Robust Network Enhancement from Flawed Networks

Jiarong Xu, Yang Yang*, Chunping Wang, Zongtao Liu, Jing Zhang, Lei Chen and Jiangang Lu

Abstract—Network data in real-world tends to be error-prone due to incomplete sampling, imperfect measurements, etc.; this in turn results in inaccurate results when performing network analysis or modeling, such as node classification and link prediction, on these flawed networks. In this paper, we aim to reconstruct a reliable network from a flawed, undirected, unweighted network, a process referred to network enhancement. More specifically, network enhancement aims to detect the noisy links that are observed in the network but should not exist in the real world, as well as to predict the missing links that do indeed exist in the real world yet remain unobserved. While some attempts have been made to detect either noisy links or missing links, few of these works have considered unifying these two tasks, even though they are inter-dependent and capable of mutually boosting each others' performance. In this paper, we therefore propose E-Net, an end-to-end graph neural network model, to leverage the mutual influence of these two tasks in order to achieve both goals more effectively. On one hand, detecting noisy links can benefit the performance of missing link prediction, while on the other hand, predicting missing links can provide indirect supervision for detecting noisy link detection when the labels of these noisy links are unavailable. Moreover, by proposing a subgraph extraction mechanism based on random walk with restart, the model can be scaled up to large networks and is able to preserve the local and global structural characteristics. The experimental results on several types of large networks demonstrate that the proposed model obtains an improvement of 10.7% on average in terms of F1 for predicting missing links, along with an average of 3.7% improvement in terms of precision for detecting noisy links compared with the state-of-the-art baselines.

Index Terms—Social networks, Network enhancement, Network robustness, Graph neural network

1 Introduction

Networks are ubiquitous and play a pivotal role in many real-world domains, including social network analysis, bioinformatics, chemistry, program analysis, etc. These networks offer rich topological features and generic connectivity patterns that can help us better understand the relational data. However, most networks obtained in the real world are error-prone and structurally flawed due to incomplete sampling [1], imperfect measurements [2], [3], individual non-response and dropout [4], etc. This will inevitably introduce many types of errors, including erroneous, ambiguous and redundant information. Generally speaking, these flawed structures can be caused by flawed links and flawed nodes; we aim to solve the flawed link problem in this paper. The flawed links can be categorized as *noisy links* and *missing links*. In more detail, *noisy links* are those that are observed in the constructed networks but do not actually exist in the real world (false positives); for example, links in a mobile network built upon calls between delivery drivers and customers are most likely to be noisy links, as they do not represent actual "social" relationships. Moreover, *miss-*

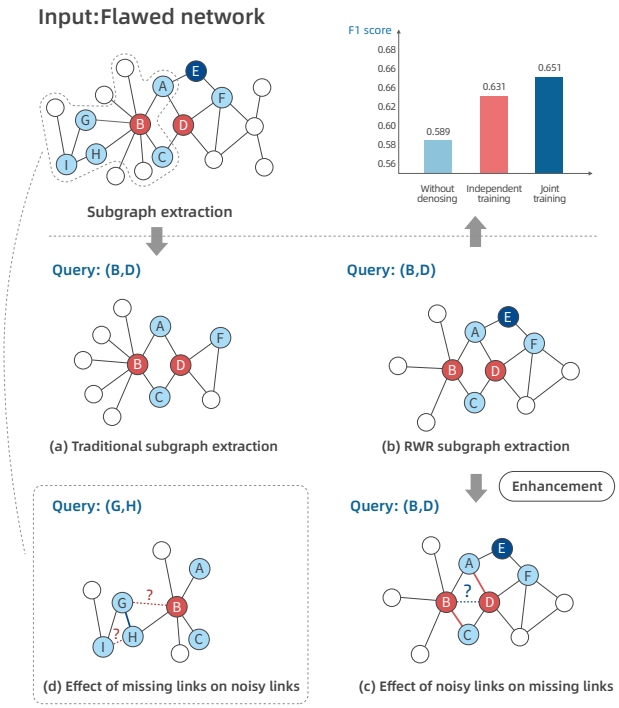


Fig. 1: Illustration of predicting missing links and detecting noisy links in a unified framework. The dashed lines denote the missing links to be predicted or the noisy links to be detected given the existing links (denoted by the solid lines).

- *J. Xu and J. Lu are with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou, Zhejiang, China, and are also with Zhejiang Laboratory, Hangzhou 311121, China. E-mail: xujr, lujg@zju.edu.cn.*
- *Y. Yang is with the College of Computer Science, Zhejiang University, Hangzhou, Zhejiang, China. E-mail: yangya@zju.edu, corresponding author.*
- *Z. Liu is with Alibaba Group. E-mail: zongtao.lzt@alibaba-inc.com.*
- *J. Zhang is with the Computer Science Department, Renmin University of China. E-mail: zhang-jing@ruc.edu.cn.*
- *C. Wang and L. Chen are with FinVolution Group. E-mail: {wangchunping02, chenlei04}@xinye.com.*

Manuscript received April 04, 2020; revised July 10, 2020.

ing links are those that do indeed exist in the real world, but are unobserved in the constructed network (false negatives). For example, if we simply establish a link between two persons in a mobile network according to the call interactions between them, some actual "social" relationships will be discarded due to the infrequent nature of these interactions. The obtained flawed networks can adversely impact how these networks are interpreted and damage the information diffusion process, resulting in misleading conclusions [1]. Hence, it is of vital and practical importance to reconstruct reliable networks from flawed networks — i.e., to remove the noisy links and complete the missing links before network analysis and network modeling is performed.

One straightforward way to deal with this problem is to adopt heuristic metrics, such as the number of common neighbors, Jaccard Coefficient, Preferential Attachment, Adamic-Adar, etc. [5], [6], [7], to simultaneously predict missing links and detect noisy links, i.e., to complete a missing link if the score measured by one of the above metrics is quite high and remove a noisy link if the score is significantly low. Some existing works have proposed additional metrics, such as node correlation [8] and link reliability [9], and used them to identify the missing and noisy links together using the method described above. The most relevant work is SEAL [12], which uses a graph neural network (GNN) to learn a suitable heuristic for link prediction (). Inspired by this work, we utilize a similar GNN structure to learn such a heuristic. However, most existing works have ignored the mutual influence of these two kinds of links, even though cooperation between the two tasks can lead to improved capability for both. We present an example in Figure 1 to illustrate the necessity of this mutual influence. In all subfigures, the dashed lines denote the links to be predicted or detected given the existing links (denoted by the solid lines). On one hand, we demonstrate the effect of noisy links on the missing links in Figure 1(c): when predicting the missing link between nodes B and D, if the links A-D and B-C have already been identified as noisy links and removed beforehand, the connections between B and D will be weakened, which will decreases the likelihood of a link being created between B and D. On the other hand, we demonstrate the effect of missing links on the noisy links in Figure 1(d) (our query is (G, H) in this example): when detecting whether or not the link B-G or H-I is noisy, if the link G-H has already been predicted to be a missing link and added beforehand, the connections between B and G or those between H and I will be strengthened, which will decrease the likelihood of the links B-G or H-I being removed. This examples indicate that the mutual influence between the two tasks can boost the performance of each task. Thus, *the main challenge to be addressed in this paper is that of how to capture the mutual influence between the missing links and noisy links*.

In addition, super-large networks prevent us from leveraging the entire network structure to infer the relationships (i.e., missing or noisy relationships) between two queried nodes. To address this issue, many researchers extracted subgraphs of the two queried nodes and inferred their relationships based on the subgraphs [10], [11], [12]. These approaches typically involve the simple extraction of the one-hop or two-hop neighbors of the two queried nodes and the relationships among them in order to compose the subgraph. However, this subgraph may still be extremely large when some hub nodes are traversed. For example, in figure 1(a), a large number of neighboring nodes will be expanded when node B is traversed. Moreover, the expanded one-hop or two-hop neighbors lose the global structural characteristics. For example, figure 1(a) presents the one-hop subgraph of the queried nodes B and D. If A-D and B-C are identified as noisy links and removed before the relationship between B and D is predicted, B and D will be disconnected in the one-hop subgraph. However, if node E (which carries more global structural characteristics than the one-hop neighbors) is included in the subgraph (cf. figure 1(b)), there will be a path B-A-E-F-D that connects B and D even if A-D and B-C are removed. Thus, *another challenge to be dealt with in this paper is that of how to extract a small-sized subgraph that contains both the local and graph structural characteristics*.

In summary, the present paper addresses the above challenges and make the following contributions:

- We propose a unified model to jointly identify noisy links and missing links due to their inter-dependence and mutual performance boosting. More specifically, we propose an end-to-end dedicated graph neural network-based model, named the Enhanced Network model (abbreviated as E-Net) which can capture these mutual influence between the two tasks. On one hand, the denoised network cleaned by the noisy link detector can benefit the performance of missing link prediction. On the other hand, the objective of missing link prediction can provide indirect supervision for noisy link detection when the labels of the noisy links cannot be easily obtained. This approach will produce a model with high resistance to noise.
- To improve the computational efficiency, we propose a RWR subgraph extraction approach based on random walk with restart to extract the subgraphs for each of the two queried nodes; this enables our model to scale up to large networks while capturing both the global and local structural characteristics.
- We conduct extensive experiments on several types of large networks. Our experimental results demonstrate that when the two tasks are trained jointly by our model, performance improvements can be achieved in terms of F1 score as high as 10.5% compared with the model that does not denoise the networks; moreover, an improvement as high as 2.6% compared with the model that predicts the missing links and detects the noisy links independently (Cf. Figure 1(e) for details).

Organization. The remainder of this paper is organized as follows. We first formulate the problem and provide the necessary definitions. We then introduce the proposed model, including the subgraph extraction procedure, the model architecture and the learning objectives. Next, we present the experimental settings and the results. Finally, we review some related works and conclude this paper.

2 Problem Formulation

Before introducing our method, we first provide the necessary definitions and formulate the problem in this section.

The key notations are listed in Table 1. In most cases, we use lower-case letters to denote scalars (e.g., l), upper-case letters for sets (e.g., V and E), bold lower-case letters for column vectors (e.g., $\mathbf{x}$), and bold upper-case letters to represent matrices (e.g., $\mathbf{A}$). When indexing the matrices, $\mathbf{A}_{ij}$ denotes the element at the i-th row and the j-th column, while $\mathbf{A}_i$ denotes the vector at the i-th row. There are also some exceptions.

In practice, we often obtain a network by sampling the nodes and edges from a complete network, or creating a network by following certain heuristic rules. It is therefore inevitable that the obtained network will contain incorrect information, especially at the edge level (when compared with edges, it is usually much easier to obtain a reliable node set) [1].

Definition 1. *Flawed Network.* We define an incomplete network with unreliable edge set as a *flawed network*, which contains two major types of flawed links: noisy links and missing links. Here, *noisy links* are defined as edges that are observed in the network but do not exist in the real world, while *missing links* are defined as the edges that do indeed exist in the real world, but are unobserved in the network.

More formally, we represent a flawed network as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes with $|\mathcal{V}|$ nodes, while $\mathcal{E}$ is the set of edges with $|\mathcal{E}|$ edges. We further denote $\mathcal{A}$ as the adjacency matrix of $\mathcal{G}$ and $\mathcal{D}$ as the degree matrix of $\mathcal{A}$. We augment $\mathcal{G}$ with the node attribute matrix $\mathcal{X}$ if nodes have certain attributes in particular applications.

However, conducting network analysis directly on flawed networks may produce misleading results. It is therefore vital to study how the input flawed network $\mathcal{G}$ can be enhanced; this process can be formally formulated as follows.

Problem 1. *Network Enhancement.* Given a flawed network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a state matrix $\mathcal{Y} = [\mathcal{Y}_{ij}]_{i,j=1...|\mathcal{V}|}$, where $\mathcal{Y}_{ij} \in \{1, 0, ?\}$ denote confirmed existing, confirmed absent and uncertain links respectively; our goal is to infer the true value (1 or 0) of the uncertain links ($\mathcal{Y}_{ij} = ?$) in $\mathcal{Y}$, i.e., to predict the missing links ($\epsilon_{ij} \notin \mathcal{E}$ and $\mathcal{Y}_{ij} = ?$) and to detect the noisy links ($\epsilon_{ij} \in \mathcal{E}$ and $\mathcal{Y}_{ij} = ?$). These two tasks (missing link prediction and noisy link detection) compose the whole problem of network enhancement.

Regarding attempts to solve the network enhancement problem (i.e., detecting both missing links and noisy links), some previous attempts have been made in totally unsupervised and supervised settings, while no such works exist in the partially supervised setting. Moreover, in most real-world scenarios, it is difficult to obtain the labels of both the noisy links and the missing links to conduct supervised network denoising. Notably, such labels can provide us valuable information enabling us to get more accurate results. Thus, we focus here on a more practical setting: namely, partially supervised network enhancement, i.e., where we can only obtain the labels of one kind of links. To reduce the difficulty of obtaining labels, in this work, we suppose that the labels of the missing links ($\epsilon_{ij} \notin \mathcal{E}$ and $\mathcal{Y}_{ij} = 1$) can be obtained or generated in some way (e.g., we hold out some existing links as the missing links), while the labels of the noisy links ($\epsilon_{ij} \in \mathcal{E}$ and $\mathcal{Y}_{ij} = 0$) are not available. We then leverage the labels of the missing links to supervise both the missing link prediction and the noisy link detection; i.e., the links that satisfy $\epsilon_{ij} \notin \mathcal{E}$ and $\mathcal{Y}_{ij} = 1$ serve as the training data of our model, where the noisy link detection is encapsulated as a component in the model without direct supervisions. Formally, we use $q \in Q$, where $Q = \{(\nu_i, \nu_j) | \epsilon_{ij} \notin \mathcal{E}, \mathcal{Y}_{ij} = ?, \nu_i \in \mathcal{V}, \nu_j \in \mathcal{V}\}$, to denote a query pair. In the following parts, a symbol marked with the superscript (m) denotes that this symbol corresponds to the m-th query, i.e., $Q = \{q^{(m)}\}_{m=1}^{M}$.

1. In this paper, we aim to solve the flawed link problem and leave the flawed node problem for the future.

TABLE 1: Description of some major notations

Notation	Description
$\mathcal{G}$, $\mathcal{G}^*$	The flawed network and the enhanced network
$\mathcal{V}$	The node set of $\mathcal{G}$
$\mathcal{E}$, ϵ_{ij}	The edge set of $\mathcal{G}$ and the edge between node i and node j in $\mathcal{G}$
$\mathcal{Y}$, $\mathcal{Y}_{ij}$	The true state matrix and state label of ϵ_{ij}
$\hat{\mathcal{Y}}$	The predicted state matrix
$\mathcal{X}$	The node attribute matrix of $\mathcal{G}$
$\mathcal{A}$	The adjacency matrix of $\mathcal{G}$
$\mathcal{D}$	The degree matrix of $\mathcal{A}$
G	The input local subgraph
V, v_i	The node set of G and node i in G
E, e_{ij}	The edge set of G and the edge between v_i and v_j in G
$\mathbf{T}$, $\mathbf{t}_i$	The relative position matrix and the position label of v_i
$\mathbf{X}$, $\mathbf{x}_i$	The node attribute matrix of G and the attribute vector of v_i
$\mathbf{A}$, $\mathbf{A}^*$	The adjacency matrix and the denoised weight matrix of G
$\mathbf{L}^*$	The Laplacian matrix of A^*
$\mathbf{p}$	The proximity vector of the starting node in RWR
$s(\cdot, \cdot)$	The noise scoring function
q, $q^{(m)}$	The query of the two nodes and the m-th query
$\boldsymbol{Z}^l$	The output of the l-th denoising graph convolution layer
$\boldsymbol{H}$	The subgraph representation
d^l	The output dimension of the l-th graph convolution layer
L	The number of graph convolution layer

3 Our Approach

In this section, we first introduce our subgraph extraction approach, which paves the way for our subsequent network enhancement methods. Then, given a query q and its corresponding extracted subgraph G, the first objective is to predict whether we should create a link between the two nodes in the query, while the second objective is to detect the noisy links in the subgraph. We encapsulate these two objectives in our proposed E-Net model, an end-to-end graph neural network architecture, in order to capture the mutual influence of the two objectives. The overall model architecture is illustrated in Figure 2.

3.1 RWR Subgraph Extraction

The first step in our approach involves extracting a subgraph for each pair of the queried nodes to represent their structural characteristics. We predict the relationship of the two queried nodes based on the extracted subgraph rather than the entire network; this is because it is costly to

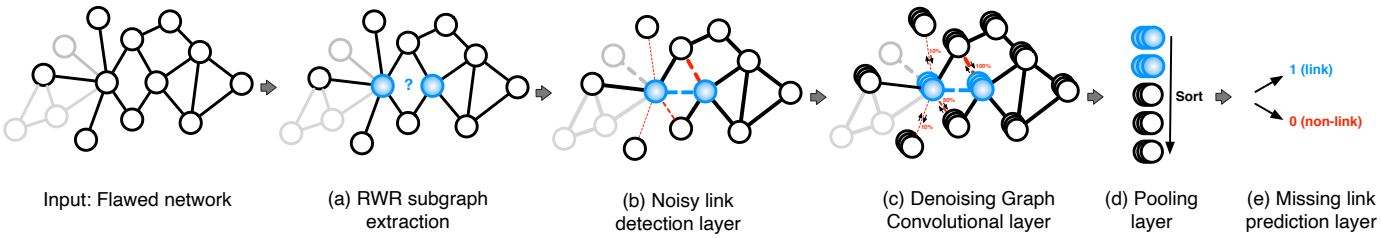


Fig. 2: E-Net model architecture. The dashed lines denote uncertain links ($\mathcal{Y}_{ij}$ =?), while the thickness of the links corresponds to their reliability.

compute for the entire network, and may even be infeasible to complete such a computation when the entire network is too large. Some studies [10], [11], [12] have verified that computing on local subgraphs can provide a good approximation of a wide range of heuristic network metrics, such as Common Neighbors, Jaccard Coefficient, Preferential Attachment, etc., which are computed on the entire network. Another theoretical study [13] has proven that the PageRank algorithm computed on the entire network has a bounded approximation error under the condition of a n-hop subgraph. Moreover, since the input graph is noisy, utilizing global information from the entire graph may inevitably introduce error messages from distant parts of the network, potentially resulting in misleading predictions. Thus, we consider it unwise to take too many global metrics into consideration. Inspired by these studies, we extract a local subgraph around a pair of the queried nodes, then infer the relationship between them based on the subgraph.

Some existing works have already utilized subgraphs to represent the nodes' structural information. For example, Zhang et al. [14] were the first to use local enclosing subgraphs to learn the nodes' structure information. However, these authors truncated the subgraphs to be of the same size, which resulted in some topographical information being omitted. Another work, SEAL [12], kept the complete one-hop or two-hop neighbors of the queried nodes; however, most of these extracted fixed-size subgraphs and were restricted to within the one-hop or two-hop local structures, which were narrowly focused and totally ignored the global structural characteristics. Moreover, the one-hop or two-hop subgraphs can be extremely large when certain hub nodes are traversed, which may seriously reduce the computational efficiency of the learning algorithm. Other sampling methods, such as GraphSAGE [15], uniformly sampled neighbors and thus ignored the reliability/noise of links.

To avoid the limitations of the above methods and capture the reliable local and global structural characteristics to a greater extent through sampling, we propose a RWR subgraph extraction approach based on random walk with restart (RWR) [35]. More specifically, given a pair of the queried nodes, rather than keeping all the κ-hop neighbors when κ is a fixed hop number, we conduct several random walks with restart from each of the two queried nodes respectively. The nodes and edges traversed in these random walks form the corresponding subgraph of this pair of queried nodes. Formally, we aim to find a solution to the following linear system,

$$\mathbf{p} = \lambda \mathcal{A}\mathcal{D}^{-1}\mathbf{p} + (1-\lambda)\mathbf{e} \tag{1}$$

where $\mathbf{p}$ is the proximity vector of the starting node, with $\mathbf{p}[i]$ denoting the probability of being at node i; moreover, $\mathbf{e}$ is a unit vector having $\mathbf{e}[i] = 1$ if i is the starting node (else 0). $\mathcal{D}$ is the degree matrix with each diagonal value $\mathcal{D}_{ii} = \sum_j \mathcal{A}_{ij}$. The teleport (or restart) probability $\lambda \in (0, 1]$ is a parameter that controls the probability of going to the starting node (with probability $1-\lambda$) or jumping to a randomly chosen neighbor (with probability λ), which enables both the local and global topological structures to be preserved. More specifically, the starting vector $\mathbf{e}$ allows us to preserve the node's local topological structure even in a limit distribution, while $\mathcal{A}\mathcal{D}^{-1}$ allows us to further visit their neighborhoods. The rate of decrease as we move away from the starting node can be adjusted by λ. Note that this equation is also similar to that of personalized Pagerank, where $\mathbf{I}$ characterizes the nodes' personalized preferences and is filled with real values [16].

By following the proposed sampling method, we extract multiple subgraphs for each query to avoid overfitting. More specifically, for the m-th query $q^{(m)} \in Q$, we extract c corresponding subgraphs, forming $\left\langle G_1^{(m)}, q^{(m)} \right\rangle$, $\left\langle G_2^{(m)}, q^{(m)} \right\rangle$, ..., $\left\langle G_c^{(m)}, q^{(m)} \right\rangle$ in the format of $\langle \text{subgraph}, \text{query} \rangle$ pair. The resulting $M \times c$ $\langle \text{subgraph}, \text{query} \rangle$ pairs for all the M queries then constitute our input data of the model. Intuitively, the corresponding subgraphs of a query can be treated as its surroundings, i.e., the context that provides it with the structural evidence, which capture the topological information and are thus crucial for predicting whether or not a link between the two queried nodes should be created.

The advantages of the proposed RWR subgraph extraction method can be summarized as follows. 1) the restart mechanism in the random walk algorithm (i.e., $\mathbf{e}$ in Eq. 1) enables us to capture both local and global structural information over the long term; 2) when encountering high-degree nodes, visiting only a certain proportion of their neighbors will save significantly on space and time costs (see Figure 1(a) and (b) for comparison); 3) the extraction of multiple subgraphs can prevent our model from overfitting and enhance its generalizability. Meanwhile, sampling different subgraphs from the same query is in line with popular data augmentation approaches.

Given a query pair and its corresponding extracted subgraph, we can turn the objective of predicting whether a link should exist between two nodes into a graph (subgraph) classification problem. This graph classification setting is

inductive, since the test instances will never be seen during training. This therefore situates our model in an inductive setting; what we only need is to extract the corresponding subgraphs of unseen links for inference.

3.2 Model Architecture

The second step of our approach involves jointly predicting the missing link between the two queried nodes and the noisy links included in the extracted subgraphs of these queried nodes. The key insight here is that the two objectives of predicting missing links and detecting noisy links can influence each other. For example, a noisy link will propagate noisy information to its neighbors, which may result in incorrect node-level and even graph-level representation and further degrade the performance of missing link prediction. We therefore integrate the two objectives into a carefully designed end-to-end graph neural network framework that aims to capture the correct information while reducing the impact of the flawed information.

Consider a subgraph $G = (V, E)$ extracted from $\mathcal{G}$ for a query q, where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_{ij} | v_i, v_j \in V\}$ denote the node set and edge set of subgraph G respectively, and n denotes the number of nodes in V. The corresponding adjacency matrix is denoted as $\mathbf{A} \in \mathbb{R}^{n \times n}$. The attribute matrix is denoted as $\mathbf{X}$, where $\mathbf{x}_i$ indicates the attribute vector of v_i. We further record the relative position of each node in G to the query q as additional features, as the nodes closer to the queried nodes will provide more important information. Here we apply Double-Radius Node Labeling (DRNL), proposed by [12], to calculate the relative position. More specifically, for the two queried nodes v_i and v_j, we assign the label $t = 1$ to each of them. Then, for any node $v_k \in G$ with $(d(v_k, v_i), d(v_k, v)_j)) = (1, 1)$, we assign the label $t = 2$, where $d(\cdot, \cdot)$ represents the shortest path between two nodes. Nodes with double-radius $(1, 2)$ or $(2, 1)$ get the label $t = 3$, while nodes with $(2, 2)$ get 4, and so on and so forth. We then transform the label into a one-hot vector $\mathbf{t}_i$ and concatenate it with the attribute vector $\mathbf{x}_i$ to represent the input feature of node v_i. The position label vector $\mathbf{t}_i$ of all nodes in G makes up the relative position matrix $\mathbf{T}$.

Our learning model contains a series of layers, as follows: 1) a noisy link detection layer, which aims to detect noisy links via a noise scoring function; 2) denoising graph convolution layers, which learn robust node representations by preventing or decreasing the flow of information along noisy links; 3) a pooling layer, which learns a function that maps the node-level representations to graph-level representations; 4) a missing link prediction layer, to predict whether there is a missing link between two queried nodes.

Noisy link detection layer. Efficiently choosing heuristic scores with different properties for unsupervised noisy link detection has been a difficult proposition for some time, since no supervision can be given. We therefore tactfully utilize the missing links to provide indirect supervision. We use a *noise scoring function*, $s(\cdot, \cdot)$, to measure the reliability of each link in the subgraph: the lower the score, the higher the likelihood that the target link will be a noisy link. In this paper, we instantiate the function as a weighted average of K heuristic score functions with learnable weights, as follows,

$$s(v_i, v_j) = \sum_{k=1}^{K} w_k \cdot s_k(v_i, v_j), \tag{2}$$

where $s_k(\cdot, \cdot)$ is the k-th heuristic score function to estimate the similarity between nodes, while w_k is its corresponding learnable weight, which can be indirectly guided by the objective of predicting missing links. We use Common Neighbors, Jaccard Coefficient, Preferential Attachment [18], Adamic-Adar [20], Resource Allocation [6] and the cosine similarity of the two nodes' attributes as these heuristic score functions, which can guarantee the detection capability. Although we do not have any labels for the noisy links, we can optimize the weights in the noise scoring function via the indirect supervision of the missing link prediction objective.

Given the noise scoring function $s(\cdot, \cdot)$ of each link, the subgraph G can be transformed into a denoised weight matrix $\mathbf{A}^* \in \mathbb{R}^{n \times n}$, in which the (i, j)-th entry $\mathbf{A}^*_{ij} = s(v_i, v_j) * \mathbf{A}_{ij}$. Later, we will conduct a subsequent study study on the denoised weight matrix $\mathbf{A}^*$.

Denoising graph convolutional layers. We next introduce the denoising graph convolutional layers, which can be understood as a generalization of the classical graph convolutional network. The general idea here is as follows: when performing message aggregation for a node, rather than simply absorbing the messages from all its neighbors, we reduce or prevent the aggregation of messages from noisy links. Although the Graph Attention Neural Network (GAT) [17] also allocated different levels of attention to different neighbors, the attention coefficient represents the contribution of a node to another, and the contribution of node i to node j is different from that of node j to node i. Moreover, it has been experimentally verified that GAT performed the same as or worse than GCN in noisy graphs [19]; this is because GAT introduces too many parameters when learning attention coefficients, which leads to overfitting to noise. By contrast, we have only k learnable parameters (w_k). In addition, our noise scoring is based on general heuristics to guarantee detection capability, while this is hard and unexplainable for GAT.

However, in our model, the influence of node i on node j is assumed to be equivalent to that of node j on node i, where this influence is calculated by our noise scoring function $s(\cdot, \cdot)$. Our rationale for adopting this aggregation method is that we assume noisy links exert a global influence on the entire graph; thus the denoised weight matrix $\mathbf{A}^*$ can be considered as our global weight function when conducting node aggregation.

We then use multiple denoising graph convolution layers to update the node representation. To be more specific, the layer-wise forward-pass updating of node representations can be expressed as follows,

$$\boldsymbol{Z}^{l+1} = f\left(\frac{\tilde{\mathbf{A}^*}}{\left\|\tilde{\mathbf{A}^*}\right\|_2}\boldsymbol{Z}^l\boldsymbol{W}^l\right), \tag{3}$$

where the initial embedding of each node is represented by its attribute and relative position label, i.e., $\boldsymbol{Z}^0 = [\mathbf{X}; \mathbf{T}]$. The notation $\boldsymbol{Z}^l \in \mathbb{R}^{n \times d^l}$ denotes the output embeddings of the l-th graph convolution layer. Moreover, notation $\tilde{\mathbf{A}^*} = \mathbf{A}^* + \mathbf{I}$, which means that the self-effect of each node is included when performing aggregation. Notation $\boldsymbol{W}^l \in \mathbb{R}^{d^l \times d^{l+1}}$ is a layer-specific weight matrix of trainable graph convolution parameters, while f is a nonlinear activation function. With the inclusion of $\mathbf{A}^*$, the information flow along the noisy links will be decreased or prevented, which helps us to learn more robust node representation.

Sequentially stacking too many graph convolutional layers might lead to an over-smoothing problem, which could result in the representation learned from the last graph convolutional layer being too coarse to capture the structural information [21], [22]. Therefore, rather than using only the node representations generated by the last denoising graph convolution layer, we instead use the node representations generated by each of the L denoising graph convolution layers to generate high-quality node representations. Our node representations are obtained by concatenating the output $\boldsymbol{Z}^l, l = 1, ..., L$ horizontally, as follows,

$$\boldsymbol{Z}^{1:L} := \left[\boldsymbol{Z}^1, ..., \boldsymbol{Z}^L\right], \tag{4}$$

where L represents the number of graph convolution layers and $\boldsymbol{Z}^{1:L} \in \mathbb{R}^{n \times \sum_1^L d^l}$.

Pooling layer. After the node-level representations are generated, the next step is to learn a graph-level representation based on these. Since our goal is to predict the existence of a link between the two queried nodes, the graph-level representation should be able to both capture the local structure of the queried nodes and place more emphasis on the two queried nodes.

We adopt SortPooling [23] to obtain the graph-level representation that can be compared between the different subgraphs. More specifically, we fix the dimension of the last denoising graph convolutional layer d^L as 1. Subsequently, all nodes in the subgraph can be sorted according to the continuous output values $\boldsymbol{Z}^L \in \mathbb{R}^{n \times 1}$ of the last denoising graph convolution layer. We then concatenate the node embeddings of the top $\bar{n}$ nodes as $\boldsymbol{Z} \in \mathbb{R}^{\bar{n} \times \sum_1^L d^l}$.

Moreover, since the queried nodes' attributes can provide additional information, we add additional attributes of the queried nodes into the graph embedding. Finally, the graph-level representation is obtained as follows,

$$\boldsymbol{H} = [\mathrm{CONV}\left(\boldsymbol{Z}\right); \mathbf{x}_q^{(1)}, \mathbf{x}_q^{(2)}], \tag{5}$$

where $\mathbf{x}_q^{(1)}$ and $\mathbf{x}_q^{(2)}$ represent the node attributes of the two queried nodes that serve as our additional evidence, while CONV denotes the several 1-D convolutional layers and maxpooling layers applied to the concatenated node embeddings $\boldsymbol{Z}$.

Missing link prediction layer. Finally, we aim to predict whether a missing link exists between the two queried nodes. Since we have already obtained the corresponding subgraph representation of our queried nodes, we can make prediction directly based on this representation, as follows,

$$\hat{\mathcal{Y}}_q = \mathrm{softmax}\left(\mathrm{MLP}\left(\boldsymbol{H}\right)\right), \tag{6}$$

where MLP denotes a multilayer perceptron, which is followed by a softmax layer used to get the graph classification result. In fact, these graph classification results can also be taken as the results of our missing link prediction.

3.3 Model Learning

Main objective. The primary goal of our learning procedure is to minimize the gap between the obtained probability of missing links and their ground truth. Accordingly, we use cross-entropy to minimize the following loss function, $\mathcal{L}_{ce}$,

$$\mathcal{L}_{ce} = -\sum_{m=1}^{M}\sum_{b=1}^{d^L} \mathcal{Y}_{q^{(m)}b} \cdot \ln \hat{\mathcal{Y}}_{q^{(m)}b} \tag{7}$$

where $\mathcal{Y}_{q^{(m)}}$ and $\hat{\mathcal{Y}}_{q^{(m)}}$ are, respectively, the ground truth of missing links and the predicted probability corresponding to the query $q^{(m)}$.

Auxiliary denoising objective. Recall that we obtained the denoised weight matrix $\mathbf{A}^*$ in Section 3.2, which can only be guided by the loss of the missing link prediction ($\mathcal{L}_{missing}$) in our current setting. In practice, it is difficult to learn a suitable denoised network that is supervised only by the indirect signals from the missing links. To resolve this issue, we add an auxiliary denoising objective. More specifically, we propose to minimize the graph Laplacian quadratic form on node attribute $\mathbf{X}$, which serves as our additional regularization,

$$\mathcal{L}_{denoise} = tr(\mathbf{X}^T\mathbf{L}^*\mathbf{X}), \tag{8}$$

$$\begin{aligned} tr(\mathbf{X}^T\mathbf{L}^*\mathbf{X}) &= \sum_{i=1}^{n}\sum_{e_{uv}\in E}\left(\mathbf{x}_{ui} - \mathbf{x}_{vi}\right)^2\mathbf{A}^*_{uv} \\ &= \sum_{e_{uv}\in E}\left\|x_u - x_v\right\|_2^2\mathbf{A}^*_{uv} \end{aligned} \tag{9}$$

where Eq. 9 is the Laplacian quadratic form on signal $\mathbf{X}$, i.e., a smooth graph signal model. This approach has been widely utilized to address various learning problems, such as regularization and semi-supervised learning on graphs [24], [25]. We have adopted this additional objective for the following reasons.

- **Global Smoothing.** Global smoothing is conceptually similar to the argument that smoothness is an indispensable property of model robustness [26]. The global smoothing objective used here can be considered as a penalty applied when two nodes with dissimilar attributes are close to each other in the learned denoised network $\mathbf{A}^*$. Because, according to Eq. 9, if v_i and v_j have dissimilar attributes, the link e_{ij} is more likely to be noisy, and will thus guide the entry $\mathbf{A}^*_{ij}$ to become a smaller probability.
- **Sparsity.** In most scenarios, it is desirable that the obtained graph should be a sparse one [27], [28]. Similarly, we also want our learned denoised network $\mathbf{A}^*$ to be more

sparse; i.e., some entries of $\mathbf{A}^*_{ij}$ can be reduced to zero if the edge e_{ij} has a high probability of being noisy. Note that the sparsity discussed here is a kind of weighted sparsity where each edge e_{uv} is weighted by $\|x_u - x_v\|_2^2$; i.e., it prefers to encourage the link between two nodes with dissimilar attributes to be weakened or removed. By contrast, when two nodes share similar attributes, $\|x_u - x_v\|_2^2$ will become small, meaning that $\mathcal{L}_{denoise}$ will not overpower $\mathcal{L}_{ce}$ when updating $\mathbf{A}^*$ (i.e., $\mathcal{L}_{denoise}$ will have little effect on $\mathbf{A}^*$ in this case). More specifically, when we cannot obtain node attributes and assume that $\mathbf{X} = \mathbf{I}$, the regularization term will be exactly equal to a commonly defined sparsity penalty $tr(\mathbf{L}^*) = \sum_{i \neq j} \mathbf{A}^*_{ij}$. This is because $tr(\mathbf{X}^T\mathbf{L}^*\mathbf{X}) = tr(\mathbf{L}^*\mathbf{X}\mathbf{X}^T) = tr(\mathbf{L}^*) = tr(\mathbf{I} - \mathbf{A}^*) = n - \mathrm{tr}(\mathbf{A}^*) = \sum_{i \neq j} \mathbf{A}^*_{ij}$ when $\mathbf{X} = \mathbf{I}$.

Joint loss function. We conduct joint optimization over the cross-entropy objective as well as the auxiliary denoising objective, which enables our model to capture the semantic meaning (cross-entropy loss) while retaining the noise-calibrated representation (auxiliary denoising loss). Finally, the total loss function $\mathcal{L}$ can be formulated as follows,

$$\mathcal{L} = \mathcal{L}_{ce} + \alpha \mathcal{L}_{denoise} \tag{10}$$

where $\alpha > 0$ is a positive coefficient to trade off the two losses: $\mathcal{L}_{ce}$ and $\mathcal{L}_{denoise}$.

4 Experiments

In this section, we evaluate the benefits of our proposed method with the aim of answering the following questions:

- **Q1.** Can the missing links be predicted by E-Net?
- **Q2.** Can the noisy links be detected by E-Net?
- **Q3.** Does our proposed RWR subgraph extraction strategy improve the training efficiency and effectiveness?
- **Q4.** How superb is the enhanced network reconstructed by E-Net, when evaluated with downstream task node classification?
- **Q5.** Is E-Net sensitive to the choices of hyperparameters?

More details about the datasets and baselines are introduced below. Our code is publicly available at: https://github.com/zjunet/E-Net.

4.1 Experimental Setup

Datasets. We employ five datasets for evaluation. Three are widely used citation networks in the network analysis literature, namely Cora, Citeseer and Pubmed, while the fourth is a large-scale user network dataset provided by FinVolution Group [2]. The fifth dataset is a protein network, namely PPI network, which is undirected and unweighted in nature. The statistics of the datasets are presented in Table 2.

- *User network.* This is a large-scale user network between 83,286 anonymous registered users, without any identifiable information. Here nodes are users and edges correspond to the connections by call. For each anonymous user, we can obtain their demographic information provided at registration (including age, sex, birthplace, educational level, marital status, work type and so on) as user attributes. To quantify the ability to identify missing links, we further randomly remove 10% of links, which will serve as our missing link samples. Fortunately, FinVolution Group has also provided us with some limited information regarding anomalous nodes that are identified as fraudsters, intermediaries or delivery drivers; thus, we can intuitively choose those links from anomalous nodes, as well as those with short call duration user, as our limited noisy link ground truth. The user network with the above-mentioned missing and noisy links is the input flawed network $\mathcal{G}$; after removing the simulated missing links and the identified noisy links, we obtain the real clean network $\mathcal{G}_{\text{clean}}$.
- *Citation network.* We also validate our method across three popular citation networks: Cora, Citeseer and Pubmed. The Cora and Citeseer networks mainly contain machine learning papers, while the Pubmed network contains scientific publications pertaining to diabetes. Here nodes are documents, while edges are the citation links between two documents. Each node has a human-annotated topic as the class label as well as a feature vector. For Cora and Citeseer, the feature vector is a sparse bag-of-words representation of the document, while for Pubmed, the feature vector is assigned real values that indicate the term-frequency-inverse document frequency (TFIDF) of the corresponding word from a dictionary. These nodes are all labelled to differentiate their topic categories across the three datasets. Note that these networks have been well-constructed and are widely used; therefore, in the following, we assume that each of these networks is error-free and can thus serve as our clean network $\mathcal{G}_{\text{clean}}$. Moreover, to quantify the ability to identify flawed links, we generate flawed networks by randomly removing links (creating the missing link samples), and randomly adding nonexistent links (constituting the noisy link samples). After these simulated noisy links and missing links are added, we get the input flawed network $\mathcal{G}$.
- *PPI network [61].* This is a subgraph of the Homo Sapiens Protein-Protein Interaction (PPI) network that is preprocessed in [30], where nodes are proteins and edges represent the interactions between two proteins. Each node belongs to a gene set, which serves as our node attributes. We create missing links and noisy links in the same manner as for the citation networks.

Baselines. We compare our model with several baselines:

- *HEU [6], [18], [20], [29].* We extract five popular heuristics (HEU), including the number of Common Neighbors (CN), Jaccard Coefficient, Preferential Attachment (PA) [18], Adamic-Adar (AA) [20] and Resource Allocation (RA) [6], and train a logistic regression classifier on these heuristic scores. This method only takes the graph structure into consideration.
- *ATT.* This method is based on the attributes (ATT) of queries and does not consider graph structure at all. We then train a MLP classifier on these attributes.
- *ENS.* We further create an ensemble (ENS) of the above node attributes and heuristic scores and train a MLP classifier on them.

2. FinVolution Group is a leading Fintech Corporation in China.

TABLE 2: Dataset statistics

Dataset	Type	#Nodes	#Edges	#Attributes	Noisy Rate	Missing Rate
FinV	User network	83,286	114,422	776	10.6%	10%
Cora	Citation network	2,708	5,429	1,433	10%	10%
Citeseer	Citation network	3,327	4,732	3,703	10%	10%
Pubmed	Citation network	19,717	44,338	500	10%	10%
PPI	Protein network	3,890	76,584	50	10%	10%

TABLE 3: Experimental results with standard deviation on missing link prediction.

	FinV		Cora		Citeseer		Pubmed		PPI	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
HEU	0.593 ± 0.04	0.345 ± 0.08	0.745 ± 0.03	0.561 ± 0.15	0.692 ± 0.06	0.509 ± 0.09	0.639 ± 0.05	0.424 ± 0.14	0.804 ± 0.04	0.400 ± 0.06
ATT	0.860 ± 0.04	0.598 ± 0.07	0.570 ± 0.02	0.286 ± 0.02	0.584 ± 0.07	0.286 ± 0.05	0.820 ± 0.03	0.500 ± 0.03	0.704 ± 0.05	0.388 ± 0.04
ENS	0.834 ± 0.06	0.603 ± 0.07	0.754 ± 0.05	0.420 ± 0.05	0.720 ± 0.06	0.398 ± 0.07	0.813 ± 0.02	0.489 ± 0.02	0.809 ± 0.04	0.402 ± 0.05
Node2vec	0.736 ± 0.05	0.443 ± 0.06	0.741 ± 0.09	0.406 ± 0.09	0.707 ± 0.09	0.389 ± 0.10	0.908 ± 0.07	0.609 ± 0.08	0.812 ± 0.05	0.485 ± 0.06
SEAL	-	-	0.781 ± 0.10	0.496 ± 0.17	0.772 ± 0.03	0.521 ± 0.05	0.962 ± 0.01	0.750 ± 0.03	-	-
SEAL (RWR)	0.892 ± 0.03	0.615 ± 0.03	0.808 ± 0.12	0.589 ± 0.18	0.793 ± 0.05	0.552 ± 0.06	**0.969 ± 0.02**	0.751 ± 0.04	0.831 ± 0.06	0.529 ± 0.06
GAT (RWR)	0.884 ± 0.04	0.608 ± 0.03	0.795 ± 0.12	0.535 ± 0.17	0.791 ± 0.06	0.548 ± 0.06	0.932 ± 0.04	0.642 ± 0.05	0.819 ± 0.08	0.502 ± 0.07
E-Net (n)	0.888 ± 0.04	0.611 ± 0.04	0.801 ± 0.11	0.548 ± 0.15	0.795 ± 0.06	0.552 ± 0.06	0.945 ± 0.04	0.687 ± 0.04	0.821 ± 0.07	0.517 ± 0.06
E-Net (fix)	0.918 ± 0.01	0.697 ± 0.02	0.890 ± 0.02	0.631 ± 0.04	0.871 ± 0.01	0.598 ± 0.03	0.952 ± 0.01	**0.762 ± 0.02**	0.855 ± 0.02	0.603 ± 0.02
E-Net (s-)	0.921 ± 0.02	0.703 ± 0.03	**0.898 ± 0.01**	**0.651 ± 0.04**	**0.883 ± 0.01**	0.607 ± 0.04	0.943 ± 0.01	0.720 ± 0.01	0.854 ± 0.03	**0.606 ± 0.03**
E-Net	**0.930 ± 0.03**	**0.712 ± 0.03**	0.891 ± 0.02	0.633 ± 0.03	0.875 ± 0.03	**0.614 ± 0.05**	0.947 ± 0.01	0.741 ± 0.01	**0.857 ± 0.02**	0.600 ± 0.03

- *Node2vec [30].* A network embedding method, which learns latent topological features from network structures.
- *SEAL [12].* A link prediction method based on graph neural networks. This method is also applied to the extracted subgraphs.
- *SEAL (RWR).* SEAL using RWR subgraph extraction for ablation study.
- *GAT (RWR).* Graph Attention Neural Network (GAT) [17] is also conducted on the extracted RWR subgraphs to facilitate comparison with our denoising graph convolutional layers. Here, the other components in this setting remain the same as ours.
- *E-Net (n).* A variant of our method. When learning edge reliability, this method is based on node attributes and Node2vec embeddings rather than the K heuristics. We learn a linear mapping $f : [\mathbf{x}_i; \mathbf{e}_i; \mathbf{x}_j; \mathbf{e}_j] \rightarrow s(v_i, v_j)$ and follow a sigmoid function, where ; denotes concatenation and $\mathbf{e}_i$ is the Node2vec embedding of node v_i.
- *E-Net (fix).* Another variant of our method, which does not consider the mutual influence between noisy links and missing links. More specifically, in this variant, we do not perform end-to-end training to identify noisy and missing links. Instead, we first calculate six similarity scores: cosine similarity of node attributes, Common Neighbors score, Jaccard score, Preferential Attachment score [18], Adamic-Adar score [20] and Resource Allocation score [6]. We then average these scores to obtain a general score. Based on these results, we take the calculated score rather than training the noise scoring function $s(\cdot, \cdot)$.
- *E-Net (s-).* A third variant of our method, where the auxiliary denoising regularization is removed.

Our flawed link candidates in the test set contain all the links with $\mathcal{Y}_{ij} =?$. We control the number of real and fake missing links with a proportion of around 1:5, as well as the number of real and fake noisy links, also a proportion of around 1:5. For each dataset, we consistently use 80% of the data as a training set, 10% as a validation set and 10% as a testing set. We further implement an early stopping strategy, where we stop the training if the performance ceases to improve or only improves in a small range ($1e$-3) for seven successive epochs on the validation set. All models are implemented in Pytorch with the Adam optimizer used for optimization [31]. All experiments are conducted on a single machine with an Intel Xeon E5 and one NVIDIA TITAN GPU. In the following, the reported results are all averaged over 10 runs with random training/validation/testing splits and random weight matrix initializations.

4.2 Experimental Results

Missing Link Prediction. Table 3 presents the results. It can be seen from the table that our model consistently achieves the best or the second best performance across all datasets, which provides an affirmative answer to our motivating question **Q1**. Firstly, we compare *E-Net* with methods that only use node attributes and heuristics measures (i.e., *HEU*, *ATT* and *ENS*). We observe that our method achieves much better performance, as *E-Net* is able to discover new structural and node-specific attributes. Secondly, we compare *E-Net* with latent feature-based methods (i.e., *Node2vec* and *SEAL*); here, *SEAL* also performs the graph convolutional operations on an extracted subgraph, but does not denoise the network at all. We can see that *E-Net* also outperforms these latent feature-based methods. One reason for this is that *E-Net* prevents a proportion of the messages from being propagated along flawed links, thus significantly decreasing the adverse effect of incorrect structure information. Interestingly, *SEAL* cannot be successfully applied to the FinV and PPI datasets; that is, *SEAL* cannot get the results for more than one entire day because of memory and time costs. The reason for this is that *SEAL* extracts all two-hop neighbors of query nodes when

TABLE 4: Precision score on noisy link detection. Note that we set the number of selected links (i.e., the denominator) as the number of real noisy links; this means that precision is equal to recall, F1 and accuracy metrics under this setting.

	FinV	Cora	Citeseer	Pubmed	PPI
ATT	0.173	0.139	0.143	0.166	0.163
CN	0.297	0.297	0.267	0.297	0.472
Jaccard	0.288	0.288	0.242	0.288	0.472
PA	0.218	0.218	0.117	0.218	0.677
AA	0.241	0.241	**0.271**	0.241	0.471
RA	0.237	0.237	0.260	0.273	0.470
ENS	0.144	0.237	0.117	0.156	0.679
NE	-	0.117	0.051	0.024	0.174
E-Net	**0.348**	**0.319**	0.248	**0.298**	**0.692**

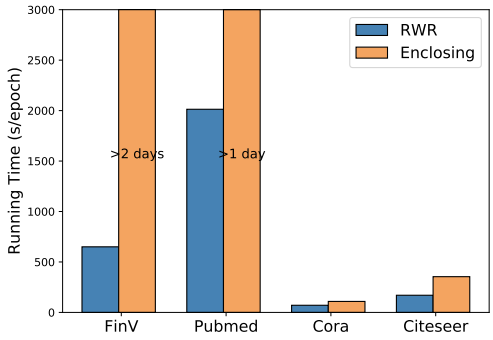


Fig. 3: Running time comparison of different subgraph extraction approaches.

constructing subgraphs; since a large number of hub nodes exist in the FinV dataset, while the PPI dataset is a much denser graph, this results in increased space and time costs when learning from large subgraphs. We further conduct an ablation study on *SEAL (RWR)* to demonstrate that our RWR subgraph extraction mechanism is more effective than fixed-size subgraph extraction. Furthermore, the comparison with *GAT (RWR)* highlights the significance of our denoising graph convolutional layers. Another interesting finding is that that none of the baseline methods can perform well on all datasets, while by contrast *E-Net* performs consistently well (+5.5% in terms of AUC and +10.7% in terms of F1). Another observation is based on a simplified version of our model, namely *E-Net (fix)*: that is, if we do not perform joint training, the performance will decrease to some extent in most cases. This is because *E-Net (fix)* ignores the mutual influence between noisy links and missing links.

It also should be noted here that *E-Net (s-)* sometimes outperforms *E-Net*. This is because the performance of the auxiliary denoising regularization term relies on the quality of the node attributes $\mathbf{X}$. For example, we can observe that *ATT* largely outperforms *HEU* on the FinV dataset, which to a certain extent reflects that node attributes play a more important role than graph structure in this FinV dataset. Thus, *E-Net*, which applies this regularization term, performs better than *E-Net (s-)*. On the contrary, we can see that *HEU* performs much better than *ATT* on the Cora datasets. This finding suggests that graph structure plays a more important role than node attributes on the Cora dataset; accordingly, *E-Net (s-)*, which lacks this regularization term, performs better than *E-Net* on the Cora dataset. Therefore, we suggest that it is better not to add this regularization term if the node attributes $\mathbf{X}$ are not sufficient in the graph.

Noisy Link Detection. Unlike the missing link prediction task, the noisy link detection task is based on a list of all observed links ordered according to their noise scores $s(\cdot,\cdot)$ (see Section 3.2). Here, we compare with eight unsupervised methods and take them as our noisy scores: 1) *ATT*, a cosine similarity between two query nodes; 2) *CN, Jaccard, PA, AA, and RA*, the five heuristic measures mentioned before; 3) *ENS*, the mean value of these six scores; 4) *NE* [32]. Since NE is designed for weighted networks, we here apply NE to unweighted networks to fit our problem setting and adapt the denoised edge weights as our noise scores; that is, we detect noisy links by picking the last C links with the smallest weights. To quantify the ability to identify noisy links by means of our proposed method, we adopt a standard metric, namely precision, which is widely used in some related works [8], [33]. Precision is defined as the ratio of true positive noisy links to the overall number of selected links. More specifically, we pick up the last C links according to $s(\cdot,\cdot)$ (which serve as our selected links), while C_n is the number of the truly observed noisy links within this selected set; thus, precision equals C_n/C. In this experiment, we set C as the number of all the actual noisy links in each dataset, meaning that the number of selected links is equal to the number of real noisy links. Under these settings, precision is therefore also equal to thes recall, F1 and accuracy metrics.

Table 4 summarizes the performance measured by precision/recall/F1/accuracy score. These results provide us with an answer to **Q2**: in short, our method can maintain a stable performance across all datasets, while the other baseline methods sometimes perform very poorly on specific datasets. For example, Preferential Attachment (*PA*) achieves poor performance on the Citeseer dataset, with a result only slightly better than chance. Moreover, node attribute similarity also performs poorly, which indicates that this kind of side information is not always available in all scenarios. Unexpectedly, we find that using the mean value of all heuristic scores (*ENS*) sometimes yields worse performance than using the individual score, except on the PPI dataset; this reflects the fact that the weights between these scores are very difficult to learn. We therefore suggest that averaging is not a good choice in some cases, and should therefore be carefully considered before it is applied. By contrast, the missing link objective in our method can provide useful guidance for learning more reasonable weights among different scores in noisy link detection. Furthermore, we find that NE is unable to deal with the FinV dataset at all; this is because NE conducts optimization directly on the adjacency matrix, which results in high computational complexity.

The Effect of RWR Subgraph Extraction. We here conduct a comparison with the *enclosing subgraph extraction* method proposed in [14], where all two-hop neighbors of query

TABLE 5: Experimental results on missing link prediction with different subgraph extraction approaches.

	Cora		Citeseer	
	AUC	F1	AUC	F1
Enclosing	0.883	0.621	0.858	0.584
RWR	**0.891**	**0.633**	**0.875**	**0.614**

TABLE 6: Performance of node classification on Cora.

Networks	Macro-F1	Micro-F1	Weighted-F1
Flawed network $\mathcal{G}$	0.613	0.629	0.626
Enhanced network $\mathcal{G}^*$	**0.662**	**0.677**	**0.676**
Clean network $\mathcal{G}_{\text{clean}}$ (**Ground Truth**)	0.778	0.786	0.785

nodes are extracted according to the settings provided in the source paper.

Figure 3 presents the running time of two different subgraph extraction approaches, both of which are applied on *E-Net*. We retain the same hyperparameters and model structure for both. As can be seen from the results, the enclosing method is much slower than the proposed RWR subgraph extraction method on all datasets due to the former's greedy extraction scheme. In particular, on the FinV dataset, the enclosing method takes more than two full days to train an epoch; this is because there are a large number of hub nodes in this dataset, which makes the subgraphs extremely large. By contrast, the RWR extraction method deftly avoids this problem.

We further assess the effectiveness of two subgraph extraction approaches in Table 5 (we do not evaluate the enclosing version on Pubmed and FinV due to the high associated space and time costs), which verifies the RWR version's superior performance. All of these results suggest that in practical applications, the RWR version is performed as a fast and accurate approach to extract subgraphs, which is particularly well-suited to large graphs and graphs containing lots of hub nodes (such as FinV). Thus, the question **Q3** is answered here.

Node Classification on the Enhanced Network. In this section, we aim to evaluate whether the enhanced network can improve the performance of downstream tasks on the Cora dataset. Here, we take node classification as our task. Given an input flawed network $\mathcal{G}$, we will reconstruct the enhanced network $\mathcal{G}^*$ via *E-Net* and further compare it with the real clean network $\mathcal{G}_{\text{clean}}$. To be more specific, when constructing the enhanced network $\mathcal{G}^*$, we first remove the links in noisy link candidates whose ranking is in the last C according to $s(\cdot,\cdot)$, where we set C as the number of real noisy links, then add the missing links predicted by our model in the set of missing link candidates. We then apply node2vec [30] to each network and use the learned embedding of each node to predict its label. Each label belongs to one of seven classes, representing its topic category. We select 80% of nodes for the training set and allocate the remaining 20% to the testing set. With regard to the target classifier, we use multilayer perceptron (MLP). We also apply early stopping to avoid overfitting. The sizes of the hidden layers are set to 64, 16, 16 and 4, respectively, and ReLU is used as the activation function. For the embedding learning, we adopt the default hyper-parameters of node2vec[3] and set the size of embedding as 64.

Table 6 presents the comparison between the performance on the different networks constructed based on the Cora dataset. As the node classification task on the Cora dataset is a multi-class classification problem, we here use Marco-F1, Micro-F1 and Weighted-F1 as our evaluation metrics. Unsurprisingly, the performance on the clean network $\mathcal{G}_{\text{clean}}$ surpasses that on the other networks, which demonstrates the importance of enhancing the flawed networks. This will also serve as our ground truth result for this classification task. Moreover, for the results on the clean network the clean network $\mathcal{G}_{\text{clean}}$, we find that the performance on the enhanced network $\mathcal{G}^*$ clearly exceeds that on $\mathcal{G}$ by a significant margin; this demonstrates that enhancing the flawed network is very helpful for the downstream tasks.

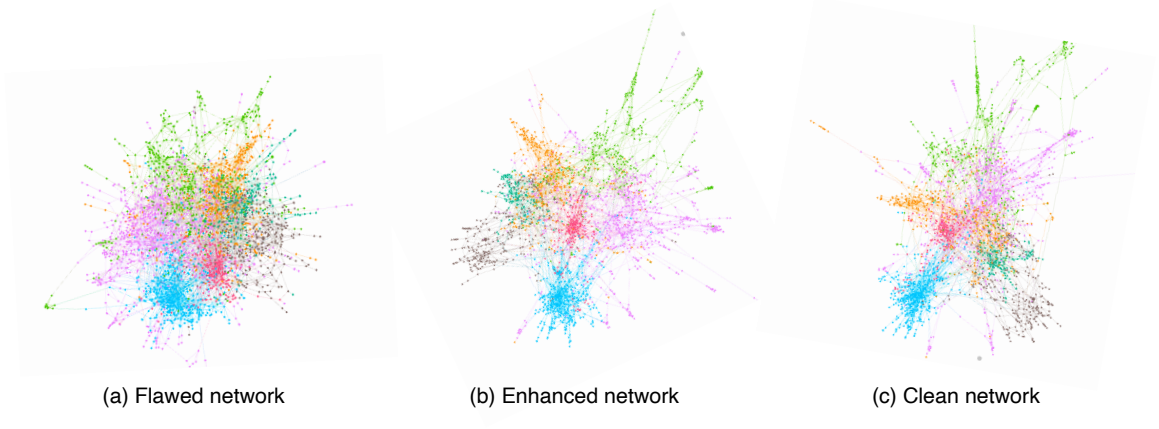
To facilitate improved understanding, we plot the visualization of the input flawed network $\mathcal{G}$, the enhanced network $\mathcal{G}^*$ by *E-Net* and the real clean network $\mathcal{G}_{\text{clean}}$ in Figure 4. From the figure, we can discern the following pattern: our enhanced network $\mathcal{G}^*$ (Figure 4 (b)) exhibits more discernible clustering compared with the input flawed network $\mathcal{G}$ (Figure 4 (a)). This is straightforward, because we have shown that enhancing the flawed network does aid the node classification task, which is closely related with this clustering visualization result. Another interesting observation from the visualization is that our enhanced network $\mathcal{G}^*$ exhibits a natural structure that is intuitively very similar to the clean network $\mathcal{G}_{\text{clean}}$ (Figure 4 (c)). Based on the above discussed results, we can now easily answer the question **Q4**.

Parameter Analysis. In this section, we analyze four crucial hyper-parameters: the number of RWR when extracting subgraph n_{RWR}, the sparsity coefficient α, the hidden dimension of GNN layers d^l (here, we let all GNN layers have the same dimension), and the number of GNN layers L.

In order to further determine the optimal parameters settings, we vary the values of n_{RWR}, α, d^l and L in order to observe more closely how the performance of missing link prediction will change. In detail, we study $n_{\text{RWR}} \in \{10, 20, 30, 40, 50\}$, $\alpha \in \{1e\text{-}8, 1e\text{-}7, 1e\text{-}6, 1e\text{-}5, 1e\text{-}4, 1e\text{-}3\}$, $d_l \in \{8, 16, 32, 64, 128\}$, $L \in \{1, 2, 3, 4, 5\}$ respectively. It should be noted here that the the number of RWR n_{RWR} remains at 30 while studying α, d^l and L, that the sparsity coefficient α remains at $1e\text{-}4$ while studying n_{RWR}, d^l and L, that the hidden dimension of GNN layers d^l remains at 32 while studying n_{RWR}, α and L, and that the number of GNN layers L remains at 4 while studying n_{RWR}, α and d^l. Here, we present some suitable values of these parameters for reference. We report the F1 score on the Cora dataset for the missing link prediction task in Figure 5. As we can see, and remarkably, E-Net can achieve relatively good performance regardless of changes in parameters (still better than the baseline methods), which answers the question **Q5**.

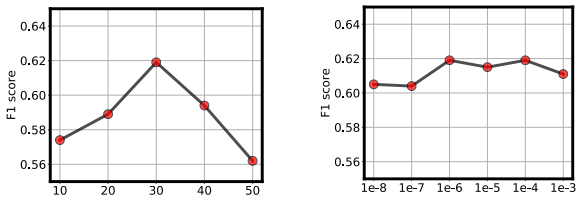
More specifically, from Figure 5(a), we observe that our

3. https://github.com/aditya-grover/node2vec



(a) Flawed network (b) Enhanced network (c) Clean network

Fig. 4: Visualization of Cora networks, where different colors represent different node categories.



(a) n_{RWR}: # of RWR (b) α: sparsity coefficient

(c) d^l: hidden dimension of GNN (d) L: # of GNN layers

Fig. 5: Hyper-parameter analysis, where the y-axes are fixed in the same range.

model yields good results when n_{RWR} is equal to 30. This can be explained with reference to the fact that only a small amount of local structural information is captured when n_{RWR} is set to be small, preventing us from capturing sufficient information, while too many observations would in turn prevent us from capturing the most important elements of graph structure. From Figure 5(b), we can see that we can maintain our results at above 0.61 when α is set to be larger than or equal to $1e$-6, which suggests the necessity of our auxiliary denoising objective. From Figure 5(c), the results seem to be less sensitive to the hidden dimension of GNN layers. From Figure 5(d), the performance improves on the whole if more GNN layers are used, and decreases only slightly after stacking 5 layers. This makes intuitive sense, since GNN with deeper layers gives the model a greater capacity to represent graphs. However, when the layer number is too high (equal to 5), the performance starts to drop slightly. This observation is due to the fact that GNN tends to result in over-smoothing after stacking too many layers, which is consistent with many conclusions in previous works [22], [34], [36]. Moreover, we concatenate all the outputs of each GNN layer to generate the final high-quality node representation, which avoids this problem to some extent.

5 Related Work

Network enhancement in our work is mostly relevant to the following three topics: 1) the link prediction methodologies used to evaluate the tendency of links to exist between the pairwise nodes; 2) network enhancement, which aims to remove noisy links and add missing links; 3) graph neural networks (GNNs), which apply deep models to relational data in order to learn high-level node representations. In the below, we review studies and applications relevant to these topics.

Link Prediction. Link prediction is the most straightforward means of identifying link existence in a network [29], [37], [38], [39]. Here, we will discuss three categories of link prediction algorithms. The first one focuses on computing the pairwise node similarity scores using heuristic measures, such as common neighbors (CN), Jaccard, preferential attachment (PA) [18], Adamic-Adar (AA) [20], resource allocation (RA) [6] etc. However, these predefined heuristics are handcrafted and only consider limited topological patterns. Accordingly, we bring them together as an ensemble and train a logistic regression classifier on multiple heuristic measures simultaneously in order to obtain better performance (this serves as our first baseline: *HEU*). The second category focuses on side information, such as node and edge attributes, and does not consider the network structure at all; thus, these approaches cannot explain how the networks are formed. For example, in a mobile network, a user's personal information can be considered as a type of side information on nodes (here, the nodes are users), while call duration can be considered a kind of side information relating to edges. The most common approach is to combine all side information together and train a multilayer perceptron classifier on it (this serves as our second baseline: *ATT*). While this approach can sometimes achieve superb performance when there is some annotation or preference on the nodes or edges, it is also important to note that this kind of side information is hard to obtain in many situations.

The third type of methods are based on latent features, i.e., the latent properties or representations of nodes. A low-dimensional embedding of nodes can be learned by skip-gram models [15], [40], [41], [42], [43], [44] or matrix factorization methods [30], [45], [46], [47]. Skip-gram models have achieved promising results in recent studies. Grover et al. [30] conducted experiments to show that this type of network embedding method outperformed traditional methods to a large extent on link prediction tasks. For their part, the latter methods are commonly derived from adjacency or Laplacian matrices, which are computationally expensive when large-scale networks are involved. While this type of method can capture more global topological properties, these methods are also conducted in a transductive way, meaning that they need to be retrained when they encounter new nodes or new networks. In addition, they are also more difficult to interpret than heuristic features. The existing work on link prediction that is most related to ours is SEAL [12], the authors of which verified that computing on local subgraphs can well approximate a range of heuristics and proposed using a GNN to learn from local subgraphs for link prediction purposes. However, these authors also ignored the mutual influence between noisy links and missing links. In addition, their extracted enclosing subgraph can become extremely large if hub nodes are traversed and is only able to capture local structure evidence. Inspired by this work, we utilize a similar GNN structure, but take the mutual information of noisy links and missing links into consideration and thereby propose an improved RWR subgraph extraction approach.

Network Enhancement. The concept of network enhancement (NE) was first proposed by Wang et al. [32]. Its original purpose was to improve the signal-to-noise ratio of a flawed input network. NE is conducted by directly optimizing the adjacency matrix, which is a process with high computational complexity; as a result, it is mainly applied on biology networks in a limited scale and is difficult to extend to larger networks. NE takes a flawed, undirected, weighted network as input and outputs a denoised network by removing the weak links and retaining the strong links. This method can also be utilized to detect missing links, although the authors did not further analyze this point. Here, we extend this concept to our setting, i.e., removing noisy links and completing missing links in a flawed, undirected, unweighted network, an approach also referred to as network enhancement. While several other works [8], [9] have also dealt with missing link prediction and the noisy link detection in one framework, none of these considered the dependency between missing links and noisy links. Moreover, there is another line of research that might share similar ideas to ours [48], [49], [50], [51]. These works aimed to conduct network embedding when encountering noisy networks. However, they only concentrate on how to learn a reliable node representation. By contrast, our work focuses more on how to identify flawed links explicitly and then reconstruct a clean network from a noisy one; this reconstructed network can then be applied to various downstream tasks, including but not limited to network embedding.

Furthermore, some works from the bioinformatics area have also utilized similar ideas to reconstruct Protein-Protein Interaction (PPI) networks [56], [57], [58], [59], [60]. Hulovatyy et al. [58] analyzed several existing link prediction measures and introduced novel sensitive measures of the topological similarity of extended neighborhoods in order to denoise PPIs, although they only evaluated their measures on missing links. Lei et al. [56] proposed reconstructing PPIs by computing topological similarity metrics via a random walk-based procedure. Following this idea, Alkan et al. [57] further employed a local neighborhood evaluation of these similarity metrics and accordingly proposed an iterative network reconstruction process. The iterative nature of these processes also enables the capture of some aspects of the mutual influence between missing links and noisy links to a certain extent. However, most of these works are designed or evaluated only on PPIs, and the computation of the similarity matrix suffers due to high memory cost.

Graph Neural Networks. As our paper does not have a major focus on graph neural network (GNN) innovations, but rather concentrates on a novel application of GNNs, we only introduce GNNs briefly here. GNNs, a new type of neural network, have been used to learn node representations from graph structure and node attributes by extending neural networks to capture the messages passed along edges, an approach that has attracted increasing interests [15], [17], [36], [52], [53], [54], [55]. The learned node or graph representation can then be applied to various downstream tasks, such as missing link prediction. The early work proposed in [52] applied a convolution filter to graph data based on the graph Laplacian spectrum, which is both not spatially localized and computationally expensive. To address these limitations, several subsequent works have focused on designing efficient localized filters [53], [54]. Another trend involves the use of neighborhood aggregations to learn from local neighborhoods rather than the entire graph, which takes advantage of both the properties of the neighbors and local topological structures [15], [17], [36], [55]. GCN [36] and GAT [17] are the two models most relevant to our approach. Graph Convolution Networks (GCN) [36] integrate local graph structures and the features of nodes to obtain node representation from the hidden layers. Graph Attention Networks (GAT) [17] employ a self-attention mechanism over the target node and its neighbors to learn adaptive weights of the neighbors. Most GNN structures can be summarized as a message-passing framework; however, none of them consider learning from noisy data. Since flawed links can substantially harm the information diffusion process, it is vital to prevent the aggregation of messages from noisy links, which is achieved by our extended version of GNN.

6 Conclusion

In this work, we study the problem of network enhancement, i.e., deleting noisy links and completing missing links. We propose E-Net, an end-to-end GNN model, to unify these two tasks due to their inter-dependence and ability to mutually boost each others' performance. More specifically, on one hand, detecting noisy links can benefit the performance of missing link prediction; on the other

hand, predicting missing links can provide indirect supervision for noisy link detection when the labels of the noisy links are unavailable. Moreover, to further reduce the computational cost, we propose a RWR subgraph extraction method. Compared with the existing enclosing method, the RWR subgraph extraction method is better able to capture global and local structure information and can also reduce the space and time cost. We apply our model on several types of networks, including a large real-world network, and evaluate the effectiveness and efficiency of our model in various ways.

Studying network enhancement is an important problem, and this work provides essential insights for further study. In terms of future work, it will be interesting to see whether E-Net or its extension can be adopted as a defense strategy to make networks more robust against network adversarial attacks, which situates the problem in a much tougher situation than the noisy one. We also hope that this work will inspire further works to investigate well-defined methods in order to study and enhance the network robustness.

Acknowledgments. This work is supported by NSFC (U1611461, 61702522), the Fundamental Research Funds for the Central Universities (Zhejiang University NGICS Platform), and the Major Scientific Project of Zhejiang Lab (Grant No. 2020MC0AE01).

REFERENCES

[1] K. Gueorgi, "Effects of missing data in social networks," *Social Networks*, 2006, pp.247–268.
[2] C. T. Butts, "Network inference, error, and informant (in)accuracy: A Bayesian approach," *Social Networks*, 2003, pp.103–140.
[3] G. M. Namata and L. Getoor, "Identifying graphs from noisy and incomplete data," *In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, 2009, pp.23–29.
[4] J. Schafer and J. Graham, "Missing data: Our view of the state of the art," *Psychological Methods*, 2002.
[5] L. Lü, C. Jin, and T. Zhou, "Similarity index based on local paths for link prediction of complex networks," *Physical review E, Statistical, Nonlinear, and Soft Matter Physics 80*, 2009.
[6] T. Zhou, L. Lü, and Y. Zhang, "Predicting missing links via local information," *The European Physical Journal B*, 2009.
[7] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," *In Proceedings of the 12th ACM International Conference on Information and Knowledge Management (CIKM'03)*, 2003.
[8] L. Pan, T. Zhou, L. Lü, and C. Hu, "Predicting missing links and identifying spurious links via likelihood analysis," *Scientific Reports*, 2016.
[9] R. Guimerà and M. Sales-Pardo, "Missing and spurious interactions and the reconstruction of complex networks," *Proceedings of the National Academy of Sciences*, 2016.
[10] Y. Chen, Q. Gan, and TorsT.ten Suel. "Local Methods for Estimating Pagerank Values," *In Proceedings of the 13rd ACM International Conference on Information and Knowledge Management (CIKM'04)*, 2004.
[11] J. Xu, H. Liu, Z. Li, J. He, X. Du, and Y. Cai. "Local methods for estimating pagerank values," *Proceedings of the 12th Asia Pacific Web Conference (APWeb'10)*, 2010.
[12] M. Zhang and Y. Chen. "Link prediction based on graph neural networks," *Proceedings of the 32th International Conference on Neural Information Processing Systems (NIPS'18)*, 2018.
[13] Z. Bar-Yossef and L.Mashiach. "Local approximation of pagerank and reverse pagerank," *Proceedings of the 17th ACM International Conference on Information and Knowledge Management (CIKM'08)*, 2008.
[14] M. Zhang and Y. Chen. "Weisfeiler-lehman neural machine for link prediction," *Proceedings of the 23rd International Conference on Knowledge Discovery and Data Mining (KDD'17)*, 2017.
[15] W. L. Hamilton, R. Ying, and J. Leskovec. "Inductive representation learning on large graphs," *Proceedings of the 31th International Conference on Neural Information Processing Systems (NIPS'17)*, 2017.
[16] G. Jeh and J. Widom. "Scaling personalized web search," *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*, 2003.
[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio. "Graph attention networks," *Proceedings of the 8th International Conference on Learning Representations (ICLR'18)*, 2018.
[18] A. Barabási and R. Albert. "Emergence of scaling in random networks," *Science*, 1999.
[19] D. Zhu, Z. Zhang, P. Cui and W. Zhu. "Robust graph convolutional networks against adversarial attacks," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'19)*, 2019.
[20] L. A. Adamic and E. Adar. "Friends and neighbors on the web," *Social Networks*, 2001.
[21] Y. Li, R. Zemel, M. Brockschmidt and D. Tarlow. "Gated graph sequence neural networks," *Proceedings of the 6th International Conference on Learning Representations (ICLR'16)*, 2016.
[22] Q. Li, Z. Han and X. Wu. "Deeper insights into graph convolutional networks for semi-supervised learning," *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, 2018.
[23] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. "An end-to-end deep learning architecture for graph classification," *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, 2018.
[24] A. J. Smola. "Kernels and regularization on graphs," *Learning Theory and Kernel Machines*, 2003.
[25] D. Zhou and B. Schölkopf. "A regularization framework for learning from graph data," *Proceedings of the 19th International Conference on Machine Learning (ICML'04)*, 2004.
[26] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin and N. Usunier. "Parseval networks: Improving robustness to adversarial examples," *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, 2017.
[27] F. Lin, M. Fardad and M. R. Jovanović. "Identification of sparse communication graphs in consensus networks," *Proceedings of 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton'12)*, 2012.
[28] G. Gnecco, R. Morisi and A. Bemporad. "Sparse solutions to the average consensus problem via various regularizations of the Fastest Mixing Markov-Chain Problem," *IEEE Transactions on Network Science and Engineering*, 2015.
[29] D. Liben-Nowell and J. Kleinberg. "The link-prediction problem for social networks," *Journal of the Association for Information Science and Technology*, 2007.
[30] A. Grover and J. Leskovec. "Node2Vec: Scalable feature learning for networks," *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining (KDD'16)*, 2016.
[31] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization," *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*, 2015.
[32] B. Wang, A. Pourshafeie, M. Zitnik, J. Zhu, C. D. Bustamante, S. Batzoglou, and J. Leskovec. "Network enhancement: A general method to denoise weighted biological networks," *Nature Communications*, 2018.
[33] J. L. Herlocker, J. A. Konstan, L. G. Terveen and J. T. Riedl. "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, 2004.
[34] J. Klicpera, A. Bojchevski and S. Günnemann. "Predict then propagate: Graph neural networks meet personalized pageRank," *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*, 2018.
[35] J. Pan, H. Yang, C. Faloutsos and P. Duygulu. "Automatic multimedia cross-modal correlation discovery," *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*, 2004.
[36] T. N. Kipf and M. Welling. "Semi-Supervised classification with graph convolutional networks," *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*, 2017.
[37] L. Lü and T. Zhou. "Link prediction in complex networks: A survey," *Physica A*, 2012.

[38] A. De, S. Bhattacharya, S. Sarkar, N. Ganguly and S. Chakrabarti. "Discriminative link prediction using local, community, and global signals," *IEEE Transactions on Knowledge and Data Engineering*, 2016.
[39] L. Duan, S. Ma, C. Aggarwal, T. Ma and J. Huai. "An ensemble approach to link prediction," *IEEE Transactions on Knowledge and Data Engineering*, 2017.
[40] P. Cui, X. Wang, J. Pei and W. Zhu. "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, 2019.
[41] B. Perozzi, R. Al-Rfou and S. Skiena. "DeepWalk: Online learning of social representations," *Proceedings of the 20th International Conference on Knowledge Discovery and Data Mining (KDD'14)*, 2014.
[42] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan and Q. Mei. "LINE: Large-scale information network embedding," *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*, 2015.
[43] Y. Lai, C. Hsu, W. Chen, M. Yeh and S. Lin. "PRUNE: Preserving proximity and global ranking for network embedding," *Proceedings of the 31th International Conference on Neural Information Processing Systems (NIPS'17)*, 2017.
[44] A. G. Duran and M. Niepert. "Learning graph representations with embedding propagation," *Proceedings of the 31th International Conference on Neural Information Processing Systems (NIPS'17)*, 2017.
[45] M. Nickel, X. Jiang and V. Tresp. "Reducing the rank in relational factorization models by including observable patterns," *Proceedings of the 28th Advances in Neural Information Processing Systems (NIPS'14)*, 2014.
[46] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. "Mixed membership stochastic blockmodels," *Proceedings of the 23rd International Conference on Neural Information Processing System (NIPS'09)*, 2009.
[47] Y. Koren, R. Bell and C. Volins. "Matrix factorization techniques for recommender systems," *IEEE Computer Society Press*, 2009.
[48] S. B. N. Lokesh and M. N. Murty. "Outlier Aware Network Embedding for Attributed Networks," *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, 2019, pp.12-19.
[49] J. Liang, P. Jacobs, J. Sun and R. Parthasarathy. "Semi-supervised embedding in attributed Networks with Outliers," *Proceedings of the 18th Industrial Conference on Data Mining (ICDM'18)*, 2018.
[50] A. Okuno and H. Shimodaira. "Robust graph embedding with noisy link weights," *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS'19)*, 2019.
[51] Z. Qiu, W. Hu, J. Wu, Z, Tang and X. Jia. "Noise-resilient similarity preserving network embedding for social networks," *Proceedings of the 28th International Joint Conferences on Artificial Intelligence (IJCAI'19)*, 2019.
[52] J Bruna, W. Zaremba, A. Szlam and Y. Lecun. "Spectral networks and locally connected networks on graph," *Proceedings of the 4th International Conference on Learning Representations (ICLR'14)*, 2014.
[53] M. Defferrard, X. Bresson and P. Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering," *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*, 2016.
[54] M. Henaff, J. Bruna and Y. Lecun. "Deep convolutional networks on graph-structured data," *arXiv: Learning*, 2015.
[55] J. Chen, J. Zhu and L. Song. "Stochastic training of graph convolutional networks with variance reduction," *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*, 2018.
[56] C. Lei and J. Ruan. "A novel link prediction algorithm for reconstructing protein-protein interaction networks by topological similarity," *Bioinformatics*, 2013.
[57] F. Alkan and C. Erten. "RedNemo: Topology-based PPI network reconstruction via repeated diffusion with neighborhood modifications," *Bioinformatics*, 2017.
[58] Y. Hulovatyy, R. Solava and T. Milenković. "Revealing missing parts of the interactome via link prediction," *PLoS One*, 2014.
[59] J. Zhao, L. Miao, J. Yang, H. Fang, Q. Zhang, M. Nie, P. Holme and T. Zhou. "Prediction of links and weights in networks by reliable routes," *Scientific Reports*, 2015.
[60] C. V. Cannistraci, G. Alanis-Lobato and T. Ravasi. "Minimum curvilinearity to enhance topological prediction of protein interactions by network embedding," *Bioinformatics*, 2013.
[61] B. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood, K. Dolinski and M. Tyers. "The BioGRID interaction database: 2008 update," *Nucleic Acids Research*, 2008.

Jiarong Xu received her BS degree in information science and technology from the Donghua University, China, in 2016. She is currently working towards her PhD degree in the College of Control Science and Engineering at Zhejiang University. Her current research interests include data mining algorithms, social network theories and reinforcement learning.

Yang Yang received his PhD degree from Tsinghua University in 2016. He is an associate professor in the College of Computer Science and Technology, Zhejiang University. His main research interests include data mining and social network analysis. He has been visiting scholar at Cornell University and Leuven University. He has published over 20 research papers in major international journals and conferences, including KDD, WWW, AAAI, and TOIS.

Chunping Wang received her PhD degree in Machine Learning from Duke University. She started her professional career in Opera Solutions as a Data Scientist, and is currently a Principal Data Scientist in FinVolution Group. Her current interests include the mining of both structured and unstructured data via machine learning technology to empower the financial industry.

Zongtao Liu received his master degree from the College of Computer Science and Technology, Zhejiang University. He is currently a senior algorithm engineer in Alimama, which is the advertising department of Alibaba Group. His main research interests include data mining and social network analysis.

Jing Zhang received her master and PhD degree from the Department of Computer Science and Technology, Tsinghua University. She is an assistant professor in Information School, Renmin University of China. Her research interests include social network mining and deep learning.

Lei Chen received his master degree from Shanghai Jiao Tong University in 2009. He is a vice president of FinVolution Group. He is currently leading the department of big data and AI to empower the finance business via technology. He was honored with the title of Senior Expert in AI by the City of Shanghai.

Lu Jiangang received the BS and PhD degrees from the Zhejiang University, in 1989 and 1995, respectively. He is currently a professor in the College of Control Science and Engineering, Zhejiang University. His research interests include artificial intelligence and industrial automation.