Link Prediction in Dynamic Networks by Combining GIN with LSTM

Jianwei Zhang
Iwate University
Morioka, Iwate, Japan
zhang@iwate-u.ac.jp

Takeru Oda
Iwate University
Morioka, Iwate, Japan

Yuta Sasaki
Iwate University
Morioka, Iwate, Japan

Lin Li
Wuhan University of Technology
Wuhan, Hubei, China
cathylilin@whut.edu.cn

ABSTRACT

In recent years, graph neural network (GNN) has been actively studied to solve graph-related tasks including link prediction. However, although many GNNs can obtain node embeddings that take into account the structure of surrounding nodes, it is complicated to learn graph embedding that takes into account the structure of the entire network. Some models are also proposed for link prediction corresponding to dynamic networks, but their performance is still to be improved. In this study, graph isomorphism network (GIN) is explored to generate node embeddings and graph embedding that take into account both the surrounding and overall structures of the network. Node embeddings are integrated with graph embedding to obtain high-level representation. We propose a model EvolveGIN, which combines GNN (typically GIN) with RNN (typically LSTM) to achieve dynamic link prediction. This model adapts GIN along the temporal evolution to capture the graph dynamism in dynamic networks by using LSTM to update the GIN weights. The experiments on both a synthetic dataset and a real-world dataset show the proposed model can outperform the baseline method.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Supervised learning**; **Knowledge representation and reasoning**; • **Information systems** → **Data mining**.

KEYWORDS

Link Prediction, Dynamic Network, Graph Isomorphism Network (GIN), Long Short Term Memory (LSTM), Node Embedding, Graph Embedding

ACM Reference Format:
Jianwei Zhang, Takeru Oda, Yuta Sasaki, and Lin Li. 2018. Link Prediction in Dynamic Networks by Combining GIN with LSTM. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Networks are a type of graph, which can show the entities represented by nodes and a variety of relationships represented by links. There are two types of networks: static networks in which the whole structure remains same and dynamic networks in which the whole structure is reshaped with the passage of time. Link prediction identifies the missing links in static networks or predicts the likelihood of future links in dynamic networks. Since real-world networks always evolve over time, link prediction in dynamic networks attracts more and more concerns in recent years. Link prediction in dynamic networks has a variety of applications, such as friend recommendation, citation recommendation, routing in networks, inference of biological interactions, identification of evolution patterns, and so on.

There are a variety of techniques for link prediction. Graph neural network (GNN)-based methods have significantly advanced the state of the art in the past few years. Graph neural network [7, 36, 37] adds graph operations to the traditional deep learning models and applies the structural information and attribute information of the graph to tackle the graph complexity, thus achieving high performance in many tasks such as link prediction, node classification and graph classification. When GNN is adopted to link prediction in dynamic networks, many researches utilize graph convolutional network (GCN) [14] and combine it with recurrent neural network (RNN) [34] to capture the evolution nature of dynamic networks. Peraja et al. proposed a model called EvolveGCN [23], in which the weights of the GCN are updated with RNN to capture the dynamism of the graph sequence. Although this method lifted the restriction of requiring node knowledge over the whole time span and achieved higher performance compared with related approaches, the link prediction accuracy is still limited.

Generally, GNN generates node embeddings that take into account the surrounding structure by aggregating the embeddings of neighbouring nodes and recursively conducting the aggregation. Therefore, multiple iterations are required in order to consider a wider range of structures. However, when embedding aggregations are repeated with a great many iterations, an over-smoothing problem occurs, which usually causes performance degradation

for the target tasks. This is because node embeddings are averaged out [17]. This drawback hinders many GNNs from generating node embeddings which can infuse global structure information of the network.

On the other hand, graph isomorphism network (GIN) [38] generates graph embeddings and performs well for graph classification. The model can generate node embeddings aggregated from neighbouring nodes, and obtain highly representative embeddings of the entire graph by pooling. GIN takes into account the structure of the entire network and can avoid the over-smoothing problem.

In this paper, we propose a model called EvolveGIN, which combines GNN (typically GIN) with RNN (typically LSTM) to handle link prediction in dynamic networks. By using GIN, node embeddings that reflect the surrounding structure and graph embedding that aggregates the structure of the entire network are generated. They are then combined to obtain the high-level representation of graph nodes. Furthermore, by using LSTM, GIN weights are updated to capture the temporal evolution in dynamic networks. Node representation infused with the entire network structure and graph sequence dynamism is used for the classification of link existence between two nodes in a graph at a target time step.

The main contributions of this work are summarized below:

(1) We propose a model EvolveGIN that combines GIN with LSTM to achieve link prediction in dynamic networks. The experiments on two datasets show the proposed model can outperform the baseline.
(2) The effectiveness of integrating graph embedding to node embeddings is verified in terms of the performance improvement for dynamic link prediction.
(3) It is proved that increasing MLP layers in GIN helps improve the performance of link prediction in some cases.

The rest of this paper is organised as follows. Section 2 reviews related work. Section 3 introduces the task of dynamic link prediction and some prior knowledge. Section 4 describes the proposed model for dynamic link prediction. Section 5 reports the evaluation experiments. Finally, Section 6 concludes this paper.

2 RELATED WORK

Researchers have studied a variety of link prediction techniques [15, 16, 31, 35], which can be grouped into several categories, like similarity-based indices, probabilistic models, classifier-based approaches, network embedding-based methods, and so on. Before 2010, mainstream methods were similarity-based indices due to their simplicity and high efficiency. However, these methods cannot make full use of nodes and network topology information. Probabilistic models extract the underlying structure from a network and estimate the parameters that can best fit the data of the network. However, these methods are computationally expensive and thus unsuitable for real large networks. Many classification algorithms such as SVM, KNN, and MLP are applicable for link prediction. Although classifier-based approaches with appropriate features can obtain high performance results, these methods face the problem of class imbalance due to the sparsity of real large networks. The most advanced network embedding-based methods , also known as graph representation learning, intend to address the deficiencies of the traditional methods. Network embedding aims to reduce the dimensionality of feature representation and capture the characteristics of the network at the same time.

In the survey paper [35], according to different encoding techniques, network embedding methods are further divided into matrix factorization-based methods (e.g., [3, 20]), random walk-based methods (e.g., [9, 24]), graph neural network-based methods, and other methods. As a representative class of methods, the graph neural network methods have been studied in recent years, which add graph operations to the traditional deep learning models. Hamilton et al. proposed a model called GraphSAGE [12], which learns a function for node information aggregation by sampling the features of neighbouring nodes. Zhang et al. proposed link prediction frameworks WLNM [39] and SEAL [40] to automatically learn network topology features by extracting and encoding a subgraph for a target link. Cluster-GCN [6] improved computational efficiency by using a graph clustering algorithm to identify a subgraph and restricting the search space. mLink [2] and MSVGAE [11] transformed subgraphs to different scales, providing supplementary embedding information for more effective link prediction. PLACN [25] and HyConvE [30] exploited the powerful information extraction capabilities of convolutional neural networks for effective link prediction and achieved the superior performance.

Most network embedding-based methods mainly focus on static networks, while many real-world networks always evolve over time. Link prediction in dynamic networks thus has attracted increasing attention. CDNE [18] tackled the problem of dynamic network embedding as a minimization of a loss function, which intends to maximally preserve the global node structures, local link structures and community dynamics. E-LSTM-D [5] proposed a deep learning model to learn structural and temporal features in a unified framework, which can handle long-term prediction problems. LP-ROBIN [1] exploited incremental embedding to predict new links, which means after the arrival of new data, it does not retrain the model from scratch, but updates the latent representation of old nodes and links to reflect changes in the network structure. Spatial-temporal graph neural networks were constructed to deal with the tasks in criminology field [21] and for traffic forecasting [10]. Two works [4, 19] combined graph convolution network (GCN) and recurrent neural network (RNN). For each snapshot, GCN is applied to learn the local structural properties of nodes and the relationships between them. RNN is adopted as the framework to capture the evolution nature of all snapshots of a dynamic network. EvolveGCN [23] is also a combination model of GCN and RNN, but is different from other similar direction methods in that it utilizes RNN to regulate not node embeddings but GCN parameters.

GCN generates node embeddings that consider the surrounding structure by aggregating information from neighbouring nodes and recursively conducting the process. To obtain node embeddings that can capture the information of distant nodes using these methods, many iterations of information aggregation processes are required. However, the phenomenon of over-smoothing occurs due to the repetitive processing of information aggregation, which weakens the representation ability of node embeddings and in turn degrades the prediction performance. Therefore, in this study, we

explore GIN to also generate graph embedding and integrate it with node embeddings to obtain high-level representation. Dynamic link prediction is achieved by combining GIN with RNN (typically LSTM). Following the idea of EvovleGCN [23] that focuses on the model itself without resorting to node embeddings, our work adopts the similar architecture that introduces LSTM to update GIN parameters. Because GCN's simplicity may hinder a satisfied accuracy of link prediction in dynamic networks, we intend to improve the performance with the combination of GIN and LSTM.

3 PRELIMINARIES

This section describes the task of link prediction in dynamic networks and introduces the prior knowledge about GNN, GCN, GIN and EvolveGCN.

3.1 Dynamic link prediction

In this study, we perform link prediction in dynamic networks. The dynamic networks consist of snapshots of time-varying graphs at each time step, represented by $\mathbb{G} = \{G_1, ..., G_T\}$. $G_t = (V, E_t, \mathbf{H}_t)$ represents the graph at time step t, $V = \{v_1, ..., v_N\}$ is the node set[1], $E_t = \{(u, v)_t | u, v \in V\}$ is the link set, $\mathbf{H}_t \in \mathbb{R}^{N \times D}$ is the node feature matrix, where N is the number of nodes and D is the number of dimensions of the node feature. In this study, we predict links between any two nodes $(u, v)_t$ in G_t at time step t, using the dynamic networks $\mathbb{G}_{t-1} = \{G_1, ..., G_{t-1}\}$ as input. That is to say, it is a binary classification task for link existence at time step t.

3.2 Graph neural network (GNN)

GNN is a generic term for neural network models handling data with a graph structure. There are two main steps in a GNN process: the first step is to collect information from neighbouring nodes; the second step is to combine the information collected from neighbouring nodes with the information of the target node. These two steps together form a layer of GNN. By stacking this layer k times, it is possible to indirectly capture the information of nodes that are k-hop away.

3.3 Graph convolutional network (GCN)

GCN [14] is a type of GNN proposed as a node classification method in 2017 by Kipf et al. The model performs convolution on the information of neighbouring nodes for all nodes in one layer by using the following formula to generate node embeddings:

$$\mathbf{H}^{(l)} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l-1)}\mathbf{W}^{(l-1)}) \tag{1}$$
$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$$
$$\tilde{\mathbf{D}} = diag\left(\sum_j \tilde{A}_{ij}\right)$$

where $\mathbf{A}$ is the adjacency matrix for the input of an undirected graph, $\mathbf{I}$ is the unit matrix and $\mathbf{W}^{(l-1)}$ is the learned weights of the $(l-1)$-th layer. $\sigma(\cdot)$ is the activation function, usually implemented with $ReLU(\cdot) = max(0, \cdot)$. Denoting the dimension of node embeddings in the l-th layer as D [2], the matrix of node embeddings can be represented as $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times D}$, where $\mathbf{H}^{(0)}$ at $l = 0$ is the input node features. By repeating these layers, information can be indirectly aggregated even for the nodes far from the target node. This model is usually used for node classification by feeding the generated node embeddings to a classifier.

3.4 Graph isomorphism networks (GIN)

GIN [38] is a type of GNN proposed as a graph classification method in 2019 by Xu et al. The model is expected to achieve the ability as the Weisfeiler-Lehman (W-L) graph isomorphism test [27]. The W-L graph isomorphism test determines whether two graphs G_1 and G_2 are isomorphic. GIN consists of two components: a node embedding generator and a graph embedding generator. The representation for the node embedding part is generated by the following formula:

$$\mathbf{h}_v^{(l)} = MLP^{(l)}\left(\left(1 + \epsilon^{(l)}\right) \cdot \mathbf{h}_v^{(l-1)} + \sum_{u \in N(v)} \mathbf{h}_u^{(l-1)}\right) \tag{2}$$

where $\mathbf{h}_v^{(l)}$ is the node embedding vector of the target node v in the l-th layer and $\mathbf{h}_v^{(0)}$ at $l = 0$ is the input feature vector of node v. $N(v)$ is the neighbour node set of the target node v. $MLP^{(l)}$ is the multilayer perceptrons and $\epsilon^{(l)}$ is a training parameter of layer l. By designing the model in this way, the conditions of the W-L graph isomorphism test are satisfied. In addition, the node embeddings can be used directly for tasks such as link prediction and node classification.

The representation for the graph embedding part is generated by the following formula:

$$\mathbf{h}_G = CONCAT\left(\mathbf{h}^{(l)} | l = 0, 1, ..., L\right) \tag{3}$$
$$\mathbf{h}^{(l)} = READOUT\left(\mathbf{h}_v^{(l)} | v \in G\right)$$

where $\mathbf{h}_G$ is the graph embedding and $CONCAT$ is the concatenation operation. $READOUT$ indicates the pooling of node embeddings for each layer. By concatenating such embeddings from the 0-th layer to the L-th layer, the same ability can be achieved with that of the W-L graph isomorphism test. Graph classification can be achieved by feeding the graph embedding to a classifier.

3.5 EvolveGCN

EvolveGCN [23] is a type of GNN for dynamic graphs proposed as the method of dynamic link prediction and node classification in 2020 by Pareja et al. The model combines a static GCN and an RNN, which can handle time-series data. As a feature of this model, EvolveGCN updates the weights of the GCN using RNN.

$$\mathbf{W}_t^{(l)} = RNN^{(l)}(\mathbf{W}_{t-1}^{(l)}) \tag{4}$$

[1] V is invariant over time and there exist nodes with no links.

[2] For simplicity, the dimension of $\mathbf{H}^{(l)}$ is marked as a same D, but different layers may set different dimensions.

where $\mathbf{W}_t^{(l)}$ is the GCN weights in the l-th layer at time step t, and $RNN^{(l)}(\cdot)$ is the recurrent neural network that adapts GCN weights to reflect the temporal dynamism of the graph sequence.

4 PROPOSED MODEL

4.1 Model overview

The overview of the proposed model EvolveGIN is shown in Figure 1. Our model extends the model proposed by Pareja et al. [23] by exploring GIN to obtain high-level representation of node embeddings and using LSTM [13] to dynamically update GIN weights. In EvolveGIN, only the LSTM weights are trained. The GIN weights are not trained but are computed by using the learned LSTM.

EvolveGIN is structured as follows. At each time step, there is a GIN component GIN-COM, which consists of a node embedding generator (green part in Figure 1) and a graph embedding generator (blue part in Figure 1) [3]. Each node embedding generator is composed of L GIN-NEs that intend to aggregate node embeddings from L-hop neighbourhoods. Each GIN-NE is a w-layer MLP. Furthermore, LSTMs are horizontally connected to dynamically update the weights of each-layer MLP in each GIN-NE. Thus, totally there are $L \times w$ LSTMs in the model. At the prediction time step, the node embeddings from GIN-NEs are integrated with the graph embedding from the graph embedding generator and are then fed to a classifier to perform the link prediction.

The process flow of EvolveGIN is as follows. At time step $t-1$, graph G_{t-1} is input to GIN-NEs to obtain node embeddings (summarized as Step 1 in Section 4.2). Then, for each layer of MLP in each GIN-NE, the MLP weights at time step t are computed by feeding the MLP weights at time step $t-1$ to LSTM that has been trained to learn the dynamism of graph sequence (Step 2). After that, at time step t, same as the process of Step 1, GIN-NEs generate node embeddings with each layer of MLP weights updated with Step 2. At the prediction time step, graph embedding is also generated (Step 3) and integrated into node embeddings (Step 4). Finally, any two nodes are selected and link prediction between them is performed by feeding their output node embeddings to a classifier (Step 5).

4.2 Model details

This section describes the details of each part of EvolveGIN.

4.2.1 Step 1: Node embedding generation with GIN-NEs. Node embeddings are generated with GIN-NEs. At each time step $t-1$, the input to the l-th GIN-NE is the matrix of node embeddings $\mathbf{H}_{t-1}^{(l-1)}$ that is the output of the $(l-1)$-th GIN-NE. $\mathbf{H}_{t-1}^{(0)}$ is the matrix of input node features of graph G_{t-1}. The l-th GIN-NE computes the embeddings for node v at time step $t-1$ as follows:

$$\mathbf{H}(v)_{t-1}^{(l)} = MLP_{t-1}^{(l)}\left(\left(1+\epsilon_{t-1}^{(l)}\right)\cdot \mathbf{H}(v)_{t-1}^{(l-1)} + \sum_{u\in N(v)_{t-1}} \mathbf{H}(u)_{t-1}^{(l-1)}\right) \tag{5}$$

where $\mathbf{H}(v)_{t-1}^{(l)} \in \mathbb{R}^{1\times D}$ is the embedding of node v at time step $t-1$ computed by the l-th GIN-NE, and D is the dimensions of node embeddings. $N(v)_{t-1}$ is the set of neighbouring nodes of node v at time step $t-1$. $\epsilon_{t-1}^{(l)}$ is the learning parameter of the l-th GIN-NE at time step $t-1$, which is fixed to 0 in this model because it is verified less effective in the previous study [38]. $MLP_{t-1}^{(l)}$ represents the multilayer perceptron of the l-th GIN-NE at time step $t-1$.

Node embeddings are generated by applying the above formula for all the nodes in the graph. By stacking multiple GIN-NEs, node embeddings are generated taking into account the surrounding structure of nodes.

4.2.2 Step 2: Update of MLP weights in GIN-NEs with LSTM. The MLP weights in GIN-NEs are updated with LSTM trained to reflect the dynamism of graph sequence. Input with $\mathbf{W}(w)_{t-1}^{l}$ (i.e., the weights of the w-th layer of MLP in the l-th GIN-NE at time step $t-1$), LSTM computes the weights $\mathbf{W}(w)_t^l$ of the same w-th layer of MLP in the same l-th GIN-NE at the next time step t [4]:

$$\mathbf{W}(w)_t^{(l)} = LSTM_w^{(l)}\left(\mathbf{W}(w)_{t-1}^{(l)}\right) \tag{6}$$

where $LSTM_w^{(l)}(\cdot)$ is the LSTM for the w-th layer of MLP in the l-th GIN-NE. By updating each layer of MLP weights in each GIN-NE in such a way, GIN-NEs can generate node embeddings that can also capture the time-series change in dynamic networks.

4.2.3 Step 3: Graph embedding generation from node embeddings. At time step t, the time point for link prediction, graph embedding is also generated from node embeddings as shown in the blue part of Figure 2. The generation method is as follows:

$$\begin{aligned}\mathbf{h}_{G_t} &= CONCAT(\mathbf{h}_t^{(l)} | l = 0, 1, ..., L)\\ \mathbf{h}_t^{(l)} &= SUM\left(\mathbf{H}(v)_t^{(l)} | v \in G_t\right)\end{aligned} \tag{7}$$

where $\mathbf{h}_{G_t} \in \mathbb{R}^{1\times((L+1)*D)}$ is graph embedding at time step t and $\mathbf{h}_t^{(l)}$ is the sum pooling results of node embeddings generated by the l-th GIN-NE. $CONCAT$ is the concatenation operation and SUM is the operation that sums up the embeddings of the nodes. These operations generate the embedding of the entire graph from the node embeddings with a simple architecture.

4.2.4 Step 4: Integration of graph embedding into node embeddings. In order to infuse the global information into node embeddings, generated graph embedding is integrated into node embeddings as shown in the yellow part of Figure 2. Our method employs a mechanism whereby the influence degree of graph embedding can be varied by applying a coefficient α to graph embedding. First, a linear transformation is performed to reform graph embedding to the same dimension as the node embeddings. Then, the reformed graph embedding is multiplied by the coefficient α and is concatenated to the embedding of each node, i.e., each row of $\mathbf{H}_t^{(L)}$. The

[3] It is possible to generate graph embedding for each time step. However, since it is actually needed only at the prediction time step for training and inference, the graph embedding generator is shown only at the last time step in Figure 1.

[4] A general formula for LSTM is $\mathbf{h}_t, \mathbf{c}_t = LSTM(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1})$. In EvolveGIN, $\mathbf{c}_t$ and $\mathbf{c}_{t-1}$ are not considered, $\mathbf{h}_{t-1}$ is a column vector in the matrix of each-layer MLP weights, and $\mathbf{x}_t$ is set as same with $\mathbf{h}_{t-1}$. That is to say, node embeddings are not used for the update of MLP weights. Although a vector is a general input to LSTM, the matrix input to LSTM in Formula 6 can be understood by using the same LSTM to process each column of the matrix.

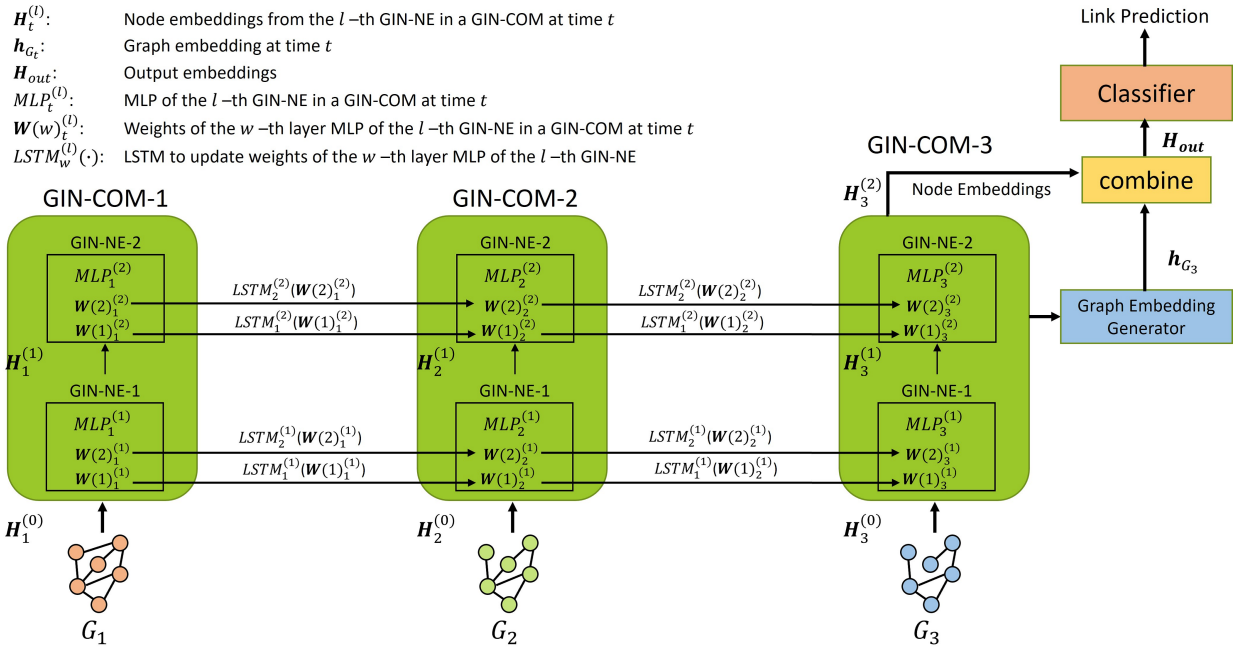

Figure 1: Overview of EvolveGIN with an example of $L = 2, w = 2, t = 3$. L is the number of GIN-NEs, w is the number of layers in each GIN-NE, and t is the time step.

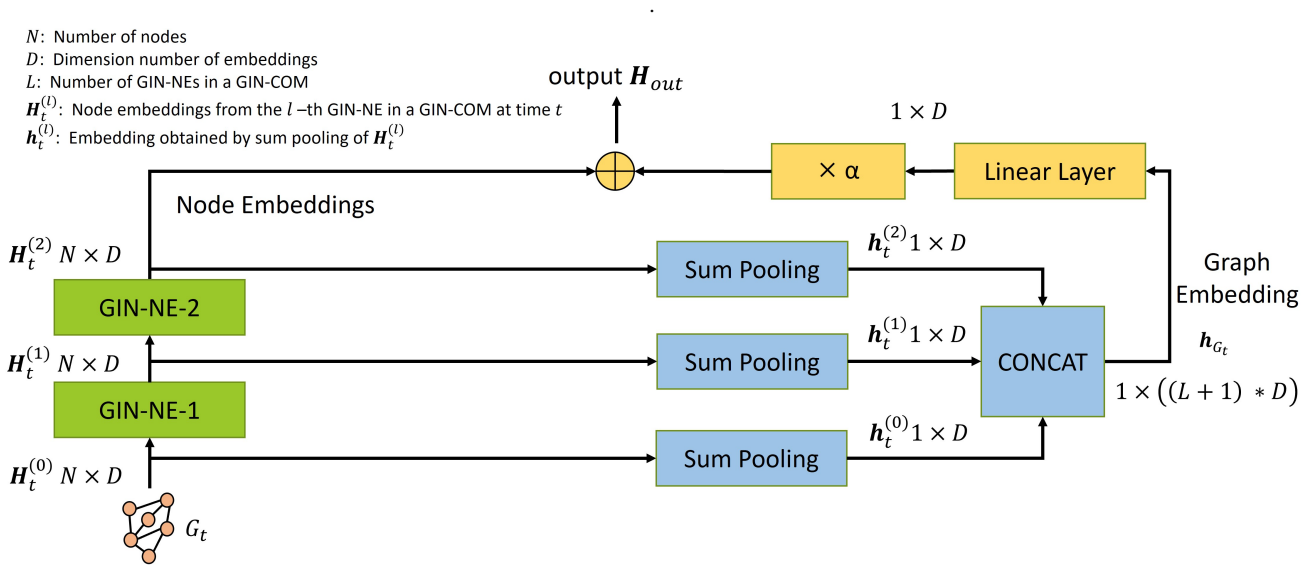

Figure 2: Graph embedding generation from node embeddings and their combination

output representation is denoted as $\mathbf{H}_{out} \in \mathbb{R}^{N \times 2D}$, where N is the number of nodes and D is the number of dimensions of node embeddings. Thus, the integrated node embeddings may contain information from both the surrounding structure and the structure of the entire graph.

4.2.5 Step 5: Link prediction with a classifier. Finally, a classifier is applied to link prediction fed with the embeddings $\mathbf{H}_{out}$ generated at the prediction time step. After all node pairs $(u, v)_t$ in G_t are created, the embeddings corresponding to two nodes in each pair are concatenated and passed to a classifier to calculate the probability that the link may exist between these two nodes.

Table 1: Datasets

	# of nodes	# of links	Time steps (Train / Validation / Test)
SBM	1,000	4,870,863	35 / 5 / 10
UCI	1,899	59,835	62 / 9 / 17

Table 2: Hyperparameters in EvolveGIN

Hyperparameter	Value for SBM	Value for UCI
# of epochs	40	40
Learning rate	0.005	0.001
Embedding size	100	100

Table 3: Hyperparameters in EvolveGCN

Hyperparameter	Value for SBM	Value for UCI
Cross-entropy weights	(0.15,0.85)	(0.1,0.9)
# of epochs	100	100
Learning rate	0.005	0.001
Embedding size	100	100

5 EXPERIMENTS

This section describes the evaluation experiments. We first introduce the experimental setup and then report the experimental results.

5.1 Experimental setup

5.1.1 Datasets. We conduct the experiments on both a synthetic SBM dataset and a real-world UCI dataset. The SBM dataset is generated synthetically [23] based on a popularly used random graph model Stochastic Block Model [33], which simulates graph structures and evolutions. Each node has its community. The probabilities of linking two nodes from the same community and linking two nodes from different communities are set to different values. Furthermore, in order to reflect the dynamism, at each time step some nodes are randomly selected and their communities are changed. The parameters for generating this dataset refer to the previous research [8]. The UCI dataset is a standard dataset for link prediction tasks[5]. It is formed from a real-world online community of students at the University of California, Irvine [22]. The nodes of this social network are the students and the links indicate message exchanges between them.

The number of nodes, the number of links and the time steps used for Train/Validation/Test are shown in Table 1. The node feature vector, i.e., the input to the first GIN-NE, is generated through one-hot encoding. It means that the dimension of each node's feature vector is the type number of distinct degrees of all the nodes. For a node, the element of the dimension with the corresponding degree is set to 1 and all the others are set to 0. In the experiments, the graphs from time step 1 to $t-1$ are the input and whether a link exists between a node pair $(u,v)_t$ in G_t at time step t is predicted. The time step windows are set to 6 for SBM and 11 for UCI, respectively.

5.1.2 Development environment. Our experiments are conducted using 51 GB RAM and a Tesla T4 GPU in Google Colaboratory[6]. Google Colaboratory is one of Google's research projects to promote machine learning mainly in education and research institutions. This service enables Python programs to be executed with GPUs or TPUs through the browser, making it easy to perform large-scale computations such as deep learning. The deep learning library Pytorch[7] and Deep Graph Library[8] [32] are used for the implementation of EvolveGIN.

5.1.3 Model parameters. EvolveGIN combines GIN with LSTM to achieve dynamic link prediction. The number of GIN-NEs at each time step is fixed to $L = 2$, and the number of MLP layers in each GIN-NE is tested and compared with different values $w = 2, 3, 4$. ReLU is adopted as the activation function in GIN-NEs. The classifier for link prediction at time step t is composed of a two-layer MLP and a softmax activation function, which can output the probability that a link between two nodes exists. Because in the datasets Class 1 (linked) has far less learning data than Class 0 (unlinked), a weighted cross-entropy function is used as the objective function during training. The cross-entropy weights are also tested in the experiments. The hyperparameters used for EvolveGIN are shown in Table 2. As the compared baseline, EvolveGCN is implemented with $L = 2$ in our experimental environment. Other hyperparameters are shown in Table 3, referring to the settings in EvolveGCN [23].

5.1.4 Evaluation metric. We use Mean Average Precision (MAP) as the evaluation metric, which ranges from 0 to 1. The higher MAP value indicates better link prediction with high performance. MAP is calculated by the following formula:

$$MAP = \frac{AP(Class0) + AP(Class1)}{2}$$
$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{8}$$

where Class 0 and Class 1 correspond to "unlinked" and "linked" respectively, and R_n and P_n are the recall and the precision at each predicted probability n.

5.1.5 Experimental contents. Three experiments are conducted. The first experiment verifies the effectiveness of graph embedding by changing the coefficient α of integrating graph embedding into node embeddings. The second experiment investigates the performance change by testing different numbers of MLP layers in GIN-NEs. The third experiment compares the performance between our proposal EvolveGIN and the baseline EvolveGCN.

5.2 Experimental results

5.2.1 Effectiveness of graph embedding. We verify the effectiveness of graph embedding for dynamic link prediction by varying

[5] http://konect.cc/networks/opsahl-ucsocial/
[6] https://colab.research.google.com/
[7] https://pytorch.org/
[8] https://www.dgl.ai

Figure 3: Influence of graph embedding on SBM

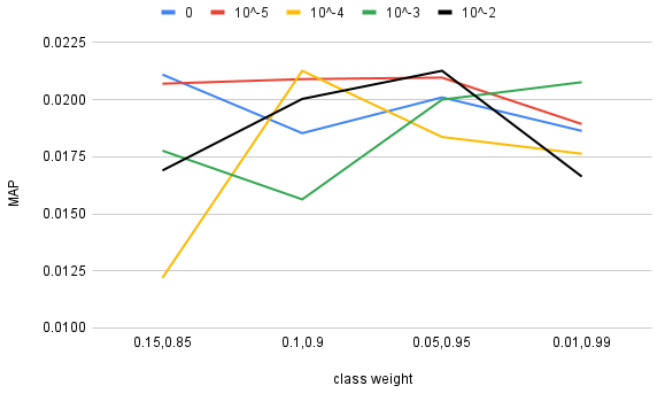

Figure 4: Influence of graph embedding on UCI

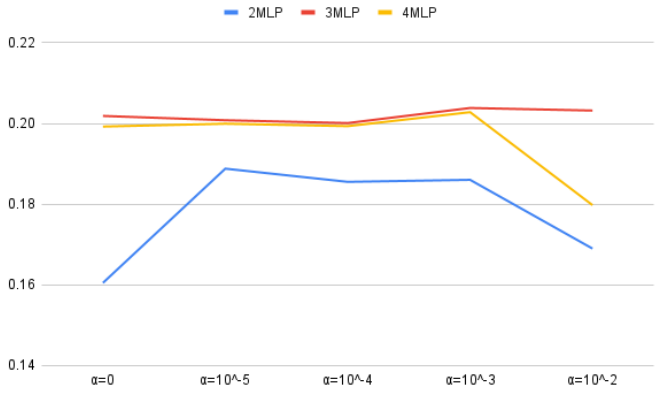

Figure 5: Influence of MLP layers in GIN-NEs on SBM

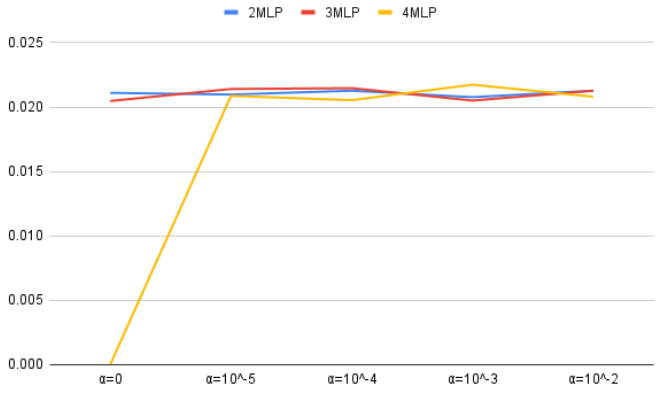

Figure 6: Influence of MLP layers in GIN-NEs on UCI

the coefficient α of integrating graph embedding into node embeddings. Five different coefficients α are tested for comparison: $0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}$. In this experiment, each GIN-COM consists of two GIN-NEs, each of which is a two-layer MLP. Cross-entropy weights (Class 0: unlinked, Class 1: linked) are tested for SBM: (0.2, 0.8), (0.15, 0.85), (0.1, 0.9), (0.05, 0.95), (0.01, 0.99) and for UCI: (0.15, 0.85), (0.1, 0.9), (0.05, 0.95) and (0.01, 0.99)[9]. The experiment is performed three times for each setting and the evaluation results are averaged.

The MAP of EvolveGIN for SBM is shown in Figure 3. The observation shows that peaks often occur when the cross-entropy weights are (0.1, 0.9) and (0.01, 0.99), and more low values are recorded at weights (0.2, 0.8). When $\alpha = 0$ without additional graph embedding, the MAP remains around 0.15 from weights (0.2, 0.8) to (0.1, 0.9) and gradually decreases at other weights. When $\alpha = 10^{-5}$ with graph embedding slightly added, the MAP is relatively stable above 0.15, achieving a maximum value of 0.1888 in the graph at weights (0.1, 0.9).

[9](0.2, 0.8) for UCI is not listed, because if the weight for Class 0 is set larger than 0.15, all the data in this dataset are classified as Class 0 with any of the five α.

The MAP of EvolveGIN for UCI is shown in Figure 4. We can observe that peaks occur at weights (0.15, 0.85) and (0.05, 0.95). When $\alpha = 0$ without additional graph embedding, the MAP achieves the highest value at weights (0.15, 0.85) and then gradually decreases at other weights. The maximum value of 0.0217 in the graph is recorded at "$\alpha = 10^{-2}$ and (0.05, 0.95)" and at "$\alpha = 10^{-4}$ and (0.1, 0.9)". When $\alpha = 10^{-5}$ with graph embedding slightly added, the MAP is often larger than or almost equal to those values with other α.

On both datasets, the maximum values are obtained when α is not 0, which indicates a performance improvement can be expected by integrating graph embedding into node embeddings. In Figure 3 and Figure 4, higher MAP is often achieved when $\alpha = 10^{-5}$, which indicates that adding a small amount of graph embedding may be more appropriate.

5.2.2 Influence of MLP layers in GIN-NEs. We examine the performance change by varying the number of MLP layers in GIN-NEs. The number of MLP layers in GIN-NEs is set to 2, 3 and 4. The experiment is conducted three times for each setting and the evaluation results are averaged. For each number of MLP layers and each α, the maximum value at all weights is compared.

Table 4: Comparison of MAP between EvolveGIN and EvolveGCN

Model	SBM	UCI
EvolveGIN (2MLP)	0.1888	0.0213
EvolveGIN (3MLP)	**0.2038**	0.0215
EvolveGIN (4MLP)	0.2028	**0.0217**
EvolveGCN	0.1938	0.0164

Figure 5 shows that the MAP obtained by models with a three-layer MLP and a four-layer MLP hovers around 0.20 and is always higher than the model with a two-layer MLP on SBM. This may be because increasing the number of MLP layers in GIN-NEs can obtain higher-level representations of the nodes.

Figure 6 shows that the MAP obtained by models with a two-layer MLP and a three-layer MLP hovers around 0.020, while the model with a four-layer MLP records a minimum value of 0 when $\alpha = 0$. The performance does not vary significantly between the different layers of MLP. However, for the model with a four-layer MLP when $\alpha = 0$, i.e., no graph embedding is added, the performance is significantly decreased. This may be dataset dependent. Checking the total number of links in Table 1, UCI is a very biased dataset with a very small number of links. From another point of view, graph embedding may be effective in preventing extreme performance degradation.

5.2.3 Comparison between EvolveGIN and EvolveGCN. We compare the performance between our proposal EvolveGIN and the baseline EvolveGCN. The parameters of EvolveGCN are based on [23]. The numbers of components in both EvolveGCN and EvolveGIN at each time step are the same with $L = 2$. The number of MLP layers in GIN-NEs of EvolveGIN is set to $w = 2$, 3 and 4. The coefficient α is set to 0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}. The cross-entropy weights are set for SBM to (0.2, 0.8), (0.15, 0.85), (0.1, 0.9), (0.05, 0.95), (0.01, 0.99), and for UCI to (0.15, 0.85), (0.1, 0.9), (0.05, 0.95) and (0.01, 0.99). The experiment is conducted three times for each setting and the evaluation results are averaged. For each setting with a w-layer MLP, varying the coefficient α and the cross-entropy weights, the highest value is adopted for performance comparison.

The results are shown in Table 4. On SBM, although the MAP of the EvolveGIN with a two-layer MLP is inferior to EvolveGCN, the models with a three-layer and a four-layer MLP achieve higher performance than EvolveGCN. The best result is obtained by the model with a three-layer MLP. On UCI, all three EvolveGIN models outperform EvolveGCN and the best result is obtained when using a four-layer MLP in GIN-NEs. This may be because node embeddings obtained with more layers of MLP in GIN-NEs can provide higher-level representation than GCN.

6 CONCLUSION

In this paper, we proposed a model EvolveGIN for link prediction in dynamic networks. GIN is used to generate node embeddings that capture the surrounding information and graph embedding that captures the structure of the entire graph. The integration of graph embedding into node embeddings is devised to obtain high-level node representation. LSTM is adopted to update GIN weights to reflect the dynamism of graph sequence. Through the experiments on both a synthetic dataset and a real-world dataset, we verified that EvolveGIN can improve the prediction performance by integrating a small amount of graph embedding into node embeddings and by increasing MLP layers in GINs, and hence outperform the baseline.

Compared with recurrent methods modelling graph evolution as our current work does, graph attention mechanism [29] may be effective for achieving competitive performance [26, 28]. In future work, we plan to explore such graph attention models and evaluate prediction performance with more metrics such as MRR and AUC. Furthermore, since dynamic link prediction is usually conducted on unbalanced data, we will also consider the classifiers that incorporate anomaly detection techniques.

REFERENCES

[1] Emanuele Pio Barracchia, Gianvito Pio, Albert Bifet, Heitor Murilo Gomes, Bernhard Pfahringer, and Michelangelo Ceci. 2022. LP-ROBIN: Link prediction in dynamic networks exploiting incremental node embedding. *Inf. Sci.* 606 (2022), 702–721.

[2] Lei Cai and Shuiwang Ji. 2020. A Multi-Scale Approach for Graph Link Prediction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020.* 3308–3315.

[3] Guangfu Chen, Haibo Wang, Yili Fang, and Ling Jiang. 2022. Link prediction by deep non-negative matrix factorization. *Expert Syst. Appl.* 188 (2022), 115991.

[4] Jinyin Chen, Xueke Wang, and Xuanheng Xu. 2022. GC-LSTM: graph convolution embedded LSTM for dynamic network link prediction. *Appl. Intell.* 52, 7 (2022), 7513–7528.

[5] Jinyin Chen, Jian Zhang, Xuanheng Xu, Chenbo Fu, Dan Zhang, Qingpeng Zhang, and Qi Xuan. 2021. E-LSTM-D: A Deep Learning Framework for Dynamic Network Link Prediction. *IEEE Trans. Syst. Man Cybern. Syst.* 51, 6 (2021), 3699–3712.

[6] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *The 25th International Conference on Knowledge Discovery and Data Mining, KDD 2019.* 257–266.

[7] Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Junjie Xu, Zhimeng Guo, Hui Liu, Jiliang Tang, and Suhang Wang. 2022. A Comprehensive Survey on Trustworthy Graph Neural Networks: Privacy, Robustness, Fairness, and Explainability. *CoRR* abs/2204.08570 (2022).

[8] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. DynGEM: Deep Embedding Method for Dynamic Graphs. *CoRR* abs/1805.11273 (2018).

[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016.* 855–864.

[10] Kan Guo, Yongli Hu, Zhen Sean Qian, Yanfeng Sun, Junbin Gao, and Baocai Yin. 2022. Dynamic Graph Convolution Network for Traffic Forecasting Based on Latent Network of Laplace Matrix Estimation. *IEEE Trans. Intell. Transp. Syst.* 23, 2 (2022), 1009–1018.

[11] Zhihao Guo, Feng Wang, Kaixuan Yao, Jiye Liang, and Zhiqiang Wang. 2022. Multi-Scale Variational Graph AutoEncoder for Link Prediction. In *The Fifteenth ACM International Conference on Web Search and Data Mining, WSDM 2022.* 334–342.

[12] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Annual Conference on Neural Information Processing Systems, NIPS 2017.* 1024–1034.

[13] Sepp Hochreiter and Jürgen Schmidhuber. [n. d.]. Long Short-Term Memory. *Neural Comput.* 9, 8 ([n. d.]), 1735–1780.

[14] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017.*

[15] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. 2020. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications* 553 (2020), 124289.

[16] Jiahao Li, Linlan Liu, and Jian Shu. 2022. Progresses in Link Prediction: A Survey. In *The 2022 11th International Conference on Computing and Pattern Recognition, ICCPR 2022.* 651–655.

[17] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *The Thirty-Second*

AAAI Conference on Artificial Intelligence, (AAAI-18). 3538–3545.

[18] Lijia Ma, Yutao Zhang, Jianqiang Li, Qiuzhen Lin, Qing Bao, Shanfeng Wang, and Maoguo Gong. 2020. Community-aware dynamic network embedding by using deep autoencoder. *Inf. Sci.* 519 (2020), 22–42.

[19] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognit.* 97 (2020).

[20] Aditya Krishna Menon and Charles Elkan. 2011. Link Prediction via Matrix Factorization. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011*. 437–452.

[21] Shengjie Min, Zhan Gao, Jing Peng, Liang Wang, Ke Qin, and Bo Fang. 2021. STGSN - A Spatial-Temporal Graph Neural Network framework for time-evolving social networks. *Knowl. Based Syst.* 214 (2021), 106746.

[22] Tore Opsahl and Pietro Panzarasa. 2009. Clustering in weighted networks. *Soc. Networks* 31, 2 (2009), 155–163.

[23] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*. 5363–5370.

[24] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014*. 701–710.

[25] Kumaran Ragunathan, Kalyani Selvarajah, and Ziad Kobti. 2020. Link Prediction by Analyzing Common Neighbors Based Subgraphs Using Convolutional Neural Network. In *24th European Conference on Artificial Intelligence, ECAI 2020*. 1906–1913.

[26] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. In *The Thirteenth ACM International Conference on Web Search and Data Mining, WSDM 2020*. 519–527.

[27] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *J. Mach. Learn. Res.* 12 (2011), 2539–2561.

[28] Min Shi, Yu Huang, Xingquan Zhu, Yufei Tang, Yuan Zhuang, and Jianxun Liu. 2021. GAEN: Graph Attention Evolving Networks. In *The Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021*. 1541–1547.

[29] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018*.

[30] Chenxu Wang, Xin Wang, Zhao Li, Zirui Chen, and Jianxin Li. 2023. HyConvE: A Novel Embedding Model for Knowledge Hypergraph Link Prediction with Convolutional Neural Networks. In *The ACM Web Conference 2023, WWW 2023*. 188–198.

[31] Meihong Wang, Linling Qiu, and Xiaoli Wang. 2021. A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry* 13, 3 (2021), 485.

[32] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2020. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *CoRR* abs/1909.01315 (2020).

[33] Yuchung J. Wang and George Y. Wong. 1987. Stochastic Blockmodels for Directed Graphs. *J. Amer. Statist. Assoc.* 82, 397 (1987), 8–19.

[34] Paul J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[35] Haixia Wu, Chunyao Song, Yao Ge, and Tingjian Ge. 2022. Link Prediction on Complex Networks: An Experimental Survey. *Data Sci. Eng.* 7, 3 (2022), 253–278.

[36] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. 2022. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Nature Singapore.

[37] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24.

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019*.

[39] Muhan Zhang and Yixin Chen. 2017. Weisfeiler-Lehman Neural Machine for Link Prediction. In *The 23rd International Conference on Knowledge Discovery and Data Mining, KDD 2017*. 575–583.

[40] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Annual Conference on Neural Information Processing Systems, NeurIPS 2018*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 5171–5181.