure 1 may remind us the classical image inpainting problem, where we need to reconstitute or retouch the missing or damaged regions of an image to make it more legible and to restore its unity (Bertalmio et al., 2000). We emphasize that the link prediction problem addressed here is fundamentally different from the image inpainting problem. For image inpainting, we generally know the locations of the damaged regions of an image, whereas for link prediction, we do not know which links are missing in a network. In fact, teasing them out is exactly the task of link prediction.

At the first glance, our DGM-based link prediction method seems to heavily rely on the existing patterns in the adjacency matrix of the original network. After all, we are treating a network as an image. But do we have to sophisticatedly label the nodes in the network to ensure the success of our method? To address this concern, we perform the following numerical experiment. We start from an original network with an appropriate node labeling such that the adjacency matrix looks exactly as the binary image of letter **E** without any missing pixels. (See Supplementary Information Figure S2 for a more complicated synthetic network generated by the stochastic block model.) Then we relabel η fraction of the nodes in the network so that the binary image associated with its adjacency matrix looks much more random than the letter **E** (see insets of Figure 2A). Note that the network structure is fixed, while we just label the nodes differently so that the resulting adjacency matrices (or binary images) look quite different. We then randomly remove 10% links of those five networks as probe set to evaluate the performance of our method at different η values, as well as the performance of two classical link prediction methods for directed networks that do not depend on the node labeling at all. Hereafter, to quantify the performance of any link prediction method, we employ the standard AUC statistic, i.e., the area under the receiver operating characteristic curve (Clauset et al., 2008; Guimerà and Sales-Pardo, 2009). To calculate the AUC, we first randomly split the link set ε into two parts (see Figures S3 and S4 for details): (1) a fraction f of links as the test or probe set ε^P, which will be removed from the network; and (2) the remaining fraction $(1-f)$ of links as the training set ε^T, which will be used to recover the removed links. The AUC statistic is defined to be the probability that a randomly chosen link in the probe set ε^P is given a higher score by the link prediction method than that of a randomly chosen nonexistent link (see Supplementary

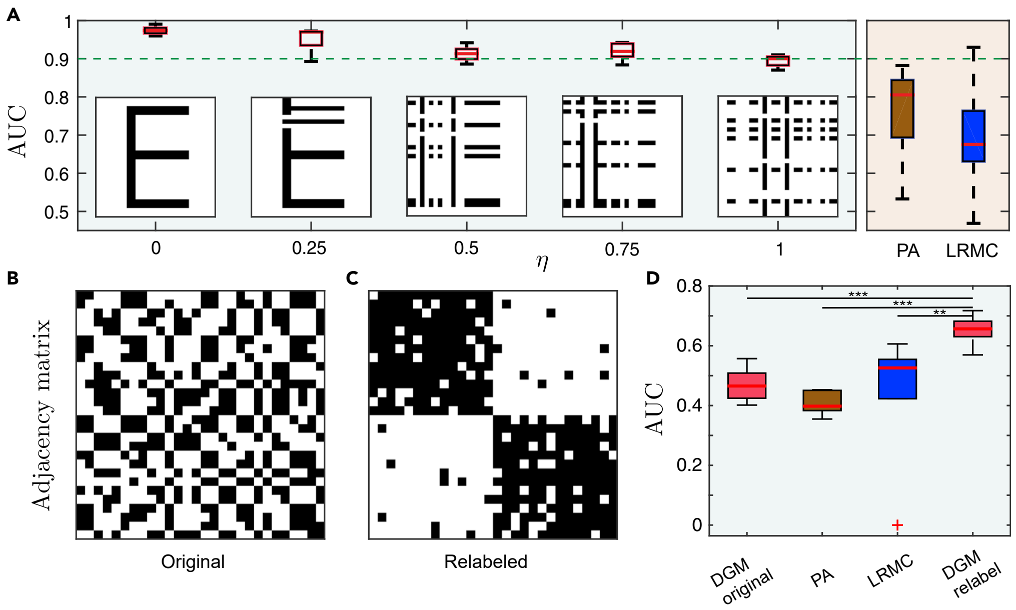


Figure 2. Impact of Node Labeling on the Performance of Our DGM-Based Link Prediction Method

(A) A randomly selected fraction of η nodes are relabeled in a directed network whose original adjacency matrix looks exactly as the binary image of letter (E) We randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% links as the training set. We perform link prediction using three different methods: DGM (deep generative model based), PA (preferential attachment based), and LRMC (low rank matrix completion). In this example, we choose M = 1000 for our DGM-based method. Even after we relabel all the nodes so that the adjacency matrix does not display prominent features, the median AUC of our DGM-based method is still around 0.9, whereas it is 0.85 for the PA method and 0.7 for the LRMC method. Inset: The adjacent matrices corresponding to different relabeling fractions, where black pixels represent existing links.

(B–D) AUCs of DGM-based and other traditional methods in the link prediction of a directed modular network (N = 28) generated by the stochastic block model (Girvan and Newman, 2002) with within-module connection probability p_{in} = 0.5 and between-module connection probability p_{out} = 0.05. The adjacent matrices before (or after node relabeling) is shown in (B) (or C), respectively. Asterisks in (D) show whether the AUC of our DGM-based link prediction method is significantly higher than that of the other three traditional algorithms (paired-sample t test).

Significance levels: p value <0.01(**), <0.001(***).

Information Section 2 and Figure S5 for details). For each network, we performed 20 independent random splittings unless otherwise stated. We find that, for this small directed network, the performance of our method degrades only slightly even after we relabel 25% nodes (Figure 2A). When we relabel more nodes, the performance is actually quite stable. Even if we relabel all the nodes, the AUC of our method is still about 0.9, which is higher than that of other link prediction methods for directed networks, such as the preferential attachment (PA) (Barabási and Albert, 1999)-based method (with AUC~0.85) and the low-rank matrix completion (LRMC) (Pech *et al.*, 2017) method (with AUC~0.7). In Supplementary Information Figure S6, we further show that the AUC of our method is generally above 0.9 with different completely random node labeling of this network. This is simply because even after completely random node labeling, small-scale patterns (e.g., many short line segments in the relabeled image of **E**) can still be readily leveraged by our method.

The results presented in Figure 2A indicate that, for those networks that have strong structural patterns, our DGM-based link prediction does not heavily rely on the detailed node labeling. However, to optimize the performance of our method, one should still label the nodes accordingly. This can be achieved by extracting community structure in the network (Newman and Girvan, 2004; Radicchi *et al.*, 2004), for example, using the classical Louvain method (Blondel *et al.*, 2008). (Note that nodes within a community can be labeled randomly and the test set should not be involved in the node labeling.) To test this simple idea, we consider a directed modular network generated from the stochastic block model (Girvan and Newman, 2002), where any two nodes within the same module are randomly connected with probability p_{in} and any two nodes between different modules are randomly connected with probability p_{out} (see Figure S7 for the performance of DGM with connection probability). We apply our method as well as various traditional methods to this modular network (N = 28) with random node labeling (Figure 2B). We find that no link prediction method performs significantly better than random guess for this directed network (Figure 2D). However, applying the Louvain method first (here we treat the directed network as an undirected one) will capture some structural patterns in the network (i.e., the community structure in the adjacency matrix, see Figure 2C), which will significantly improve the performance of our method (Figure 2D; paired-sample t test). This result suggests that any structural patterns should be exploited for our DGM-based link prediction.

Demonstration Using a Real Network

Real-world complex networks certainly display more prominent structural patterns than ER random graphs. Thanks to the deep neural networks in the DGM, our method can actually leverage structural patterns in a real network at different scales all together, from small subgraphs to community structure (Girvan and Newman, 2002). To demonstrate this, we consider the character co-occurrence network of Victor Hugo's *Les Misérables*. As shown in Figure 3A, this network displays many interesting structural patterns, e.g., stars, cliques, and communities. After node labeling using the Louvain method (Blondel *et al.*, 2008), those structural patterns naturally emerge in the matrix (image) presentation. In particular, those stars show up as line segments, cliques and communities appear as blocks in the image (Figure 3B). After training, deep neural networks with many layers are able to extract the most important structural patterns of the network as the key features of the corresponding image. Note that, at the same layer of the deep neural network, different filters can actually learn different feature representations: some focus more on lower level features such as line segments, whereas others focus more on higher level features such as blocks (Figure 3C). Deeper layers will typically capture higher level features or more global structural patterns (Figure 3D). Leveraging those features at different levels or structural patterns learned at different scales, our DGM-based link prediction performs very well (see Supplementary Information Section 4.2.2 and 4.2.3 for more details). Indeed, for this particular network, with a fraction f = 0.1 of links as test set, we have AUC$\sim$0.95, much higher than that of other link prediction methods, e.g., CN (with AUC$\sim$0.7) and SEAL (with AUC$\sim$0.85).

Systematic Benchmarking Using Real Networks

To systematically demonstrate the advantage of our DGM-based link prediction in real-world applications, we compare the performance of our method with that of both classical and state-of-the-art link prediction methods for a wide range of real-world networks (LeBlanc et al., 1975; Baird et al., 1998; Christian and Luczkovich, 1999; Krebs, 2002; Zhang *et al.*, 2005), from social, economic, technological to biological networks (see Supplementary Information Section 6 and Table S1 for brief descriptions of real networks analyzed in this work). For undirected networks (Figure 4A), we find that generally global similarity indices (e.g., Katz, ACT) and SBM-based link prediction methods perform better than local similarity indices (e.g., CN, PA, RA)-based methods. But the performances of those heuristics-based methods vary a lot over different network domains. Some of them actually perform even worse than random guess, especially when the training set is small (corresponding to large f). By contrast, our DGM-based method displays very robust and high performance for various undirected networks (see Figures S8 and S9 for comparing the performance with additional three methods). It also outperforms several state-of-the-art link prediction methods based on non-negative matrix factorization (Chen *et al.*, 2017), network embedding (Perozzi et al., 2014), and graph neural networks (Zhang and Chen, 2018). For directed networks (Figure 4B), most of the existing methods (especially those state-of-the-art methods) are actually not applicable, except two classical methods: PA and LRMC (see Figure S10 for comparing the performance with modified RA method). We compare the performance of our method with those of PA and LRMC. Again, we find that our method displays more robust and better performance than PA and LRMC for various directed networks (see Figure S11 for detailed statistical test). We also use AUPRC (area under the precision-recall curve) as the performance evaluation metrics (see Table S2, Figures S12 and S13).

We emphasize that, before applying our DGM-based link prediction to each of the real-world networks tested in Figure 4, we performed node labeling (also known as matrix reordering in the literature) to get the matrix (or image) presentation of the network. For the sake of simplicity, we just labeled nodes in a network based on its community structure. In particular, for undirected networks, we apply the Louvain method; for directed networks, we apply the method proposed in Arenas et al. (2008). We emphasize that our approach does not rely on the presence of communities in a network. Any structural patterns at different scales (from small subgraphs to communities, see Figure 3) can be and should be leveraged all together. A reasonable node labeling can actually be achieved in many different ways other than just community detection. We found that for real-world networks the performance of our DGM-based link

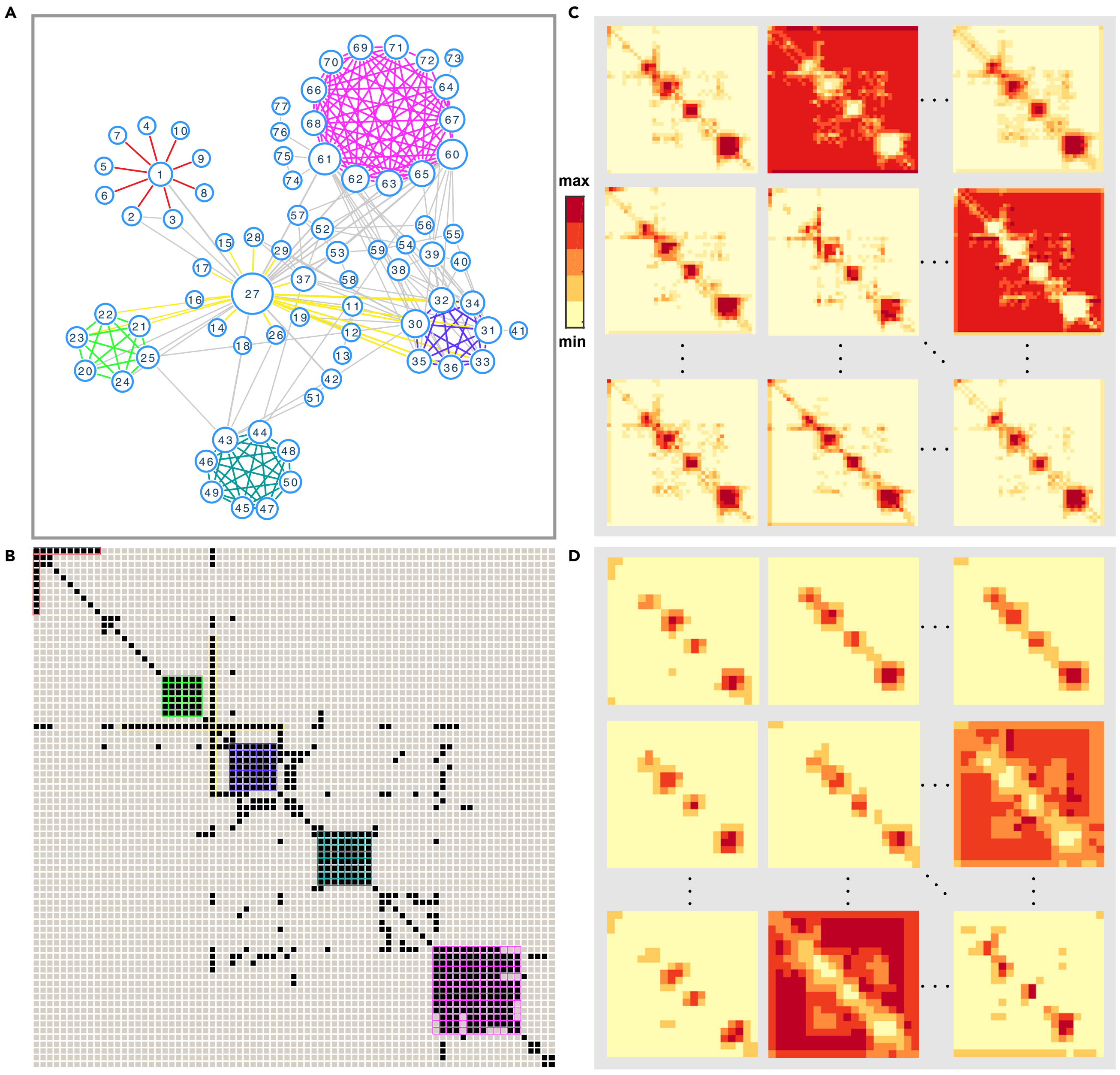


Figure 3. Deep Neural Networks in the DGM Are Able to Learn Different Structural Patterns of a Network at Different Scales

(A) The character co-occurrence network of Victor Hugo's *Les Misérables* (with 77 nodes) contains several interesting structural patterns such as stars, cliques, and community structure.

(B) The matrix (image) representation of the network, with node labeling based on the Louvain method. Those structural patterns are highlighted in different colors.

(C) Learned feature maps from the first convolutional layer with trained filters. There are in total 64 feature maps. Each of them is of size 40 × 40.

(D) Learned feature maps from the second convolutional layer with trained filters. There are in total 128 feature maps. Each of them is of size 20 × 20. For each feature map, higher (or lower) values are shown in redder (or yellower) color.

prediction actually does not heavily depend on the specific node labeling algorithm (see Supplementary Information Section 5). This is consistent with the results presented in Figure 2A, where we show that as long as the network has strong structural patterns, then any reasonable node labeling will offer a plausible matrix (or image) presentation of the network, which can be used for our DGM-based link prediction.

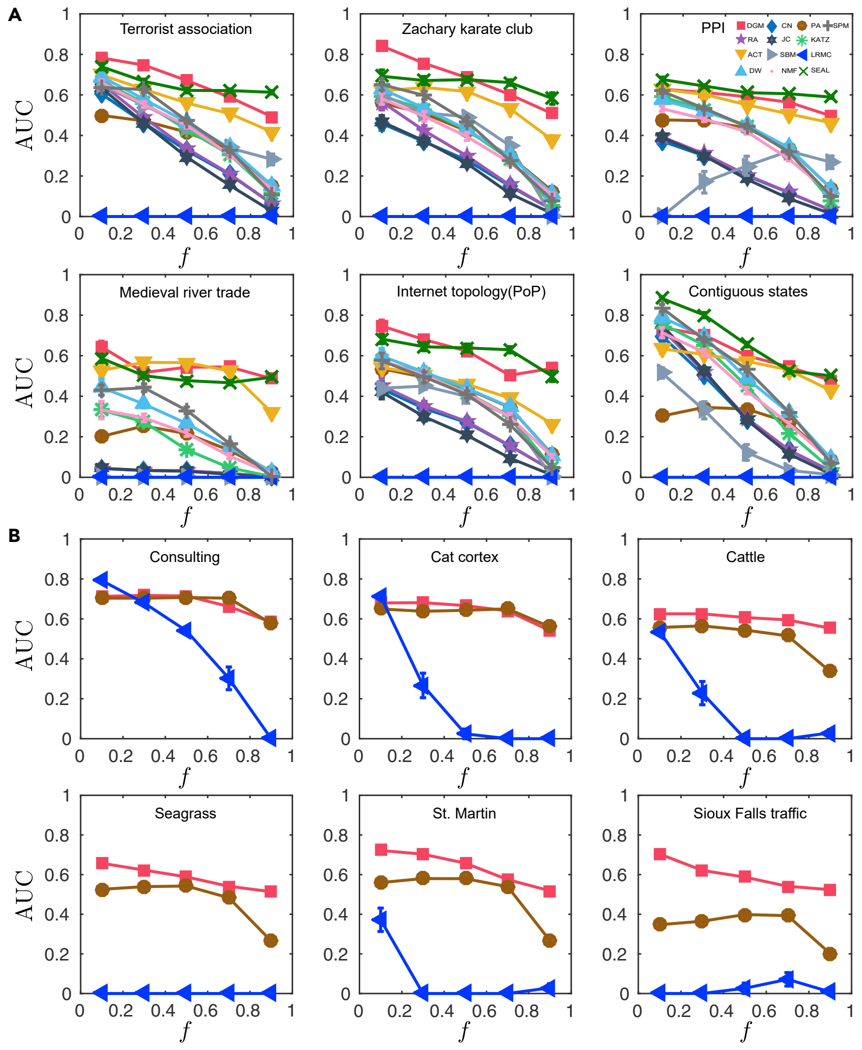


Figure 4. Our DGM-Based Link Prediction Displays Very Robust and High Performance for Both Undirected and Directed Real-World Networks

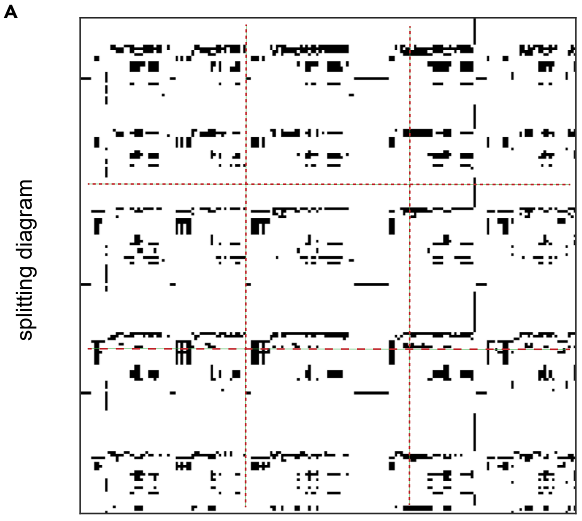
DGM, deep generative model based link prediction; CN, common neighbors; PA, preferential attachment; RA, resource allocation; JC, Jaccard index; KATZ, Katz index; ACT, average commute time; SBM, stochastic block model; LRMC, low rank matrix completion; DW, deep-walk embedding method; NMF, non-negative matrix factorization; SEAL, learning from Subgraphs, Embeddings, and Attributes for Link prediction; SPM, structural perturbation method (see Supplementary Information Section 1 for details of each algorithm).

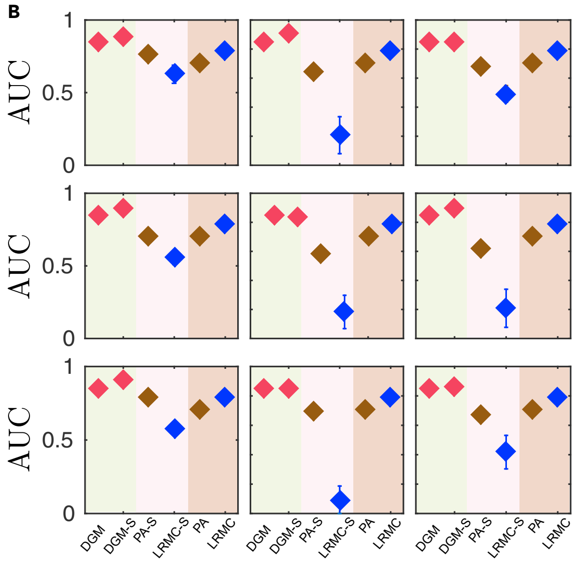
(A) Undirected networks. Top: Terrorist association network, Zachary karate club, Protein-protein interaction (PPI) network (a subnetwork of PPIs in *S. cerevisiae*). Bottom: Medieval river trade network in Russia, Internet topology (at the PoP level), Contiguous states in the United States.

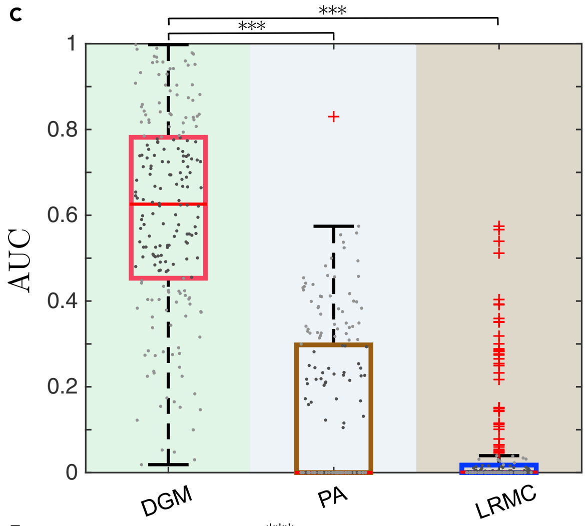
(B) Directed networks. Top: Consulting (a social network of a consulting company), cat cortex (the connection network of cat cortical areas), cattle (a network of dominance behaviors of cattle). Bottom: Seagrass food web, St. Martin food web, Sioux Falls traffic network. AUC of our DGM-based method is the average AUC over the last 20 epochs of the total 150 epochs for all of networks. Here an epoch is one full training cycle on the training set. For all the undirected real networks, we apply the Louvain method first to label the nodes appropriately. Directed networks are labeled by the method proposed in Arenas et al., (2008). Error bar represents the standard error of the mean.

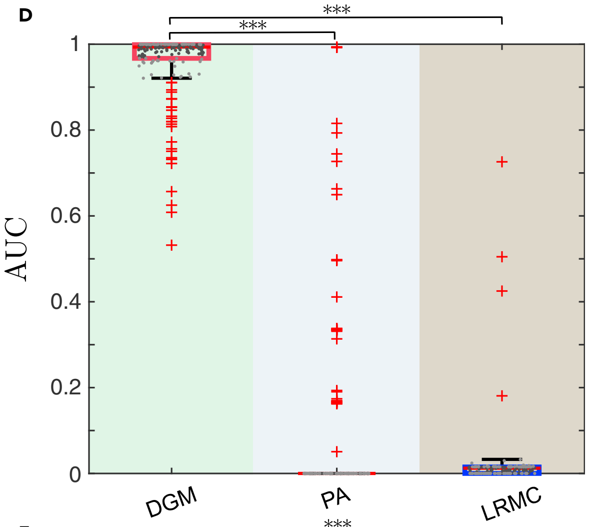
Parallelization Based on Image Splitting

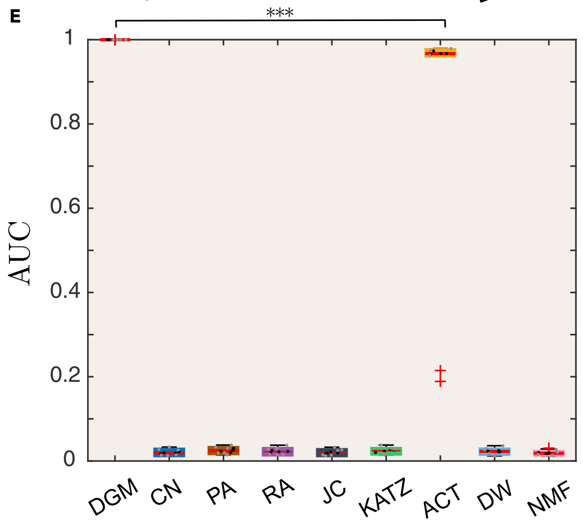
Since our method essentially treats a network as an image, it can be easily parallelized by splitting a large network (image) into several small subnetworks (subimages) and then performing link prediction for each subnetwork (subimage) in parallel (Figure 5A). Note that node labeling is the first step of our approach. In this step, we always treat the whole network as an image (by any reasonable node labeling algorithm),

A

splitting diagram

B

AUC
1
0.5
0
DGM
DGM-S
PA-S
LRMC-S
PA
LRMC

C

***
***
AUC
1
0.8
0.6
0.4
0.2
0
DGM
PA
LRMC

D

***
***
AUC
1
0.8
0.6
0.4
0.2
0
DGM
PA
LRMC

E

***
AUC
1
0.8
0.6
0.4
0.2
0
DGM
CN
PA
RA
JC
KATZ
ACT
DW
NMF

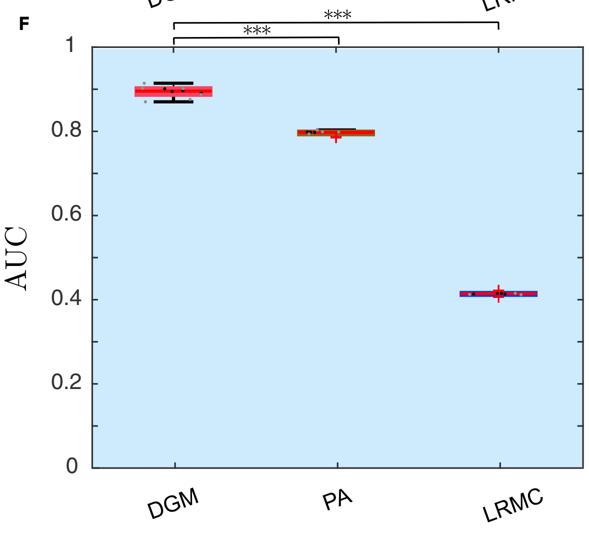
F

***
***
AUC
1
0.8
0.6
0.4
0.2
0
DGM
PA
LRMC

Figure 5. The DGM-Based Link Prediction Method Can Infer Missing Links of Large Networks and Arbitrarily Selected Subnetworks within Large Networks

(A) The adjacency matrix of a real network: The Little Rock food web. The network (image) is split into 9 subnetworks (subimages).

(B) AUC of DGM, PA, and LRMC on the original network and AUC of DGM, PA, and LRMC on each subnetwork (DGM-S, PA-S, LRMC-S). Error bar represents the standard error of the mean.

(C and D) We perform link prediction for 200 randomly selected subnetworks (of size 60) chosen from two large-scale real networks: (E) Facebook wall posts (with 46,952 nodes and 87,6993 links) and (F) Google+ (with 23,628 nodes and 39,242 links), respectively. We randomly divide the links of the relabeled networks into two parts: a fraction of 10% links are chosen as probe set and the remaining 90% fraction of links as training set (here, each subnetwork contains 15 links at least). Asterisks at the top of each panel shows whether the AUC of our DGM-based link prediction model is significantly higher than that of the other two traditional algorithms (paired-sample t test). Significance levels: p value <0.001(***).

(E and F) AUC of DGM and other scalable link prediction methods in two large-scale networks: (E) Facebook-NIPS (with $N = 2,888$ nodes and 2,981 links), and (F) US airports (with $N = 1,574$ nodes and 28,236 links).

regardless of the network size. Training the DGM is the second step of our approach. In this step, if the network/image is small, we train the DGM and hence perform link prediction for the whole image. Only if the network/image is too large, for which the DGM cannot be easily trained, we need to split the image into subimages, train DGM, and perform link prediction for different subimages in parallel. This splitting typically does not decrease the overall link prediction performance, compared with the result of treating the large network as a whole (Figure 5B). For each subnetwork, when only the information of the subnetwork is provided, our method outperforms other methods (Figure 5B). In fact, even if other methods (e.g., PA and LRMC) use the information of the whole network to perform link prediction for a subnetwork, our method that only relies on the information of the subnetwork still displays better performance (Figure 5B).

The image representation of complex networks also allows us to focus on any specific subnetwork of interest and just predict the missing links in that subnetwork. For example, we perform link prediction for 200 subnetworks of size 60 randomly selected from two large real networks: Facebook wall posts (Viswanath et al., 2009) (with 46,952 nodes and 87,6993 links) and Google+ (Leskovec and Mcauley, 2012) (with 23,628 nodes and 39,242 links). To get each subnetwork, we randomly select a subimage of size 60 × 60 from the whole image (i.e., the adjacency matrix $A=(a_{ij})$ of the network) by choosing rows (i+1) to (i+60) and columns (j+1) to (j+60). Then we convert this subimage into a small network with adjacency matrix $A^s=(a_{pq}{}^s)$, where $a_{pq}{}^s = 1$ if $a_{i+p-1,j+q-1} = 1$. We find that our method shows much higher AUC than other methods (Figures 5C and 5D). All these results suggest that our method holds great promise in link prediction for large-scale real-world networks. To directly demonstrate the performance of our method in analyzing large-scale networks, we consider an undirected network: Facebook-NIPS (with $N = 2,888$ nodes), and a directed network: US airports (with $N = 1,574$ nodes). We randomly remove a fraction ($f = 0.1$) of links as test set. To facilitate the training process of DGM and speed up the link prediction, we still split each large network (image) into several small subnetworks (subimages) and then perform link prediction for each subnetwork. But, in the end, to have a fair comparison with other methods (that always treat the large network as a whole), we calculate the AUC of our method from the whole network (constructed by merging subnetworks/subimages generated by the DGM). We find that clearly for both large-scale real networks (Facebook-NIPS and US airports) our method outperforms other existing methods (Figures 5E and 5F). See Supplementary Information Figure S14 for results of large-scale model networks.

DISCUSSION

In summary, our DGM-based link prediction shows superior performance against existing methods for various types of networks, be they technological, biological, or social in nature. Since our method treats the adjacency matrix of a network as an image, it can be naturally extended to solve the link prediction problem for bipartite graphs, multi-layer networks, and multiplex networks, where the adjacency matrices have certain inherent structure. With small modification, it can also be used to perform link prediction in weighted graphs (see Supplementary Information Figure S15). To achieve that, we need to normalize the link weights so that they can be treated as existent probabilities of the corresponding links. In principle, any DGM can be utilized in our method. But we find that, for the link prediction purpose, GANs perform much better than other DGMs, e.g., variational autoencoder (Sohn et al., 2015) (see Supplementary Information Figure S16). There are several hyperparameters in training the GANs (see Supplementary Information Section 4 for details). In this work, we use the same set of hyperparameters for all the networks to show a conservative AUC estimation of our method. The performance of our method can certainly be further improved by carefully tuning those hyperparameters for a specific network of interest (see Figure S17). We feel this is beyond the scope of the current work and hence leave it as a future one.

Limitations of the Study

We should admit that, although our DGM-based link prediction displays superior performance in various real-world networks, it has several limitations. *First*, its time complexity is higher than traditional heuristic-based methods (e.g., Common Neighbors [Zhou et al., 2010], Preferential Attachment [Barabási and Albert, 1999]) and embedding-based methods (e.g., DeepWalk [Perozzi et al., 2014], node2vec [Grover and Leskovec, 2016, p. 2]). (See Supplementary Information Section 4.2.4 for detailed analysis of its time complexity.) Such a *speed-accuracy trade-off* deserves a very careful consideration in real-world applications. For certain link prediction applications, such as recommendation system in e-commerce or online social media with daunting network sizes, speed is the major concern, hence traditional link prediction methods still have big advantages. For applications in biomedicine (e.g., inferring protein-protein interactions or drug-target interactions) or criminal intelligence analysis (e.g., identifying hidden accomplice in criminal activity), those networks are much smaller than social media networks, and accuracy is way more important than speed. In those cases, we anticipate that our DGM-based link prediction should have an unparalleled advantage. Furthermore, we suggest that one should definitely exploit graphics process unit parallelism to train the GANs (Im et al., 2016), which will certainly speed up our method. Finally, we emphasize that, in real-world applications of link prediction, any additional side information, such as node attributes, can be incorporated into our method to further improve the link prediction. *Second*, we should admit that, since we treat the adjacency matrix of a network as a binary image, our method is by definition not permutation invariant. Recently, the notion of permutation-invariance has been discussed a lot in the deep learning literature (Wood and Shawe-Taylor, 1996; Kondor and Trivedi, 2018; Maron et al., 2018, 2019; Bloem-Reddy and Teh, 2019; Behrisch et al., 2016). A network method is called permutation-invariant if it produces the same output regardless of the node labeling used to encode the adjacency matrix of the network. Although our method is not permutation invariant in theory, we emphasize that it is approximately permutation invariant in practice. As shown in Figure S18, node labeling does not significantly affect the performance of our link prediction method, as long as the node labeling method leverages existing structure features in the network.

Resource Availability

Lead Contact

Yang-Yu Liu (yyl@channing.harvard.edu).

Materials Availability

This study did not generate new unique reagents.

Data and Code Availability

The code is available on the cloud-based reproducibility platform: Code Ocean (https://codeocean.com/), with the compute capsule entitled "*Link Prediction through Deep Generative Model*" (https://codeocean.com/capsule/6854770/tree/v1).

METHODS

All methods can be found in the accompanying Transparent Methods supplemental file.

SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.isci.2020.101626.

ACKNOWLEDGMENTS

We thank Yizhou Sun, Tina Eliassi-Rad, Pan Zhang, Changjun Fan, Nima Dehmamy, Santo Fortunato, Yong-Yeol Ahn, Huawei Shen, and Marco Tulio Angulo for valuable discussions. Research reported in this publication was supported by grants R01AI141529, R01HD093761, UH3OD023268, U19AI095219, and U01HL089856 from National Institutes of Health.

AUTHOR CONTRIBUTIONS

Y.-Y.L. conceived and designed the project. X.-W.W. and Y.C. did the analytical and numerical calculations. X.-W.W. analyzed all the real networks. All authors analyzed the results. Y.-Y.L. and X.-W.W. wrote the manuscript. Y.C. edited the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing financial interests. Correspondence and requests for materials should be addressed to Y.-Y.L. (yyl@channing.harvard.edu).

Received: May 5, 2020
Revised: June 19, 2020
Accepted: September 24, 2020
Published: October 23, 2020

REFERENCES

Albert, R., and Barabási, A.-L. (2002). Statistical mechanics of complex networks. Rev. Mod. Phys. *74*, 47.

Arenas, A., Fernández, A., and Gómez, S. (2008). Analysis of the structure of complex networks at different resolution levels. New J. Phys. *10*, 053039.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. arXiv. http://arxiv.org/abs/1701.07875.

Baird, D., Luczkovich, J., and Christian, R.R. (1998). Assessment of Spatial and temporal variability in ecosystem Attributes of the St Marks National Wildlife Refuge, Apalachee Bay, Florida. Estuar. Coast. Shelf Sci. *47*, 329–349.

Barabási, A.-L., and Albert, R. (1999). Emergence of scaling in random networks. Science *286*, 509–512.

Behrisch, M., Bach, B., Henry Riche, N., Schreck, T., and Fekete, J.-D. (2016). Matrix reordering methods for table and network visualization. Comput. Graph. Forum *35*, 693–716.

van den Berg, R., Kipf, T.N., and Welling, M. (2017). Graph convolutional matrix completion. arXiv. arXiv:1706.02263.

Berlusconi, G., Calderoni, F., Parolini, N., Verani, M., and Piccardi, C. (2016). Link prediction in criminal networks: a tool for criminal intelligence analysis. PLoS One *11*, e0154244.

Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. (2000). Image inpainting. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (ACM Press/Addison-Wesley Publishing Co.), pp. 417–424.

Blagus, N., Šubelj, L., and Bajec, M. (2012). Self-similar scaling of density in complex real-world networks. Physica A Stat. Mech. Appl. *391*, 2794–2802.

Bloem-Reddy, B., and Teh, Y.W. (2019). Probabilistic symmetry and invariant neural networks. arXiv. arXiv:1901.06082.

Blondel, V.D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. J. Stat. Mech. *2008*, P10008.

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D. (2006). Complex networks: structure and dynamics. Phys. Rep. *424*, 175–308.

Campillos, M., Kuhn, M., Gavin, A.C., Jensen, L.J., and Bork, P. (2008). Drug target identification using side-effect similarity. Science *321*, 263–266.

Chaney, A.J., Blei, D.M., and Eliassi-Rad, T. (2015). A probabilistic model for using social networks in personalized item recommendation. In Proceedings of the 9th ACM Conference on Recommender Systems (ACM), pp. 43–50.

Chen, B., Li, F., Chen, S., Hu, R., and Chen, L. (2017). Link prediction based on non-negative matrix factorization. PLoS One *12*, e0182968.

Chiu, C., and Zhan, J. (2018). Deep learning for link prediction in dynamic networks using weak estimators. IEEE Access *6*, 35937–35945.

Christian, R.R., and Luczkovich, J.J. (1999). ‘Organizing and understanding a winter’s seagrass foodweb network through effective trophic levels’. Ecol. Model. *117*, 99–124.

Clauset, A., Moore, C., and Newman, M.E. (2008). Hierarchical structure and the prediction of missing links in networks. Nature *453*, 98–101.

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. (1999). Learning probabilistic relational models. In International Joint Conferences on Artificial Intelligence (Sweden: Stockholm), pp. 1300–1309.

Girvan, M., and Newman, M.E. (2002). Community structure in social and biological networks. Proc. Natl. Acad. Sci. U S A *99*, 7821–7826.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Advances in Neural Information Processing Systems, Z. Ghahramani, M. Welling, and C. Cortes, eds., pp. 2672–2680.

Grover, A., and Leskovec, J. (2016). node2vec: scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’16. The 22nd ACM SIGKDD International Conference, San Francisco, California, USA (ACM Press), pp. 855–864.

Guimerà, R., and Sales-Pardo, M. (2009). Missing and spurious interactions and the reconstruction of complex networks. Proc. Natl. Acad. Sci. U S A *106*, 22073–22078.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems, *30*, I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds. (Curran Associates, Inc.), pp. 1024–1034.

Han, J.D., Dupuy, D., Bertin, N., Cusick, M.E., and Vidal, M. (2005). Effect of sampling on topology predictions of protein-protein interaction networks. Nat. Biotechnol. *23*, 839–844.

Heckerman, D., Meek, C., and Koller, D. (2007). ‘Probabilistic Entity-Relationship Models, PRMs, and Plate Models’, Introduction To Statistical Relational Learning (MIT Press), pp. 201–238.

Hulovatyy, Y., Solava, R.W., and Milenković, T. (2014). Revealing missing parts of the interactome via link prediction. PLoS One *9*, e90073.

Im, D.J., Ma, H., Kim, C.D., and Taylor, G. (2016). Generative adversarial parallelization. arXiv. arXiv:1612.04021.

Katz, L. (1953). A new status index derived from sociometric analysis. Psychometrika *18*, 39–43.

Kondor, R., and Trivedi, S. (2018). On the generalization of equivariance and convolution in neural networks to the action of compact groups. arXiv. arXiv:1802.03690.

Kovács, I.A., Luck, K., Spirohn, K., Wang, Y., Pollis, C., Schlabach, S., Bian, W., Kim, D.K., Kishore, N., Hao, T., et al. (2019). Network-based prediction of protein interactions. Nat. Commun. *10*, 1–8.

Krebs, V.E. (2002). Mapping networks of terrorist cells. Connections *24*, 43–52.

LeBlanc, L.J., Morlok, E.K., and Pierskalla, W.P. (1975). An efficient approach to solving the road network equilibrium traffic assignment problem. Transport. Res. *9*, 309–318.

Leskovec, J., and Mcauley, J.J. (2012). Learning to discover social circles in ego networks. In Advances in Neural Information Processing Systems, pp. 539–547.

Liben-Nowell, D., and Kleinberg, J. (2007). ‘The link-prediction problem for social networks’. J. Am. Soc. Inf. Sci. Technol. *58*, 1019–1031.

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. *7*, 76–80.

Lü, L., and Zhou, T. (2011). Link prediction in complex networks: a survey. Physica A Stat. Mech. Appl. *390*, 1150–1170.

Luo, Y., Zhao, X., Zhou, J., Yang, J., Zhang, Y., Kuang, W., Peng, J., Chen, L., and Zeng, J. (2017). A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. Nat. Commun. *8*, 573.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. (2018). Invariant and equivariant graph networks. arXiv. arXiv:1812.09902.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. (2019). On the universality of invariant networks. arXiv. arXiv:1901.09342.

Martínez, V., Berzal, F., and Cubero, J.-C. (2016a). A survey of link prediction in complex networks. ACM Comput. Surv. *49*, 69.

Martínez, V., Berzal, F., and Cubero, J.-C. (2016b). A survey of link prediction in complex networks. ACM Comput. Surv. *49*, 1–33.

Murphy, R.L., Srinivasan, B., Rao, V., and Ribeiro, B. (2019). Relational pooling for graph representations. arXiv. arXiv:1903.02541.

Newman, M.E.J. (2003). The structure and function of complex networks. SIAM Rev. *45*, 167–256.

Newman, M.E., and Girvan, M. (2004). Finding and evaluating community structure in networks. Phys. Rev. E Stat. Nonlin. Soft Matter Phys. *69*, 026113.

Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning Convolutional Neural Networks for Graphs. International conference on machine learning. 2014–2023.

Pan, L., Zhou, T., Lü, L., and Hu, C.K. (2016). Predicting missing links and identifying spurious links via likelihood analysis. Sci. Rep. *6*, 22955–23010.

Pech, R., Hao, D., Pan, L., Cheng, H., and Zhou, T. (2017). Link prediction via matrix completion. EPL *117*, 38002.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). DeepWalk: online learning of social representations. arXiv, 701–710, https://doi.org/10.1145/2623330.2623732.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. Proc. Natl. Acad. Sci. U S A *101*, 2658–2663.

Sahni, N., Yi, S., Taipale, M., Fuxman Bass, J.I., Coulombe-Huntington, J., Yang, F., Peng, J., Weile, J., Karras, G.I., Wang, Y., et al. (2015). Widespread macromolecular interaction perturbations in human genetic disorders. Cell *161*, 647–660.

Sarukkai, R.R. (2000). Link prediction and path analysis using Markov chains1. Comput. Netw. *33*, 377–386.

Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2017). Modeling Relational Data with Graph Convolutional Networks. arXiv. http://arxiv.org/abs/1703.06103.

Scholtes, I., Wider, N., Pfitzner, R., Garas, A., Tessone, C.J., and Schweitzer, F. (2014). Causality-driven slow-down and speed-up of diffusion in non-Markovian temporal networks. Nat. Commun. *5*, 5024.

Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. In Advances in Neural Information Processing Systems, C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, eds., pp. 3483–3491.

Srinivasan, B., and Ribeiro, B. (2019). On the equivalence between node embeddings and structural graph representations. arXiv. arXiv:1910.00452.

Tavakoli, S., Hajibagheri, A., and Sukthankar, G. (2017). Learning social graph topologies using generative adversarial neural networks. In International Conference on Social Computing, Behavioral-Cultural Modeling & Prediction.

Viswanath, B., Mislove, A., Cha, M., and Gummadi, K.P. (2009). On the evolution of user interaction in facebook. In Proceedings of the 2nd ACM Workshop on Online Social Networks (ACM), pp. 37–42.

Von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S.G., Fields, S., and Bork, P. (2002). 'Comparative assessment of large-scale data sets of protein–protein interactions'. Nature *417*, 399–403.

Wood, J., and Shawe-Taylor, J. (1996). A unifying framework for invariant pattern recognition. Pattern Recognit. Lett. *17*, 1415–1422.

Zhang, B., Liu, R., Massey, D., and Zhang, L. (2005). Collecting the Internet AS-level topology. ACM SIGCOMM Comput. Commun. Rev. *35*, 53–61.

Yeh, R.A., Chen, C., Yian Lim, T., Schwing, A.G., Hasegawa-Johnson, M., and Do, M.N. (2017). Semantic image inpainting with deep generative models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5485–5493.

Zhang, M., and Chen, Y. (2018). Link Prediction Based on Graph Neural Networks. arXiv. arXiv:1802.09691.

Zhou, T., Kuscsik, Z., Liu, J.G., Medo, M., Wakeling, J.R., and Zhang, Y.C. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. Proc. Natl. Acad. Sci.U S A *107*, 4511–4515.

iScience, Volume *23*

Supplemental Information

Link Prediction through Deep Generative Model

Xu-Wen Wang, Yize Chen, and Yang-Yu Liu

CONTENTS

Part I

Transparent Methods

I. EXISTING LINK PREDICTION ALGORITHMS

A real-world network can be mathematically represented by a graph $G(\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = \{1, 2, \ldots, N\}$ is the node set and $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$ is the link set. A link is a node pair (i, j) with $i, j \in \mathscr{V}$, representing certain interaction, association or physical connection between nodes i and j. Link prediction aims to infer the missing links or predict future links between currently unconnected nodes based on the observed links Barzel and Barabási (2013); Liben-Nowell and Kleinberg (2007); Lichtenwalter et al. (2010). Many algorithms have been developed to solve the link prediction problem Al Hasan et al. (2006); Liu and Lü (2010); Lü et al. (2009); Sarukkai (2000). Here we briefly describe some classical link prediction algorithms.

A. Similarity-based algorithms

For similarity-based algorithms, each non-observed node pair (i, j) is assigned a similarity score s_{ij}. Higher score is assumed to represent higher link existence probability. The similarity score can be defined in many different ways.

(1) Common Neighbors. The common neighbors algorithm quantifies the overlap or similarity of two nodes as follows Zhou et al. (2010):

$$s_{ij} = |\Gamma(i) \cap \Gamma(j)|, \tag{S1}$$

where $\Gamma(i)$ denotes the set of neighbors of node i, $\cap$ denotes the intersection of two sets and $|X|$ denotes the cardinality or size of set X.
(2) Jaccard Index. The Jaccard index measures the overlap of two nodes with normalization Jaccard (1901):

$$s_{ij} = \frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|}, \tag{S2}$$

where $\cup$ denotes the union of two sets.
(3) Preferential Attachment Index. This index assumes that the existent likelihood of a link between two nodes is proportional to the product of their degrees Barabási and Albert (1999):

$$s_{ij} = k_i \times k_j, \tag{S3}$$

where k_i is the degree of node i.
(4) Resource Allocation Index. This index is based on a resource allocation process between pair of nodes Ou et al. (2007); Zhou et al. (2009). Consider a node pair (i, j), the similarity between i and j is defined as the amount of resource j received from i through their common neighbors:

$$s_{ij} = \sum_{m \in \Gamma(i) \cap \Gamma(j)} \frac{1}{k_m}. \tag{S4}$$

Here we assume each common neighbor has a unit of resource and will equally distribute among all its neighbors.
(5) Katz Index. The Katz index is based on a weighted sum over the collection of all paths connecting nodes i and j:

$$s_{ij} = \sum_{l=1}^{\infty} \beta^l (A^l)_{ij}, \tag{S5}$$

where β is a damping factor that gives the shorter paths more weights, and A is the adjacency matrix of the network. The $N \times N$ similarity matrix $S = (s_{ij})$ can be written in a compact form Katz (1953):

$$S = (I - \beta A)^{-1} - I, \tag{S6}$$

where I is the identity matrix. The damping factor β is a free parameter and should be lower than the reciprocal of the absolute value of the largest eigenvalue $|\lambda_{\text{max}}|$ of A (in our calculations, we choose $\beta = 0.5/|\lambda_{\text{max}}|$).

(6) Average Commute Time. The average commute time index is motivated by a random walk process on the network Liu and Lü (2010):

$$s_{ij} = \frac{1}{l^+_{ii} + l^+_{jj} - 2l^+_{ij}}, \tag{S7}$$

where l^+_{ij} is the entry of pseudoinverse of the Laplacian matrix $L \equiv D - A$, where $D = \text{diag}\{k_1, k_2, \cdots, k_N\}$ is the degree matrix and A is the adjacency matrix.

(7) Node embedding. Node embedding aims to represent each node $i \in \mathscr{V}$ into a low-dimensional space $\mathbb{R}^d$ by using the proximity between nodes Roweis and Saul (2000). After embedding the nodes in the network into a low-dimensional space, we can directly calculate the distance between any two nodes in the transformed space, which can then be used in many downstream analysis, such as multi-label classification, networks reconstruction as well as link prediction. There are many existing node embedding methods, such as Structural Deep Networks Embedding (SDNE) Wang et al. (2016), LINE Tang et al. (2015), DeepWalk Perozzi et al. (2014), GraRep Cao et al. (2015), and LE Belkin and Niyogi (2003). Though node embedding methods can efficiently perform link prediction for large-scale networks, the embedding process itself will cause information loss, which might affect the performance of link prediction. In addition, for sparse network, embedding methods cannot provide the representations of isolate nodes since no attribute is available. We choose DeepWalk to compare with our method and the code is downloaded from github: https://github.com/phanein/deepwalk.

(8) Non-negative matrix factorization. Suppose the adjacency matrix A of a network is also non-negative where each node is represented by the corresponding column. Then non-negative matrix factorization (NMF) is aiming to find two non-negative matrices $U_{N\times k}$ and $V_{k\times N}$ so that Duan et al. (2017); Long et al. (2005):

$$A = UV, \tag{S8}$$

where N is the network size and $k < N$ is the dimension of latent space. Then, similar to the network embedding methods, we can calculate the distance (similarity) between any two nodes in the latent space and realize the task of link prediction Chen et al. (2017).

(9) Structural perturbation method. Given an undirected network, its adjacency matrix A can be written as $A = A^{\mathrm{R}} + \Delta A$, where ΔA and A^{R} represent the symmetric perturbation matrix and the residual matrix after randomly removing E^{R} links, respectively. A^{R} can be diagonalized as: $A^{\mathrm{R}} = \sum_{k=1}^{N} \lambda_k \mathbf{x}_k \mathbf{x}_k^{\top}$, where λ_k and $\mathbf{x}_k$ are the eigenvalue and the corresponding eigenvector of A^{R}, respectively. After perturbation, the eigenvalue change can be approximated as $\Delta\lambda_k \approx \frac{\mathbf{x}_k^{\top}\Delta A \mathbf{x}_k}{\mathbf{x}_k^{\top}\mathbf{x}_k}$. Then, the perturbed matrix can be expressed as: $\tilde{A} = \sum_{k=1}^{N}(\lambda_k + \Delta\lambda_k)\mathbf{x}_k\mathbf{x}_k^{\top}$. The entry of $\tilde{A}$ can be used to quantify the similarity of any node pair. Note that the random removal of links in SPM is similar to the removal of pixels in the binary images to get a pool of perturbed images in our method.

Other similarity-based algorithms, such as SimRank Jeh and Widom (2002), Random Walks Pearson (1905), Random Walks with Restart Tong et al. (2006), Negated Shortest Path Liben-Nowell (2005), Network Paths of Length Three (L3) Kovács et al. (2019) can also be used for link prediction (see Fig. S8 for the performance of L3 in the link prediction of 6 undirected networks). Note that some similarity-based methods can be modified to be applicable for directed networks. For example, a modified Resource Allocation (RA) index Chen et al. (2018); Zhou et al. (2009):

$$s_{ij}^{\text{RA}} = \sum_{z\in\Gamma(i)\cap\Gamma(j)} \frac{1}{k_z^{\text{in}} \cdot k_z^{\text{out}}} \tag{S9}$$

can be used for the link prediction of directed networks. Here, z represents the common neighbors between nodes i and j, and k_z^{in} (k_z^{out}) is the in- (out-) degree of node z. We compared the performance of this RA-based method with that of our DGM-based link prediction in 6 real-world directed networks. We found that our method significantly outperforms the RA-based method in 4 of the 6 networks (p-value $< 10^{-5}$, paired t-test, see Fig. S10).

B. Maximum likelihood algorithms

The maximum likelihood algorithms assume that real networks have some structure, i.e., hierarchical or community structure. The goal of these algorithms is to select model parameters that can maximize the likelihood of the observed structure.

(10) Stochastic Block Model. As one of the most general network models, the stochastic block model (SBM) assumes that nodes are partitioned into groups and the probability of two nodes are connected depends solely on the groups they belong to Holland et al. (1983); White et al. (1976). The SBM assumes that a link with higher reliability has higher existent probability, and the reliability of a link is defined as Guimerà and Sales-Pardo (2009):

$$R_{ij}^{L} = \frac{1}{Z} \sum_{P \in \mathscr{P}} \left(\frac{l_{\sigma_i \sigma_j}^{O} + 1}{r_{\sigma_i \sigma_j} + 2} \right) \exp[-\mathscr{H}(P)], \tag{S10}$$

where $\mathscr{P}$ represents the partition space of all possible partitions, σ_i is the group that node i belongs to the partition P, $l_{\sigma_i\sigma_j}^{O}$ is the number of links between groups σ_i and σ_j in the observed network, $r_{\sigma_i\sigma_j}$ is the maximum possible number of links between them, the function $\mathscr{H}(P) \equiv \sum_{\alpha \leq \beta} [\ln(r_{\alpha\beta}+1) + \ln \begin{pmatrix} r_{\alpha\beta} \\ l_{\alpha\beta}^{O} \end{pmatrix}]$, and $Z \equiv \sum_{P \in \mathscr{P}} \exp[-\mathscr{H}(P)]$. In practice, we can use the Metropolis algorithms to sample relevant partitions that significantly contribute to the sum over the partition space $\mathscr{P}$. This allows us to calculate the link reliability efficiently.

(11) Hierarchical Structure Model. Many real networks have hierarchical structure, which can be represented by a dendrogram D. One can assign a probability p_r to each internal node r of D. Then the connecting probability of a pair of leaves is given by $p_{r'}$, where r' is the lowest common ancestor of these two leaves. Denote E_r as the number of edges in the network whose endpoints have r as their lowest common ancestor in the dendrogram D. Let L_r and R_r be the number of leaves in the left and right subtrees rooted at r, respectively. Then the likelihood of D associated with a set of probabilities $\{p_r\}$ is given by Clauset et al. (2008):

$$\mathscr{L}(D, \{p_r\}) = \prod_{r \in D} p_r^{E_r} (1 - p_r)^{L_r R_r - E_r}. \tag{S11}$$

For a specific D, the probabilities $\{\overline{p}_r\}$ that maximize $\mathscr{L}(D, \{p_r\})$ are edges between the two subtrees of r that are present in the network:

$$\{\overline{p}_r\} = \frac{E_r}{L_r R_r}. \tag{S12}$$

Evaluating the likelihood $\mathscr{L}(D, \{p_r\})$ at this maximum yields

$$\mathscr{L}(D) = \prod_{r \in D} \left[\bar{p}_r^{\bar{p}_r} (1 - \bar{p}_r)^{1-\bar{p}_r} \right]^{L_r R_r}. \tag{S13}$$

One can use the Markov chain Monte Carlo (MCMC) method to sample a large number of dendrorgams D with probability proportional to their likelihood $\mathscr{L}(D)$. For each pair of unconnected leaves i and j, we calculate the connection probability p_{ij} for each D, and then calculate the average $\langle p_{ij} \rangle$ over all the sampled dendrograms. The $\langle p_{ij} \rangle$ value yields the existent probability of the link between nodes i and j. For each nonexistent link or node pair (i, j), we calculate the average connecting probability $\langle p_{ij} \rangle$ over all sampled dendrograms and the node pairs with highest $\langle p_{ij} \rangle$ are missing links.

Since the SBM-based link prediction method introduced in Ref Guimerà and Sales-Pardo (2009) has demonstrated better performance than this hierarchical structure model, in this work we chose the former as a representative maximum likelihood algorithm to compare with our DGM-based link prediction method.

C. Other algorithms

a. **(12) Low-Rank Matrix Completion**. The goal of matrix completion is to recover a low-rank matrix L from a large matrix A, which can be used to infer the missing links of a network. The matrix L can be calculated by solving the convex optimization problem Pech et al. (2017):

$$\min_{L,S} \|L\|_* + \lambda |S|_1, \text{s.t. } A = L + S. \tag{S14}$$

Here, S is a sparse matrix. $\|\cdot\|$ denotes the nuclear norm of a matrix, and $|\cdot|$ represents the sum of the absolute values of each matrix element and λ is a positive parameter. The matrix S is defined as follows: if two nodes (i, j) are connected in the observed network, then the score $s_{ij} = A_{ij}$; otherwise, $s_{ij} = L_{ij}$.

b. **(13) Graph neural networks**. Recently, the notion of graph neural networks (GNNs) has gained increasing popularity in graph representation learning. GNNs are a type of neural networks that operate directly on the graph structure. Here, we introduce three state-of-the-art GNN-based methods: GraphSAGE (SAmple and aggreGatE) Hamilton et al. (n.d.), SEAL (learning from Subgraphs, Embeddings, and Attributes for Link prediction) Zhang and Chen (2018), and CGNN (Colliding Graph Neural Networks) Srinivasan and Ribeiro (n.d.).

GraphSAGE is a popular graph representation learning method that uses node feature information to efficiently generate node embeddings for previously unseen data. Unlike most of embedding methods that require all nodes in the graph are present, GraphSAGE trains individual embeddings for each node through learning a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. The GraphSAGE code can be downloaded from github: https://github.com/williamleif/GraphSAGE.

SEAL is a representative GCN-based link prediction method, which combines graph structure features, latent features, and explicit features into a single GCN. In particular, the input to GCN is a local subgraph around each target link. Those local subgraphs capture graph structure feature related to link existence. The latent and explicit features can be naturally combined in GCN by concatenating node embedding and node attributes in the node information matrix for each subgraph. The SEAL code can be downloaded from github: https://github.com/muhanzhang/SEAL.

CGNN uses a novel variational auto-encoder procedure to obtain node embeddings using neural networks. The observed variable A, evidence feature X and the latent variable Z are connected by the joint distribution representing the probability of A and X with given Z. The neural network is used to learn the joint probability via MCMC (Markov chain Monte Carlo) through supervised learning. The CGNN code can be downloaded from github:
https://github.com/PurdueMINDS/Equivalence.

Note that among those three methods, SEAL was designed to solve the link prediction problem, while GraphSAGE and CGNN are general graph representation learning methods. In the main text (Fig.4), we compared our DGM-based link prediction with SEAL, finding that our method outperforms SEAL in almost all the real-world networks analyzed in this work. In Fig. S9, we compared our method with GraphSAGE and CGNN, finding that our method significantly outperforms those two method in all the real-world networks analyzed in this work.

There are also many probabilistic model based link prediction methods (which typically have high time complexity Martínez et al. (2017)), such as Probabilistic Relational Model Friedman et al. (1999), Probabilistic Entity Relationship Model Heckerman et al. (2007), and Stochastic Relational Model Yu et al. (2007), as well as many other methods, such as Structural Perturbation Method Lü et al. (2015), etc. For a more complete list of existing link prediction methods, please see the review articles Hulovatyy et al. (2014); Martínez et al. (2017). In this work, we compare our DGM-based link prediction method with representative link prediction methods that have either relatively higher performance or lower time complexity. Among the methods mentioned above, PA, RA, LRMC and our DMG-base link prediction methods can be used in directed networks.

II. PERFORMANCE METRICS OF LINK PREDICTION METHODS

The two most commonly used metrics to quantify the performance of link prediction methods are AUC and Precision. To calculate those two metrics, we first split the total link set $\mathscr{E}$ into two parts: (i) a fraction of links as the test or probe set $\mathscr{E}_{\mathrm{P}}$, which will be removed from the network; and (ii) the remaining fraction of links as the training set $\mathscr{E}_{\mathrm{T}}$, which will be used to recover the removed links.

If we shuffle all the existing links and randomly select an f fraction of links from the total link set $\mathscr{E}$ as the probe set, then those links originating from hubs (i.e., high-degree nodes) will be more likely to be selected (especially for networks with strong degree heterogeneity), while those links originating from low-degree nodes will be less likely to be selected. To avoid any bias introduced by this edge-based splitting method, we employ a node-based splitting method. See Algorithm 1 for details and Fig. S3 for a demonstration of its fairness.

Algorithm 1 Node-based splitting method

```
Input:
    The network, defined by $\mathscr{G}(\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ denotes the node set and $\mathscr{E}$ denotes the link set.
Output:
    The probe set $\mathscr{E}_{\mathrm{P}}$ with a fraction of $f$ links, and training set $\mathscr{E}_{\mathrm{T}}$ with a fraction of $(1-f)$ links. $\mathscr{E}_{\mathrm{P}} \cup \mathscr{E}_{\mathrm{T}} = \mathscr{E}$
 1: Set $n = 0$
 2: while $n < \lceil f \times \mathscr{E} \rceil$ do
 3:     Randomly select one node $i$ from $\mathscr{V}$
 4:     if node $i$ has at least one link (not an isolate node) then
 5:         Randomly select one of link $(i, j)$ of node $i$
 6:         if $rand() < p^1$ then
 7:             Select link $(i, j)$ as one of probe links, and remove it from the network
 8:             $n = n + 1$
 9:         end if
10:     end if
11: end while
```

A. AUC

The AUC statistic Clauset et al. (2008) is defined to be the probability that a randomly chosen link in the probe set is given a higher score by the link prediction method than that of a randomly chosen nonexistent link. If we denote n as the total number of independent comparisons, and n' denotes the times that the score of a randomly selected link in the probe set is higher than that of the nonexistent link, then:

$$\text{AUC} = \frac{n'}{n}. \tag{S15}$$

Remark 1. *In literature Zhou et al. (2009), a slightly different definition of AUC was used:*

$$AUC = \frac{n' + 0.5n''}{n}, \tag{S16}$$

where n'' is the times that the score of a randomly selected link in the probe set is equal to that of a randomly chosen nonexistent link. We emphasize that with this definition of AUC, our DGM-based link prediction method still displays very robust and superior performance for both undirected and directed real-world networks (see Fig. S5). However, for some similarity-based link prediction methods, this definition of AUC sometimes leads to misleading results. For instance, in Fig. S5, we find that for some similarity-based link prediction methods, the AUC with larger

[2] Note: $\lceil \rceil$ represents the rounds towards the nearest integer. $rand$ denotes a random number from uniform distribution. Probability p does not affect the fairness of algorithm and we chose $p = 0.01$ for all of networks.

training set (or equivalently, smaller probe set, say $f = 0.1$) is even lower than that with smaller training set (or larger probe set, say $f = 0.9$). This spurious performance enhancement with larger probe set is due to the term n'' in Eq. (S16). Consider a specific similarity-based link prediction method such as the preferential attachment (PA) method, where the similarity score of a node pair is defined as the degree product of the two nodes. As shown in Fig. S4(b), (d), with a larger probe set, the degrees of nodes in the remaining network display very few possible values (non-negative integers), and most of the nodes have degree zero. Consequently, it is very likely that the score of a randomly selected link in the probe set will be equal to that of a randomly chosen nonexistent link. In other words, as $f \to 1$, $n' \to 0$ and $n''/n \to 1$, rendering $\mathrm{AUC} = (n' + 0.5n'')/n \to 0.5$. Note that the AUC definition in Eq. (S15) does not include the contribution of n'', hence in the case $f \to 1$, we will have $n' \to 0$, and naturally $\mathrm{AUC} = n'/n \to 0$.

In addition, one can show that the calculation of AUC using Eq. (S16) is actually equivalent to the AUC calculated directly from the whole receiver operating characteristic (ROC) curve. For example, as shown in Fig. **??***, we plotted the whole ROC curves for 12 link prediction algorithms for the 12 real-world networks analyzed in the main text (with test set fraction $f = 0.1$). The AUC values are shown in Fig.* **??***, which are consistent with what we show in Fig. S5. Since we want to systematically demonstrate the performance of various link prediction methods with different fractions of training set, showing the whole ROC curves will make the presentation very complicated. Hence in the main text, we did not use the ROC curves to present our results.*

Remark 2. *Based on definition of Eq. (S15), for some networks, most similarity-based link prediction methods yield AUC ~ 0 as shown in the main text (Fig.4). This is simply because the scores of almost all links are 0, and hence scores of missing links tend to be equal to that of nonexistent links, rendering $n' \sim 0$ and hence AUC ~ 0. Note that this doesn't mean that we can get AUC ~ 1 by simply reversing the ranks. Only if scores of missing links are strictly lower than that of nonexistent links (which also leads to $n' \sim 0$ and AUC ~ 0) can we simply invert the prediction to provide a better-than-random result. With the definition of Eq. (S16), we have shown that the AUCs of most similarity-based methods are close to 0.5 (especially for large fraction of probe set, see Fig. S5). This clearly implies that we cannot simply invert the prediction to provide a better-than-random result.*

B. Precision

To calculate the Precision of a link prediction method, we first sort the non-observed links in descending order according to their scores assigned by the link prediction method. We take the top-L non-observed links as the predicted ones, and count how many of the L links belong to the probe set, denoted as L_{p}. Then the Precision is defined as Lü and Zhou (2011):

$$\text{Precision} = \frac{L_{\mathrm{p}}}{L}. \tag{S17}$$

In the main text, we choose AUC rather than Precision as the metric to quantify the performance of various link prediction methods. The reason is that the Precision metric is quite sensitive to L and the total link set $\mathscr{E}$. For example, if we compare the Precisions of L being 10% and 20% of total links as predicted ones respectively, one can find that the Precision corresponding to $L = \lceil 0.1\mathscr{E} \rceil$ is smaller than that of $L = \lceil 0.2\mathscr{E} \rceil$. This would be rather counterintuitive, since with $L = \lceil 0.2\mathscr{E} \rceil$, we have chosen more links as the probe set, and the training set is smaller, which means that the link prediction performance should degrade. Therefore, the spurious high Precision is generally caused by the increase of L, rather than the enhancement of link prediction method Pech et al. (2017). If we just calculate the Precision according to Eq. (S17), with an $f = 0.3$ fraction of probe links, and we take the top-L ($L = \lceil 0.3\mathscr{E} \rceil$) non-observed links as predicted ones. Note that if we use Precision as the performance metric, our method still outperforms other methods in 8 of the 12 real networks studied in this work. None of the other methods can achieve this level of robust performance across different real networks (see Table. **??**).

Remark 3. *We also used two other evaluation metrics: AUPRC — the area under the precision-recall curve; and $P(100)$ — the precision of top-100 predicted links to evaluate the performance of each link prediction method. We found our methods still shows superior performance in most of real networks (Table.* **??***, Table.* **??***, Fig. S13). Note that our method does not perform well for two particular networks: Consulting and Cat cortex. It is likely due to the fact that those two networks are very dense (with mean degree 32.5 and 31.5, respectively). For such dense networks, node labeling based on community detection might not be very effective in detecting structural patterns in the networks. We emphasize that, though our method does not always display the best performance over all those networks, none of the other methods displays as robust performance as our method.*

III. DEEP GENERATIVE MODELS

As a class of models for unsupervised learning, generative models take a training set that consists of samples drawn from some unknown distribution $p_{\mathrm{data}}(\mathbf{x})$, learn a distribution $p_{\mathrm{model}}(\mathbf{x})$ that we can sample from, such that $p_{\mathrm{model}}(\mathbf{x})$ is as similar as possible to $p_{\mathrm{data}}(\mathbf{x})$. Generative models learn $p_{\mathrm{model}}(\mathbf{x})$ either explicitly or implicitly Goodfellow et al. (2016). In the former case, we explicitly define and solve for $p_{\mathrm{model}}(\mathbf{x})$. In the latter case, we learn a model $p_{\mathrm{model}}(x)$ that we can sample from it but without explicitly defining it. Deep generative models (DGMs) define distributions over a set of variables organized in multiple layers Hu et al. (2017). In this section, we describe two of the most commonly used DGMs: Variational Autoencoders (VAE) and Generative Adversarial Networks (GANs).

A. Variational Autoencoder.

Consider a dataset consisting of N samples of some continuous or discrete variable $\mathbf{x}$. VAE contains a specific probability model of data $\mathbf{x}$ and latent variable $\mathbf{z}$. We can write the joint probability distribution as $p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})$, where $\boldsymbol{\theta}$ represents neural network model parameters. VAE assumes that the data are generated as follows: (i) draw a latent variable $\mathbf{z}^{(i)}$ from some prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$; (ii) then draw the datapoint $\mathbf{x}^{(i)}$ from the likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$.

Note that the integral of the marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})\mathrm{d}\mathbf{z}$ is generally intractable, since one has to evaluate over all configurations of the latent variable $\mathbf{z}$. Consequently, the posterior density $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$ is also intractable.

VAE approximates the intractable true posterior density $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ by a recognition model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. Since the latent variable $\mathbf{z}$ is unobserved, it can be interpreted as a *code*. Similarly, the recognition model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is called an *encoder*. For a given datapoint $\mathbf{x}$, the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ will produce a distribution over the possible values of the code $\mathbf{z}$ from which the datapoint $\mathbf{x}$ could have been generated. And the likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ will be referred to as *decoder*, because for a given code $\mathbf{z}$, it will produce a distribution over the possible corresponding values of $\mathbf{x}$.

To quantify the difference between the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and the true posterior density $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, we use the Kullback-Leibler divergence that measures the information loss when using the encoder to approximate the true posterior (in units of nats):

$$\begin{aligned} D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\right) &\equiv \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log\frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\mathrm{d}\mathbf{z} \\ &= \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})\right] + \log p_{\boldsymbol{\theta}}(\mathbf{x}). \end{aligned} \tag{S18}$$

Hence we have

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\mathbf{x}) &= D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\right) - \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})\right] \\ &= D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\right) + \mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x}) \\ &\geq \mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x}). \end{aligned} \tag{S19}$$

In the last step, we have used the fact that the KL-divergence is non-negative. The above inequality (S19) suggests that: (i) $\mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x})$ is the (variational) lower bound of the log marginal likelihood; (ii) we can minimize $D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\right)$ by maximizing $\mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x})$. Note that (ii) is the key idea of VAE, since $D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\right)$ is simply intractable (due to the intractable posterior density $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$), while $\mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x})$ is tractable.

Here

$$\begin{aligned} \mathscr{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x}) &\equiv \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})\right). \end{aligned} \tag{S20}$$

Note that the first term in Eq. (S20) is the expected log-likelihood of the datapoint $\mathbf{x}$. The expectation is taken with respect to the encoder's distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ over the latent variable $\mathbf{z}$. This term encourages the decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ to learn to reconstruct the data. The second term is the Kullback-Leibler divergence between the encoder's distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and the prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$. This term encourages small error when using $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to approximate $p_{\boldsymbol{\theta}}(\mathbf{z})$.

B. Generative Adversarial Networks.

GANs are based on a game theory setup, where two players (the generator and the discriminator) compete with each other Goodfellow et al. (2014). Each player can be represented by a function that is differentiable with respect to both its inputs and its parameters. The generator directly produces fake samples $\mathrm{x} = G(\mathrm{z}; \boldsymbol{\theta}^{(g)})$ with z the input noise variable and $\boldsymbol{\theta}^{(g)}$ the parameters. Its adversary, the discriminator, attempts to distinguish between real samples drawn from the training data and fake samples generated by the generator. The discriminator takes x as input and uses $\boldsymbol{\theta}^{(d)}$ as parameters. It outputs a single scalar $D(\mathrm{x}; \boldsymbol{\theta}^{(d)})$, indicating the probability of x belonging to the real samples.

Within the GANs framework, the discriminator is trying to minimize the cost function

$$J^{(d)}(\boldsymbol{\theta}^{(d)}, \boldsymbol{\theta}^{(g)}) = -\frac{1}{2}\mathbb{E}_{\mathbf{x}\sim\mathbb{P}_{\mathrm{data}}(\mathbf{x})} \log D(\mathbf{x}, \boldsymbol{\theta}^{(d)}) - \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim\mathbb{P}_{\mathbf{z}}(\mathbf{z})} \log\left[1 - D(G(\mathbf{z}, \boldsymbol{\theta}^{(g)}); \boldsymbol{\theta}^{(d)})\right], \tag{S21}$$

which is the cross-entropy cost when training a standard binary classifier with a sigmoid input Goodfellow (2016).

For the generator, we can specify its cost function in many different ways. The simpliest way is a zero-sum game, i.e.,

$$J^{(g)}(\boldsymbol{\theta}^{(d)}, \boldsymbol{\theta}^{(g)}) = -J^{(d)}(\boldsymbol{\theta}^{(d)}, \boldsymbol{\theta}^{(g)}). \tag{S22}$$

Since $J^{(g)}$ is directly tied to $J^{(d)}$, we can summarize the entire game with a value function

$$V(\boldsymbol{\theta}^{(d)}, \boldsymbol{\theta}^{(g)}) = -J^{(d)}(\boldsymbol{\theta}^{(d)}, \boldsymbol{\theta}^{(g)}). \tag{S23}$$

During the learning, each player attempts to maximize its own payoff, and at convergence

$$\boldsymbol{\theta}^{(g)*} = \arg\min_{\boldsymbol{\theta}^{(g)}} \max_{\boldsymbol{\theta}^{(d)}} V(\boldsymbol{\theta}^{(d)}, \boldsymbol{\theta}^{(g)}). \tag{S24}$$

Unfortunately, the cost function used for the generator G in the above zero-sum game is only useful for theoretical analysis. In practice, it does not perform very well, because it may not provide sufficient gradient for G to learn well. Indeed, in the beginning of the learning process, G performs poorly and generates fake samples that can be easily rejected by D with very high confidence. In this case, $\log\left[1 - D(G(\mathbf{z}, \boldsymbol{\theta}^{(g)}); \boldsymbol{\theta}^{(d)})\right]$ saturates, rendering the generator's gradient vanish. Instead of training G to minimize $\log\left[1 - D(G(\mathbf{z}, \boldsymbol{\theta}^{(g)}); \boldsymbol{\theta}^{(d)})\right]$, we can train G to maximize $\log D(G(\mathbf{z}, \boldsymbol{\theta}^{(g)}); \boldsymbol{\theta}^{(d)})$. In other words, G aims to maximize the log-probability that D makes a mistake, rather than aiming to minimize the log-probability that D makes the correct prediction. The cost of the generator is then re-defined as

$$J^{(g)}(\boldsymbol{\theta}^{(d)}, \boldsymbol{\theta}^{(g)}) = -\frac{1}{2}\mathbb{E}_{\mathbf{z}\sim\mathbb{P}_{\mathbf{z}}(\mathbf{z})} \log D(G(\mathbf{z}, \boldsymbol{\theta}^{(g)}); \boldsymbol{\theta}^{(d)}). \tag{S25}$$

However, even this modified cost function can misbehave in case the discriminator is really good Arjovsky and Bottou (2017). Such training difficulty of GANs might be due to the divergences between GANs' training objective and generator's parameters $\boldsymbol{\theta}^{(g)}$ updating direction Arjovsky et al. (2017). To address this issue, the Earth-Mover (also called Wasserstein-1) distance $W(\mathbb{Q}, \mathbb{P})$ was proposed Arjovsky et al. (2017). Informally, $W(\mathbb{Q}, \mathbb{P})$ defines the minimum cost of transporting mass in order to transform the distribution $\mathbb{Q}$ into the distribution $\mathbb{P}$ (where the cost is mass times transport distance). Under mild assumptions, $W(\mathbb{Q}, \mathbb{P})$ is continuous everywhere and differentiable almost everywhere. Unlike the original GANs cost function, the Wasserstein GANs (or WGANs) is more likely to provide gradients that are useful for updating the generator. Yet, the cost function in the WGANs relies on the discriminator being a k-Lipschitz continuous function Arjovsky et al. (2017); Gulrajani et al. (2017). Following the setup in Arjovsky et al. (2017), the cost functions of the generator and the discriminator are:

$$J^{(g)}(\boldsymbol{\theta}^{(d)}, \boldsymbol{\theta}^{(g)}) = -\mathbb{E}_{\mathbf{z} \sim \mathbb{P}_{\mathbf{z}}(\mathbf{z})}[D(G(\mathbf{z}, \boldsymbol{\theta}^{(g)}); \boldsymbol{\theta}^{(d)})] \tag{S26a}$$

$$J^{(d)}(\boldsymbol{\theta}^{(d)}, \boldsymbol{\theta}^{(g)}) = -\mathbb{E}_{\mathbf{x} \sim \mathbb{P}_{\text{data}}}[D(\mathbf{x}, \boldsymbol{\theta}^{(d)})] + \mathbb{E}_{\mathbf{z} \sim \mathbb{P}_{\mathbf{z}}(\mathbf{z})}[D(G(\mathbf{z}, \boldsymbol{\theta}^{(g)}); \boldsymbol{\theta}^{(d)})]. \tag{S26b}$$

The entire game with a value function $V(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$ is given by:

$$\min_{\boldsymbol{\theta}^{(g)}} \max_{\boldsymbol{\theta}^{(d)}} V(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)}) = \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_{\text{data}}(\mathbf{x})}[D(x, \boldsymbol{\theta}^{(d)})] - \mathbb{E}_{\mathbf{z} \sim \mathbb{P}_{\mathbf{z}}(\mathbf{z})}[D(G(z, \boldsymbol{\theta}^{(g)}); \boldsymbol{\theta}^{(d)})]. \tag{S27}$$

IV. DGM-BASED LINK PREDICTION

The key idea of our DGM-based link prediction method is to treat the adjacency matrix of a network as the pixel matrix of an image. By perturbing the original input image in M different ways, we obtain a pool of perturbed images. Those perturbed images will be fed into a DGM to create S fake images that look similar to the input ones. The existent likelihood of the link between nodes i and j, denoted as α_{ij}, in a fake network is simply given by $\alpha_{ij} = 1 - P_{ij}$, where P_{ij} is the rescaled pixel value (ranging from 0 to 1) in its corresponding fake grayscale image. Finally, we take the average value $\langle \alpha_{ij} \rangle = 1 - \langle P_{ij} \rangle$ over all the S fake images to get the overall existent likelihood of the node pair (i, j). In the following, we describe the details of two DGMs, i.e., VAE and GANs in solving the link prediction problem.

A. VAE-based link prediction

We employed the Tensorflow implementation of Convolutional Variational Autoenconders *Convolutional Variational Autoencoder – Jason Ramapuram* (2016). In our calculation, we chose the following parameters: number of latent variables is 20, Mini-batch size for data subsampling is 128. We test the VAE-based link prediction for a directed network with 28 nodes and 118 links, whose adjacency matrix looks like the letter **E**. We randomly remove 5% links from the total links to be the probe set, and the remaining links form the training set. Then, we perturb the network by randomly removing 5 links in M different ways to acquire $M (M = 1000)$ networks as the training input to VAE. The AUC of VAE-based link prediction is shown in Fig. S16(B), which is much lower than that of GANs-based link prediction in Fig. S16(C). Moreover, the fake images generated by VAE (Fig. S16(D)) are generally more blurry than those generated by GANs (Fig. S16(E)). This partially explains why the performance of VAE-based link prediction is worse than that of the GANs-based link prediction Goodfellow (2016). (See Sec. IV B 2 for more detailed explanations.)

B. GANs-based link prediction

We implemented the Wasserstein GAN using Python 2.7 with deep learning open-source package TensorFlow Abadi et al. (2016). The stride size for both generator and discriminator is 2×2. The number of convolutional and deconvolutional kernels are 128 and 64 for generator and discriminator, respectively. Being affected by convolutional (deconvolutional) layer as well as the stride of filters, the network size should be divisible by 4, hence we add some isolate nodes (at most 3) to each network, while those nodes are excluded in the calculating of AUC. We use Mini-batch gradient descent with mini-batch size of 32 to perform optimization. The learning rate (used to control the speed in adjusting the weights of neural network of loss gradient) is set to be $\eta = 0.0002$, clipping parameter (used to clip the value of weights) $c = 10^{-7}$. In order to achieve stability, the discriminator is trained two steps and then alternatively training the generator for one step (see Algorithm 2 for training process of WGAN in link prediction).

1. Selection of hyperparameters

Note that the AUC of GANs-based link prediction can be affected by the following hyperparameters: (1) total epochs T (one epoch in stochastic gradient descent is defined as a single pass of training data through neural networks); (2) size of the input pool M; (3) perturbation strategy, including randomly adding or removing links; (4) fraction of perturbed links (adding a links or removing r links); and (5) final epoch window $(T - \Delta T, T)$ selected to calculate the average AUC.

For all of the networks (if not specified), we chose $T = 150$, $M = 1000$, $\Delta T = 20$. We systematically examined these hyperparameters in the link prediction of a directed network whose adjacency matrix looks similar to the letter **E**. We found that: (i) the larger input pool can render the AUC converge to maximum value faster, while the average AUC over final ΔT epochs is lower (Fig. S17(a), (d)); (ii) randomly removing a few links facilities the AUC of links prediction (Fig. S17(b), (e)), while randomly adding links is always inducing lower AUC (Fig. S17(c), (f)). Interestingly, for $r = 0$ (i.e., without introducing any perturbation to the network), the AUC is still very high. But we do see larger fluctuations than that of $r = 5$ in the final epoch window. Fig. S17(g)-(l) shows similar results for an undirected network (random graph).

In total, the results presented in Fig. 4 and Fig. 5 of main text are conservative estimation of the performance of our GANs-based link prediction, because we didn't fine tune hyperparameters for each network to achieve the best performance.

Remark 4. *We emphasize that the framework of our GANs-based method in link prediction is significantly different from that in computer vision, where GANs are trained to produce fake images by learning input images' distributions. Such learning scheme may lead to a potential issue of GANs unable to learn the distribution of real data indeed Arora and Zhang (2017). Here, we infer the missing pixels (links) from the generated fake images (networks) with a single input image and its various random perturbations.*

Algorithm 2 WGANs for Link Prediction

```
Input: Learning rate $\eta$, clipping parameter $c$, batch size $m$, Number of iterations for discriminator per generator iteration $n_{discri}$
Input: Initial weights $\boldsymbol{\theta}^{(d)}$ for discriminator and $\boldsymbol{\theta}^{(g)}$ for generator
  while $\boldsymbol{\theta}^{(d)}$ has not converged do
    for $t = 0, ..., n_{discri}$ do
      # Update parameter for Discriminator
      Sample batch from historical data:
      $\{x^{(i)}\}_{i=1}^{m} \; from \; \mathbb{P}_{data}$
      Sample batch from Gaussian distribution:
      $\{z^{(i)}\}_{i=1}^{m} \; from \; \mathbb{P}_{Z}$
      Update discriminator nets using gradient descent:
      $g_{\boldsymbol{\theta}^{(d)}} \leftarrow \nabla_{\boldsymbol{\theta}^{(d)}}[-\frac{1}{m}\sum_{i=1}^{m} D(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m} D(G(z^{(i)}))]$
      $\boldsymbol{\theta}^{(d)} \leftarrow \boldsymbol{\theta}^{(d)} - \eta \cdot RMSProp^{3}(\boldsymbol{\theta}^{(d)}, g_{\boldsymbol{\theta}^{(g)}})$
      $\boldsymbol{\theta}^{(d)} \leftarrow clip^{3}(\theta^{(d)}, c, 1-c)$
    end for
    # Update parameter for Generator
    Update generator nets using gradient descent:
    $g_{\boldsymbol{\theta}^{(g)}} \leftarrow \nabla_{\boldsymbol{\theta}^{(g)}} \frac{1}{m}\sum_{i=1}^{m} D(G(z^{(i)}))$
    $\boldsymbol{\theta}^{(g)} \leftarrow \boldsymbol{\theta}^{(g)} - \eta \cdot RMSProp(\boldsymbol{\theta}^{(g)}, g_{\boldsymbol{\theta}^{(g)}})$
  end while
```

2. *A heuristic explanation from deep learning perspective*

Here, we briefly explain why our GANs-based link prediction works.

Denote the $N \times N$ adjacency matrix of the input network to GANs as Y. Here, $Y_{ij} = 1$ if there is a link from node j to node i; otherwise $Y_{ij} = 0$. Denote $G(z)$ as the fake network generated by the generator G, where each element $G(z)_{ij}$ represents the existent probability of corresponding node pair (i, j) in the fake network. Then, the weighted adjacency matrix of the reconstructed network, denoted as W, is given by:

$$W \approx G(z). \tag{S28}$$

Denote the ground truth of the targeted network as R, then the probe set and non-existent link set are given by:

$$\begin{aligned} \mathscr{E}_{\rm p} &: R_{ij} = 1, Y_{ij} = 0, \\ \mathscr{E}_{\rm ne} &: R_{ij} = 0, Y_{ij} = 0. \end{aligned} \tag{S29}$$

The AUC of our GANs-based link prediction is given by:

$$\text{AUC} = P(W_{ij} > W_{i'j'}), \text{where}(i, j) \in \mathscr{E}_{\rm p}, (i', j') \in \mathscr{E}_{\rm ne}. \tag{S30}$$

We denote Γ^{ij} as the neighbor set of entry (i, j). Then, a link (i, j) belongs to one of the following cases according to the existent probability of its neighbors and itself within the input network: (i) $Y_{ij} = Y_{\Gamma^{ij}}$. For these links, the

[4] Note: $RMSProp$ denotes Root Mean Square Propagation Tieleman and Hinton (2012) and $clip$ is the operation to clip tensor values to a specified min and max.

generated existent probabilities in reconstructed networks satisfy $W_{ij} \approx W_{\Gamma^{ij}} \sim 0$, if $Y_{ij} = 0$; and $W_{ij} \approx W_{\Gamma^{ij}} \sim 1$, if $Y_{ij} = 1$. (ii) $Y_{ij} = 0, Y_{\Gamma^{ij}} = 1$. Following the training process described in Section. III B, the generator G of GANs is trained to optimize its neurons' weights $\theta^{(g)}$ such that the produced fake networks are sufficiently similar to input (real) ones. Therefore, the predicted (missing) pixels tend to have similar intensities with their surrounding pixels Yeh et al. (2016). Equivalently, the predicted existent probabilities of missing links tend to be consistent with existent probabilities of surrounding links, which can be expressed as:

$$W_{ij} = \underset{\theta^{(g)}}{\text{argmin}} \sum_{\Gamma^{ij}} [W_{ij} - W_{\Gamma^{ij}}]^2. \tag{S31}$$

Since $W_{\Gamma^{ij}} \approx Y_{\Gamma^{ij}} \sim 1$, to minimize Eq. (S31), W_{ij} will exceed 0, which means this unobserved link in the original network has higher existent probability in the generated fake networks and hence be one of predicted links. In other words, if the existent probabilities $W_{\Gamma^{ij}}$ of link (i, j)'s neighbors have higher values, then the existent probability W_{ij} of link (i, j) also tends to be higher in the generated networks. That is, if an unconnected nodes pair is surrounded by connected nodes pairs, then it has higher existent probability and vice versa (see Fig. S1 for two examples).

Remark 5. *We have introduced a notation Γ^{ij} in Eq. (S31) to represent the neighboring pixels (or surrounding links) of pixel (link) (i, j). The shape of the neighborhood of pixel (i, j) is determined by the dimension of filters in the convolutional layer. Hence, our GANs-based link prediction method can also generate various fake networks to represent different scales of neighborhood interactions. Such adoption of filters can be applied in the link prediction by tuning the dimension of filters.*

3. A heuristic explanation from network perspective

One of the big advantages of deep neural networks is that they can extract the features within the input data. Here, our key idea of DGM-based link prediction is to treat the adjacency matrix of a network as the pixel matrix of an image. Therefore, given a network, expressed by an adjacency matrix, or equivalently the pixel matrix, GANs can extract the features, denoting as a small $m \times n$ matrix, of the input pixel matrix by filters. Taking a feature example with $m = 2, n = 4$ at the i-th row and j-th column in the adjacency matrix:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1j} & a_{1,j+1} & a_{1,j+2} & a_{1,j+3} & \cdots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2j} & a_{2,j+1} & a_{2,j+2} & a_{2,j+3} & \cdots & a_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & a_{i3} & \cdots & 1 & 1 & 0 & 1 & \cdots & a_{iN} \\ a_{i+1,1} & a_{i+1,2} & a_{i+1,3} & \cdots & 1 & 1 & 1 & 1 & \cdots & a_{i+1,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & \cdots & a_{Nj} & a_{N,j+1} & a_{N,j+2} & a_{N,j+3} & \cdots & a_{NN} \end{pmatrix}. \tag{S32}$$

We can define the similarity between two nodes (u, v) by only considering the topological information within the feature (highlighted as a red submatrix in Eq. (S32)):

$$s_{uv} = |\Gamma(u) \cap \Gamma(v)|. \tag{S33}$$

We take the node pair $(i, i + 1)$ as an example, then within the red feature submatrix we have $\Gamma(i) = \{j, j + 1, j + 3\}$ and $\Gamma(i + 1) = \{j, j + 1, j + 2, j + 3\}$, then we have $s_{i,i+1} = 3$, i.e., node i and node $i + 1$ have three common neighbors (within the feature submatrix). Here, we conclude that node i is similar to node $i+1$, indicating that if there is a link from node $j + 2$ to node $i + 1$, then with high probability there exists a link from node $j + 2$ to i. Note that for a traditional similarity-based method, i.e., common neighbors (CN), it will conclude that with high probability there is a link from node i to $i + 1$, quite different from what will be predicted from our method.
Note that our method does not perform well for two particular networks: Consulting and Cat cortex. This could be due to the following facts: (1) those two networks are very dense (with mean degree 32.5 and 31.5, respectively); (2) there are unstructured long-range connections in those networks that cannot be leveraged by our DGM-based link prediction.

4. Time complexity of GANs-based link prediction

For the DGM, we leverage Generative Adversarial Networks (GANs) that consist of two deep artificial neural networks: generator and discriminator. Both are convolutional neural networks. According to the Ref He and Sun (2015), the total time complexity of all convolutional layers is:

$$O(\sum_{i=1}^{d} n_{l-1} \cdot {s_l}^2 \cdot n_l \cdot m_l^2), \tag{S34}$$

where l is the index of a convolutional layer, d is the number of convolutional layers. n_l is the number of filters in the i-th layer and n_{l-1} is the number of input channels of the l-th layer. s_l is the spatial size of the filter and m_l is the spatial size of the output feature map. For all of networks, we choose the parameters as: $d = 2$, $n_1 = 128$, $n_2 = 64$, $s_{1,2} = 5$, $m_1 \approx N/2$ and $m_1 \approx N/4$, where N is the network size (the number of nodes). We only have 1 channel, so the number of input channels for all of layers is always 1. Then the time complexity to train the GANs of a give network is proportional to N^2.

V. NODE RELABELING ALGORITHM

Apparent, Our DGM-based link prediction relies on the visual patterns within the network. As shown in Fig. S7 for synthetic networks generated by stochastic block model (SBM), with a fixed density of between-cluster, the performance of our DGM method gradually improves with increasing in-cluster density and stronger modular structure. By contrast, with a fixed in-cluster density, the performance of our DGM method deteriorates with increasing between-cluster density and weaker modular structure. To reach the optimal performance of our DGM-based link prediction, one should leverage any existing structural features in the network and label the nodes accordingly. After node labeling, those structural patterns naturally emerge in the matrix (image) presentation. Then, DGM is able to extract the most important structural patterns of the network as the key features of the corresponding image. Node labeling can be achieved in different ways.

A. Node relabeling algorithm based on Louvain community detection

Community detection aims to partition a network into communities of densely connected nodes, while nodes belonging to different communities are connected sparsely. For undirected networks, the quality of the partition can be measured by modularity Blondel et al. (2008); Newman (2006):

$$Q = \frac{1}{2w}\sum_{i,j}[w_{ij} - \frac{w_i w_j}{2w}]\delta(c_i, c_j), \tag{S35}$$

where w_{ij} is the weighted adjacency matrix of the network, $w_i = \sum_j w_{ij}$ is the sum of the edge weights connected to node i, c_i is the community index to which node i is assigned, $\delta(m,n) = 1$ if $m = n$ otherwise $\delta(m,n) = 0$ and $m = \frac{1}{2}\sum_{ij} w_{ij}$.

Ref Blondel et al. (2008) introduced an algorithm to find the partition with high modularity. This is often called the Louvain algorithm, which consists of two iterative phases. (i) Each node is assigned into different communities initially, then calculate the corresponding gain of modularity by removing node i from its community and place it in the community of its neighbor j. Place i in the community with maximum gain (i will stay in its original community if the gain is negative). Repeat this process for all nodes until no further improvement can be achieved; (ii) Build a new network within which the nodes are the communities obtained in the first phase. The links between nodes with the same community are self-loops in the new network, while the link weight between new nodes is the sum weight of links between nodes in the corresponding two communities. Reapply the procedure in phase (i) to the weighted network obtained from phase (ii). Iterate the process until reaching the maximum modularity. In this work, we employ the Louvain algorithm implemented in Python package NetworkX 1.10 Hagberg et al. (2008). For each undirected network, this algorithm assigns each node into a community, then we relabel the nodes within the same community with similar labels.

B. Other node relabeling algorithms

Besides node labeling methods based on community detection or consensus, we can also leverage various matrix reordering algorithms to reveal structural patterns in the adjacency matrix of a network, such as block pattern, off-diagonal block pattern, line/star pattern, bands pattern, noise anti-pattern and bandwidth anti-pattern. In fact, matrix reordering has many applications, such as in archaeology and anthropology; cartography, graphics, and information visualization; sociology and sociometry; psychology and psychometry; ecology; biology and bioinformatics; cellular manufacturing; and operations research Behrisch et al. (2016); Liiv (2010). Consequently, there exist many matrix reordering algorithms, such as Robinsonian Kendall (1963); Robinson (1951), Spectral Friendly (2002); Sternin (1965), Dimension Reduction Elmqvist et al. (2008); Harel and Koren (2002), Heuristic Approaches Deutsch and Martin (1971); Wilkinson (2006), Graph Theoretic Sloan (1986, 1989), Biclustering Hartigan (1972) , Interactive User-Controlled Perin et al. (2014), Cuthill-McKee algorithm Cuthill and McKee (1969), and Barycenter based permutation Mäkinen and Siirtola (2005). We also compared the performance our DGM-based method on a real social network (Zachary karate club) labeled by the Cuthill-McKee algorithm, the Louvain method and our own consensus-based method. We found that all node labeling methods yield very similar performance (see Fig. S18).

VI. DESCRIPTION OF REAL DATASET

(1) **Zachary karate club.** The social network of a karate club. Each node represents an individual and each link represents that two individuals are friends outside the club activities. The network is downloaded from: `https://icon.colorado.edu/#!/networks`.

(2) **Terrorist association.** The terrorist communication network from the attacks on the United States on September 11, 2001. Nodes represent the terrorists and links represent the communication between them. The network is downloaded from: `http://www.orgnet.com/hijackers.html`.

(3) **Medieval river trade.** The network of medieval trade and communication along the rivers of Russia. Each node represents a route or place, and each link represents the trading between two routes. The network is downloaded from: `https://icon.colorado.edu/#!/networks`.

(4) **Contiguous states.** A network of contiguous states and the District of Columbia of the USA. Nodes represent states and a link means that the two states share a land-based geographic border. The network is downloaded from: `http://konect.uni-koblenz.de/networks/contiguous-usa`.

(5) **Internet topology (PoP level).** Internet at the Point of Presence (PoP) level. Nodes represent routers and links mean that the message can transmit along them. The network is downloaded from: `http://www.topology-zoo.org/dataset.html`.

(6) **Protein-protein interactions (PPI).** A sub-network of *S. cerevisiae* protein-protein interaction network, induced by proteins annotated with the function ‘Cellular Communication’ of the Gene Ontology (GO) database. The network is downloaded from: `http://math.bu.edu/people/kolaczyk/datasets.html`.

(7) **Cat cortex.** The brain network of cat cortex, where a node represents a cortical area and a link represents large cortical tracts or functional associations. The network is downloaded from: `https://sites.google.com/site/bctnet/datasets`.

(8) **Consulting.** A social network of consulting company. Each node represents a employee and each link represents the frequency of information or advice requests. The network is downloaded from: `https://toreopsahl.com/datasets/#Cross_Parker`.

(9) **St. Martin.** A network of carbon-flow among species in St. Marks National Wildlife Refuge. Each node represents a species and each link denotes a carbon flow from one species to another. The network is downloaded from: `http://cosinproject.eu/extra/data/foodwebs/WEB.html`.

(10) **Seagrass.** The food web that represents the predatory interactions among species in a seagrass ecosystem. Each node represents a species and each link represents a predatory interaction. The network is downloaded from: `http://cosinproject.eu/extra/data/foodwebs/WEB.html`.

(11) **Sioux Falls traffic.** A network of traffic flow between intersections. Nodes represent intersections and a link represents the traffic flow between two intersections. The network is downloaded from: `https://github.com/bstabler/TransportationNetworks`.

(12) **Cattle.** The network of dominance behaviors observed between dairy cattles at the Iberia Livestock Experiment Station in Jenerette, Louisiana. Each node represents a cattle and a link represents dominance of the left cattle over the right cattle. The network is downloaded from: `http://konect.uni-koblenz.de/networks/moreno_cattle`.

(13) **Little rock.** The food web of the Little Rock Lake. Each node represents a species and each link represents a predatory interaction. The network is downloaded from: `http://konect.uni-koblenz.de/networks/maayan-foodweb`.

(14) **Facebook wall posts.** A network of subset of posts to other user’s wall on Facebook. Each node represents a Facebook user and each link represents a wall-post interaction. The network is downloaded from: `http://konect.uni-koblenz.de/networks/facebook-wosn-wall`.

(15) **Google+.** The social network of Google+ users. Each node represents a user and a link means that one user has the other user in his or her circle. The network is downloaded from: `http://konect.uni-koblenz.de/networks/ego-gplus`.

(16) **Facebook-NIPS.** The user-user friendship network. Each node represents a user and a link indicates that the node is a friend of another. The network is downloaded from: `http://konect.uni-koblenz.de/networks/ego-facebook`.

(17) **US airports.** The network of flights between US airports in 2010. Each node represents an airport and each link represents a connection from one airport to another. The network is downloaded from: `http://konect.uni-koblenz.de/networks/opsahl-usairport`.

Part II

Supplemental Tables

Name	N	L	Type
Zachary karate club Zachary (1977)	34	78	Undirected
Terrorist association Krebs (2002)	62	152	Undirected
Medieval river trade Pitts (1978)	39	53	Undirected
Contiguous state Knuth (1993)	49	107	Undirected
Internet topology (PoP) Knight et al. (2011)	41	63	Undirected
PPI Hand (2010)	104	236	Undirected
Facebook-NIPS Leskovec and Mcauley (2012)	2888	2981	Unirected
Cat cortex Scannell et al. (1999)	52	820	Directed
Consulting Cross and Parker (2004)	46	879	Directed
St. Martin Baird et al. (1998)	45	224	Directed
Seagrass Christian and Luczkovich (1999)	49	226	Directed
Sioux Falls traffic LeBlanc et al. (1975)	24	76	Directed
Cattle Schein and Fohrman (1955)	28	217	Directed
Little rock Martinez (1991)	183	2494	Directed
Facebook wall posts *Facebook wall posts network dataset – KONECT* (2017); Viswanath et al. (2009)	46952	876993	Directed
Google+ *Google+ network dataset – KONECT* (2017); Leskovec and Mcauley (2012)	23628	39242	Directed
US airports Opsahl (2011)	1574	28236	Directed

The networks of Cat cortex, Consulting network and Sioux Falls traffic, Cattle are weighted, here we treat them as an unweighted ones. The protein-protein interaction network is a sub-network of the *S. cerevisiae* protein-protein interaction network, induced by proteins annotated with the function ‘Cellular Communication’ of the Gene Ontology (GO) database.

TABLE S1: **Description of each real network,** related to Figures 4 and 5.

colorNetworks	DGM	CN	PA	RA	JC	KATZ	ACT	SBM	LRMC	DW	NMF	SEAL	GraphSAGE	CGNN
Terrorist association	0.0193 (0.0035)	0.0365 (0.0025)	0.0143 (0.0013)	**0.0483 (0.0034)**	0.028 (0.0014)	0.033 (0.0025)	0.0183 (0.0009)	0 (0)	0 (0)	0.0232 (0.0012)	0.0252 (0.0019)	0.0377 (0.0043)	0.0166 (0.001)	0.0045 (0.0005)
Zachary karate club	0.018 (0.003)	0.0297 (0.0052)	0.0309 (0.0048)	0.0415 (0.0064)	0.0174 (0.0009)	0.035 (0.0068)	0.034 (0.0036)	0 (0)	0 (0)	0.03 (0.0026)	0.0273 (0.0034)	**0.0559 (0.0115)**	0.0161 (0.0007)	0.0136 (0.0023)
PPI	0.0065 (0.0011)	0.0062 (0.0007)	0.0062 (0.0005)	0.0134 (0.0008)	0.0119 (0.0006)	0.0103 (0.0007)	0.0084 (0.0004)	0 (0)	0 (0)	0.0094 (0.0008)	0.0084 (0.0005)	**0.0137 (0.0015)**	0.0071 (0.0005)	0.0012 (0.0001)
Medieval river trade	0.0038 (0.0046)	0.0045 (0.0005)	0.0047 (0.0003)	0.0046 (0.0005)	0.0047 (0.0006)	0.0059 (0.0006)	0.01 (0.0011)	0 (0)	0 (0)	0.0078 (0.001)	0.0064 (0.0006)	**0.0122 (0.0024)**	0.0071 (0.0009)	0.0032 (0.0003)
Internet topology (PoP)	0.0096 (0.0027)	0.0207 (0.0029)	0.0182 (0.004)	**0.0236 (0.0038)**	0.0181 (0.0014)	0.0237 (0.0044)	0.0077 (0.003)	0 (0)	0 (0)	0.0199 (0.0021)	0.0215 (0.0024)	0.0214 (0.0026)	0.0088 (0.0005)	0.0129 (0.0016)
Contiguous states	0.0243 (0.0023)	0.0348 (0.002)	0.008 (0.0005)	0.0438 (0.0027)	0.0564 (0.0036)	0.0296 (0.002)	0.0168 (0.0016)	0 (0)	0 (0)	0.0408 (0.0032)	0.0225 (0.0013)	**0.0611 (0.0039)**	0.0274 (0.002)	0.0062 (0.0003)
Consulting	0.1650 (0.007)		0.131 (0.0039)						**0.2797 (0.0081)**					
Cat cortex	0.0698 (0.0022)		0.00848 (0.0033)						**0.1242 (0.004)**					
Cattle	0.0446 (0.004)		0.0416 (0.0013)						**0.0536 (0.0021)**					
Seagrass	0.0125 (0.0015)		**0.0134 (0.0009)**						0 (0)					
St. Martin	0.019 (0.0018)		0.0173 (0.001)						**0.0315 (0.0063)**					
Sioux Falls traffic	**0.0389 (0.0045)**		0.013 (0.007)						0 (0)					

For each real-world network, we first randomly remove 10% links as the predicted ones, and the remaining links form the training set, which will be used to recover the removed links. We show the mean AUPRC over 20 realizations and stand deviation of AUPRC for each method.

TABLE S2: **The AUPRCs of link prediction methods on real-world networks,** related to Figure 4.

Part III
Supplemental Figures

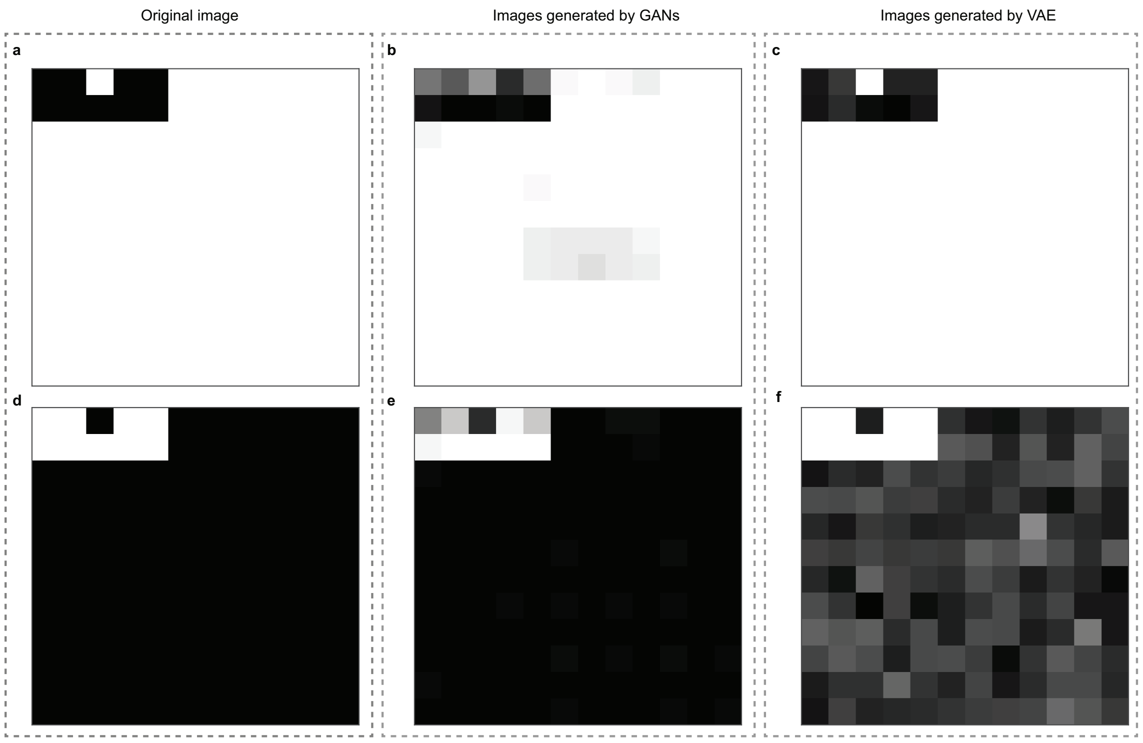


Figure S1: **Intuitive explanation of DGM-based link prediction methods,** related to Figure 1. **a,** An missing link $(1, 3)$ is surrounded by several observed links in the original network. **b,** The snapshot of the recovered network at Epoch $= 50$ by GANs. The existent probability of this missing pixel (link) generated by GANs tends to be consistent with its surrounded pixels (links). **c,** The snapshot of the recovered network at Epoch $= 50$ by VAE. **d,** An observed link $(1, 3)$ is surrounded by several missing links in the original network. **e,** The snapshot of the recovered network at Epoch $= 50$ by GANs. The existent probabilities of several missing pixels (links) tend to be consistent with this observed pixel (link). **f,** The recovered images at Epoch $= 50$ by VAE.

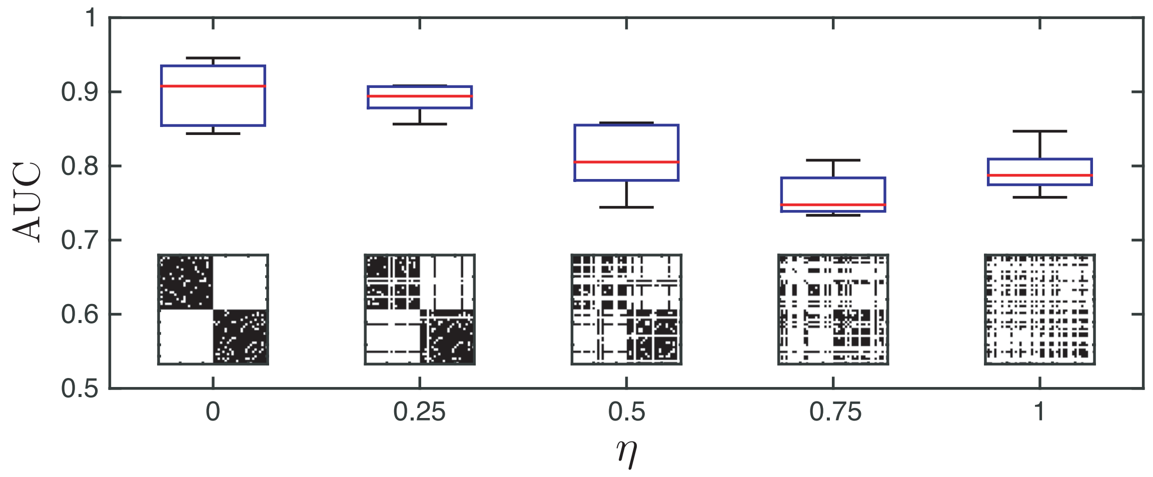
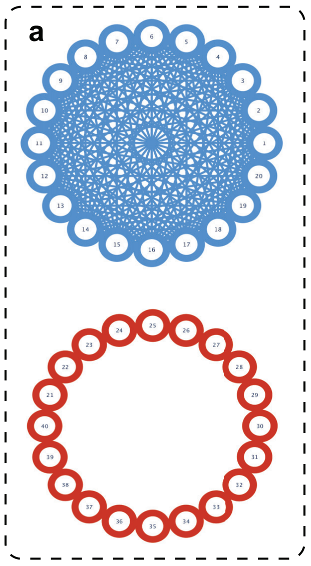


Figure S2: **Performance of our DGM-based link prediction on a network generated by the stochastic block model,** related to Figure 2. The network has $N = 48$ nodes, and two communities (with 24 nodes for each community). The between-cluster connection probability is set to be zero and within-cluster connection probability is 0.83. A randomly selected fraction of $f(0.25, 0.5, 0.75, 1.0)$ nodes are relabeled, rendering different "visual patterns". Boxplots of AUC are calculated from 10 different realizations. For each realization, we randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% links as the training set.



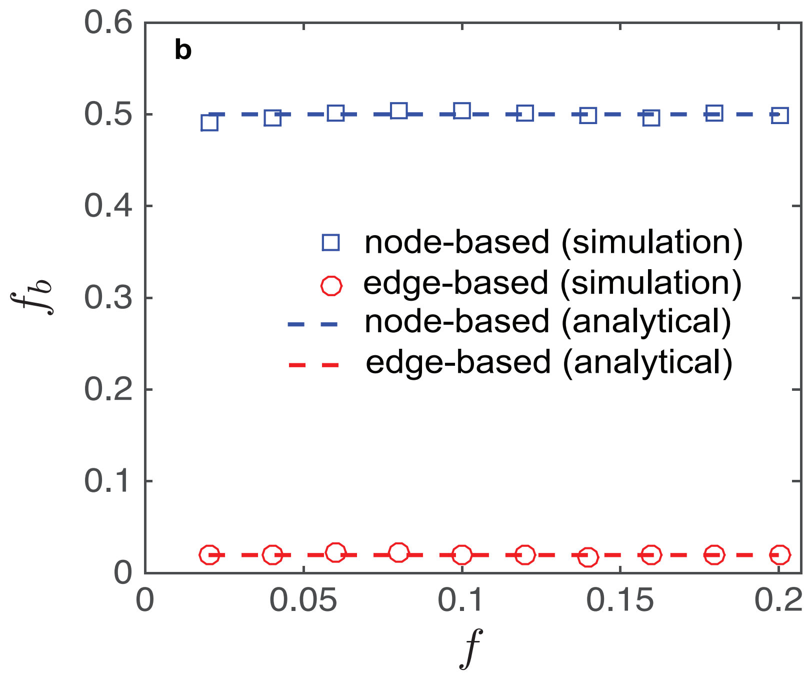


Figure S3: **Edge-based splitting tends to select links originating from hub nodes to be part of the probe set,** related to Figures 2, 4 and 5. **a,** Consider a simple network consisted of two separated subnetworks: (i) A complete graph with $N_1 = 20$ nodes; (ii) A cycle with $N_2 = 20$ nodes. The degree of each node is $k_i = N_1 - 1$ in the first subnetwork and $k_i' = 2$ in the second subnetwork. For the node-based splitting, the probability of selecting $s(s \leq N_2 - 1)$ probe links belonging to the second subnetwork is: $f_b^{\mathrm{n}} = \frac{N_2}{N_1+N_2}$. For the edge-based splitting, the probability of selecting s probe links that belong to the second subnetwork is: $f_b^{\mathrm{e}} = \frac{E_2}{E_1+E_2}$, where $E_1 = \frac{N_1(N_1-1)}{2}$ and $E_2 = N_2 - 1$ are the number of total links in the first and the second subnetwork, respectively. When $N_1 = N_2 = n$, we have $f_b^{\mathrm{n}} = 0.5$, indicating the node-based splitting is perfectly fair; whereas, $f_b^{\mathrm{e}} = \frac{2}{n+2} \to 0$ for large n, indicating that the probability of selecting probe links originating from low-degree nodes is extremely low. **b,** The probability f_b of selecting a fraction of f links as probe set that belong to the second subnetwork, using the edge-based splitting (circles) or node-based splitting (squares).

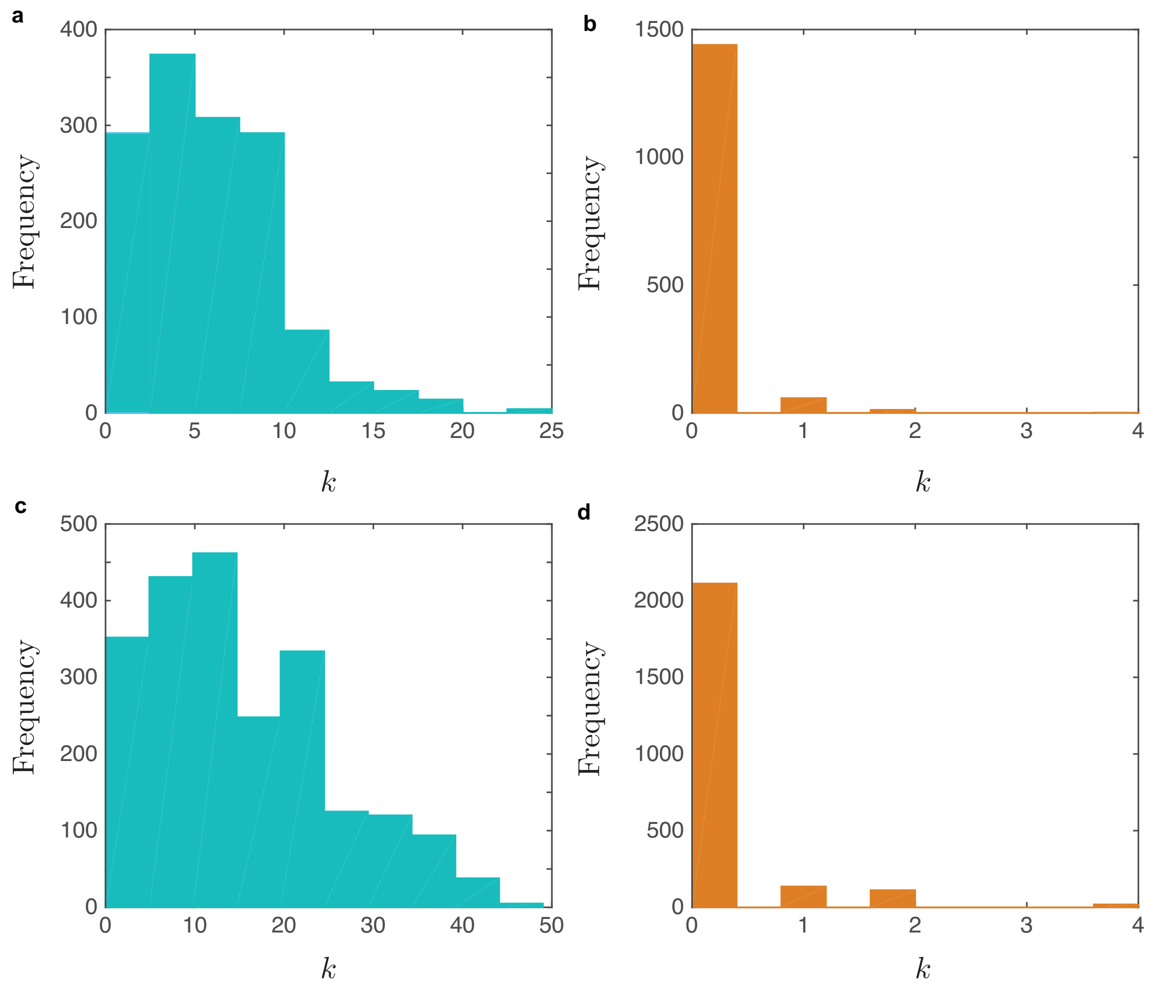


Figure S4: **The spurious enhancement of AUC (Eq. (S16)) with larger probe set for some link prediction methods is induced by the term n'',** related to Figures 2, 4 and 5. An undirected network (Medieval river trade in (**a**), (**b**)) and a directed network (Sioux Falls traffic in (**c**), (**d**)) are chosen to illustrate this point. The total links of each network are split into two parts: a fraction of f as the probe set and the remaining $1-f$ as the training set. Here, we show the degree distributions of nodes in the remaining network for $f=10\%$ (left panel) and $f=90\%$ (right panel), respectively. For a small probe set, the possible degrees of nodes in the remaining network are very diverse ((**a**), (**c**)). Consequently, it is very unlikely that the score of a randomly selected link in the probe set will be equal to that of a randomly chosen nonexistent link. In other words, the term n'' in Eq. (S16) is negligible and will not affect AUC. However, for a large probe set , the degrees of nodes in the remaining network display very few possible values ((**b**), (**d**)), and most of the nodes have degree zero. Then it is very likely that the score of a randomly selected link in the probe set and that of a randomly chosen nonexistent link are both zero, rendering a large n'' in Eq. (S16) and hence a spuriously large AUC.

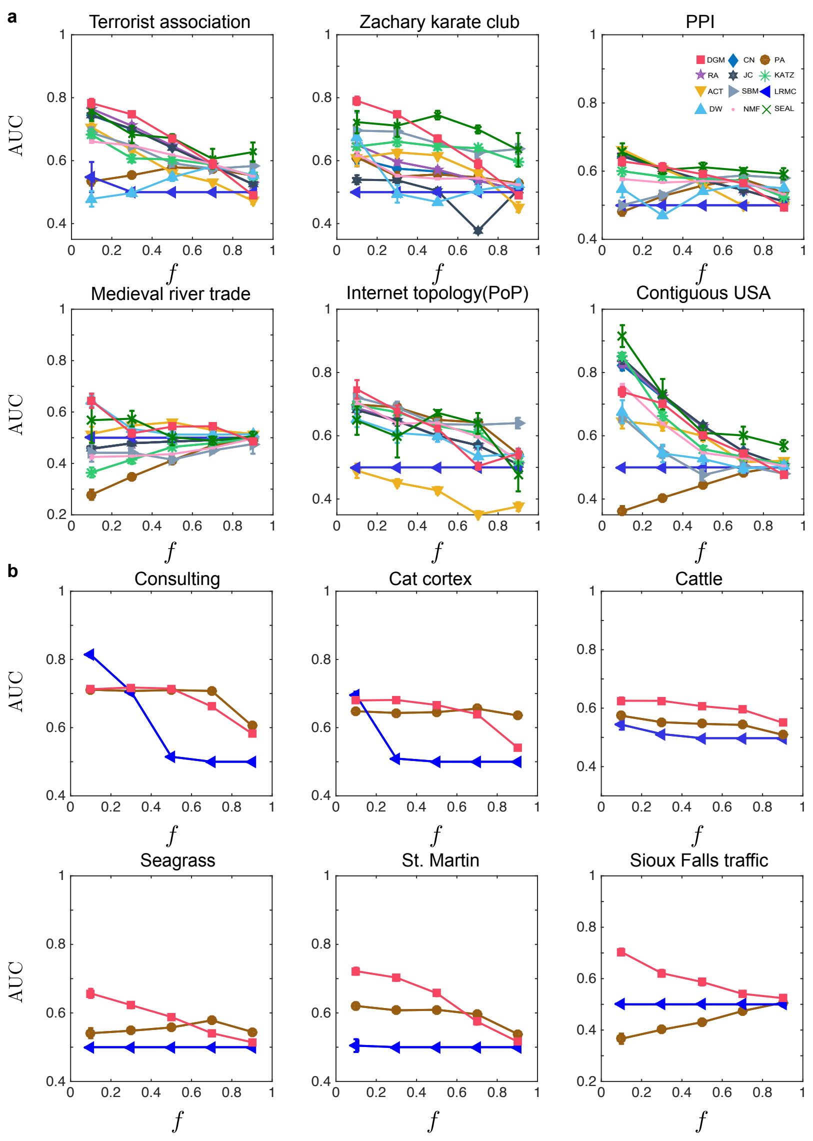


Figure S5: **The DGM-based link prediction method displays very robust and superior performance for both undirected and directed real-world networks,** related to Figure 4. Here an alternative definition of AUC (Eq. (S16)) is used. DGM: deep generative model based link prediction; CN: common neighbors; PA: preferential attachment; RA: resource allocation; JC: Jaccard index; KATZ: Katz index; ACT: average commute time; SBM: stochastic block model; LRMC: low rank matrix completion; DW: deepwalk embedding method; NMF: non-negative matrix factorization); SEAL: learning from Subgraphs, Embeddings, and Attributes for Link prediction (see Section. I for details of each algorithm). **a,** Undirected networks. Top: Terrorist association network, Zachary karate club, Protein-protein interaction (PPI) network (a subnetwork of protein-protein interactions in *S. cerevisiae*). Bottom: Medieval river trade network in Russia, Internet topology (at the PoP level), Contiguous states in USA. **b,** Directed networks. Top: Consulting (a social network of a consulting company), cat cortex (the connection network of cat cortical areas), cattle (a network of dominance behaviors of cattles). Bottom: Seagrass food web, St. Martin food web, Sioux Falls traffic network. AUC of our DGM-based method is the average AUC over the last 20 epochs of the total epochs for all of networks.

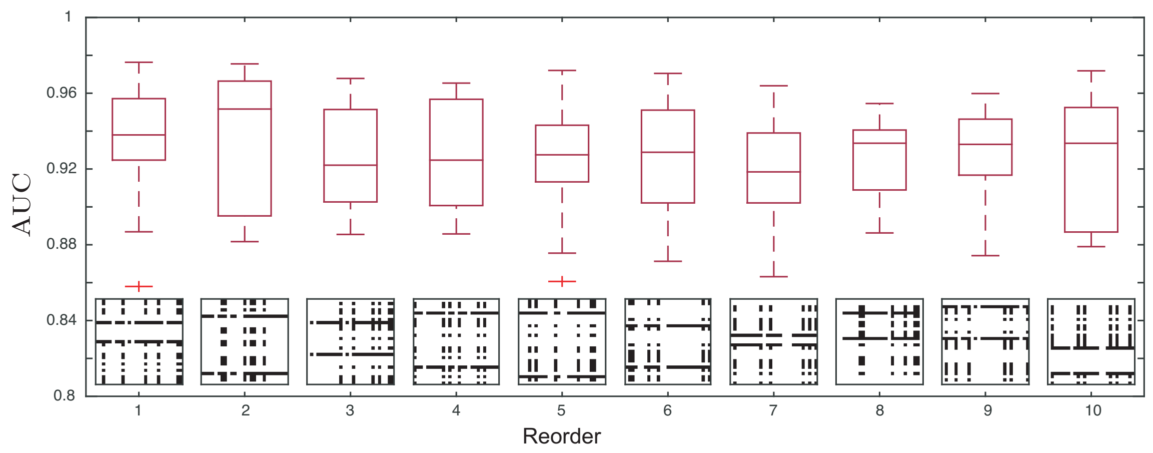


Figure S6: **Performance of our DGM-based link prediction with different random node labelings,** related to Figure 2. We randomly label the nodes in a toy network (as shown in main text Fig.2a) in 10 different ways to obtain 10 different adjacency matrices (binary images shown in inset). Then we show the performance of our method for each adjacency matrix (labelled network) in 10 realizations using boxplots of AUC. For each realization, we randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% links as the training set. We chose $M = 1,000$ and $S = 500$ for our DGM-based method.

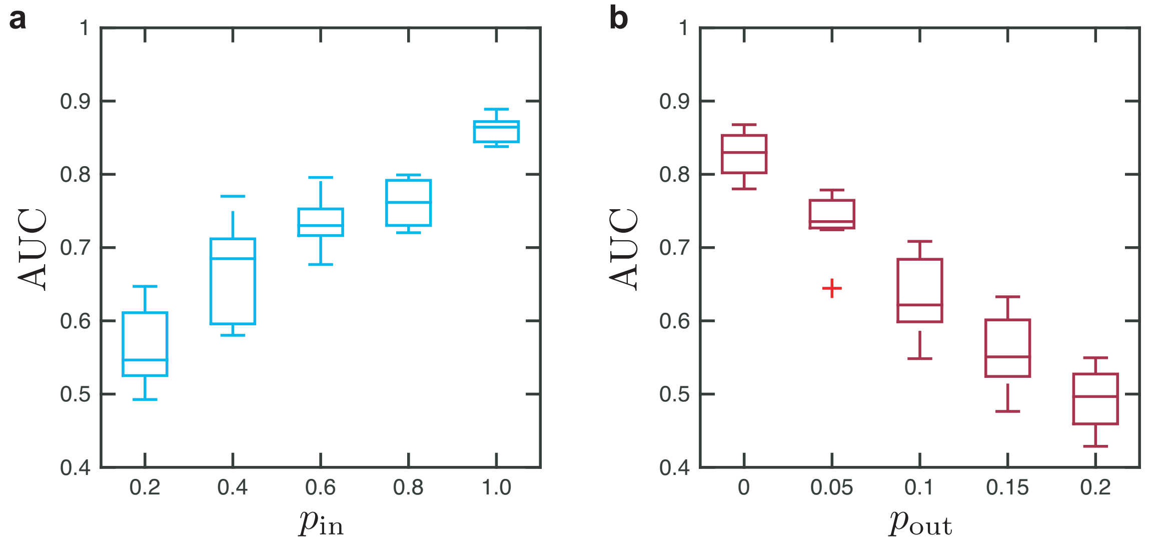


Figure S7: **Performance of our DGM-based link prediction on synthetic networks generated by stochastic block models with varying within-cluster connection probability p_{in} or between-cluster connection probability p_{out},** related to Figure 2. The networks consist of 3 clusters/communities and each cluster has 16 nodes. Boxplots of AUC are calculated from 10 different realizations. For each realization, we randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% links as the training set. **a,** AUC as a function of p_{in}, with fixed $p_{\text{out}} = 0.05$. **b,** AUC as a function of p_{out}, with fixed $p_{\text{in}} = 0.5$.

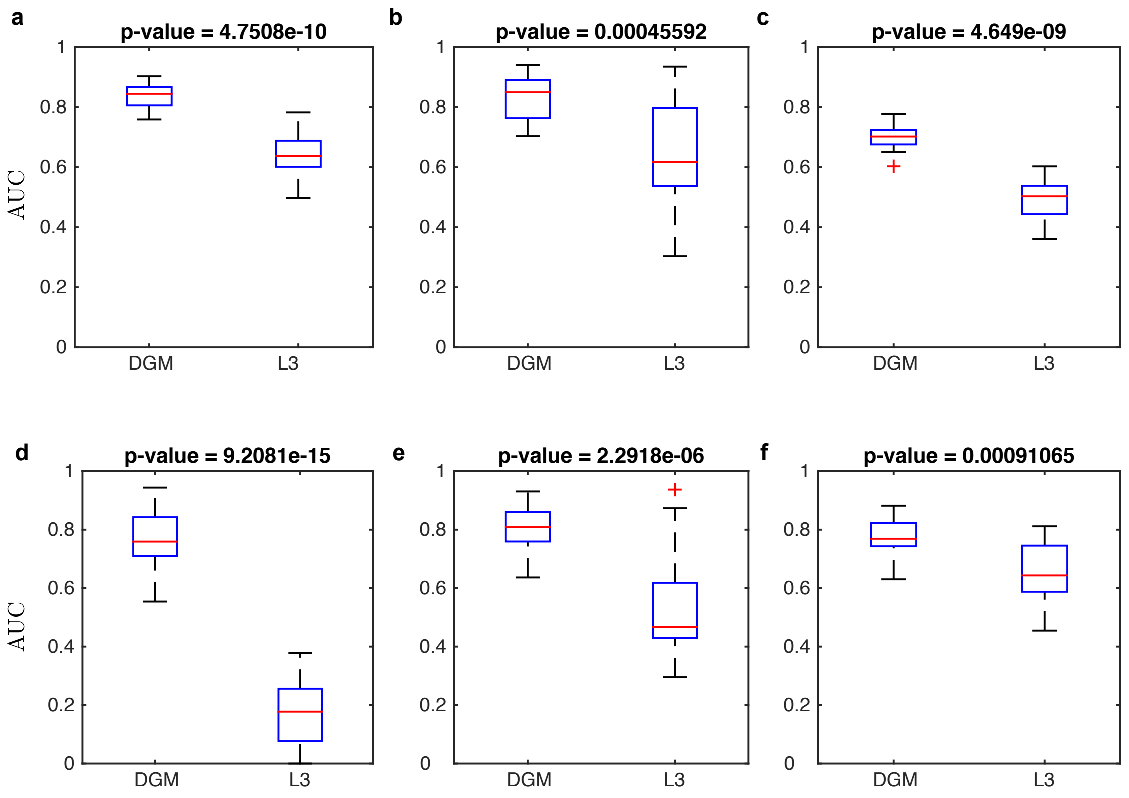


Figure S8: **Comparing the performance of our DGM-based link prediction and the Network Paths of Length Three (L3) for 6 real-world undirected networks,** related to Figure 4. **a,** Terrorist association. **b,** Zachary karate club. **c,** PPI. **d,** Medieval river trade. **e,** Internet. **f,** Contiguous states. Boxplots of AUC are calculated from 20 different realizations. For each realization, we randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% links as the training set.

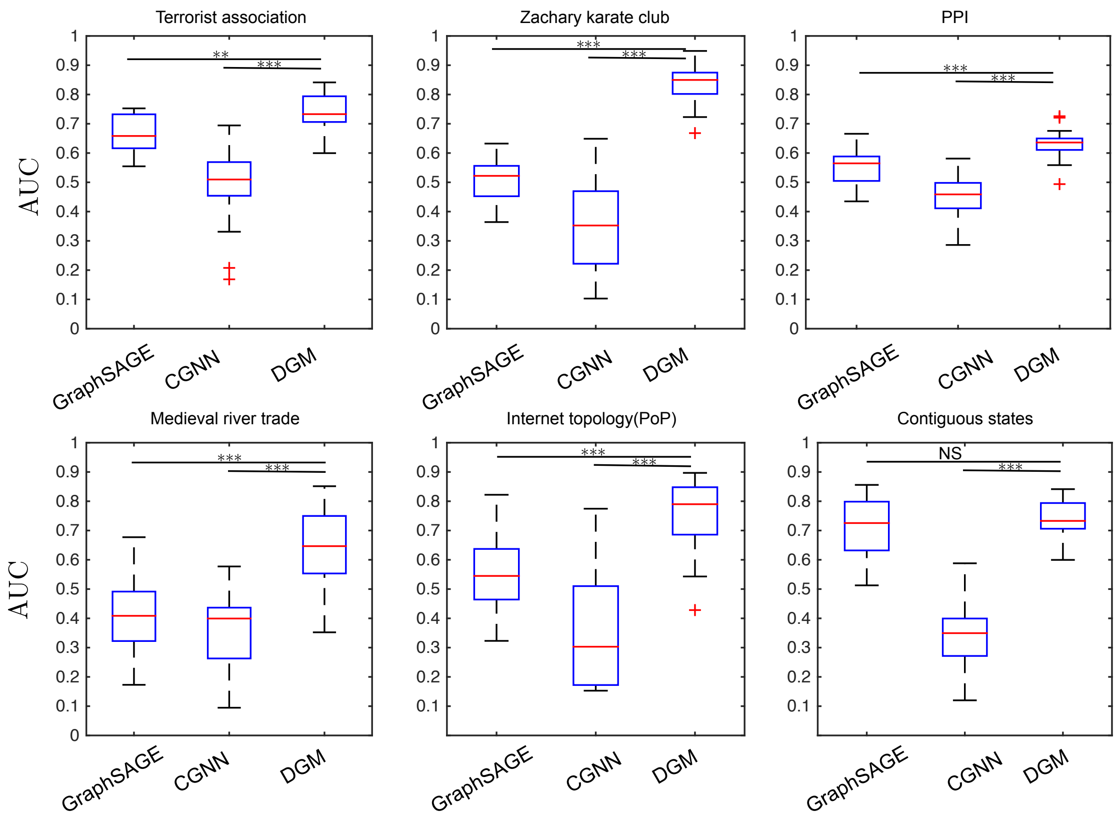


Figure S9: **AUCs of two GCN-based link prediction methods and our method in various real-world networks (see SI for the details of each network),** related to Figure 4. We randomly divide the links into two parts: a fraction of $f = 0.1$ links chosen as the probe set and the remaining $1 - f$ as training set. Significance levels (paired-sample t-test): p-value$<$0.05(*), $<$0.01(**), $<$0.001(***), not significant (NS).

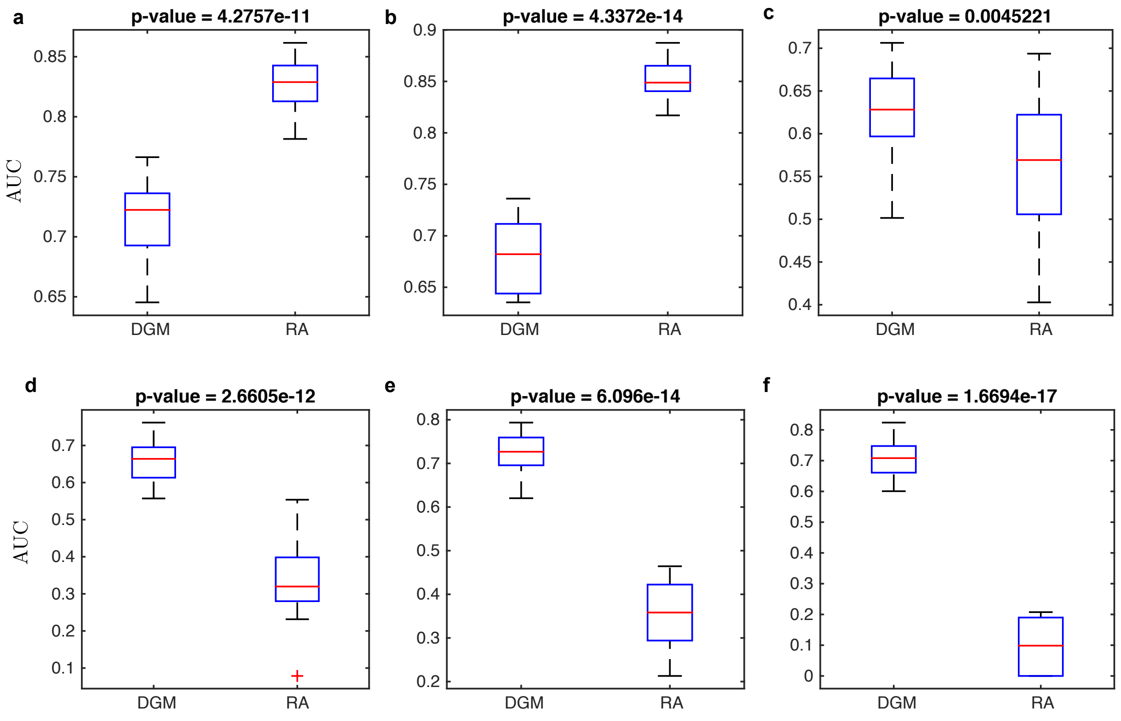


Figure S10: **Comparing the performance of our DGM-based link prediction and the Resource-Allocation (RA) index-based link prediction for 6 real-world directed networks,** related to Figure 4. **a,** Consulting. **b,** Cat cortex. **c,** Cattle. **d,** Seagrass. **e,** St. Martin. **f,** Sioux Falls traffic. Boxplots of AUC are calculated from 20 different realizations. For each realization, we randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% links as the training set.

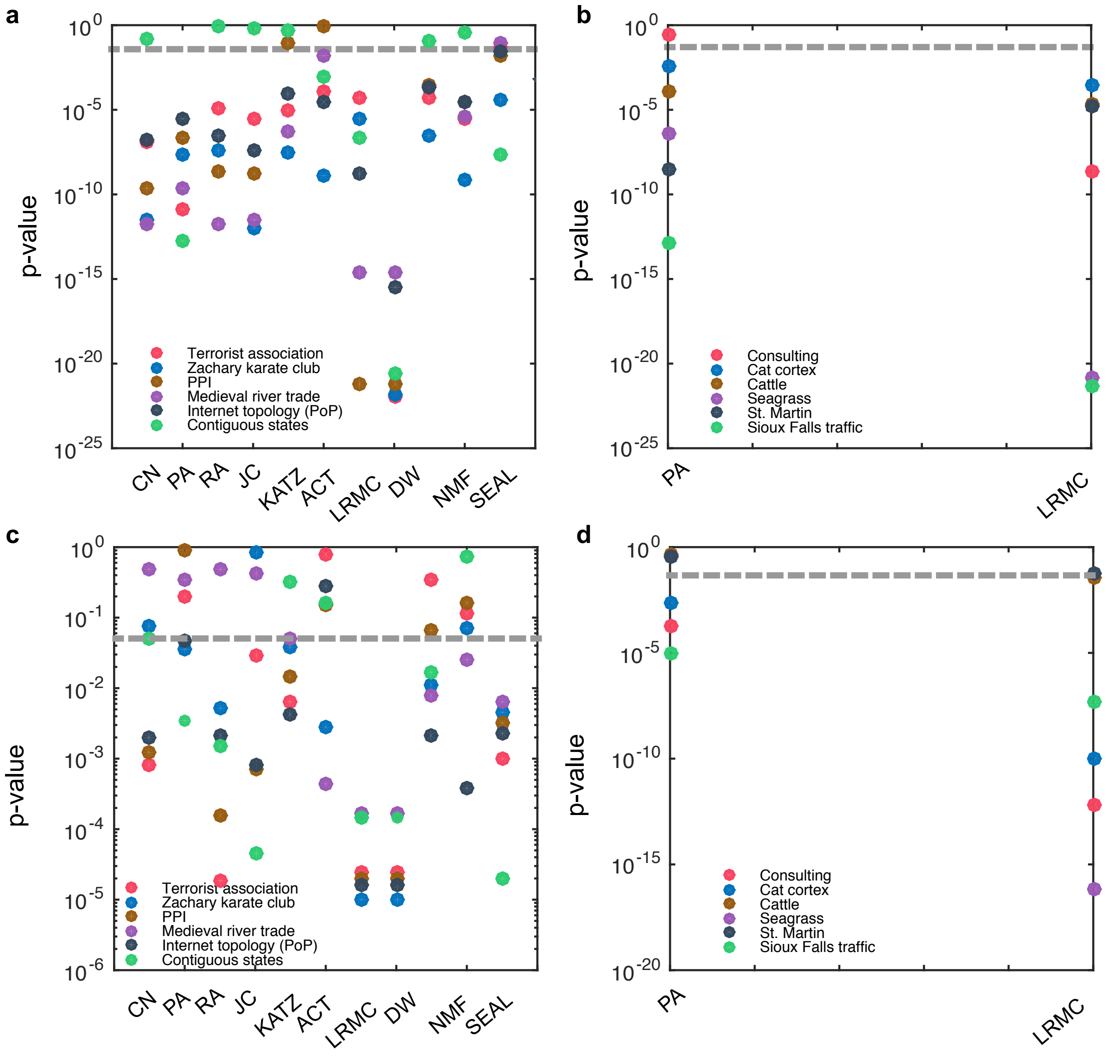


Figure S11: **Significance level of the performance of our DGM-based link prediction compared with other methods,** related to Figure 4. We calculated the p-value using paired t-test for the AUCs of our DGM-based method and other methods in **a,** 6 undirected networks; **b,** 6 directed networks. We calculated the p-value using paired t-test for the AUPRCs of our DGM-based method and other methods in **c,** 6 undirected networks; **d,** 6 directed networks.

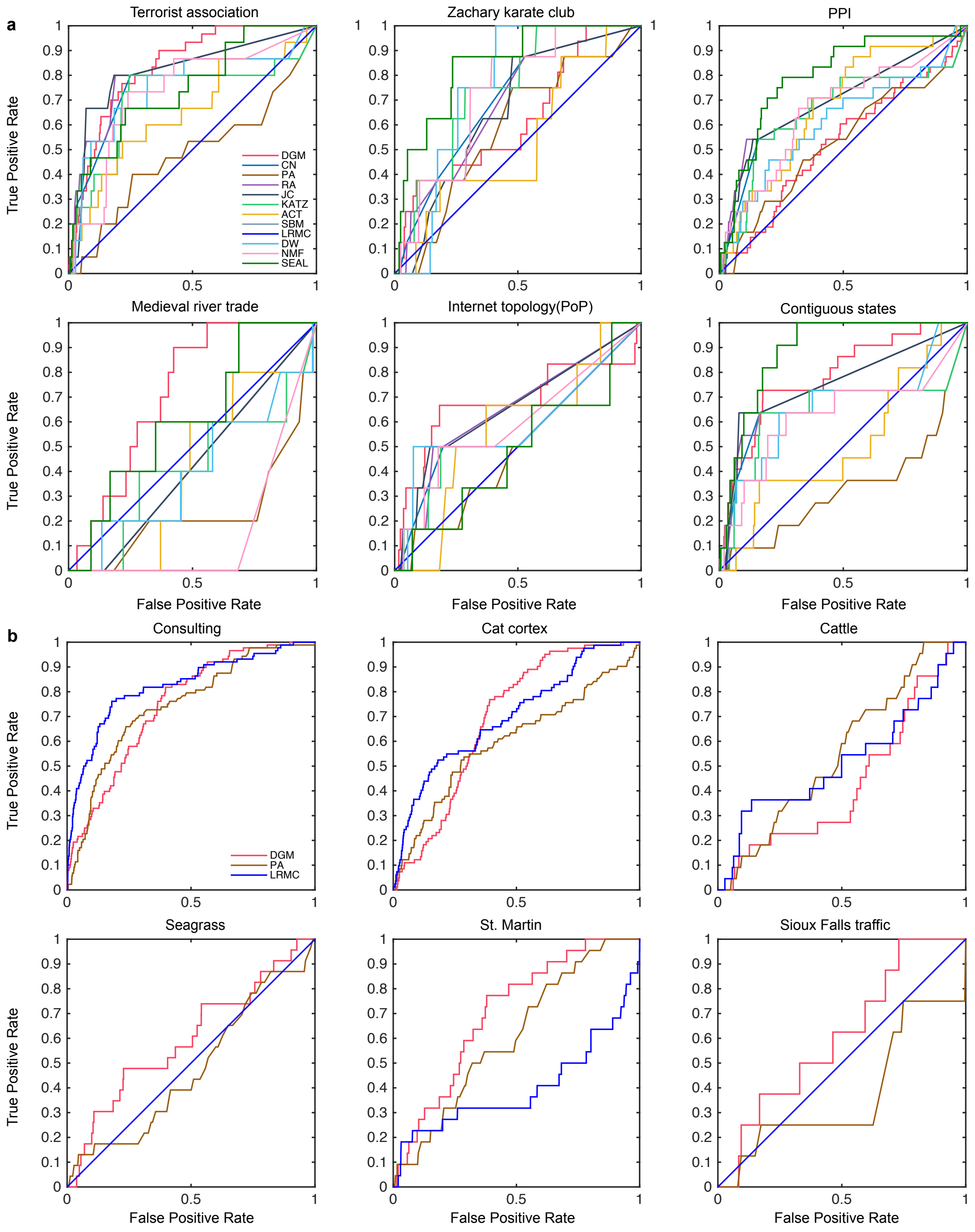
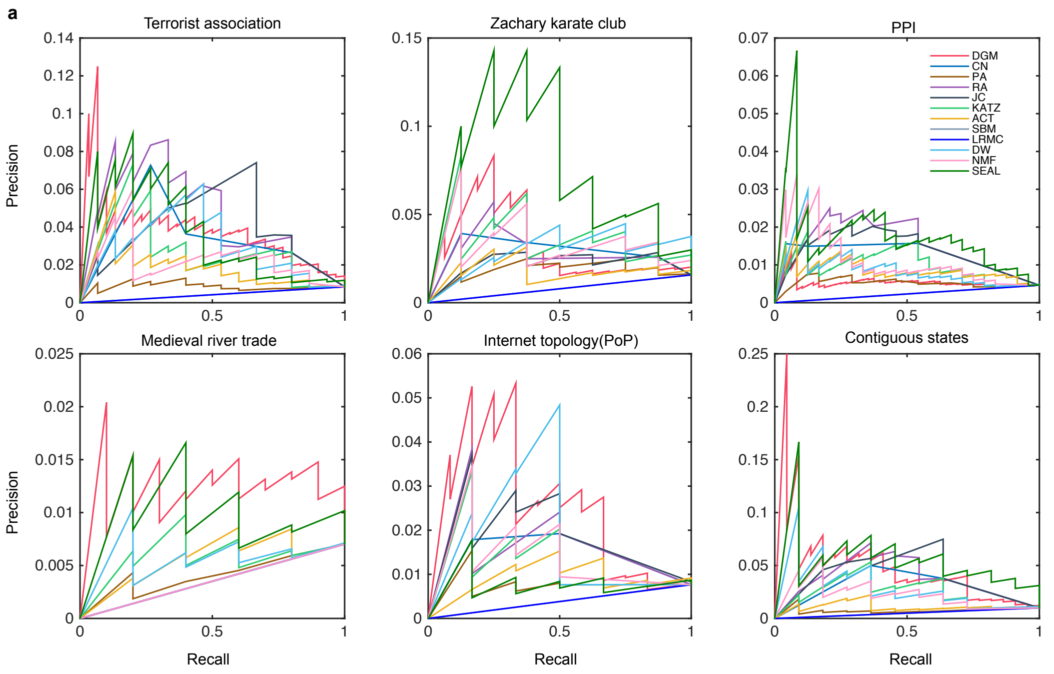


Figure S12: **ROC curves of different link prediction methods in various real-world networks,** related to Figure 4. We randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% as training set.



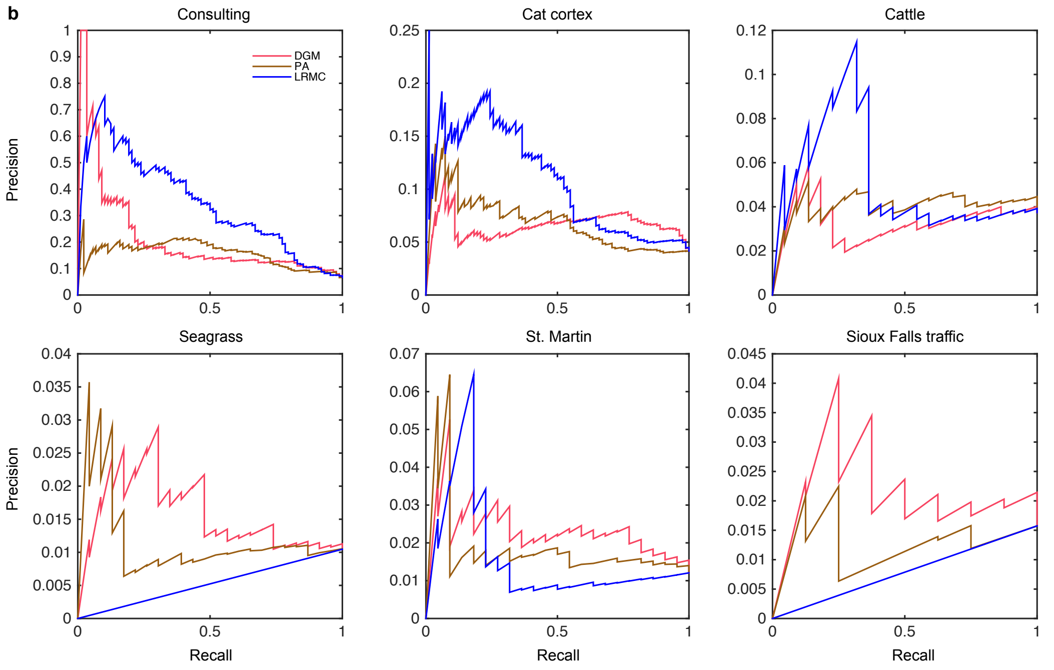


Figure S13: **Precision-Recall curves of different link prediction methods in various real-world networks (see SI for the details of each network)**, related to Figure 4. We randomly divide the links into two parts: a fraction of $f = 0.1$ links chosen as the probe set and the remaining $1 - f$ as training set.

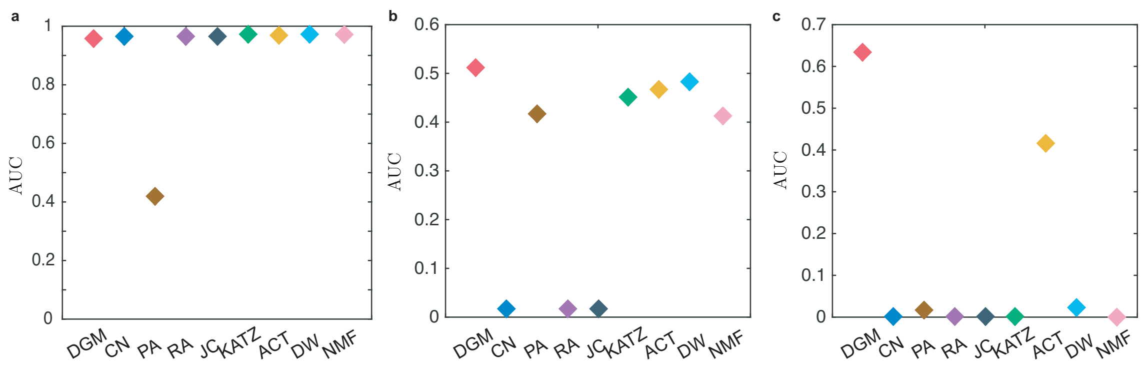


Figure S14: **Performance of DGM-based link prediction and other scalable link prediction methods on modeled networks,** related to Figure 5. **a,** A synthetic stochastic block model network with 1056 nodes (with 12 clusters and 88 nodes for each, the connection probability of within cluster is $p_{\text{in}} = 0.5$ and between clusters is $p_{\text{out}} = 0.05$). **b,** random graph with connection probability $p = 0.01$ and **c,** random graph with connection probability $p = 0.001$. The network size is $N = 440$. We randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% links as the training set.

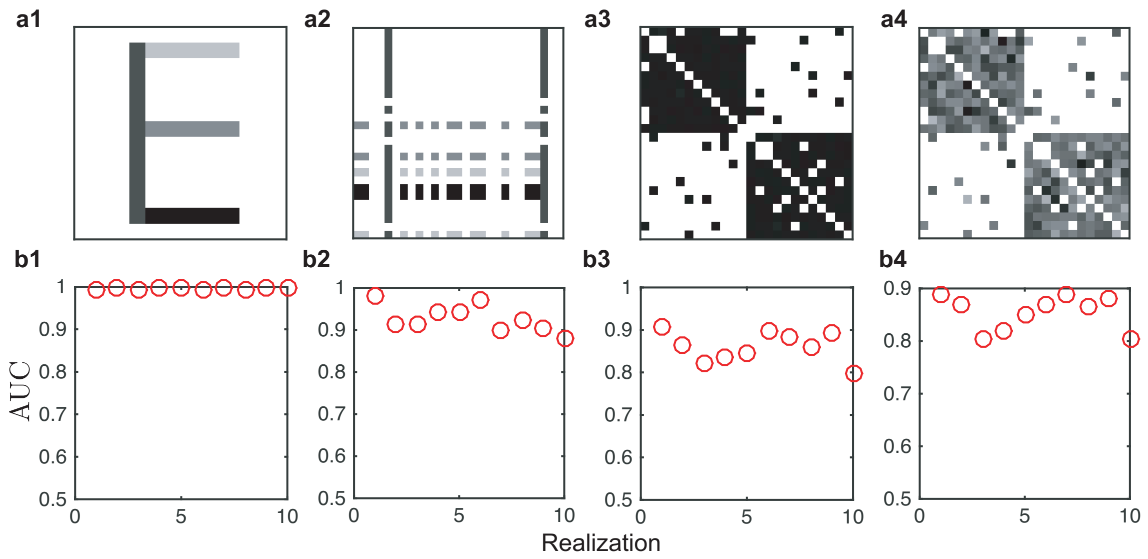


Figure S15: **DGM in link prediction of weighted networks,** related to Figures 1 and 2. **a1,** The toy model of letter **E** and there have four link weights: 0.25, 0.5, 0.75, 1.0. **a2,** The network by randomly relabeling the node in (a1). **a3,** A stochastic block model with link weights drawn from normal distribution with mean 0.5 and derivation 0.01. **a4,** A stochastic block model with link weights drawn from normal distribution with mean 0.5 and derivation 0.1. **b1-b4,** AUC of DGM in 10 different realization for the networks in (a1)-(a4). We randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% links as the training set.

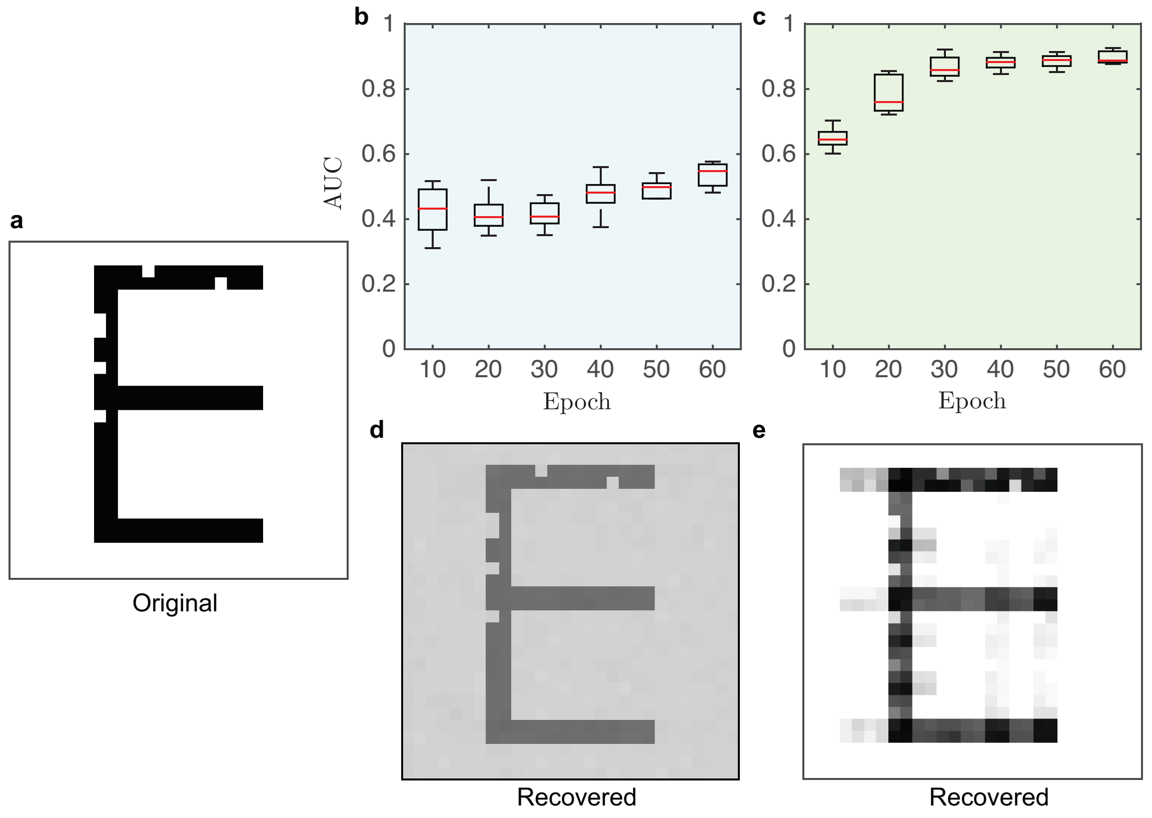


Figure S16: **The VAE-based based link prediction generally does not perform as well as the GANs-based link prediction,** related to Figure 1. As in Fig. 1 of the main text, we are testing the link prediction for a directed network whose adjacency matrix has the shape of letter **E** (with some missing pixels). We divide the links of this network (with 28 nodes and 118 links) into two parts: 5% be the probe set and the remaining 95% be the training set. We find that the AUC of VAE-based link prediction (**b**) is considerably lower than that of GANs-based link prediction (**c**). **a,** Original network. **b,** AUC of VAE-based link prediction as a function of epoch. **c,** AUC of GANs-based link prediction as a function of epoch. **d,** Recovered network by VAE-LP. **e,** Recovered network by GANs-based link prediction.

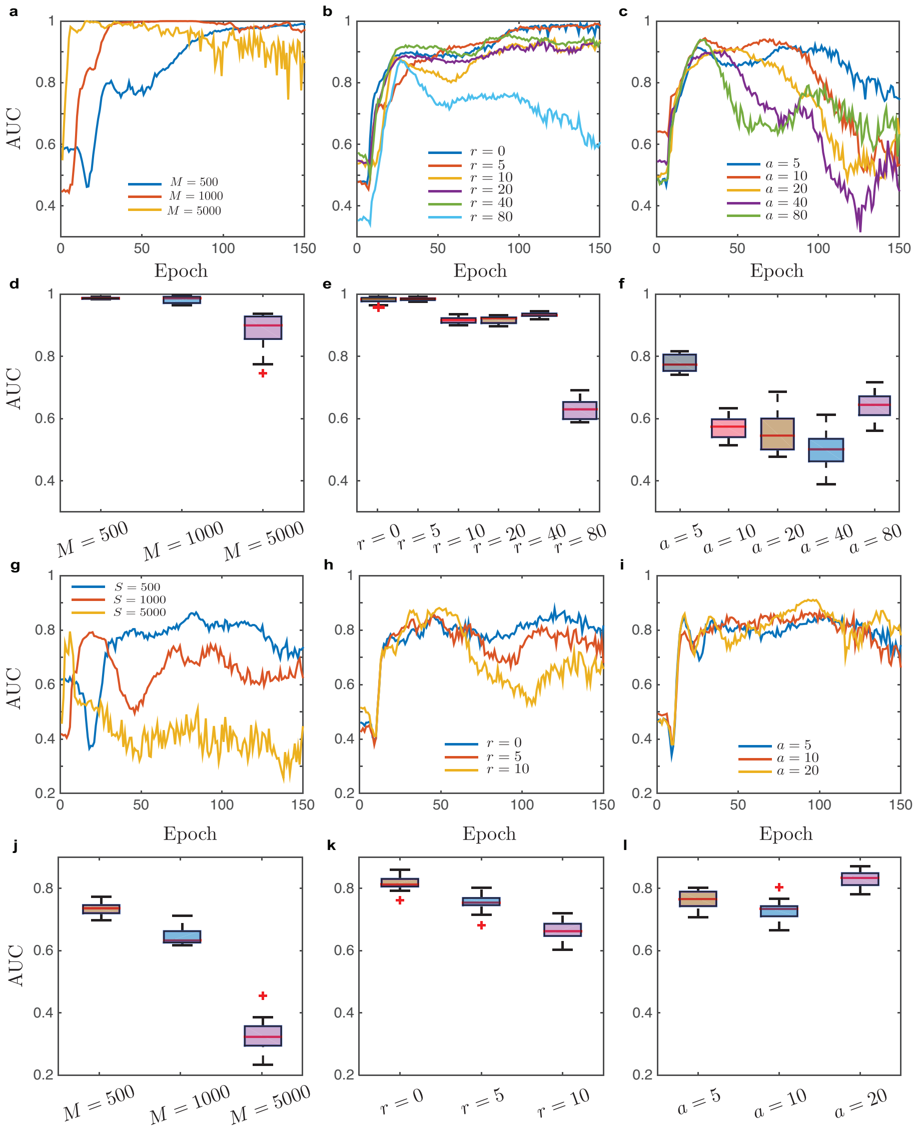


Figure S17: **Effect of hyperparameters in GANs on the performance of link prediction,** related to Figure 1. To illustrate the effect of hyperparameters on the AUC, we perform GANs-based link prediction for on two networks. We divide the links of each network into two parts: 10% be the probe set and the remaining 90% be the training set. We perturb the training set by removing r (or adding a) links at random in M different ways to obtain the input dataset, which will be fed into the GANs. **a**-**f,** A directed network (of 28 nodes and 118 links) whose adjacency matrix looks like the image of letter ***E*** (with some missing pixels). AUC as a function of epoch for different input size S (**a**), number of removed links r (**b**), number of added links (**c**). AUC of the last 20 epochs for different input size S **d**,, number of removed links r (**e**), number of added links (**f**). **g**-**l,** An undirected random network (of 48 nodes and 40 links). AUC as a function of epoch for different input size M (**g**), number of removed links r (**h**), number of added links (**i**). AUC of the last 20 epochs for different input size S (**j**), number of removed links r (**k**), number of added links (**l**).

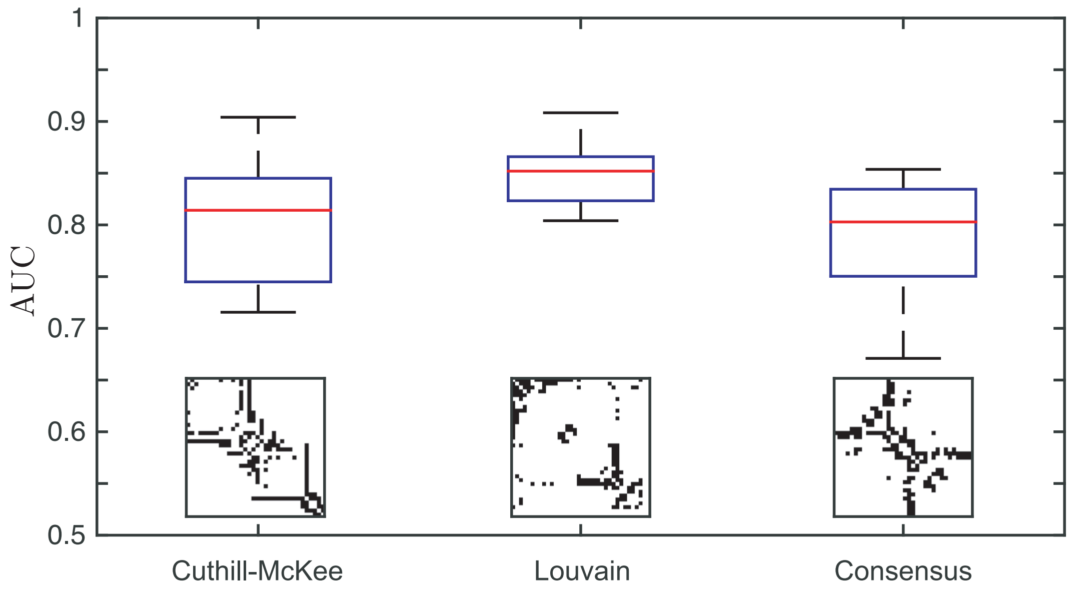


Figure S18: **Performance of our DGM-based link prediction on an example network (Zachary karate club) labeled by three different methods: Cuthill-McKee algorithm, Louvain method, and our own consensus-based method,** related to Figure 4. Insets show the adjacency matrices reordered by the three methods. Boxplots of AUC are calculated from 10 different realizations. For each realization, we randomly divide the links into two parts: a fraction of 10% links chosen as the probe set and the remaining 90% links as the training set.

Part IV

Supplemental References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. et al. (2016), 'Tensorflow: Large-scale machine learning on heterogeneous distributed systems', *arXiv preprint arXiv:1603.04467* .

Al Hasan, M., Chaoji, V., Salem, S. and Zaki, M. (2006), Link prediction using supervised learning, *in* 'SDM06: workshop on link analysis, counter-terrorism and security'.

Arenas, A., Fernandez, A. and Gomez, S. (2008), 'Analysis of the structure of complex networks at different resolution levels', *New journal of physics* **10**(5), 053039.

Arjovsky, M. and Bottou, L. (2017), 'Towards principled methods for training generative adversarial networks', *arXiv preprint arXiv:1701.04862* .

Arjovsky, M., Chintala, S. and Bottou, L. (2017), 'Wasserstein gan', *arXiv preprint arXiv:1701.07875* .

Arora, S. and Zhang, Y. (2017), 'Do gans actually learn the distribution? an empirical study', *arXiv preprint arXiv:1706.08224* .

Baird, D., Luczkovich, J. and Christian, R. R. (1998), 'Assessment of spatial and temporal variability in ecosystem learning from subgraphs, embeddings, and attributes for link predictions of the st marks national wildlife refuge, apalachee bay, florida', *Estuarine, Coastal and Shelf Science* **47**(3), 329–349.

Barabási, A.-L. and Albert, R. (1999), 'Emergence of scaling in random networks', *Science* **286**(5439), 509–512.

Barzel, B. and Barabási, A.-L. (2013), 'Network link prediction by global silencing of indirect correlations', *Nature biotechnology* **31**(8), 720–725.

Behrisch, M., Bach, B., Henry Riche, N., Schreck, T. and Fekete, J.-D. (2016), Matrix reordering methods for table and network visualization, *in* 'Computer Graphics Forum', Vol. 35, Wiley Online Library, pp. 693–716.

Belkin, M. and Niyogi, P. (2003), 'Laplacian eigenmaps for dimensionality reduction and data representation', *Neural computation* **15**(6), 1373–1396.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. (2008), 'Fast unfolding of communities in large networks', *Journal of statistical mechanics: theory and experiment* **2008**(10), P10008.

Cao, S., Lu, W. and Xu, Q. (2015), Grarep: Learning graph representations with global structural information, *in* 'Proceedings of the 24th ACM International on Conference on Information and Knowledge Management', ACM, pp. 891–900.

Chen, B., Hua, Y., Yuan, Y. and Jin, Y. (2018), 'Link prediction on directed networks based on auc optimization', *IEEE Access* **6**, 28122–28136.

Chen, B., Li, F., Chen, S., Hu, R. and Chen, L. (2017), 'Link prediction based on non-negative matrix factorization', *PloS one* **12**(8), e0182968.

Christian, R. R. and Luczkovich, J. J. (1999), 'Organizing and understanding a winter? seagrass foodweb network through effective trophic levels', *Ecological modelling* **117**(1), 99–124.

Clauset, A., Moore, C. and Newman, M. E. (2008), 'Hierarchical structure and the prediction of missing links in networks', *Nature* **453**(7191), 98–101.

Community Detection Toolbox – Athanasios (2014).
URL: *http://www.mathworks.com/matlabcentral/fileexchange/45867-community-detection-toolbox*

Convolutional Variational Autoencoder – Jason Ramapuram (2016).
URL: *https://github.com/jramapuram/CVAE*

Cross, R. L. and Parker, A. (2004), *The hidden power of social networks: Understanding how work really gets done in organizations*, Harvard Business Review Press.

Cuthill, E. and McKee, J. (1969), Reducing the bandwidth of sparse symmetric matrices, *in* 'Proceedings of the 1969 24th national conference', ACM, pp. 157–172.

Deutsch, S. B. and Martin, J. J. (1971), 'An ordering algorithm for analysis of data arrays', *Operations Research* **19**(6), 1350–1362.

Duan, L., Ma, S., Aggarwal, C., Ma, T. and Huai, J. (2017), 'An ensemble approach to link prediction', *IEEE Transactions on Knowledge and Data Engineering* **29**(11), 2402–2416.

Elmqvist, N., Do, T.-N., Goodell, H., Henry, N. and Fekete, J.-D. (2008), Zame: Interactive large-scale graph visualization, *in* 'Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific', IEEE, pp. 215–222.

Facebook wall posts network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/facebook-wosn-wall*

Friedman, N., Getoor, L., Koller, D. and Pfeffer, A. (1999), Learning probabilistic relational models, *in* 'IJCAI', Vol. 99, pp. 1300–1309.

Friendly, M. (2002), 'Corrgrams: Exploratory displays for correlation matrices', *The American Statistician* **56**(4), 316–324.

Glover, F. (1986), 'Future paths for integer programming and links to artificial intelligence', *Computers & operations research* **13**(5), 533–549.

Goodfellow, I. (2016), 'Nips 2016 tutorial: Generative adversarial networks', *arXiv preprint arXiv:1701.00160* .

Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y. (2016), *Deep learning*, Vol. 1, MIT press Cambridge.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014), Generative adversarial nets, *in* ‘Advances in neural information processing systems’, pp. 2672–2680.
Google+ network dataset – KONECT (2017).
URL: *http://konect.uni-koblenz.de/networks/ego-gplus*
Guimerà, R. and Sales-Pardo, M. (2009), ‘Missing and spurious interactions and the reconstruction of complex networks’, *Proceedings of the National Academy of Sciences* **106**(52), 22073–22078.
Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A. C. (2017), Improved training of wasserstein gans, *in* ‘Advances in Neural Information Processing Systems’, pp. 5769–5779.
Hagberg, A., Swart, P. and S Chult, D. (2008), Exploring network structure, dynamics, and function using networkx, Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).
Hamilton, W., Ying, Z. and Leskovec, J. (n.d.), Inductive representation learning on large graphs, *in* I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds, ‘Advances in Neural Information Processing Systems 30’, Curran Associates, Inc., pp. 1024–1034.
URL: *http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf*
Hand, D. J. (2010), ‘Statistical analysis of network data: Methods and models by eric d. kolaczyk’, *International Statistical Review* **78**(1), 135–135.
Harel, D. and Koren, Y. (2002), Graph drawing by high-dimensional embedding, *in* ‘International symposium on graph drawing’, Springer, pp. 207–219.
Hartigan, J. A. (1972), ‘Direct clustering of a data matrix’, *Journal of the american statistical association* **67**(337), 123–129.
He, K. and Sun, J. (2015), Convolutional neural networks at constrained time cost, *in* ‘Proceedings of the IEEE conference on computer vision and pattern recognition’, pp. 5353–5360.
Heckerman, D., Meek, C. and Koller, D. (2007), ‘Probabilistic entity-relationship models, prms, and plate models’, *Introduction to statistical relational learning* pp. 201–238.
Holland, P. W., Laskey, K. B. and Leinhardt, S. (1983), ‘Stochastic blockmodels: First steps’, *Social networks* **5**(2), 109–137.
Hu, Z., Yang, Z., Salakhutdinov, R. and Xing, E. P. (2017), ‘On unifying deep generative models’, *arXiv preprint arXiv:1706.00550* .
Hulovatyy, Y., Solava, R. W. and Milenković, T. (2014), ‘Revealing missing parts of the interactome via link prediction’, *PloS one* **9**(3), e90073.
Jaccard, P. (1901), ‘Étude comparative de la distribution florale dans une portion des alpes et des jura’, *Bull Soc Vaudoise Sci Nat* **37**, 547–579.
Jeh, G. and Widom, J. (2002), Simrank: a measure of structural-context similarity, *in* ‘Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining’, ACM, pp. 538–543.
Katz, L. (1953), ‘A new status index derived from sociometric analysis’, *Psychometrika* **18**(1), 39–43.
Kendall, D. G. (1963), ‘A statistical approach to flinders petries sequence-dating’, *Bulletin of the International Statistical Institute* **40**(2), 657–681.
Knight, S., Nguyen, H. X., Falkner, N., Bowden, R. and Roughan, M. (2011), ‘The internet topology zoo’, *IEEE Journal on Selected Areas in Communications* **29**(9), 1765–1775.
Knuth, D. E. (1993), *The Stanford GraphBase: a platform for combinatorial computing*, Vol. 37, Addison-Wesley Reading.
Kovács, I. A., Luck, K., Spirohn, K., Wang, Y., Pollis, C., Schlabach, S., Bian, W., Kim, D.-K., Kishore, N., Hao, T. et al. (2019), ‘Network-based prediction of protein interactions’, *Nature communications* **10**(1), 1–8.
Krebs, V. E. (2002), ‘Mapping networks of terrorist cells’, *Connections* **24**(3), 43–52.
LeBlanc, L. J., Morlok, E. K. and Pierskalla, W. P. (1975), ‘An efficient approach to solving the road network equilibrium traffic assignment problem’, *Transportation research* **9**(5), 309–318.
Leskovec, J. and Mcauley, J. J. (2012), Learning to discover social circles in ego networks, *in* ‘Advances in neural information processing systems’, pp. 539–547.
Liben-Nowell, D. (2005), An algorithmic approach to social networks, PhD thesis, Massachusetts Institute of Technology.
Liben-Nowell, D. and Kleinberg, J. (2007), ‘The link-prediction problem for social networks’, *Journal of the Association for Information Science and Technology* **58**(7), 1019–1031.
Lichtenwalter, R. N., Lussier, J. T. and Chawla, N. V. (2010), New perspectives and methods in link prediction, *in* ‘Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining’, ACM, pp. 243–252.
Liiv, I. (2010), ‘Seriation and matrix reordering methods: An historical overview’, *Statistical Analysis and Data Mining: The ASA Data Science Journal* **3**(2), 70–91.
Liu, W. and Lü, L. (2010), ‘Link prediction based on local random walk’, *EPL (Europhysics Letters)* **89**(5), 58007.
Long, B., Zhang, Z. M. and Yu, P. S. (2005), Co-clustering by block value decomposition, *in* ‘Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining’, ACM, pp. 635–640.
Lü, L., Jin, C.-H. and Zhou, T. (2009), ‘Similarity index based on local paths for link prediction of complex networks’, *Physical Review E* **80**(4), 046122.
Lü, L., Pan, L., Zhou, T., Zhang, Y.-C. and Stanley, H. E. (2015), ‘Toward link predictability of complex networks’, *Proceedings of the National Academy of Sciences* **112**(8), 2325–2330.
Lü, L. and Zhou, T. (2011), ‘Link prediction in complex networks: A survey’, *Physica A: statistical mechanics and its applications* **390**(6), 1150–1170.
Mäkinen, E. and Siirtola, H. (2005), ‘The barycenter heuristic and the reorderable matrix’, *Informatica (Slovenia)* **29**(3), 357–364.
Martinez, N. D. (1991), ‘Artifacts or attributes? effects of resolution on the little rock lake food web’, *Ecological monographs* **61**(4), 367–392.

Martínez, V., Berzal, F. and Cubero, J.-C. (2017), 'A survey of link prediction in complex networks', *ACM Computing Surveys (CSUR)* **49**(4), 69.
Newman, M. E. (2006), 'Modularity and community structure in networks', *Proceedings of the national academy of sciences* **103**(23), 8577–8582.
Opsahl, T. (2011), 'Why anchorage is not (that) important: Binary ties and sample selection', *online] http://toreopsahl.com/2011/08/12/why-anchorage-is-not-that-important-binary-tiesand-sample-selection (accessed September 2013)* .
Ou, Q., Jin, Y.-D., Zhou, T., Wang, B.-H. and Yin, B.-Q. (2007), 'Power-law strength-degree correlation from resource-allocation dynamics on weighted networks', *Physical Review E* **75**(2), 021102.
Pearson, K. (1905), 'The problem of the random walk', *Nature* **72**(1865), 294.
Pech, R., Hao, D., Pan, L., Cheng, H. and Zhou, T. (2017), 'Link prediction via matrix completion', *EPL (Europhysics Letters)* **117**(3), 38002.
Perin, C., Dragicevic, P. and Fekete, J.-D. (2014), 'Revisiting bertin matrices: New interactions for crafting tabular visualizations', *IEEE transactions on visualization and computer graphics* **20**(12), 2082–2091.
Perozzi, B., Al-Rfou, R. and Skiena, S. (2014), Deepwalk: Online learning of social representations, *in* 'Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 701–710.
Pitts, F. R. (1978), 'The medieval river trade network of russia revisited', *Social networks* **1**(3), 285–292.
Robinson, W. S. (1951), 'A method for chronologically ordering archaeological deposits', *American antiquity* **16**(4), 293–301.
Roweis, S. T. and Saul, L. K. (2000), 'Nonlinear dimensionality reduction by locally linear embedding', *Science* **290**(5500), 2323–2326.
Sarukkai, R. R. (2000), 'Link prediction and path analysis using markov chains', *Computer Networks* **33**(1), 377–386.
Scannell, J., Burns, G., Hilgetag, C., O'Neil, M. and Young, M. P. (1999), 'The connectional organization of the cortico-thalamic system of the cat', *Cerebral Cortex* **9**(3), 277–299.
Schein, M. W. and Fohrman, M. H. (1955), 'Social dominance relationships in a herd of dairy cattle', *The British Journal of Animal Behaviour* **3**(2), 45–55.
Sloan, S. (1986), 'An algorithm for profile and wavefront reduction of sparse matrices', *International Journal for Numerical Methods in Engineering* **23**(2), 239–251.
Sloan, S. (1989), 'A fortran program for profile and wavefront reduction', *International Journal for Numerical Methods in Engineering* **28**(11), 2651–2679.
Srinivasan, B. and Ribeiro, B. (n.d.), 'On the equivalence between node embeddings and structural graph representations'.
Sternin, H. (1965), Statistical methods of time sequencing., Technical report, STANFORD UNIV CALIF DEPT OF STATISTICS.
Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J. and Mei, Q. (2015), Line: Large-scale information network embedding, *in* 'Proceedings of the 24th International Conference on World Wide Web', International World Wide Web Conferences Steering Committee, pp. 1067–1077.
Tieleman, T. and Hinton, G. (2012), 'Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude', *COURSERA: Neural networks for machine learning* **4**(2), 26–31.
Tong, H., Faloutsos, C. and Pan, J.-Y. (2006), 'Fast random walk with restart and its applications', *in:Proceedings of the 6th International Conference on Data Mining* .
Viswanath, B., Mislove, A., Cha, M. and Gummadi, K. P. (2009), On the evolution of user interaction in Facebook, *in* 'Proc. Workshop on Online Social Networks', pp. 37–42.
Wang, D., Cui, P. and Zhu, W. (2016), Structural deep network embedding, *in* 'Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 1225–1234.
White, H. C., Boorman, S. A. and Breiger, R. L. (1976), 'Social structure from multiple networks. i. blockmodels of roles and positions', *American journal of sociology* **81**(4), 730–780.
Wilkinson, L. (2006), *The grammar of graphics*, Springer Science & Business Media.
Yeh, R., Chen, C., Lim, T. Y., Hasegawa-Johnson, M. and Do, M. N. (2016), 'Semantic image inpainting with perceptual and contextual losses', *arXiv preprint arXiv:1607.07539* .
Yu, K., Chu, W., Yu, S., Tresp, V. and Xu, Z. (2007), Stochastic relational models for discriminative link prediction, *in* 'Advances in neural information processing systems', pp. 1553–1560.
Zachary, W. W. (1977), 'An information flow model for conflict and fission in small groups', *Journal of anthropological research* **33**(4), 452–473.
Zhang, M. and Chen, Y. (2018), 'Link prediction based on graph neural networks', *arXiv preprint arXiv:1802.09691* .
Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R. and Zhang, Y.-C. (2010), 'Solving the apparent diversity-accuracy dilemma of recommender systems', *Proceedings of the National Academy of Sciences* **107**(10), 4511–4515.
Zhou, T., Lü, L. and Zhang, Y.-C. (2009), 'Predicting missing links via local information', *The European Physical Journal B-Condensed Matter and Complex Systems* **71**(4), 623–630.