Learning Universal Network Representation via Link Prediction by Graph Convolutional Neural Network

Weiwei Gu  
*College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China*

Fei Gao  
*School of Systems Science, Beijing Normal University, Beijing 100875, China*

Ruiqi Li  
*College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China*

Jiang Zhang  
*School of Systems Science, Beijing Normal University, Beijing 100875, China*

Follow this and additional works at: [https://dc.tsinghuajournals.com/journal-of-social-computing](https://dc.tsinghuajournals.com/journal-of-social-computing)

Part of the Computer Engineering Commons, Computer Sciences Commons, and the Science and Technology Studies Commons

**Recommended Citation**  

This Research Article is brought to you for free and open access by Tsinghua University Press: Journals Publishing. It has been accepted for inclusion in *Journal of Social Computing* by an authorized editor of Tsinghua University Press: Journals Publishing.
Learning Universal Network Representation via Link Prediction by Graph Convolutional Neural Network

Weiwei Gu, Fei Gao, Ruiqi Li*, and Jiang Zhang*

Abstract: Network representation learning algorithms, which aim at automatically encoding graphs into low-dimensional vector representations with a variety of node similarity definitions, have a wide range of downstream applications. Most existing methods either have low accuracies in downstream tasks or a very limited application field, such as article classification in citation networks. In this paper, we propose a novel network representation method, named Link Prediction based Network Representation (LPNR), which generalizes the latest graph neural network and optimizes a carefully designed objective function that preserves linkage structures. LPNR can not only learn meaningful node representations that achieve competitive accuracy in node centrality measurement and community detection but also achieve high accuracy in the link prediction task. Experiments prove the effectiveness of LPNR on three real-world networks. With the mini-batch and fixed sampling strategy, LPNR can learn the embedding of large graphs in a few hours.

Key words: network representation; link prediction; deep learning

1 Introduction

Most real-world data naturally come in the form of pairwise relations, such as protein-protein interactions in human cells, citation relations in scientific research, and drug-target interactions in medicine discovery[1–3]. These pairwise relations contain rich information about node properties, network structures, and network evolutions. Network analysis, as a fundamental and important task in network science, has been attracting increasing attention in academia and industry. Recently, there has been a surge of research revealing network information through network representation learning algorithms, which can automatically extract local and global graph structures. The idea behind representation learning algorithms is to learn a mapping function that embeds nodes as points in a low-dimensional space, $\mathbb{R}^d$, by optimizing an objective function, such that the learned geometric relations can reflect the structure of the original graph. After the optimization process, the learned node representations can be used as feature inputs for downstream machine learning tasks.

Over the past decades, several embedding algorithms have been proposed and they can be classified into the following categories: matrix factorization based methods, random walk based methods, graph neural network based methods, and a combination of these miscellaneous strategies. Matrix factorization based methods, such as Laplacian eigenmap[4], Isomap[5], TADW[6], and their extensions, usually perform well on small graphs, but they cannot be effectively applied on larger networks containing millions of nodes and billions of edges. Based on the Skip-Gram model, Perozzi et al.[7] proposed DeepWalk, which can learn node representations from a collection of random walks and extract high-quality node representations without doing the matrix factorization operation. Similar to DeepWalk[7], node2vec[8] preserves a high-order proximity between nodes by maximizing the probability of node occurrence of the random walk sequence. Unlike

© The author(s) 2021. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).
DeepWalk, node2vec employs biased random walks, which provide a trade-off between breath-first and depth-first walking. However, both algorithms directly update the representation vectors of the nodes without considering any shared information.

Recently, deep neural network based representation learning algorithms have achieved success in image processing\cite{9} and Natural Language Processing (NLP)\cite{10}. This demonstrates the potential of deep learning algorithms to be applied for node classification and link prediction tasks by converting network structures into low-dimensional vector representations. For example, instead of directly updating node vectors, DNGR\cite{11} and SDNE\cite{12} use complicated encoder and decoder architecture to represent graph structures. Graph Convolutional Network (GCN)\cite{13} borrows the concept of convolution from graph convolutional neural networks and directly convolves the graph according to the graph structure. Velickovic et al.\cite{14} proposed the Graph Attention Network (GAT), which obtained a high accuracy in the node classification task. Following the self-attention mechanism, GAT computes representations of nodes by combining the neighborhood vectors in an adaptive way. Node pair relations are described as adjustable attention weight parameters that can be dynamically updated according to the states of the nodes.

Nevertheless, the algorithms mentioned above and their extensions\cite{15} encounter the problem of scalability, because they take the whole graph as an input and recursively expand neighborhoods across layers. Expanding neighborhoods is a computationally expensive operation, particularly for complex graphs. Due to the scale-free property of many real networks, when a hub node is sampled as the first-order neighbors, their second-order neighbors can quickly fill up the memory and the problem of memory bottleneck prevents GAT and GCN to be applied to large-scale networks.

GraphSAGE\cite{16} solves the problem of memory bottleneck by sampling a fixed neighborhood during the training process and then applies several aggregators to combine the features of the neighborhoods into single vector representations. The sampling strategy of GraphSAGE yields impressive performance on the node classification task over several large-scale networks. FastGCN\cite{17} views GCN\cite{13} as an integral transform of embedding functions under the probability measure. FastGCN achieves a competitive node classification accuracy while getting rid of the reliance of the test data.

The performance of representation-based tasks mainly relies on the quality of supervised information. Most neural network based methods, such as GAT\cite{14}, GraphSAGE\cite{16}, FastGCN\cite{17}, and GCN\cite{13}, use node labels as the supervised information. However, in the real world, node labels are scarce and only a few networks have supervised node label information. Compared with node label information, graph linkages contain richer information about graph structure and evolution. For instance, according to the similarity and popularity theories\cite{18}, the linkages within a network can not only reveal similarity relations of the nodes\cite{19} but also encode popularity information of the nodes\cite{20}. Considering the formation of a citation network as an example, citations not only rely on similarities in the topics of papers but also are related to the popularity of the papers\cite{21}. In this paper, we claim that to achieve high-quality node representations, linkages should be extensively used as the supervised information, because they encode the popularity as well as similarity. Predicting the missing links and forecasting the future ones are two of the most important and traditional questions in network science. For example, the discovery and validation of protein interactions require significant experimental efforts in biological research. Instead of blindly checking all possible node links, link prediction algorithms can reduce the experimental cost and help scientists focus on the most probable emerging links. For World Wide Web, social networks, citation networks, and link prediction can also help in recommending relevant web pages, finding new friends, and discovering new citations\cite{22-24}. Traditional link prediction methods can be divided into several groups. Approaches predict links according to the local similarity of a graph, based on the assumption that two nodes are more likely to be connected if they have many common neighbors\cite{20, 25}. These algorithms are fast and highly parallel, because they only consider the local structure. However, these algorithms cannot achieve high link prediction accuracies. Conversely, global similarity based methods use the topological information about the whole network to calculate the similarities of the nodes\cite{25-27}. Although they have high prediction accuracies, they usually suffer from high computational complexity, which prevents them from being applied on complex graphs with millions of nodes and billions of edges. There are also some probabilistic and statistical approaches that assume a known prior network structure, such as hierarchies or circles\cite{28, 29}. However, these methods cannot address the
problem of low link prediction accuracies. Furthermore, conventional approaches can hardly reveal hidden information about the nodes and the structures behind links.

In this paper, we propose a novel network representation model, named Link Prediction based Network Representation (LPNR). This approach extends the GAT algorithm\[14\] to the link prediction task. With the mechanism of the graph neural networks, LPNR can not only improve the accuracy of link prediction but also learn meaningful node representations, on which reasonable node ranking and proper community labels can be obtained in an unsupervised way. The original GAT algorithm cannot be directly applied in predicting missing links for the following reasons: Firstly, the complexities of the node classification and link prediction tasks are $O(N)$ and $O(N^2)$, respectively, where $N$ denotes the number of nodes. The link prediction task involves a large number of node feature computations. Secondly, the original GAT algorithm requires access of the entire network to make inferences on node labels. However, we cannot directly sample the node neighbors to form a mini-batch in the training process, due to the scale-free property of most networks, and the expansion of neighborhoods of the hub nodes can quickly fill up a large portion of the graph. Although large mini-batches are preferable to reduce the communication cost, they may also slow down the convergence rate\[30\]. During optimization, a decrease in the mini-batch was associated with an increase in the convergence rate\[31\]. Moreover, the existing Graphic Processing Unit (GPU)-enabled tensor can only manipulate on the same sized neighbor, which also causes the parallel problem in GAT. In this paper, we solve the problems of memory bottleneck and convergence by incorporating the mini-batch training strategy via sampling a fixed number of neighborhoods.

Various experiments have been implemented on several representative networks. Our experimental results show that LPNR can not only achieve state-of-the-art accuracy in link prediction but also obtain high-quality node representations for various downstream tasks, such as community detection and node ranking.

Our main contributions are summarized as follows:

- We propose LPNR, which achieves competitive accuracy in the link prediction task.
- We deal with the problems of memory bottleneck and mini-batch for large-network representations with a fixed sampling size strategy, which can reduce fluctuations, increase predictive powers, and yield controllable costs for per-batch computations.
  - We obtain high-quality node representations from link predictions, on which we can rank nodes according to their importance.
  - Compared with node representations learned from a supervised classification task, LPNR is better at detecting network communities.

Language models in the NLP field can extract effective representations of words and sentences for various downstream tasks by predicting nearby words\[32,33\]. Similarly, LPNR can also learn high-quality node representations for various downstream tasks. Thus, we claim that LPNR can be regarded as the “language model” for graphs.

2 Architecture of LPNR

2.1 Encoding node representations

LPNR aims at obtaining a sufficient expressive power to transfer input features into high-level node representations. It takes a set of original node features as the input, i.e., $\mathbf{h} = \{h_1, h_2, \ldots, h_N\}$, in which $h_i \in \mathbb{R}^F$, $F$ is the number of input attributes. We use $h'_i \in \mathbb{R}^{F'}$ to represent the output of LPNR and $F'$ is the number of output attributes. As shown in Eq. (1), we first apply a learnable linear transformation parameterized by a weight matrix, $W \in \mathbb{R}^{F' \times F}$, to make LPNR capable of learning linear relations. Unlike GAT algorithm, we do not use a single-layer feed-forward neural vector to compute the attention coefficients. During the exploration process, we find that the attention coefficients between nodes increase the training time, decrease the link prediction accuracy, introduce redundancy parameters, and make it difficult to train LPNR. In LPNR, we use multi-weight parameters to stabilize the learning process. In real-world networks, many (if not the majority of) legitimate links are actually not observed and the observed first-order proximity is not sufficient to preserve the global network structures. As a complement, we consider the second-order proximity of the nodes, which is determined through the shared neighborhood instead of the observed ties\[34\]. LPNR considers a two-hop feature transformation, when computing node feature representations. The feature aggregation process is shown in Eq. (1):

$$
\begin{align*}
\mathbf{h}'_i &= \sum_{k_1=1}^{K_1} \sigma \left( \sum_{j \in N_i} W_{k_2} \mathbf{k}_2 \right) \sum_{k_2=1}^{K_2} \sigma \left( \sum_{z \in N_j} W_{k_1} \mathbf{h}_z \right)
\end{align*}
$$

(1)
where $\|\|$ is the concatenation operation, node $j$ represents a neighbor of node $i$, the neighborhood of $i$ is denoted as $N_i$, $z$ is a neighbor of node $j$, the neighborhood of $j$ is represented as $N_j$, and $\sigma$ is a potentially nonlinear sigmoid function. The number of the first layer is denoted as $K_1$ and the number of the second layer is denoted as $K_2$.

LPNR consists of two layers. The primary purpose of the first layer is to quantify the influence of the neighbors on the features of the node. In this layer, we aggregate features from the first-order neighbors of the nodes and thereby feed the aggregated features into an exponential linear unit to add the nonlinearity of the features. We then gather the representation of the nodes from different weight matrices and concatenate their feature representations to form vectors. We feed the vector representations of the nodes into the second layer to form the representations of the edges. We use the logistic regression model to evaluate the edge with existing probability between the nodes.

### 2.2 Decoding node pair relations and the fixed sampling strategy

How to encode node pair relations with feature representations is a key task in revealing the graph structure. In LPNR, we encode node vectors into edge representations by calculating the Hadamard product between the nodes. Specifically, for node pairs $i$ and $j$, the edge vector $e_{ij}$ is represented as $h_i \odot h_j$. $e_{ij}$ is a $d$-dimensional vector, and the connection probability between $i$ and $j$ is represented with Eq. (2):

$$p_{ij}(e_{ij}; \theta) = \frac{1}{1 + \exp(e_{ij}^T \theta)} \tag{2}$$

where $\theta$ is also a $d$-dimensional parameter and $e_{ij}^T \theta$ is the dot product between vectors $e_{ij}$ and $\theta$. We use Eq. (2) to test the link prediction accuracy. Specifically, we consider that edge $(i, j)$ exists if $p_{ij}(e_{ij}; \theta) \geq 0.5$ and is missing otherwise. The best estimate of $\theta$ is trained with the training set via logistic regression. We quantify the link prediction accuracy of the test set with the F1-score.

To tackle the above-mentioned memory bottleneck and parallel problems, we propose a fixed-size sampling strategy. Before the training process, we first sample a fixed number of first- and second-order neighbors for each node. In this study, we set the size of the first- and second-order neighbors to 20. For nodes with less than 20 neighbors, we use a random selection mechanism to repeatedly add duplicated nodes. LPNR samples only once and then fixes the first and second neighbors during the training process, while GraphSAGE keeps sampling neighbors during the training process. The sampling strategy of LPNR is different from that of FastGCN, where nodes within a batch share the same neighbors and the sampling probabilities are proportional to the degrees of the nodes. The fixed sampling strategy can speed up the convergence, improve the link prediction accuracy, reduce the training fluctuations, and stabilize the training process. We provide explanations to this phenomenon in Section 3.5.

### 2.3 Training LPNR

LPNR encodes the network structure by optimizing the edge existence probability of the training set aiming at correctly predicting the node pair relations of the test set. During the training process, LPNR first randomly samples an equal number of disconnected node pairs as the connected ones with the constraint that there are no common neighbors between the disconnected node pairs. This constraint not only increases the robustness of LPNR but also decreases its training difficulties. After the sampling process, we randomly select 80% node pairs to form the training set, which is denoted as $\varepsilon$. We then equally split the remaining corpus into the validation set and the test set, which is denoted as $\phi$. LPNR is trained by minimizing the following objective function:

$$\mathcal{L} = -\frac{1}{|\varepsilon|} \sum_{(i,j)\in\varepsilon} y_{i,j} \log p_{i,j} + (1 - y_{i,j}) \log (1 - p_{i,j})$$

(3)

where $|\varepsilon|$ is the size of the training set and $y_{i,j}$ denotes the connectivity, with $y_{i,j} = 0$ for disconnected node pairs and 1 for connected ones.

### 3 Result

#### 3.1 Experimental setup

In this study, we use four networks and three tasks to evaluate the performance of LPNR. An overview of the four networks is given in Table 1. In our experiments, we evaluate the link prediction accuracy on Cora, Citeseer, and PubMed networks and test the node centrality accuracy with the American Physical Society (APS) citation network. We compare the representation qualities learned from the node classification and link prediction with the community detection task on Cora, Citeseer, and PubMed. To stabilize and accelerate the training process, LPNR takes the original node feature
vectors as the input. For Cora, Citeeser, and PubMed, we use the original content features that contain information about the paper’s abstract, whereas for the APS network, we use the sparse adjacency vector of the node to form raw input features.

PubMed’s network contains a large quantity of nodes and edges, and it has a more complicated structure. Thus, we use eight weight matrices \( (K_2 = 8) \) to make the second layer when training PubMed. The aggregated weight matrices are averaged in the second layer. The outputs of the second layer are the final representations of the network. The batch sizes are 32 for Cora, Citeseer, and PubMed, and 16 for the APS network. We initialize the parameter of LPNR with the Glorot initialization scheme\(^{[35]} \) and minimize the binary cross-entropy loss, as shown in Eq. (3). For the training set, we use the Adam stochastic gradient descent optimizer\(^{[36]} \), with an initial learning rate of \( 5 \times 10^{-4} \). We also apply early stopping strategy on the validation set over the link prediction accuracy with the training patience set to 100 epochs.

### 3.2 Link prediction

We extract meaningful node feature representations with the link prediction task. As described in Section 2.3, we first use the node pair relations in the training set \( \mathcal{E} \) to train LPNR and then evaluate the performance of LPNR with the link prediction accuracies in the test set \( \mathcal{F} \). We select six well-known representative link prediction algorithms to compare with, which include the classic node similarity based algorithms, such as RA\(^{[25]} \); shallow node feature representation algorithms, such as LINE\(^{[34]} \) and node2vec\(^{[8]} \); and some GCN-based algorithms, such as GraphSAGE\(^{[16]} \) and VGAE\(^{[37]} \).

Two standard metrics, i.e., accuracy and Area Under the receiver operating characteristic Curve (AUC), are used to quantify the link prediction accuracy. As shown in Table 2, LPNR outperforms most of the baseline methods across a variety of graphs. We report the mean results averaged over 10 runs. Finally, we obtain the pretrained node features to implement the following node centrality measurement and community detection tasks.

### 3.3 Centrality measurement

Node centrality measurement is another important and well-studied task in the network field\(^{[38]} \), with a variety of applications. With the understanding that network representations are capable of revealing hidden network structure and metric space of the nodes\(^{[39]} \), we use the well-trained node features extracted from LPNR to measure node centralities. As shown in Eq. (4), we first accumulate the Euclidean distances from the focal node \( i \) to all other nodes,

\[
\tilde{c}_i = \sum_{j} \sqrt{\sum_{f=1}^{F^\prime} (h^F_i - h^F_j)^2}
\]  

(4)

\( h^F_i \) is the embedding representation of node \( i \), \( h^F_j \) is the \( f \)-th component of \( i \)'s representation, and \( F^\prime \) represents the embedding dimension. After computing the accumulated \( \tilde{c}_i \), we rank nodes in an ascending order, according to their accumulated distances. The validation of Eq. (4) is based on the assumption that the central nodes have tighter connections and shorter distances to other nodes.

To evaluate the performance of centrality measurement using LPNR, we use the citation network APS, which is a subgraph extracted from APS journals with nodes representing papers and links representing citations. We follow a paper published in the journal, Science\(^{[40]} \), and evaluate the importance of the papers by counting the number of citations they have received during the first 10 years after its publication. We select six well-known representative link prediction algorithms to compare with, which include the classic node similarity based algorithms, such as RA\(^{[25]} \); shallow node feature representation algorithms, such as LINE\(^{[34]} \) and node2vec\(^{[8]} \); and some GCN-based algorithms, such as GraphSAGE\(^{[16]} \) and VGAE\(^{[37]} \).

Two standard metrics, i.e., accuracy and Area Under the receiver operating characteristic Curve (AUC), are used to quantify the link prediction accuracy. As shown in Table 2, LPNR outperforms most of the baseline methods across a variety of graphs. We report the mean results averaged over 10 runs. Finally, we obtain the pretrained node features to implement the following node centrality measurement and community detection tasks.

### 3.3 Centrality measurement

Node centrality measurement is another important and well-studied task in the network field\(^{[38]} \), with a variety of applications. With the understanding that network representations are capable of revealing hidden network structure and metric space of the nodes\(^{[39]} \), we use the well-trained node features extracted from LPNR to measure node centralities. As shown in Eq. (4), we first accumulate the Euclidean distances from the focal node \( i \) to all other nodes,

\[
\tilde{c}_i = \sum_{j} \sqrt{\sum_{f=1}^{F^\prime} (h^F_i - h^F_j)^2}
\]  

(4)

\( h^F_i \) is the embedding representation of node \( i \), \( h^F_j \) is the \( f \)-th component of \( i \)'s representation, and \( F^\prime \) represents the embedding dimension. After computing the accumulated \( \tilde{c}_i \), we rank nodes in an ascending order, according to their accumulated distances. The validation of Eq. (4) is based on the assumption that the central nodes have tighter connections and shorter distances to other nodes.

To evaluate the performance of centrality measurement using LPNR, we use the citation network APS, which is a subgraph extracted from APS journals with nodes representing papers and links representing citations. We follow a paper published in the journal, Science\(^{[40]} \), and evaluate the importance of the papers by counting the number of citations they have received during the first 10 years (the first 10 years citation number was denoted as \( c_{10} \)) after their publication. Measuring the centrality of articles not only helps scientists find significant and valuable discoveries among thousands of publications.

### Table 1

Overview of the datasets used in our experiments. In APS, nodes represent published papers, and edges denote citation relations. We quantify the importance of the paper by computing the accumulated citations that a paper has received within the first 10 years after its publication.

<table>
<thead>
<tr>
<th>Graph name</th>
<th>Number of nodes</th>
<th>Number of edges</th>
<th>Number of features</th>
<th>Number of categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2708</td>
<td>5429</td>
<td>1433</td>
<td>7</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3327</td>
<td>4732</td>
<td>3703</td>
<td>6</td>
</tr>
<tr>
<td>PubMed</td>
<td>19,717</td>
<td>44,338</td>
<td>3</td>
<td>500</td>
</tr>
<tr>
<td>APS</td>
<td>1012</td>
<td>3336</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

### Table 2

Link prediction accuracy and AUC of different algorithms over four networks. Here, VC represents venture capital.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Link prediction accuracy/AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPNR</td>
<td>0.88/0.93 0.86/0.91 0.90/0.97 0.82/0.90</td>
</tr>
<tr>
<td>GraphSAGE-mean</td>
<td>0.83/0.89 0.84/0.90 0.89/0.91 0.81/0.87</td>
</tr>
<tr>
<td>node2vec</td>
<td>0.82/0.92 0.85/0.89 0.77/0.87 0.83/0.89</td>
</tr>
<tr>
<td>VGAE</td>
<td>0.75/0.91 0.75/0.90 0.76/0.94 0.76/0.88</td>
</tr>
<tr>
<td>LINE</td>
<td>0.69/0.76 0.67/0.73 0.78/0.84 0.68/0.74</td>
</tr>
<tr>
<td>RA</td>
<td>0.41/0.75 0.32/0.73 0.33/0.76 0.35/0.78</td>
</tr>
</tbody>
</table>
but can also help institutes quantify the achievements of the authors. We report the Spearman’s rank-order correlation between the benchmark ranking results ($c_{10}$) and the ranking computed from other node centrality measurement algorithms, such as LPNR’s Euclidean distance.

Besides PageRank and node2vec centralities, we also compare LPNR with the closeness centrality and betweenness centrality. The closeness centrality is based on the assumption that important nodes should have shorter path lengths to other nodes, whereas the betweenness centrality assumes that the most important nodes should be involved in shorter paths. We also extract the well-trained node representations from GraphSAGE and use Eq. (4) to calculate the GraphSAGE ranking of the nodes, according to their accumulated Euclidean distances. From Fig. 1, we discover that LPNR ranking can significantly outperform other ranking methods in discovering latent important nodes.

3.4 Node clustering

GAT$^{[14]}$, GCN$^{[13]}$, unsupervised GraphSAGE$^{[16]}$, and other GCN-based embedding algorithms adopt node labels as supervised information and evaluate the embedding quality with the node classification accuracy of the test set. Compared with node labels, structural information, such as linkages and co-occurrence frequencies of the nodes, contains richer network information$^{[18]}$, because graph labels encode node similarity, whereas network structures encode not only similarity but also popularity$^{[18]}$. Link prediction based embedding algorithms, such as LPNR, are supposed to outperform node classification based embedding algorithms, such as GAT.

In this study, to fairly compare representations extracted from structural learning algorithms and node labeling learning representations, we introduce community detection as a third-party evaluation task. Community detection is a fundamental task in network study, which aims at finding sets of nodes that are similar to one another within a group. Community detection is an important tool for understanding network structures. For instance, community detection in the World Wide Web helps people find websites that offer similar content, and community detection of a protein interaction network helps people identify genes with similar functions. Due to the scarcity of node labels in most networks, we compare the representation qualities with Cora, Citeseer, and PubMed networks. We first extract node feature representations learned from the well-trained LPNR, LINE, unsupervised GraphSAGE-mean, and supervised GAT, and then use the $k$-means algorithm to automatically divide nodes into different groups (communities). Afterward, we compare the clustering outcome with the Louvain algorithm$^{[41]}$, which is a widely used community detection algorithm. The Louvain algorithm tries to maximize the modular degree of the whole complex network. Moreover, it is a classical community detection algorithm with a high accuracy of community detection. Finally, we use the Normalized Mutual Information (NMI) to quantify the difference in community detection by the Louvain algorithm and the above-mentioned embedding algorithms. The larger the NMI value, the more similar the outcome of the embedding algorithm with the Louvain algorithm and the better the representation quality. The experimental results are shown in Fig. 2.

From Fig. 2, we can find that the structural learning based algorithms, such as LPNR, unsupervised
Weiwei Gu et al.: Learning Universal Network Representation via Link Prediction by Graph Convolutional Neural Networks.

3.5 Parameter sensitivity and sampling strategy

In this part, we examine how the parameters of LPNR influence its performance by tuning the embedding dimensions (the parameter size of the last layer). We vary the embedding dimensions from 8 to 100 and report the link prediction accuracy of the test set under different dimensions. As shown in Fig. 3, the link prediction accuracies of Cora and Citeseer significantly improved when the embedding dimensions increase from 2 to approximately 40. Thereafter, the accuracy becomes saturated and fluctuates as the embedding dimension grows.

We also use the link prediction task to evaluate the performance of different sampling strategies. As shown in Fig. 4, a fixed sampling strategy is more robust than the flexible ones. Moreover, the link prediction accuracy of the fixed-size sampling is slightly higher than that of the flexible strategy. We assume that at least two factors attribute to this result. The first factor is the scale-free property of most networks. A large number of nodes have a small number of neighbors, whereas a small number of nodes own a large number of connected neighbors. In our experiment, we set the size of the neighbors to 20, which cover approximately all the neighbors for most nodes. The second factor is the small-world property. For a given network, nearly all the nodes can be connected within small hops. We take a hub node for example, where even some first-order neighbors have not been sampled as long as the leftover nodes are connected to the chosen neighbors and the influence of their features can be easily transferred to the hub nodes.

4 Conclusion and Discussion

In this paper, we propose LPNR, which learns network representations based on a graph neural network and a fixed sampling strategy. LPNR can extract meaningful vertex representations and achieve state-of-the-art accuracy in the link prediction task. The byproducts of LPNR, i.e., node representations, can be used in predicting missing links, measuring node centralities, and detecting communities. LPNR outperforms other GCNs that are based on supervised node label information in the community detection task, which proves the universality of the structural learning.
based network representations.

Despite the tedious process of adjusting the hyperparameters, we still assume that structural learning representations, such as the link prediction based LPNR, can lead to quantitative and qualitative leaps in graph processing problems. Although LPNR can be used in predicting missing links, measuring node centrality, and detecting network community, we still lack a clear understanding and cannot determine the mechanisms that lead to such a good performance. Hence, our future work will mainly focus on the hidden theory behind the LPNR algorithm.

Acknowledgment

We acknowledge the support from the program of the China Scholarships Council (No. 201806040107), the Fundamental Research Funds for the Central Universities, the National Natural Science Foundation of China (Nos. 61673070 and 61903020), and the BUCT Talent Start-up Fund (No. BUCTRC 201825). We acknowledge the help from the Swarma Club.

References


[29] Z. Huang, Link prediction based on graph topology: The


**Fei Gao** received the BEng degree in mathematics and applied mathematics from University of Science and Technology Beijing in 2017. He is currently pursuing the PhD degree at the School of Systems Science, Beijing Normal University. He joined Tencent in 2018 and Microsoft Research in 2019 as a research intern. His main research interests include graph embedding, graph neural networks, and their applications in complex network control.

**Weiwei Gu** received the PhD degree from Beijing Normal University in 2020. She was a visiting PhD student at Indiana University from 2018 to 2020. She is currently an assistant professor at College of Information Science and Technology, Beijing University of Chemical Technology. Her research areas include network representation, network dynamic learning, and social network analysis.

**Jiang Zhang** is a professor at School of Systems Science, Beijing Normal University and the founder of Swarma Club and Swarma Campus (Beijing) Science and Technology Co. Lt. He also provides consulting services for Tencent Research Institute. He received the PhD degree from Beijing Jiaotong University in 2006. His major research interests include machine learning on graphs, modelling complex systems, etc. His works have been published on international journals including *Nature Communications, Scientific Reports, Physical Review E, Journal of Theoretical Biology*, etc. He is enthusiastic in teaching and science propagation for public. He teaches several popular courses in Beijing Normal University, such as complexity thinking, Artificial Intelligence (AI), and MATLAB. He initiated Swarma Club in 2003, which is a well-known non-profit organization for inter-disciplinary scientific research in China. The club has stimulated some innovative ideas and even incubated startup AI teams (Colorful Cloud AI). Several academic papers were published on top journals like *Nature, Nature Communications, Nature Machine Intelligence, Scientific Reports*, and so forth by the core members of Swarma Club. They also authored some popular science books for citizens, including *the ultimate in science –Talking about artificial intelligence, approaching 2050 – About attention, internet, and artificial intelligence, deep thinking (translation), and principles of deep learning and practice for PyTorch*. 

**Ruiqi Li** received the BS degree in computer science from University of Electronic Science and Technology of China in 2013, and the PhD degree in systems science from Beijing Normal University in 2018. He was a visiting PhD student at MIT in 2016 and a visiting scholar at Boston University in 2017. He is currently an associate professor at College of Information Science and Technology, Beijing University of Chemical Technology. He is the director of UrbanNet Lab, which focuses on gaining better understanding on urban systems with network science, big data, and advanced technologies. His main research interests include urban modeling and computation, social network analysis, and epidemic spreading dynamics.