Line Graph Neural Networks for Link Prediction

Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji, *Senior Member, IEEE*

Abstract—We consider the graph link prediction task, which is a classic graph analytical problem with many real-world applications. With the advances of deep learning, current link prediction methods commonly compute features from subgraphs centered at two neighboring nodes and use the features to predict the label of the link between these two nodes. In this formalism, a link prediction problem is converted to a graph classification task. In order to extract fixed-size features for classification, graph pooling layers are necessary in the deep learning model, thereby incurring information loss. To overcome this key limitation, we propose to seek a radically different and novel path by making use of the line graphs in graph theory. In particular, each node in a line graph corresponds to a unique edge in the original graph. Therefore, link prediction problems in the original graph can be equivalently solved as a node classification problem in its corresponding line graph, instead of a graph classification task. Experimental results on fourteen datasets from different applications demonstrate that our proposed method consistently outperforms the state-of-the-art methods, while it has fewer parameters and high training efficiency.

Index Terms—Deep learning, graph analysis, link prediction, graph neural networks, line graphs.

1 Introduction

Link prediction models are used to learn the distribution of links in graphs and predict the existence of potential links [1], [2], [3], [4]. In many real-world applications, the input data is represented as graphs, and link prediction models can be applied to tasks like friend recommendation in social networks [5], product recommendation in e-commerce [6], [7], [8], knowledge graph completion [9], protein interaction analysis [10], and metabolic network reconstruction [11].

To solve the link prediction problem, various heuristic methods were proposed to measure the similarity between two target nodes and predict the existence of link [12]. However, the heuristic functions in these methods are often manually designed for a specific network, limiting their applicability to diverse areas. For example, the number of common neighbors [5] is employed as a first-order heuristic function to predict the potential friendship relations in social networks and achieves satisfactory performance. However, this heuristic may not work well on protein-protein interaction networks, since two proteins sharing many common neighbors may have a low probability of interacting [13].

Many heuristic methods have been proposed to solve graph link prediction problems from different areas. However, there still exist challenges to select heuristic functions given a new network. To tackle these challenges, the link prediction model based on graph neural networks (SEAL) was proposed to learn heuristic functions from $h-$hop neighborhood automatically [14]. This method achieved the state-of-the-art performance on a variety of graphs. The SEAL model extracts an h-hop enclosing subgraph centered on the two target nodes and predicts the existence of link based on the topology of enclosing subgraph. Therefore, the link prediction task is converted to the graph classification problem, where the model takes the enclosing subgraph as inputs and predicts the existence of link between them. Normally, graph neural networks [15] are employed to learn features to represent the topology of the subgraph. Therefore, graph pooling layers [16], [17], [18] are required to compute a fixed-size feature vector from the whole graph while some information may be lost in this operation. For example, in sort pooling operations [15], only partial nodes can be selected to represent the graph. In addition, a graph neural network with pooling layers often requires more training time to converge.

Although the SEAL works well in many types of graphs, it still has some limitations, due to the usage of pooling operations in the graph neural network. To solve the information loss in pooling layers, we propose to learn the features of the target link directly instead of extracting features from the whole enclosing subgraph. Compared with extracting features from the whole graph, learning node embedding is more effective. Graph convolution layers [19], [20], [21], [22] have shown promising performance for learning node embeddings. However, graph convolution layers are not effective enough to learn edge embeddings from graphs. To address this issue, we propose to convert the original enclosing subgraph into a corresponding line graph. Each node in the line graph has a unique corresponding edge in the original graph. In addition, the topology information can be well preserved during the transformation. Therefore, graph convolution layers can be directly applied to learn the node embeddings in the line graph. The node embeddings in the line graph are used as features for the edges in the original graph to predict the existence of links. Therefore, the link prediction task can be regarded as the node classification

- *Lei Cai is with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164. E-mail: lei.cai@wsu.edu.*
- *Jundong Li is with the Department of Electrical and Computer Engineering, Department of Computer Science, and School of Data Science, University of Virginia, Charlottesville, VA 22904. E-mail: jundong@virginia.edu.*
- *Jie Wang is Department of Electronic Engineering and Information Science, University of Science and Technology of China, China 230026. Email: jiewangx@ustc.edu.cn.*
- *Shuiwang Ji is with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843. E-mail: sji@tamu.edu.*

Manuscript received April 19, 2005; revised August 26, 2015.

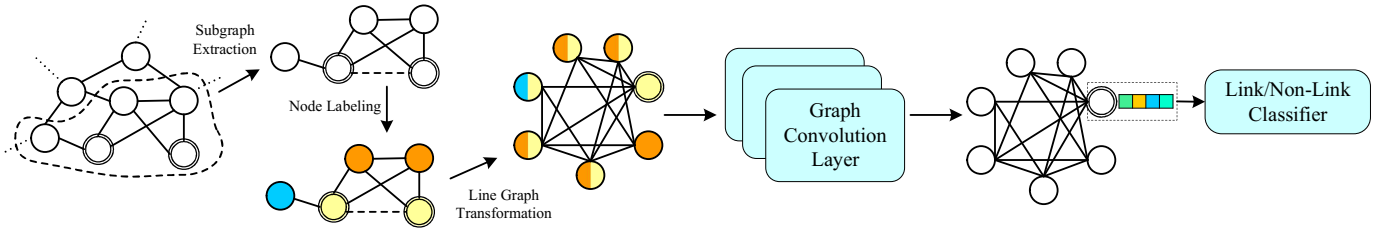


Fig. 1. Illustration of our proposed model based on line graph neural networks. The two target nodes in the graph are marked with double circles. To predict the existence of the link, an h-hop enclosing subgraph centered on two target nodes is extracted. A node labeling function is employed to assign the label for each node to represent the structural importance to the target link. To learn the feature of the target link, we transform the enclosing subgraph into a corresponding line graph. The graph convolution networks are used to learn the feature that is employed to predict the existence of link.

problem in our proposed framework. Our contributions can be summarized as follows:

1) We analyze the limitations of exiting deep learning based link prediction methods due to graph pooling operations and envision the need to develop a new method that do not have the information loss problem.
2) We propose a line graph neural networks model for the link prediction task, where the original graph is transformed into a corresponding line graph to enable efficient feature learning for the target link.
3) We conduct experiments on 14 datasets from different areas. Our proposed method can achieve promising performance on different datasets and outperform all baseline methods, including the previous state-of-the-art model.
4) Our proposed method can achieve promising performance only with graph convolution layers, and thus requires fewer parameters. In addition, the neural network consisting of graph convolution layers converges significantly faster than the state-of-the-art model.

2 Related Work

Link prediction models can be grouped into three categories – heuristic methods, embedding methods, and deep learning methods.

Heuristic Methods: The key idea of heuristic methods is to compute the similarity score from the neighborhood of two target nodes. Based on the maximum hop of neighbors used in the computation procedure, heuristic methods can be categorized into three groups, including first-order, second-order, and high-order heuristics. Common neighbors and preferential attachment [12] are typical first-order heuristics since only one-hop neighbors are employed to compute the similarity. Second-order heuristic methods that involve two-hop neighbors include Adamic-Adar [5] and resource allocation [12], [23]. In addition, high-order heuristics, including Katz [24], rooted PageRank [25], and SimRank [26] were proposed to compute the similarity score between a pair of nodes using the whole graph. High-order heuristic methods can often achieve better performance than low-order heuristics but require more computation cost. Since many heuristic methods were proposed to handle different graphs, selecting a favorable heuristic method becomes a challenging problem.

Embedding Methods: The similarity between two target nodes can also be calculated based on node embeddings [27]. Therefore, embedding methods that can learn the features of nodes from graph topology were also employed to solve the link prediction task, and typical methods along this line include matrix factorization [6] and stochastic block [10] etc. Recently, inspired by world embedding methods in natural language processing tasks, recent advances such as deepwalk [28], LINE [29], and node2vec [30] were proposed to learn node embedding via the skip-gram method. Deepwalk generates random walks for each vertex with a given length and picks the next visited node uniformly from the neighbors of the current node. Later on, the skip-gram method is employed to learn node embeddings from the generated node sequence. Variational graph auto-encoder (GAE) [31] is proposed to learn the node embedding through graph convolution neural networks to reconstruct the graph topology. The node embedding methods can learn informative features from the graph and thus achieve satisfactory performance for the link prediction task. However, the performance of link node embedding methods can be affected if the graph becomes very sparse.

Deep Learning: To overcome the limitations of heuristic methods, deep learning based methods were proposed to learn the distribution of links from the graph automatically [14], [32], [33]. Weisfeiler-Lehman Neural Machine was proposed to predict the existence of a link using a fully-connected neural network based on a fixed-size enclosing subgraph centered on the two target nodes [32]. To predict the existence of a link from a general enclosing subgraph, SEAL [14] converts the link prediction task to a graph classification problem and solve it using graph neural networks. Due to the promising learning ability of graph neural networks, the SEAL model achieves the state-of-the-art performance for the link prediction problem. Later on, a multi-scale link model was also proposed to extend SEAL to achieve better performance on plain graphs [33].

3 The Proposed Methods

3.1 Problem Formulation

In the link prediction task, we are often given a network represented as an undirected graph $G = (V, E)$ which consists of a set of vertices $V = \{v_1, v_2, ..., v_n\}$ and a set of links $E \subseteq V \times V$. The graph can also be represented by the adjacency matrix A. If there exists a link between vertex i and j, then $A_{i,j} = 1$ and $A_{i,j} = 0$ otherwise. The goal of

link prediction is to predict potential or missing links that may appear in a foreseeable future.

3.2 Overall Framework

Deep learning based link prediction models were proposed to learn the link distribution from the existing links and determine whether a link exists between two target nodes in the graph. For example, when we predict if there exists a link between two users in a social network, the number of mutual friends is commonly considered as a main criterion. If two users share many mutual friends, they are more likely to be connected. In this sense, if the 1-hop subgraph induced from two target nodes are densely connected, we will have higher chances to observe a link between them. Considering the variation of networks from different areas, deep learning based methods were proposed to learn the topology feature of subgraphs automatically and predict the existence of links [14]. In particular, the deep learning based link prediction models generally consist of the following three components:

1) Enclosing subgraph extraction: The existence of a potential link can be determined by the topology of a local enclosing subgraph centered on two target nodes. To seek a balance between computation cost and prediction performance, an $h-$hop enclosing subgraph is extracted for learning features and predicting the existence of potential links.
2) Node labeling: Given an enclosing subgraph, we are required to identify the role of each node in the graph before learning features and predict the existence of the link. That is, we need to identify the target nodes and mark the structural importance of other nodes. A favorable labeling function is of great importance for the further feature learning procedure.
3) Feature learning and link prediction: The output of node labeling function can be used as the attribute of each node in the graph. The attribute can indicate the structural importance of the link to be predicted. Graph neural networks are commonly employed to learn features from the given enclosing subgraph, which can be further used to predict the existence of a link.

In this work, we propose line graph neural networks for the link prediction task. Our proposed model can be illustrated in Figure 1. Following the general framework of deep learning based link prediction models, we extract an h-hop enclosing subgraph centered on two target nodes and assign each node with a label that can represent the structural importance to the target link. The key contribution of our proposed method is the feature learning component. In the previous state-of-the-art model, graph convolution and graph pooling layers are employed to obtain a fixed-size feature vector to predict the existence of the link considering the scale variation of different graphs. Since this graph pooling layer is employed in the state-of-the-art model, only part of graph information can be preserved for further prediction. To overcome the limitations of the SEAL method, we propose to convert the enclosing subgraph to a line graph where each node corresponds to a unique link in the original graph. The feature of the link can be learned directly using the entire input from line graph representation. Thus the proposed method can greatly improve the performance of the link prediction.

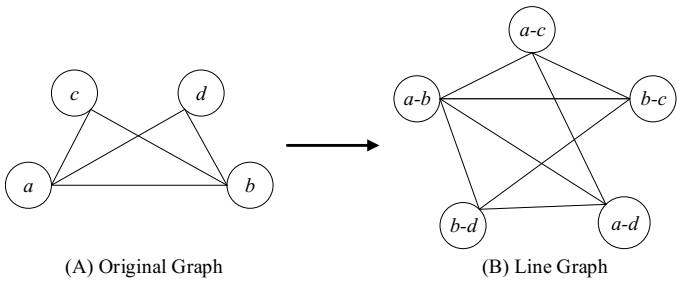


Fig. 2. Illustration of the line graph transformation procedure. Each node in the line graph corresponds to a unique edge in the original graph and is marked with the name of two end nodes.

3.3 Line Graph Neural Networks

Line Graph Space Transformation In order to predict the existence of a link, graph neural networks are employed to learn features from a given enclosing subgraph $G^h_{v_1,v_2}$, where $G^h_{v_1,v_2}$ is an h-hop enclosing subgraph centered on two target nodes v_1 and v_2, and each node in the enclosing subgraph is associated with a label that can indicate the structural importance to the target link. Different enclosing subgraphs commonly contain a different number of nodes. To extract a fixed-size feature vector for the further prediction, we will lose some information during the procedure. To overcome this challenge, we propose to convert the enclosing subgraph to the line graph, which represents the adjacencies between edges of the original graph. Thus, the feature of the link to be predicted can be learned directly in the line graph representation using graph convolution neural networks.

The line graph $L(G)$ of a given undirected graph G is proposed to represent the adjacencies between edges of G [34], [35]. The definition of line graph $L(G)$ can be defined as follows.

Definition 1. The edges in the original graph G are considered as nodes in the line graph $L(G)$. Two nodes in $L(G)$ are connected if and only if the two corresponding links share the same node.

An example of the line graph transformation procedure is illustrated in Figure 2. The original undirected graph G contains four nodes and five edges. Therefore, the line graph $L(G)$ contains five nodes. The node $(a-b)$ and $(a-c)$ in the line graph are connected since the corresponding edges in the original graph G share a common node a based on the definition of the line graph.

Based on the definition of the line graph, we can obtain the following property of $L(G)$: *Given a graph G with m nodes and n edges, the number of nodes of the line graph $L(G)$ equals to n. The number of edges in $L(G)$ is $\frac{1}{2}\sum_{i=1}^{m} d_i^2 - n$, where d_i is the degree of node i in graph G.*

This property guarantees that learning features in the line graph space will not increase the computation complexity significantly. In additional, converting a graph G to line graph $L(G)$ only costs linear time complexity [36], [37].

Node Label Transformation An enclosing subgraph can be converted into the corresponding line graph through transformation. However, this procedure can only transform the topology of a given graph. Each node in the enclosing subgraph also contains a label $l \in \mathbb{R}$ generated by labeling function as node attributes that can represent the structural importance. During the transformation procedure, edges in the original graph are represented as nodes in the line graph. The label l is only assigned for nodes in the original graph. To transfer the node label from the original graph directly, a transformation function is required to convert the node label to edge attribute. Thus the edge attribute can be assigned directly as the node attribute in the line graph. In this work, we propose to generate the edge attribute from the node label through the following function:

$$l_{(v_1,v_2)} = \text{concate}(\min(f_l(v_1), f_l(v_2)), \max(f_l(v_1), f_l(v_2))), \tag{1}$$

where $f_l(\cdot)$ is the node labeling function, v_1 and v_2 are the two end nodes of the edge, and concate$(\cdot)$ represents the concatenation operation for the two inputs. Since we only consider undirected graph link prediction in this work, the attribute of edge $(v1, v2)$ and $(v2, v1)$ should be the same. It is easy to prove that the edge attribute generated by equation (1) is consistent when switching the end nodes. In addition, the structural importance information of the node can be well preserved in the function.

The proposed method in equation (1) can well address the edge attribute transformation in plain graphs. In some cases, graphs are commonly provided with node attributes. For example, in citation networks, the node attribute describing a summary of the paper can be provided in the graph. For attributed graphs, node attributes also play an important role in the link prediction task. Therefore, the edge attribute transformation function should be generalized to deal with attributed graph. Following the edge attribute transformation function in equation (1), we can concatenate the original node attribute with the node label as the edge attribute. But the edge attribute will not be consistent when we switch the order of two end nodes in the undirected graph. To overcome this limitation, we propose to deal with the original node attribute and node label in different ways by:

$$l_{(v_1,v_2)} = \text{concate}(\min(f_l(v_1), f_l(v_2)), \max(f_l(v_1), f_l(v_2)), X_{v_1} + X_{v_2}), \tag{2}$$

where X_{v_1} and X_{v_2} are the original attribute of node v_1 and v_2. We propose to combine the node attribute using summation operation, which can guarantee the invariance of edge attributes when switching the end nodes. The generated edge attribute $l_{(v_1,v2)}$ can be used as the node attribute directly in the line graph. Therefore, the link prediction task is converted to a node classification problem which can be solved by graph convolution neural networks.

Feature Learning by Graph Neural Networks With recent progress in graph neural networks, learning the graph feature becomes a favorable solution that has been explored in various graph analytical tasks [17], [18], [20]. In this work, we employ graph convolution neural networks to learn the node embedding in the line graph, which can represent an edge in the original graph. Thus, the node embedding in the line graph can be used to predict whether a potential link is likely to exist in the network.

Given a line graph representation of the enclosing subgraph $L(G^h_{v1,v2})$, the node embedding of (v_i, v_j) in the k-th layer of the graph convolution neural network is indicated as $Z^{(k)}_{(v_i,v_j)}$. Then the embedding of (v_i, v_j) in the $(k+1)$-th layer is given by:

$$Z^{(k+1)}_{(v_i,v_j)} = (Z^{(k)}_{(v_i,v_j)} + \beta \sum_{d \in \mathcal{N}_{(v_i,v_j)}} Z^{(k)}_d) W^{(k)}, \tag{3}$$

where $\mathcal{N}_{(v_i,v_j)}$ is the set of neighbors of node (v_i, v_j) in the line graph, $W^{(k)}$ is the weight matrix for the k-th layer, β is a normalization coefficient. The input for the first layer of graph convolution neural network is set to node attribute in the line graph as $Z^0_{(v_i,v_j)} = l_{(v1,v2)}$. We then consider the link prediction task as a binary classification problem and train the neural network by minimizing the cross-entropy loss for all potential links as:

$$\mathcal{L}_{CE} = -\sum_{l \in L_t} (y_l \log(p_l) + (1 - y_l) \log(1 - p_l)), \tag{4}$$

where L_t is the set of target links to be predicted, p_l is the probability that the link l exists in the graph, and $y_l \in \{0, 1\}$ is the label of a target link that indicating whether the link exists or not.

Connection with Learning on Original Graphs The key idea of our proposed method is to learn edge features from the enclosing subgraph and predict the existence of edge using the features. In this work, the feature of edge $e = (v_1, v_2)$ is learned based on attributes of two end nodes as:

$$f_e = g(f_l(v1), f_l(v2)), \tag{5}$$

where f_e is the edge feature, $g(\cdot)$ is the graph neural network function. Although graph convolutional layers are performed on the line graph, it still has connections with the same operation on the original graph. We use the first layer graph convolution layer as an example to illustrate this relationship. We reformulate the equation (3) as:

$$\begin{aligned} Z^1_{(v_i,v_j)} = (l_{(v1,v2)} &+ \beta \sum_{d_1 \in \mathcal{N}_{v1}} \sum_{d_2 \in \mathcal{N}_{d1}} l_{(d1,d2)} \\ &+ \beta \sum_{d_3 \in \mathcal{N}_{v2}} \sum_{d_4 \in \mathcal{N}_{d3}} l_{(d3,d4)}) W^{(0)}. \end{aligned} \tag{6}$$

The graph convolution operation learns embedding for each node by aggregating node embedding from its $1-$hop neighbors. It can be seen from equation (6) that the graph convolution on the line graph can aggregate the node embedding from $2-$hop neighbors. In the line graph transformation procedure, each node attribute is derived from two corresponding node attributes. That is, the attribute of each node in the line graph contains attributes from two nodes in the original graph. Therefore, aggregating information from $1-hop$ neighbors is equivalent to performing the same operation on $2-$hop neighbors. It also shows that learning node embedding through graph convolution in the line graph is more efficient than that in the original graph in terms of neighbor embedding aggregation.

3.4 The Proposed Algorithm

In this section, we provide a detailed description of the three components for our proposed framework. There is no strict restriction for the three steps. The given enclosing subgraph extraction and graph topology labeling function in this section can work well for most networks.

Enclosing Subgraph Extraction The existence of the link between two nodes can be determined by the graph topology centered on them. In general, we can achieve better performance when more topology information is involved. However, it will incur more computation cost. To seek a balance between performance and computation cost, we predict the existence of the link between node v_i and v_j using $2-$hop enclosing subgraph as:

$$G^2_{(v_i,v_j)} = \{v | \min(d(v,v_i), d(v,v_j) \leq 2)\}, \tag{7}$$

where $d(v,v_i)$ is the shortest path/geodesic distance between v and v_i.

Node Labeling Given an enclosing subgraph, we only know the topology of the graph. Before we learn features of the target link, we need to identify the role of each node in the graph through a labeling function. The node labeling function must satisfy the following criteria: (1) Identifying the two target nodes. (2) Provide the structural importance of each node to the target nodes. In this work, we employ an effective node labeling function proposed by [14] as:

$$f_l(v) = 1+\min(d(v,v_1), d(v,v_2))+(d_s/2)[(d_s/2)+(d_s)\%2-1], \tag{8}$$

where $d_s = d(v,v_1) + d(v,v_2)$, $(d_s/2)$ and $(d_s\%2)$ are the integer quotient and remainder of d divided by 2, respectively. In addition, the two target nodes v_1 and v_2 are assigned with label 1 as $f_l(v_1) = 1$ and $f_l(v_2) = 1$. For any node v satisfying $d(v,v_1) = \infty$ or $d(v,v_2) = \infty$, it will be assigned with label 0 as $f_l(v) = 0$. The node labeling function provides a label $f_l(\cdot) \in \mathbb{R}$. In practice, the node label is represented as a one-hot vector. As discussed above, the edge is represented as an order invariant pair. The edge feature pair is represented as a concatenation of two one-hot vectors. Algorithm 1 shows the link prediction procedure using our proposed framework.

Algorithm 1 Link Prediction Model Based on Line Graph Neural Networks

1: **Input**: Target link (v_1, v_2), Graph G
2: **Output**: Prediction result
3: Extract $h-$hop enclosing subgraph $G^h_{(v_1,v_2)}$
4: Apply node labeling function (8) to $G^h_{(v_1,v_2)}$
5: Generate edge attribute using equation (1) or (2)
6: Transform $G^h_{(v_1,v_2)}$ to line graph $L(G^h_{(v_1,v_2)})$
7: Apply graph neural networks to extract node embeddings on $L(G^h_{(v_1,v_2)})$ to predict the existence of link

4 Experiments

In this section, we evaluate our proposed method on 14 different datasets for the link prediction task. Two evaluation metrics, including area under the curve (AUC) and average precision (AP) are employed in this work to measure the performance of different models. The code and dataset used in this work are available at https://github.com/divelab/LGLP.

TABLE 1
Summary of datasets used in our experiments. The number of node, link, average node degree, and graph type are provided for each dataset.

Name	#Nodes	#Links	Degree	Type
BUP	105	441	8.4	Political Blogs
C.ele	297	2148	14.46	Biology
USAir	332	2126	12.81	Transportation
SMG	1024	4916	9.6	Co-authorship
EML	1133	5451	9.62	Shared Emails
NSC	1461	2742	3.75	Co-authorship
YST	2284	6646	5.82	Biology
Power	4941	6594	2.669	Power Network
KHN	3772	12718	6.74	Co-authorship
ADV	5155	39285	15.24	Social Network
GRQ	5241	14484	5.53	Co-authorship
LDG	8324	41532	9.98	Co-authorship
HPD	8756	32331	7.38	Biology
ZWL	6651	54182	16.29	Co-authorship

4.1 Datatsets and Baseline Models

In this work, we perform our proposed line graph link prediction (LGLP) model on 14 different datasets, including BUP, C.ele, HPD, YST, SMG, NSC, KHN, GRQ, LDG, ZWL, USAir, EML, Power, and ADV[1] [38], [39]. To demonstrate that our proposed method can work well in different areas, 14 datasets are collected from 6 areas. In addition, graphs in different scales, including the number of nodes and links, are used in the experiments. The details of the datasets are shown in Table 1.

In this work, we compare our proposed method with three high-order heuristic methods including Katz [24], PageRank (PR) [25], SimRank (SR) [26]. In addition, graph embedding method node2vec (N2V) [40], variational graph auto-encoder (GAE) [31], and the state-of-the-art method SEAL [14] are selected as baseline methods. To explore the performance of line graph neural network, we add another baseline method using neural relational inference (NRI) [41] to learn the target link embedding.

4.2 Experimental Setup

To assess the performance of our proposed method, we randomly select 50% of existing links as positive training samples, and the rest are used as positive test samples. In addition, the same number of non-existed links are randomly selected from the graph as negative samples for training and testing. To demonstrate the effectiveness of our proposed method with a different number of training samples, we also select 80% training links for the experiments.

The parameters of baseline methods are tuned to achieve the best performance on datasets. The damping factor in Katz method is set to 0.001. The damping factor in PageRank is set to 0.85. The constant factor in the SimRank is set to 0.8. The dimension of node embedding for node2vec is set to 128.

1. https://noesis.ikor.org/datasets/link-prediction

TABLE 2
AUC comparison with baseline methods (80% training links).

Model	BUP	C.ele	USAir	SMG	EML	NSC	YST
Katz	87.10(±2.73)	84.84(±2.05)	92.01(±0.88)	86.09(±1.06)	88.45(±0.68)	98.00(±0.31)	80.56(±0.78)
PR	90.13(±2.45)	**89.14**(±1.35)	93.74(±1.01)	89.13(±0.90)	89.46(±0.63)	98.05(±0.29)	81.40(±0.75)
SR	85.47(±2.75)	75.65(±2.24)	79.21(±1.50)	78.39(±1.14)	86.90(±0.71)	97.19(±0.48)	73.93(±0.95)
N2V	80.25(±5.55)	80.08(±1.52)	85.40(±0.96)	78.30(±1.22)	83.06(±1.42)	96.23(±0.95)	77.07(±0.36)
GAE	90.16(±1.65)	83.73(±0.75)	91.80(±0.86)	85.88(±0.90)	86.78(±1.07)	98.83(±0.33)	77.07(±0.36)
SEAL	93.32(±0.84)	87.44(±1.21)	95.21(±0.77)	91.53(±0.46)	92.01(±0.38)	99.55(±0.01)	82.07(±0.96)
NRI	94.77(±0.60)	90.13(±0.82)	96.44(±0.38)	91.18(±0.35)	91.83(±0.30)	99.80(±0.01)	91.63(±0.25)
LGLP	**95.24**(±0.53)	**90.16**(±0.76)	**97.44**(±0.32)	**92.53**(±0.29)	**92.03**(±0.28)	**99.82**(±0.01)	**91.97**(±0.12)
Model	Power	KHN	ADV	LDG	HPD	GRQ	ZWL
Katz	59.59(±1.51)	84.60(±0.79)	92.13(±0.21)	92.96(±0.19)	85.47(±0.35)	89.81(±0.59)	96.42(±0.12)
PR	59.88(±1.51)	88.43(±0.80)	92.78(±0.18)	94.46(±0.19)	87.19(±0.34)	89.98(±0.57)	97.20(±0.12)
SR	70.18(±0.75)	79.55(±0.90)	86.18(±0.22)	90.95(±0.14)	81.73(±0.37)	89.81(±0.58)	95.97(±0.16)
N2V	70.37(±1.15)	82.21(±1.19)	77.70(±0.83)	91.88(±0.56)	79.61(±1.14)	91.33(±0.53)	94.38(±0.51)
GAE	69.84(±0.96)	84.37(±0.39)	90.55(±0.23)	93.84(±0.21)	85.21(±0.45)	91.15(±0.45)	95.46(±0.30)
SEAL	81.37(±0.93)	92.69(±0.14)	95.07(±0.13)	96.44(±0.13)	92.26(±0.09)	97.10(±0.12)	97.46(±0.02)
NRI	82.15(±0.50)	92.63(±0.11)	94.53(±0.10)	96.49(±0.08)	91.63(±0.08)	97.21(±0.11)	97.44(±0.01)
LGLP	**82.17**(±0.57)	**93.30**(±0.09)	**95.40**(±0.10)	**96.70**(±0.07)	**92.58**(±0.08)	**97.68**(±0.10)	**97.76**(±0.01)

TABLE 3
AP comparison with baseline methods (80% training links).

Model	BUP	C.ele	USAir	SMG	EML	NSC	YST
Katz	85.94(±3.46)	85.94(±3.46)	93.51(±0.79)	87.68(±0.90)	90.54(±0.53)	98.02(±0.43)	85.76(±0.64)
PR	89.53(±3.11)	87.96(±1.69)	94.30(±1.27)	91.07(±0.59)	91.01(±0.67)	98.08(±0.34)	86.34(±0.72)
SR	81.10(±3.31)	66.43(±2.39)	69.80(±1.99)	70.39(±1.67)	87.24(±0.84)	96.55(±1.14)	77.56(±1.09)
N2V	81.47(±4.48)	77.98(±1.54)	82.53(±1.12)	77.01(±1.79)	83.08(±1.36)	96.81(±0.86)	78.48(±1.03)
GAE	89.26(±2.10)	82.53(±1.51)	93.41(±0.67)	85.95(±0.67)	88.73(±0.92)	98.93(±0.31)	82.65(±0.86)
SEAL	93.58(±0.68)	86.49(±1.08)	95.46(±0.59)	91.90(±0.31)	91.93(±0.31)	99.51(±0.01)	91.85(±0.20)
NRI	94.88(±0.50)	89.58(±0.65)	96.68(±0.31)	91.23(±0.26)	92.28(±0.28)	99.80(±0.01)	92.57(±0.23)
LGLP	**95.46**(±0.43)	**89.70**(±0.53)	**97.37**(±0.25)	**92.92**(±0.21)	**92.61**(±0.23)	**99.82**(±0.01)	**92.98**(±0.10)
Model	Power	KHN	ADV	LDG	HPD	GRQ	ZWL
Katz	74.29(±0.83)	88.27(±0.32)	93.72(±0.16)	94.91(±0.27)	89.52(±0.32)	93.08(±0.29)	97.08(±0.09)
PR	74.74(±0.81)	92.17(±0.24)	94.03(±0.24)	96.26(±0.22)	91.01(±0.23)	93.18(±0.34)	97.69(±0.08)
SR	70.69(±0.67)	77.16(±0.81)	83.31(±0.35)	88.71(±0.79)	84.16(±0.42)	92.97(±0.31)	95.44(±0.15)
N2V	76.55(±0.75)	83.26(±0.79)	79.02(±0.65)	92.12(±0.50)	80.57(±0.81)	93.92(±0.31)	93.82(±0.39)
GAE	75.04(±0.87)	87.52(±1.17)	90.87(±0.26)	95.24(±0.19)	86.62(±0.39)	93.78(±0.33)	95.79(±0.27)
SEAL	83.91(±0.83)	93.40(±0.13)	95.18(±0.12)	96.55(±0.11)	93.41(±0.09)	97.86(±0.11)	97.54(±0.02)
NRI	83.90(±0.52)	93.52(±1.21)	94.73(±0.11)	96.86(±0.07)	92.81(±0.08)	97.76(±0.11)	97.59(±0.01)
LGLP	**84.78**(±0.53)	**94.14**(±0.09)	**95.72**(±0.08)	**96.86**(±0.06)	**93.65**(±0.08)	**98.14**(±0.10)	**97.91**(±0.01)

For the SEAL framework, we employ the same setting as the original paper [14]. The $2-hop$ enclosing subgraph is extracted for the SEAL framework, and the labeling function is the same as equation (8) in this work. Three graph convolution layers are employed to compute node embeddings, and the sort pooling [15] is used to generate a fixed-size feature vector for the enclosing subgraph. The output feature map for three graph convolution layers is set to 32. The ratio of the sort pooling layer is set to 0.6. Two 1-D convolution layers with the number of output channels as 16 and 32, and two fully connected layers are employed as a classifier to predict the existence of a link. The SEAL model is trained for 50 epochs on each dataset.

To guarantee the comparison between our proposed method and SEAL model is fair, we employ the same graph neural network architecture to compute node embeddings in the line graph. It is worth noting that our proposed method does not employ graph pooling and 1-D convolution layers. Therefore, the number of parameters in our proposed method is much fewer than that in the SEAL model. Our proposed method is trained for 15 epochs on each dataset.

4.3 Results and Analysis

Plain Graph Link Prediction We perform our proposed method and baseline methods on 14 datasets to compare the performance of each model. We randomly split each dataset into training and testing dataset for ten times. The averaged AUC and standard deviations using 80% training links are shown in Table 2. The results in terms of AP are shown in Table 3. It can be seen from results that heuristic methods cannot achieve satisfactory performance on all datasets since the heuristic function is manually designed thus cannot handle different cases. We find that the state-of-the-art model SEAL always outperforms all heuristic methods and embedding methods since it can learn the distribution of links automatically from datasets. The NRI method can outperform the SEAL method and the results also show that learn the embedding of target link directly is better than learning graph embedding. Our proposed LGLP model can consistently achieve better performance than all baseline methods, including SEAL and NRI in terms of two evaluation metrics. It shows that our proposed method can learn better features to represent the target link for prediction in line graph space. In addition, our proposed method is more stable than other baseline methods.

TABLE 4
AUC comparison with baseline methods (50% training links).

Model	BUP	C.ele	USAir	SMG	EML	NSC	YST
Katz	81.61(±3.40)	79.99(±0.59)	88.91(±0.39)	80.65(±0.58)	84.16(±0.64)	95.99(±0.62)	77.28(±0.37)
PR	84.07(±3.39)	**84.95**(±0.58)	90.57(±0.39)	84.59(±0.45)	85.43(±0.63)	96.06(±0.60)	77.90(±3.69)
SR	80.98(±3.03)	76.05(±0.80)	81.09(±0.59)	75.28(±0.74)	83.05(±0.64)	95.59(±0.68)	73.71(±0.41)
N2V	80.94(±2.65)	75.53(±1.23)	84.63(±1.58)	73.50(±1.22)	80.15(±1.26)	94.20(±1.25)	73.62(±0.74)
GAE	82.31(±1.34)	80.34(±0.76)	89.71(±0.63)	84.36(±0.71)	82.04(±0.55)	97.79(±0.57)	80.06(±0.59)
SEAL	85.10(±0.82)	81.23(±1.52)	93.23(±1.46)	86.56(±0.53)	85.83(±0.46)	99.07(±0.02)	85.56(±0.28)
NRI	87.88(±0.69)	83.62(±0.95)	94.59(±0.82)	88.53(±0.48)	86.53(±0.37)	99.28(±0.01)	87.42(±0.26)
LGLP	**88.57**(±0.52)	84.60(±0.82)	**95.18**(±0.33)	**89.54**(±0.36)	**86.77**(±0.26)	**99.33**(±0.01)	**87.63**(±0.15)
Model	Power	KHN	ADV	LDG	HPD	GRQ	ZWL
Katz	57.34(±0.51)	78.99(±0.20)	90.04(±0.17)	88.61(±0.19)	81.60(±0.12)	82.50(±0.21)	93.72(±0.06)
PR	57.34(±0.52)	82.34(±0.21)	90.97(±0.15)	90.50(±0.19)	83.15(±0.17)	82.64(±0.22)	95.11(±0.09)
SR	56.16(±0.45)	75.87(±0.19)	84.87(±0.14)	87.95(±0.14)	78.88(±0.22)	82.68(±0.24)	94.00(±0.10)
N2V	55.40(±0.84)	78.53(±0.72)	74.67(±0.98)	88.82(±0.44)	75.84(±1.03)	84.24(±0.35)	92.06(±0.61)
GAE	56.75(±1.93)	82.65(±0.38)	90.12(±0.17)	89.95(±0.23)	83.71(±0.34)	83.18(±0.44)	94.14(±0.23)
SEAL	65.80(±1.10)	87.43(±0.17)	92.75(±0.14)	92.98(±0.16)	88.05(±0.10)	90.07(±0.15)	94.94(±0.02)
NRI	66.94(±0.61)	88.17(±0.16)	92.86(±0.12)	92.80(±0.13)	88.07(±0.09)	90.75(±0.12)	95.10(±0.01)
LGLP	**66.94**(±0.60)	**88.88**(±0.13)	**93.28**(±0.10)	**93.43**(±0.11)	**88.65**(±0.09)	**91.31**(±0.11)	**95.51**(±0.01)

TABLE 5
AP comparison with baseline methods (50% training links).

Model	BUP	C.ele	USAir	SMG	EML	NSC	YST
Katz	85.94(±2.03)	83.99(±0.79)	93.51(±0.35)	87.68(±0.79)	80.54(±0.31)	98.02(±0.53)	81.63(±0.41)
PR	89.53(±2.58)	**87.96**(±0.86)	94.30(±0.49)	91.07(±0.69)	91.01(±0.52)	98.08(±0.59)	82.08(±0.46)
SR	81.09(±2.57)	66.43(±1.17)	69.78(±0.84)	70.39(±0.96)	87.24(±0.52)	96.55(±0.75)	76.02(±0.49)
N2V	76.05(±3.20)	73.37(±1.23)	81.03(±1.18)	73.32(±1.34)	81.12(±0.92)	95.32(±1.08)	76.61(±0.94)
GAE	81.30(±2.14)	79.75(±0.92)	91.00(±0.59)	84.96(±0.68)	84.58(±1.59)	98.20(±0.37)	81.35(±0.68)
SEAL	84.17(±0.62)	83.94(±1.31)	94.31(±1.13)	86.76(±0.41)	87.45(±0.41)	99.09(±0.02)	86.45(±0.25)
NRI	87.99(±0.58)	83.72(±0.92)	94.72(±0.73)	89.02(±0.34)	88.33(±0.36)	99.30(±0.01)	88.84(±0.23)
LGLP	**89.03**(±0.41)	84.80(±0.63)	**94.89**(±0.33)	**90.23**(±0.26)	**88.49**(±0.23)	**99.38**(±0.01)	**89.22**(±0.13)
Model	Power	KHN	ADV	LDG	HPD	GRQ	ZWL
Katz	57.63(±0.51)	83.04(±0.38)	91.76(±0.15)	91.57(±0.17)	85.73(±0.89)	86.59(±0.20)	95.12(±0.05)
PR	57.61(±0.56)	87.18(±0.26)	92.43(±0.17)	93.53(±0.14)	87.20(±0.15)	86.73(±0.20)	96.24(±0.05)
SR	56.19(±0.49)	75.87(±0.66)	83.22(±0.20)	88.11(±0.25)	81.07(±0.18)	86.27(±0.20)	94.26(±0.11)
N2V	60.46(±0.86)	80.60(±0.74)	76.70(±0.82)	89.57(±0.64)	77.66(±0.54)	88.70(±0.26)	91.61(±0.49)
GAE	60.50(±2.26)	85.29(±0.34)	90.60(±0.16)	92.63(±0.16)	85.60(±0.28)	88.15(±0.29)	94.95(±0.18)
SEAL	68.67(±0.98)	90.37(±0.16)	93.52(±0.13)	94.33(±0.15)	90.25(±0.10)	92.80(±0.12)	95.88(±0.02)
NRI	68.53(±0.52)	90.42(±0.13)	93.34(±0.11)	94.24(±0.12)	89.92(±0.09)	92.41(±0.11)	95.83(±0.01)
LGLP	**69.41**(±0.50)	**90.83**(±0.11)	**93.82**(±0.10)	**94.63**(±0.10)	**90.34**(±0.09)	**93.01**(±0.10)	**96.19**(±0.01)

To demonstrate our proposed method can still achieve satisfactory performance with limited training samples, we conduct experiments on all datasets using 50% training links. The averaged AUC and AP are shown in Table 4 and Table 5, respectively. It can be seen from the results that our proposed method outperforms all baseline methods significantly on most datasets. We find that our proposed method can still perform well, even using 50% training links. The AUC and AP are close to that of using 80% training links. To better illustrate the performance of our proposed method, we extracted the output of penultimate fully connected layer as the feature of each edge and visualize edge features using t-distributed stochastic neighbor embedding (t-SNE) [42]. The visualization results are shown in Figure 5. We show the visualization on EML, ADV, HPD, and GRQ datasets for 20% test edges. Positive links are marked as green, and negative links are marked as cyan. It can be seen from the results that features learned from our proposed model can be easily classified.

In the experiments, we dynamically take 30%, 40%, 50%, 60%, 70%, and 80% of all the links in G as the training set and the rest as the test set, respectively. We conduct experiments with different training percentages and describe the AUC results in Figure 4. The AUC value of our proposed method is marked with a sold line, and other baseline methods are marked with dashed lines in different colors. It can be seen from the results that our proposed method can outperform all baseline methods with different percentages of the training data. In addition, the performance of our proposed method is not sensitive to the number of training samples.

TABLE 6
Comparison on Cora dataset using plain graph and attributed graph (50% training links).

	Attribute		Plain	
	AUC	AP	AUC	AP
SEAL	75.33	77.69	79.95	82.91
LGLP	**81.45**	**81.99**	**79.96**	**83.30**

Attributed Graph Link Prediction We also conduct experiments on attributed graphs. Since the heuristic method can only be applied to plain graphs, we mainly focus on the comparison between our proposed method and SEAL. In the SEAL framework, the attribute is concatenated with the node label as the input for graph neural networks. In this

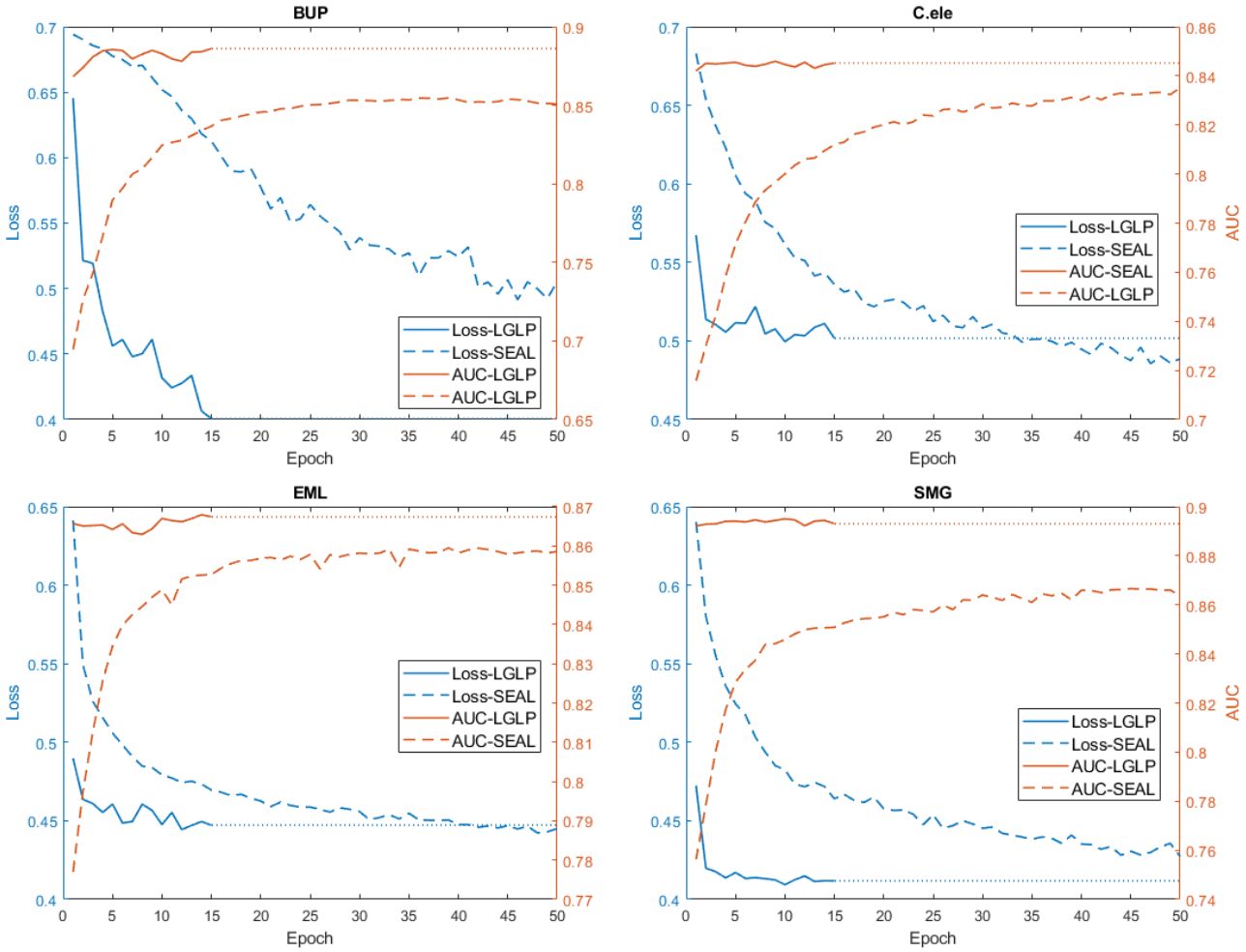


Fig. 3. Training loss and test AUC comparison between our proposed LGLP and SEAL method. The training loss and testing AUC on BUP, C.ele, EML, and SMG dataset. The training loss and testing AUC of LGLP are marked with blue and orange solid lines. Those of SEAL are marked with blue, orange dashed lines.

work, we propose a new function to combine the node label and node attributes. We perform the experiment on Cora dataset [43] that contains 2,708 nodes and 5,429 links. Each node in the Cora dataset is associated with an attribute vector in 1433 dimensions. We conduct the experiments without node attribute first, and then involve the node attributes to analyze the performance. The results are shown in Table 6. We can find both AUC and AP decrease after using node attributes as input in the SEAL model. In our proposed method, the performance does not change significantly. It shows that our proposed method can work well for both plain and attributed graphs.

Convergence Speed Analysis Our proposed method can learn features for the target link directly in the line graph. Therefore, only graph convolution layers are required to extract features. In the SEAL model, the procedure is completed in the original graph and thus requires graph convolution and pooling layers to achieve this goal. Compared with the SEAL model, our proposed method contains fewer parameters and converges faster. To analyze the converging speed of two models, we run the models on different datasets and collect the loss and test AUC value for each epoch. The result is shown in Figure 3. The loss and AUC of our proposed method are marked with solid lines. Those of the SEAL model are marked with dashed lines. It can be seen from the results that our proposed model can converge faster than the SEAL. Only 10 to 15 epochs are required to achieve the best performance for our proposed method. It takes 50 epochs for SEAL to converge. Therefore, our proposed method saves training time and requires fewer model parameters.

5 Conclusion

In this work, we propose a novel link prediction model based on line graph neural networks. Graph neural networks have achieved promising performance for the link prediction task. To deal with graphs in different scales, graph pooling layers are employed to extract a fixed-size feature vector in predicting the existence of a link. However, valuable information can be ignored in the pooling operation. In addition, graph neural networks with pooling layers commonly require more training time to converge. To overcome these limitations, we propose to transform the original input graph into line graph and thus the feature of the target link can be learned directly in the line graph without pooling operation. Experimental results on 14 datasets

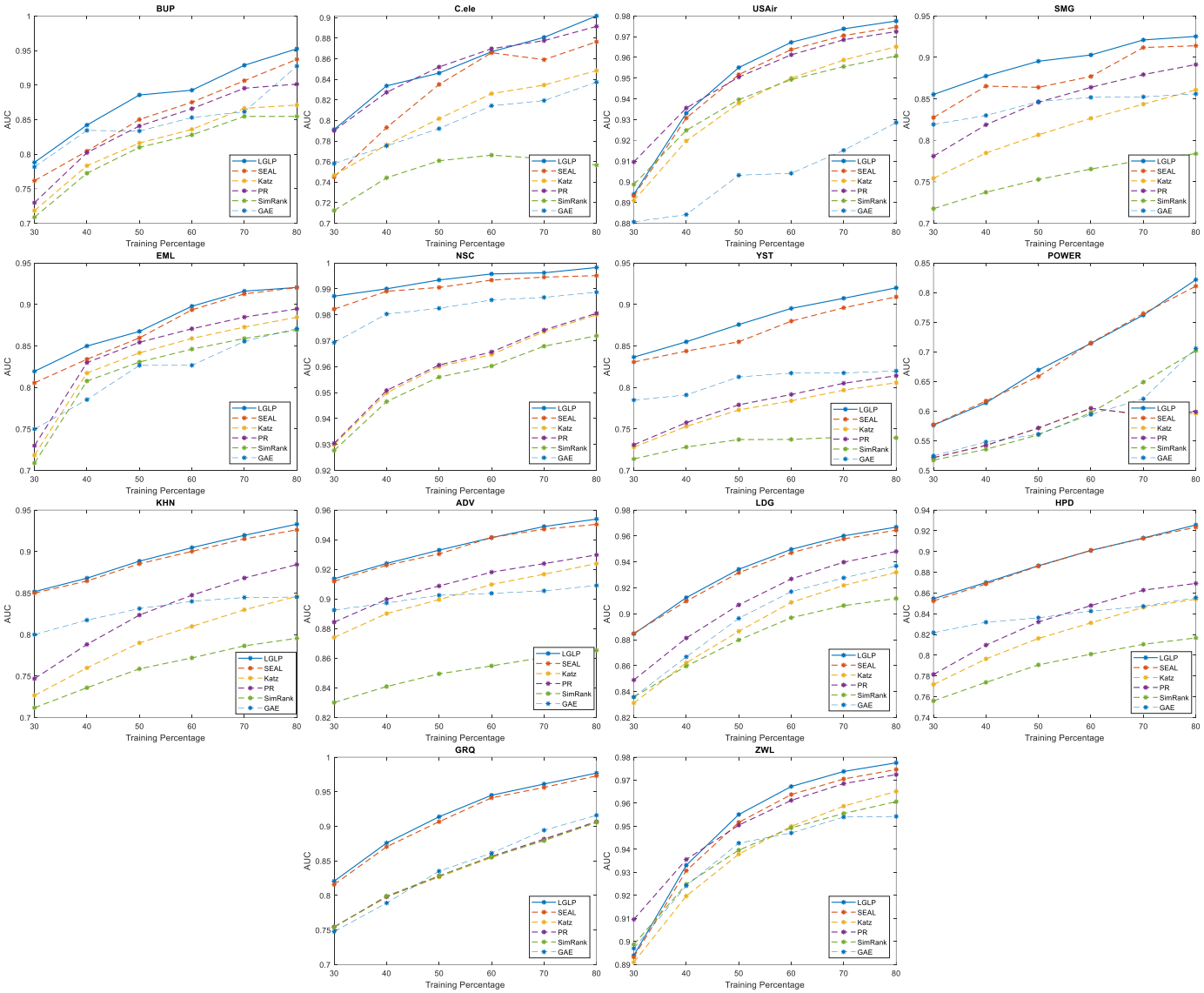


Fig. 4. AUC comparison on all datasets for Katz, PR, SR, SEAL, and LGLP using different percent of training links. On each dataset, we take 30%, 40%, 50%, 60%, 70%, and 80% of all the links in G as the training set. Our proposed method LGLP is marked with solid line and all baseline methods are marked with dashed lines in different colors.

from different areas demonstrate that our proposed method can outperform all baseline methods, including the state-of-the-art models. In addition, our proposed method can converge faster than the state-of-the-art model significantly.

ACKNOWLEDGMENT

This work was supported in part by National Science Foundation grant DBI-2028361.

REFERENCES

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[2] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[3] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," in *SDM06: workshop on link analysis, counter-terrorism and security*, vol. 30, 2006, pp. 798–805.

[4] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[5] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[7] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 968–977.

[8] J. You, Y. Wang, A. Pal, P. Eksombatchai, C. Rosenburg, and J. Leskovec, "Hierarchical temporal convolutional networks for dynamic recommender systems," in *The world wide web conference*, 2019, pp. 2236–2246.

[9] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.

[10] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *Journal of machine learning research*, vol. 9, no. Sep, pp. 1981–2014, 2008.

[11] T. Oyetunde, M. Zhang, Y. Chen, Y. Tang, and C. Lo, "Boostgapfill: Improving the fidelity of metabolic network reconstructions

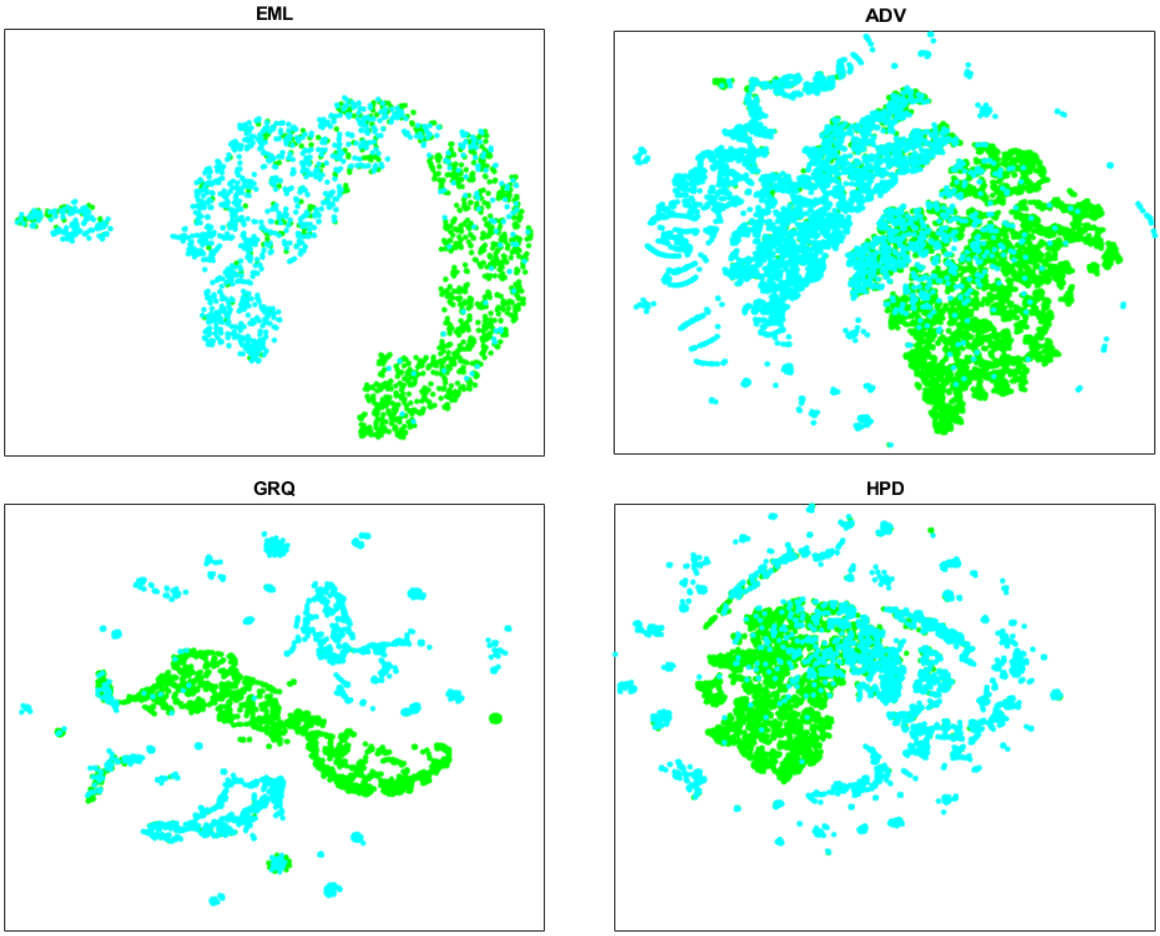


Fig. 5. t-SNE visualization on EML, ADV, HPD, and GRQ datasets. Positive links are shown as green, and negative links are shown as cyan.

through integrated constraint and pattern-based methods," *Bioinformatics*, vol. 33, no. 4, pp. 608–611, 2017.

[12] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[13] I. A. Kovács, K. Luck, K. Spirohn, Y. Wang, C. Pollis, S. Schlabach, W. Bian, D.-K. Kim, N. Kishore, T. Hao *et al.*, "Network-based prediction of protein interactions," *Nature communications*, vol. 10, no. 1, p. 1240, 2019.

[14] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5165–5175.

[15] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification." in *AAAI*, vol. 18, 2018, pp. 4438–4445.

[16] H. Gao and S. Ji, "Graph u-nets," in *International Conference on Machine Learning*, 2019, pp. 2083–2092.

[17] H. Yuan and S. Ji, "Structpool: Structured graph pooling via conditional random fields," in *International Conference on Learning Representations*, 2019.

[18] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in neural information processing systems*, 2018, pp. 4800–4810.

[19] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1416–1424.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[21] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2016, pp. 2014–2023.

[22] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[23] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *The European Physical Journal B*, vol. 71, no. 4, pp. 623–630, 2009.

[24] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[25] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Computer networks*, vol. 56, no. 18, pp. 3825–3833, 2012.

[26] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 538–543.

[27] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.

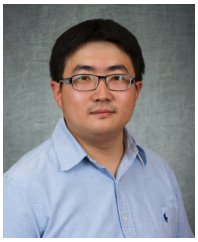
[28] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[29] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[30] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *Proceedings of the Eleventh ACM International*

Conference on Web Search and Data Mining. ACM, 2018, pp. 459–467.
[31] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.
[32] M. Zhang and Y. Chen, "Weisfeiler-lehman neural machine for link prediction," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 575–583.
[33] L. Cai and S. Ji, "A multi-scale approach for graph link prediction." in *AAAI*, 2020, pp. 3308–3315.
[34] F. Harary and R. Z. Norman, "Some properties of line digraphs," *Rendiconti del Circolo Matematico di Palermo*, vol. 9, no. 2, pp. 161–168, 1960.
[35] Z. Chen, L. Li, and J. Bruna, "Supervised community detection with line graph neural networks," in *International Conference on Learning Representations*, 2019.
[36] N. D. Roussopoulos, "A max {m, n} algorithm for determining the graph h from its line graph g," *Information Processing Letters*, vol. 2, no. 4, pp. 108–112, 1973.
[37] P. G. Lehot, "An optimal algorithm to detect a line graph and output its root graph," *Journal of the ACM (JACM)*, vol. 21, no. 4, pp. 569–575, 1974.
[38] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, p. 440, 1998.
[39] M. E. Newman, "The structure of scientific collaboration networks," *Proceedings of the national academy of sciences*, vol. 98, no. 2, pp. 404–409, 2001.
[40] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.
[41] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2688–2697.
[42] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, pp. 857–864, 2002.
[43] L. Šubelj and M. Bajec, "Model of complex networks based on citation dynamics," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 527–530.

Lei Cai received the PhD degree in computer science from Washington State University, Pullman, WA, in 2020. Currently, he is a Data & Applied Scientist in Microsoft. His research interests include data mining, machine learning and deep learning.



Jundong Li is an Assistant Professor at the University of Virginia. Prior to joining UVA, he received his Ph.D. degree in Computer Science at Arizona State University in 2019, M.Sc. degree in Computer Science at the University of Alberta in 2014, and B.Eng. degree in Software Engineering at Zhejiang University in 2012. His research interests are broadly in data mining and machine learning, with a particular focus on graph representation learning and causal inference.

Jie Wang received the B.Sc. degree in electronic information science and technology from University of Science and Technology of China, Hefei, China, in 2005, and the Ph.D. degree in computational science from the Florida State University, Tallahassee, FL, in 2011. He is currently a professor in the Department of Electronic Engineering and Information Science at University of Science and Technology of China, Hefei, China. His research interests include reinforcement learning, knowledge graph, large-scale optimization, deep learning, etc..

Shuiwang Ji received the PhD degree in computer science from Arizona State University, Tempe, Arizona, in 2010. Currently, he is an Associate Professor in the Department of Computer Science and Engineering, Texas A&M University, College Station, Texas. His research interests include machine learning, deep learning, data mining, and computational biology. He received the National Science Foundation CAREER Award in 2014. He is currently an Associate Editor for IEEE Transactions on Pattern Analysis and Machine Intelligence, ACM Transactions on Knowledge Discovery from Data, and ACM Computing Surveys. He regularly serves as an Area Chair or equivalent roles for data mining and machine learning conferences, including AAAI, ICLR, ICML, IJCAI, KDD, and NeurIPS. He is a senior member of IEEE.