GCN-GAN: A Non-linear Temporal Link Prediction Model for Weighted Dynamic Networks

Kai Lei[†,§], Meng Qin[†], Bo Bai[‡,*], Gong Zhang[‡], Min Yang[¶,*]

[†]ICNLAB, School of Electronics and Computer Engineering (SECE), Peking University, Shenzhen, China
[§]PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China
[‡]Future Network Theory Lab, 2012 Labs, Huawei Technologies, Co. Ltd., Hong Kong, China
[¶]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[†]leik@pkusz.edu.cn, [†]mengqin_az@foxmail.com, [‡]{baibo8, nicholas.zhang}@huawei.com, [¶]min.yang@siat.ac.cn
[*]Corresponding Authors

Abstract—In this paper, we generally formulate the dynamics prediction problem of various network systems (e.g., the prediction of mobility, traffic and topology) as the temporal link prediction task. Different from conventional techniques of temporal link prediction that ignore the potential non-linear characteristics and the informative link weights in the dynamic network, we introduce a novel non-linear model GCN-GAN to tackle the challenging temporal link prediction task of weighted dynamic networks. The proposed model leverages the benefits of the graph convolutional network (GCN), long short-term memory (LSTM) as well as the generative adversarial network (GAN). Thus, the dynamics, topology structure and evolutionary patterns of weighted dynamic networks can be fully exploited to improve the temporal link prediction performance. Concretely, we first utilize GCN to explore the local topological characteristics of each single snapshot and then employ LSTM to characterize the evolving features of the dynamic networks. Moreover, GAN is used to enhance the ability of the model to generate the next weighted network snapshot, which can effectively tackle the sparsity and the wide-value-range problem of edge weights in real-life dynamic networks. To verify the model's effectiveness, we conduct extensive experiments on four datasets of different network systems and application scenarios. The experimental results demonstrate that our model achieves impressive results compared to the state-of-the-art competitors.

Index Terms—Temporal Link Prediction, Weighted Dynamic Networks, Generative Adversarial Networks, Graph Convolutional Networks

I. INTRODUCTION

Dynamics is a significant factor that hinders the performance of most network systems. The prediction of mobility, traffic and topology has been considered as an effective technique to cope with the problem. For instance, the dynamics of communication links in ad hoc networks makes the design of routing protocol a challenging problem, where the prediction of the dynamic topology plays an important role to achieve a more efficient and reliable communication [1]. In data center networks, traffic prediction technique could be utilized to effectively schedule the highly parallel network flows while avoiding the performance degradation due to resource shortages [2]. For cellular networks, the prediction of users' locations can help to reduce the resource consumption (e.g., bandwidth) and achieve better Quality of Services (QoS) [3]. In a word, if the dynamics of the network system can be

accurately predicted, the key resources can be effectively pre-allocated to ensure the system's high performance.

Although numerous studies have been developed to deal with the dynamics of network systems [2]–[5], most of them only focus on a specific application scenario (e.g., flow prediction in data center networks), failing to be generalized to other different scenarios.

In fact, the dynamics prediction problem of various network systems can be generally formulated as the temporal link prediction task, where the system's behavior is described by an abstracted dynamic graph. For example, one can model each host in a data center as a node (entity), and the dynamic traffic between a pair of hosts can be regarded as the changed weighted link (relation) between the corresponding entity pair. Given the graph snapshots of previous time slices, the temporal link prediction task tries to construct the graph topology in the next time slice [6].

Several recent techniques have been proposed to predict the temporal links in dynamic graphs from different perspectives [2]–[5]. Despite their effectiveness, we argue that temporal link prediction remains a challenging task for two primary reasons.

First, to the best of our knowledge, most of the existing approaches merely consider the link prediction in unweighted networks, determining the existence and absence of a link between a certain node pair. However, the link weights are essential in real networks, which bring significant information about the network's behavior. For example, the link weights may contain some useful information about delay, flow, signal strength or distance of the network systems. Under such circumstance, the temporal link prediction technique should not only determine the existence of links but also consider the corresponding weights, which is a more challenging problem that most of the conventional methods cannot tackle.

Second, non-linear transformations over time are commonly observed in dynamic networks since the formation process of most networks is complicated and highly non-linear [7]. However, conventional methods are almost based on typical linear models (e.g., non-negative matrix factorization (NMF) [8]), ignoring the potential non-linear characteristics of dynamic networks. These linear models may have limited performances for some network inference tasks, including the temporal

link prediction, because the linear data representation cannot capture the different latent factors of variation behind the network. To that end, it would be highly desirable to exploit the composition of multiple non-linear transformations of networks to improve the link prediction performance.

To alleviate the aforementioned limitations, we propose a novel deep learning based model for the temporal link prediction of weighted dynamic networks. Our model combines the strengths of the deep neural networks (i.e., graph convolutional network (GCN) [9] and long short-term memory (LSTM) network [10]) as well as generative adversarial network (GAN) [11] to strengthen the representation learning of the network data and generate the high-quality graph snapshot in next time slice. Concretely, we first utilize the GCN to capture the characteristics of topological structure hidden in each single graph snapshot. Then, the learned network representations are fed into an LSTM network to capture the evolving patterns of the weighted dynamic network with multiple successive time slices. Moreover, GAN is applied to generate high-quality and plausible graph snapshot with adversarial training. In the adversarial process, we train a generative model G to predict the weighted links in the next time slice based on the historical data sequentially. A discriminative model D is also trained to distinguish the generated list of links from the real records. G and D are jointly optimized with a minimax two-player game, enabling the model to generate high-quality weighted links.

We summarize our main contributions as follow:

- We formulate the dynamics prediction of various network systems as the temporal link prediction problem and discuss the challenges for the prediction of weighted dynamic networks.
- We employ deep neural networks (i.e., GCN and LSTM) to explore the non-linear characteristics of topological structure and evolving patterns hidden in the network.
- To the best of our knowledge, we are the first to utilize GAN to tackle the temporal link prediction of weighted dynamic networks, by generating high-quality next weighted links based on the historical snapshots.
- Besides the standard mean square error (MSE) metric, we introduce two additional metrics (i.e., edge-wise KL-divergence and the mismatch rate) to investigate the sparsity of the relations among entities in network systems and the wide-value-range property of edge weights.
- To verify the effectiveness of our model, we conduct extensive experiments on four datasets of various network systems, where our model consistently outperforms other competitors for the temporal link prediction task of weighted dynamic networks.

The rest of this paper is organized as follows. In Section II, we briefly introduce the related work. A formal definition of the temporal link prediction problem is given in Section III. Section IV presents the proposed GCN-GAN model in details. In Section V, we describe the experiments, including the performance evaluation on four datasets of network systems and a case study of the proposed model's refining effect.

Section VI concludes this paper and indicates our future work.

II. RELATED WORK

The dynamics of real network systems has received considerable attention in recent years. Several techniques have been developed to tackle the performance degradation caused by the system's dynamics. In [2], the authors introduced a convolutional neural network framework, which can forecast the short-term traffic load in data center networks. In [3], a hidden Markov model was constructed to predict users' locations in mobile cellular networks. To improve the quality of service (QoS) for users in wireless mesh backbone networks, [4] proposed a network traffic prediction model by integrating the deep belief neural network and spatiotemporal compressive sensing method. For the traffic matrix estimation problem, a novel approach with multiple low-rank matrices was advocated in [5], achieving a better performance compared to the conventional gravity model. However, most of the dynamics prediction techniques of network systems (including the above methods) only utilize the unique patterns or characteristics of a specific application scenario (e.g., data center network), lacking the significant ability to be generalized to other different scenarios.

On the other hand, the dynamics prediction problem of network systems can be generally modeled as the temporal link prediction task, and a brief overview about the task can be found in [12] and [13].

Conventional temporal link prediction methods are almost based on the collapsed network model [14], [15]. In the model, the network snapshots of multiple successive time slices are linearly combined to construct a single comprehensive snapshot named as the collapsed network. The characteristics of the dynamic network are extracted by conducting a certain matrix decomposition process on the collapsed snapshot. Nevertheless, such conventional models may ignore the critical information hidden in the dynamic network with multiple network snapshots resulting in limited prediction performance.

To avoid collapsing the temporal networks, authors of [16] represented the dynamic network as a third-order tensor, and the temporal information was explored by conducting a tensor factorization process. In [17], a model based on the non-negative matrix factorization (NMF) framework [8] was developed, where the dynamic information of historical snapshots was incorporated by utilizing the graph regularization technique. As discussed in [18], each network snapshot in the dynamic network could be described as a corresponding NMF component. A unified model was proposed based on the combination of multiple NMF components, where a novel adaptive parameter was introduced to consider the intrinsic correlation between single snapshot and the dynamic network.

However, the aforementioned approaches still have limited room for the improvement of prediction accuracy, because they are almost based on the traditional linear model, ignoring the potential non-linear characteristic of the dynamic network. Although several non-linear methods based on the restricted Boltzmann machine (RBM) [18] and graph embedding [19]

are proposed, most of them can only be applied to the prediction of unweighted networks but cannot deal with the challenging case of weighted networks.

III. PROBLEM DEFINITION

A dynamic network can be defined as a sequence of graph snapshots $G = \{G_1, G_2, \cdots, G_\tau\}$, in which $G_t = (V, E_t, W_t)$ is the snapshot at a certain time slice t ($t \in \{1, 2, \cdots, \tau\}$) with a node set V, a edge set E_t and a weight set W_t (we use the subscript τ to represent current time slice). In this study, we only consider the case of undirected weighted networks with all the graph snapshots sharing the same node set V.

For the snapshot of time slice t, we use an adjacency matrix $\mathbf{A}_t \in \Re^{|V| \times |V|}$ to describe the corresponding static topological structure. Specially, when there is an edge between node i and j ($(i,j) \in E_t$) with weight $W_t(i,j)$, we let $(\mathbf{A}_t)_{ij} = (\mathbf{A}_t)_{ji} = W_t(i,j)$, and $(\mathbf{A}_t)_{ij} = (\mathbf{A}_t)_{ji} = 0$ otherwise.

Given the adjacency matrices of previous l time slices and current time slice $\{\mathbf{A}_{\tau-l}, \mathbf{A}_{\tau-l+1}, \cdots, \mathbf{A}_\tau\}$ (with $l+1$ network snapshots in total), the goal of the temporal link prediction task is to predict the topology of the next time slice ($\tau + 1$), which can be formally described below:

$$\tilde{\mathbf{A}}_{\tau+1} = f(\mathbf{A}_{\tau-l}, \mathbf{A}_{\tau-l+1} \cdots, \mathbf{A}_\tau), \quad (1)$$

where $f(\cdot)$ is the model that we need to construct in this paper while $\tilde{\mathbf{A}}_{\tau+1}$ represent the prediction result. For the convenience of discussion, we utilize the simplified notation $\mathbf{A}_{\tau-l}^\tau$ to represent the sequence $\{\mathbf{A}_{\tau-l}, \cdots, \mathbf{A}_\tau\}$.

IV. METHODOLOGY

A. The Model Architecture

In this study, we introduce a novel non-linear model GCN-GAN for the temporal link prediction of weighted dynamic networks. The proposed model, depicted in Fig. 1, consists of three main components: (i) Graph Convolutional Network (GCN) [9], (ii) Long Short-Term Memory (LSTM) [10] and (iii) Generative Adversarial Nets (GAN) [11].

First, we utilize the GCN to explore the local topology characteristics of each single graph snapshot. Then, the comprehensive representations given by the GCN are fed into an LSTM network to capture the evolving patterns of the dynamic graph. Moreover, we apply the GAN to generate high-quality predicted graph snapshot with an adversarial process, where we use the GCN as well as the LSTM to construct a generative network G (bottom side of Fig. 1) and introduce another full-connected discriminative network D (top side of Fig. 1). In the adversarial process, G is trained to predict the next snapshot based on the dynamic graph's historical topology, while D is trained to distinguish the generated weighted links from the real records. By applying this minimax two-player game, the adversarial process eventually adjusts G to generate plausible and high-quality prediction result.

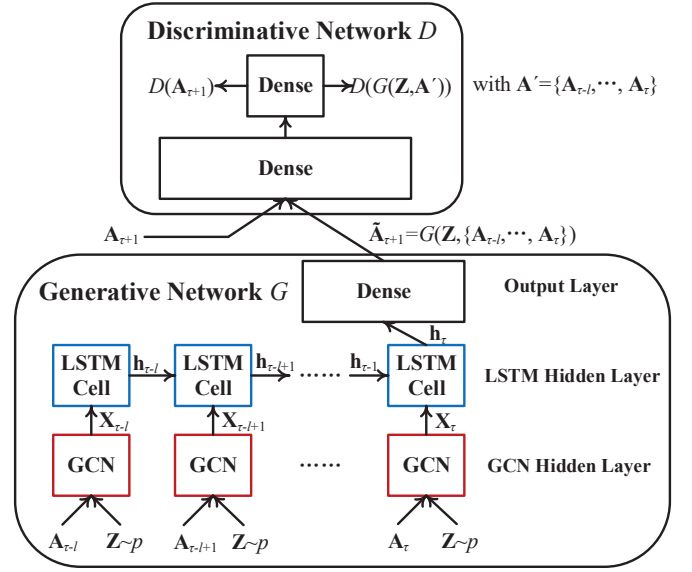In the rest of this section, we elaborate on the three main components of GCN-GAN in detail.

Fig. 1. The architecture of the GCN-GAN temporal link prediction model with a generative network G (bottom side) and a discriminative network D (top side). The generative network consists of a GCN hidden layer, an LSTM hidden layer and a full-connected output layer, while the discriminative network takes the form of full-connected feedforward network.

B. The GCN Hidden Layer

We utilize GCN to model the local topology structure of each single graph snapshot in the dynamic network. GCN is an efficient variant of convolutional neural networks that can operate directly on graphs. Formally, assume that there are N nodes with M-dimensional features (or attributes) in a static graph. The topological structure and node attributes can be respectively represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ and a feature matrix $\mathbf{Z} \in \mathbb{R}^{N \times M}$ (in which the i-th row of $\mathbf{Z}$ corresponds to the feature vector of node i). A typical GCN unit takes the feature matrix $\mathbf{Z}$ as the input and conducts the spectral graph convolution operation (according to $\mathbf{A}$) with localized first-order approximation on it. The final output is generated in the same way with a standard fully-connected layer. The overall operation of a specific GCN unit can be briefly defined as follows:

$$\mathbf{X} = GCN(\mathbf{Z}, \mathbf{A}) = f\left(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{Z}\mathbf{W}\right), \quad (2)$$

where $\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}$ is the approximated graph convolution filter with $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ ($\mathbf{I}_N$ is an N-dimensional identity matrix) and $\hat{\mathbf{D}}_{ii} = \sum_{j=1}^{N}\hat{\mathbf{A}}_{ij}$; $\mathbf{W}$ represents the weight matrix; $f(\cdot)$ is the activation function; $\mathbf{X}$ is the representation output given by the GCN unit.

For the temporal link prediction task that considers more than one static graph, the GCN-GAN model maintains a GCN unit $GCN(\mathbf{Z}, \mathbf{A}_t) = \mathbf{X}_t$ for each graph snapshot input $\mathbf{A}_t$ ($t \in \{\tau - l, \cdots, \tau\}$). In our model, the feature matrix $\mathbf{Z}$ is set to be the noise input of the generative network, where the value of $\mathbf{Z}$ is generated according to a certain probability distribution p (e.g. the uniform distribution). Based on the

input of $\mathbf{Z}$ and $\mathbf{A}_{\tau-l}^{\tau} = \{\mathbf{A}_{\tau-l}, \cdots, \mathbf{A}_{\tau}\}$, the GCN layer of the generative network outputs a sequence of representations notated as $\mathbf{X}_{\tau-l}^{\tau} = \{\mathbf{X}_{\tau-l}, \cdots, \mathbf{X}_{\tau}\}$.

C. The LSTM Hidden Layer

In the GCN-GAN model, the learned comprehensive network representations $\mathbf{X}_{\tau-l}^{\tau}$ are fed into an LSTM layer, which has a powerful capacity to learn the long-term dependencies of sequential data, to capture the evolving patterns of the weighted dynamic networks. The standard LSTM architecture can be described as an encapsulated cell with several multiplicative gate units. For a certain time step t, the LSTM cell takes current input vector $\mathbf{x}_t$ as well as the state vector of last time step $\mathbf{h}_{t-1}$ as the input, and then output the state vector in current time step $\mathbf{h}_t$:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_x^i \mathbf{x}_t + \mathbf{W}_h^i \mathbf{h}_{t-1} + \mathbf{b}^i\right) \tag{3}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_x^f \mathbf{x}_t + \mathbf{W}_h^f \mathbf{h}_{t-1} + \mathbf{b}^f\right) \tag{4}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_x^o \mathbf{x}_t + \mathbf{W}_h^o \mathbf{h}_{t-1} + \mathbf{b}^o\right) \tag{5}$$

$$\mathbf{s}_t = \mathbf{f}_t \odot \mathbf{s}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{s_t}} \tag{6}$$

$$\tilde{\mathbf{s_t}} = \sigma\left(\mathbf{W}_x^s \mathbf{x}_t + \mathbf{W}_h^s \mathbf{h}_{t-1} + \mathbf{b}^s\right) \tag{7}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{s}_t\right) \tag{8}$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ and $\mathbf{s}_t$ represent the input gate, forget gate, output gate and memory cell, respectively; $\{\mathbf{W}_x, \mathbf{W}_h, \mathbf{b}\}$ are the parameters of the corresponding unit; $\sigma\left(\cdot\right)$ is the sigmoid activation function; $\odot$ denotes the element-wise multiplication.

Eventually, we treat the last hidden state $\mathbf{h}_{\tau+1}$ as the distributed representation of the historical snapshots and feed it into a fully-connected layer to generate the prediction result $\tilde{\mathbf{A}}_{\tau+1}$.

Due to the capacity of learning temporal information of sequential data, one can directly use the LSTM framework (with multiple inputs and single output) to tackle the temporal link prediction task. However, there remain some limitations for the prediction of the weighted dynamic networks. Specifically, to learn the temporal information of the dynamic network, LSTM is usually trained by using the Mean Square Error (MSE) loss function. However, the MSE loss cannot reflect the sparsity and wide-value-range of the link weights in the real network systems, which is empirically demonstrated in Section V.

D. The Generative Adversarial Network

To cope with the sparsity and wide-value-range problem of the dynamic network's edge weights, we utilize the GAN framework to enhance the generative capacity of LSTM.

Typically, GAN consists of a generative model G and a discriminative model D that compete in a minimax game with two players. First, D tries to distinguish real data in the training set from the data generated by G. On the other hand, G tries to fool D and generate high-quality samples (data). Formally, such process can be described as follow (with two alternative optimization steps):

$$\min_G \max_D \left(\begin{array}{c} \mathrm{E}_{x \sim p_{data}(x)}\left[\log D\left(x\right)\right] + \\ \mathrm{E}_{z \sim p(z)}\left[\log\left(1 - D\left(G\left(z\right)\right)\right)\right] \end{array} \right), \tag{9}$$

where x is the input data from the training set, and z represents the noise generated via a certain probability distribution $p\left(z\right)$ (e.g., the uniform distribution).

Like the above standard GAN framework, our model also optimizes two neural networks (i.e., the generative network G and the discriminative network D) with a minimax two-player game. In the model, D tries to distinguish the real graph snapshot in the training data from the snapshot generated by G, while G maximizes the probability of D to make a mistake. Hopefully, this adversarial process can eventually adjust G to generate plausible and high-quality weighted links. We further elaborate such two neural networks below.

a) The Discriminative Network D: As depicted in Fig. 1 (top side), we implement the discriminative model D via a full-connected feedforward neural network with one hidden layer and one output layer. In the training process, D alternatively takes G's output $\tilde{\mathbf{A}}_{\tau+1}$ or the ground-truth $\mathbf{A}_{\tau+1}$ as the input. Since each input data of the full-connected neural network is usually represented as a vector (but not in the form of matrix), we reshape the matrix input (i.e., $\tilde{\mathbf{A}}_{\tau+1}$ or $\mathbf{A}_{\tau+1}$) into a corresponding row-wise long vector when feeding it into D. Moreover, as we adopt the Wasserstein GAN (WGAN) framework [20], [21] to train the model (which is discussed later in this section), we set the output layer to be a linear layer, which directly generates the output without a non-linear activation function. Briefly, the details of the discriminative network D can be formulated as follow:

$$D\left(\mathbf{A}'\right) = \left(\sigma\left(\mathbf{a}' \mathbf{W}_h^D + \mathbf{b}_h^D\right) \mathbf{W}_o^D + \mathbf{b}_o^D\right), \tag{10}$$

where $\mathbf{A}' \in \left\{\mathbf{A}_{\tau+1}, \tilde{\mathbf{A}}_{\tau+1}\right\}$ with $\mathbf{a}'$ as the corresponding reshaped row-wise long vector; $\left\{\mathbf{W}_h^D, \mathbf{b}_h^D\right\}$ and $\left\{\mathbf{W}_o^D, \mathbf{b}_o^D\right\}$ are respectively the parameters of the hidden layer and the output layer; $\sigma\left(\cdot\right)$ represents the sigmoid activation function of the hidden layer.

Since the edge weights of a given network snapshot may have a large value range (e.g., [0, 2,000]), we normalize the element value of $\mathbf{A}_{\tau+1}$ into the range of [0, 1] when selecting $\mathbf{A}_{\tau+1}$ as the input of D. The original prediction result $\tilde{\mathbf{A}}_{\tau+1}$ given by G is defined within the range [0, 1], so it can be directly utilized as the input of D.

b) The Generative Network G: As depicted in Fig. 1 (bottom side), the generative model G consist of a GCN layer, an LSTM layer and a full-connected output layer. The GCN layer takes the graph snapshots sequence $\mathbf{A}_{\tau-l}^{\tau}$ as well as the noise $\mathbf{Z}$ as the input, and outputs the representations sequence $\mathbf{X}_{\tau-l}^{\tau}$ which is later fed into the LSTM layer. Note that each adjacency matrix input $\mathbf{A}_t$ $(t \in \{\tau - l, \cdots, \tau\})$ should be normalized into the range of [0, 1] before being fed into the GCN layer. Moreover, we adopt sigmoid as the activation function of all the GCN units and let the noise input $\mathbf{Z}$ follow a uniform distribution within [0, 1] (notated as $\mathbf{Z} \sim \mathrm{U}\left(0, 1\right)$).

The LSTM layer takes the representations sequence $\mathbf{X}_{\tau-l}^{\tau}$ given by the GCN layer as the input, and outputs the hidden states $\mathbf{h}_{\tau-l}^{\tau} = \{\mathbf{h}_{\tau-l}, \cdots, \mathbf{h}_{\tau}\}$. Note that each matrix input $\mathbf{X}_t$ $(t \in \{\tau - l, \cdots, \tau\})$ should be reshaped into a row-

wise vector $\mathbf{x}_t$ when being fed into the LSTM, since each LSTM cell treats the input data as a vector. Finally, the last hidden state $\mathbf{h}_\tau$ is fed into the output layer to generate the graph snapshot $\tilde{\mathbf{A}}_{\tau+1}$ (with the corresponding row-wise vector form) of the next time slice. In particular, the elements of a certain generated result are within the range of [0, 1]. The final predicted snapshot $\tilde{\mathbf{A}}_{\tau+1}$ with the correct value range of the dynamic network can be obtained by conducting the inverse process of normalization.

In the rest of this paper, we utilize the following simplified notation to represent the generative network G (with noise $\mathbf{Z}$ and network snapshot sequence $\mathbf{A}_{\tau-l}^\tau$ as the input):

$$\tilde{\mathbf{A}}_{\tau+1} = G\left(\mathbf{Z}, \mathbf{A}_{\tau-l}^\tau\right). \tag{11}$$

E. Model Optimization

Since the network's topology dynamically changes over time, the GCN-GAN model should constantly update its parameters to adapt to the network's evolution. Moreover, it's usually assumed that the network snapshots close to the next time slice $(\tau + 1)$ can be considered to have more similar characteristics to the ground-truth compared to those far from it [6]. Based on such reasonable assumption, we utilize the following optimization strategy. When it comes to a new time slice τ, the model first conducts the training process by utilizing the previous network graph sequence $\mathbf{A}_{\tau-l-1}^{\tau-1}$ as the input and current snapshot $\mathbf{A}_\tau$ as the ground-truth. After training the model for current time slice τ, we conduct the prediction process to generate the next graph snapshot $\tilde{\mathbf{A}}_{\tau+1}$ with sequence $\mathbf{A}_{\tau-l}^\tau$ as the input. The details of the training and predicting process are elaborated in the rest of the section.

For the temporal link prediction task, directly utilizing the standard adversarial training process is inappropriate, because G may generate a plausible network snapshot that can successfully fool D but it is not consistent with the next graph snapshot. In fact, we expect that the prediction result should be as close as possible to the ground-truth $\mathbf{A}_{\tau+1}$. In order to tackle such possible problem, we introduce another pre-training process for G with the following loss function:

$$\min_{\theta_G} h\left(\theta_G; \mathbf{Z}, \mathbf{A}_{\tau-l-1}^{\tau-1}, \mathbf{A}_\tau\right) = \left\|\mathbf{A}_\tau - G\left(\mathbf{Z}, \mathbf{A}_{\tau-l-1}^{\tau-1}\right)\right\|_F^2 + \frac{\lambda}{2}\left\|\theta_G\right\|_2^2 , \tag{12}$$

where θ_G represents the parameters of G and λ is the parameter to control the effect of the L_2-regularization term. In (12), G tries to reconstruct current graph snapshot $\mathbf{A}_\tau$ given by the snapshot sequence $\mathbf{A}_{\tau-l-1}^{\tau-1}$ and noise $\mathbf{Z}$. Such process can help G to fully capture the latest temporal information of the dynamic network, which is considered as the most similar characteristics to the real snapshot of $\mathbf{A}_{\tau+1}$.

After the pre-training procedure, G has the initial ability to generate the prediction result. The adversarial training process can be further developed to enhance G's generative capacity to cope with the sparsity and wide-value-range problem of weighted dynamic networks. Especially, we utilize the Wasserstein GAN (WGAN) framework [20], [21], which has been proved to have a more reliable performance than the standard GAN, to achieve a relatively stable training process.

In such procedure, we first utilize the gradient descent method to update D's parameters (notated as θ_D) with G's parameters fixed via the following loss function:

$$\min_{\theta_D} h_D\left(\theta_D; \mathbf{Z}, \mathbf{A}_{\tau-l-1}^{\tau-1}, \mathbf{A}_\tau\right) = \\ \mathrm{E}\left[D\left(\mathbf{A}_\tau\right)\right] - \mathrm{E}\left[D\left(G\left(\mathbf{Z}, \mathbf{A}_{\tau-l-1}^{\tau-1}\right)\right)\right]. \tag{13}$$

After updating the parameters of D, their values should be further clipped into a pre-defined range of $[-c, c]$. Then, we update the parameters of G (denoted as θ_G) with D's parameters fixed by using the following loss function:

$$\min_{\theta_G} h_G\left(\theta_G; \mathbf{Z}, \mathbf{A}_{\tau-l-1}^{\tau-1}\right) = \\ -\mathrm{E}\left[D\left(G\left(\mathbf{Z}, \mathbf{A}_{\tau-l-1}^{\tau-1}\right)\right)\right]. \tag{14}$$

In the experiment, we adopted the RMSProp algorithm to alternatively update the parameters of θ_D and θ_G until converge.

After finishing the training process, G can be utilized to generate the prediction result $\tilde{\mathbf{A}}_{\tau+1}$ with $\mathbf{A}_{\tau-l}^\tau$ and $\mathbf{Z}$ as the input. Note that the original $\tilde{\mathbf{A}}_{\tau+1}$'s elements are within the range of [0, 1], and another renormalization process is needed to recover its values to the real range of the network's edge weights. Furthermore, some tricks can be used to further refine the prediction result, which are formulated as follow:

$$\tilde{\mathbf{A}}_{\tau+1} \leftarrow \left(\tilde{\mathbf{A}}_{\tau+1} + \tilde{\mathbf{A}}_{\tau+1}^T\right)\big/ 2, \tag{15}$$

$$\left(\tilde{\mathbf{A}}_{\tau+1}\right)_{ii} \leftarrow 0 \text{ for } i \in \{1, 2, \cdots, |V|\}, \tag{16}$$

$$\left(\tilde{\mathbf{A}}_{\tau+1}\right)_{ij} \leftarrow 0 \text{ if } \left(\tilde{\mathbf{A}}_{\tau+1}\right)_{ij} < \varepsilon. \tag{17}$$

First, we use (15) to make $\tilde{\mathbf{A}}_{\tau+1}$ symmetric, as we only consider the case of undirected networks. Then, by using (16), we set the diagonal elements of $\tilde{\mathbf{A}}_{\tau+1}$ to be 0 to remove the effect of self-connected edges. Finally, the elements whose values are less than a small threshold ε can be set to 0 to reflect the sparsity of edge weights.

We summarize the overall training and predicting procedures of the GCN-GAN model (when the network system comes to a new time slice τ) in Table I.

V. EXPERIMENTAL EVALUATION

A. Datasets

In order to evaluate the effectiveness of our model, we conduct experiments on three real datasets and one simulation dataset of different network systems. The detailed statistics of these four datasets are presented in Table II, where N, T and Max denote the number of nodes, the number of time slices and the maximum edge weight value for a certain dataset after necessary pre-processing.

UCSB[1] [22] is a link quality dataset of a wireless mesh network; *NumFabric*[2] [23] is a simulation flow dataset of data

---

[1]https://crawdad.org/ucsb/meshnet/20070201/

[2]The data can be generated by running the simulation code released in https://knagaraj@bitbucket.org/knagaraj/numfabric.git

The GAN-CAN Algorithm

Input: $\{\mathbf{A}_{\tau-l-1}, \cdots, \mathbf{A}_{\tau-1}, \mathbf{A}_\tau\}$, $\{\alpha_0, \alpha_D, \alpha_G\}$, $\{n_0, n\}$, c.
Output: $\tilde{\mathbf{A}}_{\tau+1}$.
//$\{\mathbf{A}_{\tau-l-1}, \cdots, \mathbf{A}_{\tau-1}, \mathbf{A}_\tau\}$: the network snapshot sequence
//used to train the model and predict the next network snapshot;
//$\{\alpha_0, \alpha_D, \alpha_G\}$: learning rate for pre-training and formal training;
//$\{n_0, n\}$: number of iterations for pre-training and formal training;
//c: clipping bound; $\tilde{\mathbf{A}}_{\tau+1}$: prediction result.
  //Train the GAN-GAN model
  //with $\mathbf{A}_{\tau-l-1}^{\tau-1} = \{\mathbf{A}_{\tau-l-1}, \cdots, \mathbf{A}_{\tau-1}\}$ as the input
  //and $\mathbf{A}_\tau$ as the ground-truth
  normalize the values of $\{\mathbf{A}_{\tau-l-1}, \cdots, \mathbf{A}_\tau\}$ into [0, 1]
  for i **from** 1 **to** n_0 //Pre-train G
    generate the noise input $\mathbf{Z} \sim \mathrm{U}[0, 1]$
    $\varsigma_{\theta_G} \leftarrow \nabla_{\theta_G} h\left(\theta_G; \mathbf{Z}, \mathbf{A}_{\tau-l-1}^{\tau-1}, \mathbf{A}_\tau\right)$
    $\theta_G \leftarrow \theta_G - \alpha_0 \cdot \mathrm{RMSProp}\left(\theta_G, \varsigma_{\theta_G}\right)$
  for i **from** 1 **to** n //Formally train the model
    generate the noise input $\mathbf{Z} \sim \mathrm{U}[0, 1]$
    $\varsigma_{\theta_D} \leftarrow \nabla_{\theta_D} h_D\left(\theta_D; \mathbf{Z}, \mathbf{A}_{\tau-l-1}^{\tau-1}, \mathbf{A}_\tau\right)$
    $\theta_D \leftarrow \theta_D - \alpha_D \cdot \mathrm{RMSProp}\left(\theta_D, \varsigma_{\theta_D}\right)$
    clip the element value of θ_D into $[-c, c]$
    generate the noise input $\mathbf{Z} \sim \mathrm{U}[0, 1]$
    $\varsigma_{\theta_G} \leftarrow \nabla_{\theta_G} h_G\left(\theta_G; \mathbf{Z}, \mathbf{A}_{\tau-l-1}^{\tau-1}\right)$
    $\theta_G \leftarrow \theta_G - \alpha_G \cdot \mathrm{RMSProp}\left(\theta_G, \varsigma_{\theta_G}\right)$
  //Generate the prediction result $\tilde{\mathbf{A}}_{\tau+1}$
  //with $\mathbf{A}_{\tau-l}^\tau = \{\mathbf{A}_{\tau-l}, \cdots, \mathbf{A}_\tau\}$ as the input
  generate the noise input $\mathbf{Z} \sim \mathrm{U}[0, 1]$
  $\tilde{\mathbf{A}}_{\tau+1} \leftarrow G\left(\mathbf{Z}, \mathbf{A}_{\tau-l}^\tau\right)$
  renormalize $\tilde{\mathbf{A}}_{\tau+1}$ into the real value range
  refine $\tilde{\mathbf{A}}_{\tau+1}$ via (15), (16) and (17)

Datasets	N	T	Max	Description
UCSB	38	1,000	2,000	Wireless mesh net link quality
KAIST	92	500	250	Human mobility position data
BJ-Taxi	256	500	2,000	Vehicle mobility position data
NumFabric	128	350	20,000	Simulation data center flow

center; *KAIST*[3] [24] and *BJ-Taxi*[4] [25] are the position datasets of a human mobility network and a vehicle mobility network, respectively. (Both *KAIST* and *BJ-Taxi* are the subsets of the original datasets.) For each dataset, we pre-process the dynamic network into multiple successive graph snapshots.

With regard to *UCSB* and *NumFabric*, the hosts in the network systems can be described as the nodes in the dynamic networks. Moreover, the link quality or flow in a certain time slice can be directly represented as the link weight between the corresponding pair of hosts in the specific snapshot.

For the position datasets *KAIST* and *BJ-Taxi*, we treated each user (person or vehicle) as a node in the abstracted dynamic network and calculated the distance between each pair of nodes for all the time slices. Particularly, we constructed a distance matrix $\mathbf{D}_t$ for a specific time slice t,

with $(\mathbf{D}_t)_{ij} = (\mathbf{D}_t)_{ji}$ representing the distance between node i and j. In real network systems of human mobility and vehicle mobility, the pair of users who are close to each other should be given more interest or attention compared to those with relatively large distance. Hence, we constructed an abstracted weighed network based on the distance matrix $\mathbf{D}_t$, in which the link weights were inversely proportional to the corresponding distance. When pre-processed the *KAIST* and *BJ-Taxi* datasets, we utilized the following formulation to construct the weighted adjacency matrix $\mathbf{A}_t$ of the graph snapshot in time slice t:

$$(\mathbf{A}_t)_{ij} = \begin{cases} 0, & \text{if } (\mathbf{D}_t)_{ij} \geq \delta \\ \delta - (\mathbf{D}_t)_{ij}, & \text{if } (\mathbf{D}_t)_{ij} < \delta \end{cases}, \quad (18)$$

where we set the link weight to be 0 if the corresponding distance was larger than a pre-set threshold δ. Particularly, δ is the maximum edge weight for a certain network (with $\delta = 250$ for *KASIT* and $\delta = 2,000$ for *BJ-Taxi* in our experiments).

Furthermore, we also statistically analyze the sparsity and distribution of the link weights for the above four networks.

The sparsity of link weights. According to our observation, most of the graph snapshots are relatively sparse, which means there is a non-ignorable portion of zeros in the adjacency matrix of a certain time slice. For example, the average portions of zero elements in all the adjacency matrices for *UCSB*, *KASIT*, *BJ-Taxi* and *NumFabric* are **0.52**, **0.92**, **0.94** and **0.50**, respectively. Although the sparsity degree of different network systems may be significantly different, such statistical result can still reveal the fact that there are usually non-ignorable pairs of entities in the network system that may not have the defined relation (e.g., the relation of data transmission in the data center).

For the weighted temporal link prediction task, $(\mathbf{A}_t)_{ij} = 0$ means there is no edge between node i and j. On the other hand, $(\mathbf{A}_t)_{ij}$ with small value means there is still an edge between this pair of nodes but the weight is small. Such two circumstances are entirely different, but to distinguish them remains a challenging problem for most of the conventional temporal link prediction methods.

In fact, the mistake that a prediction model fails to distinguish small edge weights and zero values is relatively serious for most network systems. It may mistakenly direct the system to (i) pre-allocate the key resources for the nonexistence links or (ii) not allocate resources for the existent links, resulting in more waste of system overhead than the normal prediction error of the existent links' weights.

The distribution of edge weights. The wide value range of link weights (e.g., from 0 to 20,000) is another significant property we have observed. To ensure the learning ability of the model, most link prediction approaches normalize the link weights input into a certain small range (e.g. $[0, 1]$). However, when recovering the value of the predicted snapshot, the very slight errors (between 0 and 1) may still result in large errors in the final results with the mean square error evaluation metric.

In this study, we also investigate the distribution of edge weights for the four datasets. The statistic of $UCSB$ is

illustrated in Fig. 2 as an example, in which w (the horizontal axis) represents the possible edge weight value of the dataset, while $c(w)$ represents the number of edges with value w. Note that we use the logarithm of $c(w)$ as the vertical axis.
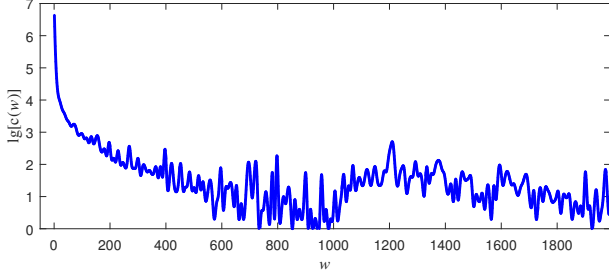

Fig. 2. The distribution of the edge weights of *UCSB*, in which the horizontal axis (w) represents the possible edge weight value while the logarithm of $c(w)$ is used as the vertical axis. $c(w)$ is the number of edges with value w.

As shown in Fig. 2, a large proportion of edges have small weights in the network. However, the conventional Mean Square Error (MSE) metric used to train and evaluate the temporal link prediction model is only sensitive to edges with large weights. It cannot reflect the dynamic changing of the majority of the edges with small weights, making the temporal link prediction of weighted networks a challenging problem.

B. Evaluation Metrics

To quantitatively evaluate our temporal link prediction model, we follow previous work [6], [12] to use the Mean Square Error (MSE) scores for comparison, which is defined as:

$$MSE = \left\| \mathbf{A}_{\tau+1} - \tilde{\mathbf{A}}_{\tau+1} \right\|_F^2 \Big/ (|V| \times |V|). \qquad (19)$$

Moreover, in order to further evaluate the capacity of our model to cope with the sparsity and wide-value-range problem discussed above, we introduce two additional metrics (i.e., the edge-wise KL-divergence and the mismatch rate).

a) The Edge-wise KL-Divergence: For some dynamic networks, the edge weights of a snapshot may have a wide value range (e.g., [0, 2,000]), in which the majority of the edges are with relatively small weights. Nevertheless, the MSE score may be sensitive to the large edge weights only and suffers from distinguishing the difference of magnitude that is important for small weights. For instance, the magnitude difference between 2 and 1 should be much larger than the difference between 2,000 and 1,990, even though the latter case results in larger MSE error. To alleviate the mentioned issue, we introduce the edge-wise KL-divergence to further consider the magnitude difference of link weights.

Firstly, we derive two auxiliary matrices $\mathbf{P}$ and $\mathbf{Q}$ to represent the normalized values of the ground-truth graph snapshot $\mathbf{A}_{\tau+1}$ and the prediction result $\tilde{\mathbf{A}}_{\tau+1}$, respectively. We formulate $\mathbf{P}$ and $\mathbf{Q}$ as follows:

$$\mathbf{P}_{ij} = \frac{(\mathbf{A}_{\tau+1})_{ij}}{\sum_{i,j=1}^{N} (\mathbf{A}_{\tau+1})_{ij}}, \mathbf{Q}_{ij} = \frac{\left(\tilde{\mathbf{A}}_{\tau+1}\right)_{ij}}{\sum_{i,j=1}^{N} \left(\tilde{\mathbf{A}}_{\tau+1}\right)_{ij}}. \qquad (20)$$

Then, the edge-wise KL-divergence is defined as:

$$KL\left(\mathbf{P} \,\|\, \mathbf{Q}\right) = \sum_{i,j=1}^{N} f\left(\mathbf{P}_{ij}, \mathbf{Q}_{ij}\right), \qquad (21)$$

where $f\left(\mathbf{P}_{ij}, \mathbf{Q}_{ij}\right) = \mathbf{P}_{ij} \log\left(\mathbf{P}_{ij}/\mathbf{Q}_{ij}\right)$ (with the standard form of the KL-divergence) if $\mathbf{P}_{ij} > 0$ and $\mathbf{Q}_{ij} > 0$; otherwise $f\left(\mathbf{P}_{ij}, \mathbf{Q}_{ij}\right) = 0$. Note that when $\mathbf{P}_{ij} = 0$ or $\mathbf{Q}_{ij} = 0$, we simply set their KL-divergence to 0 since the zero value may cause the NaN or Inf exception, and we consider such special cases in the definition of the mismatch rate below.

b) The Mismatch Rate: According to our observation on the datasets, the sparsity issue of the edge weights in the weighted dynamic network is also significant, which need to be specially discussed. We consider the following two cases:
(a) $(\mathbf{A}_{\tau+1})_{ij} = 0$ but $\left(\tilde{\mathbf{A}}_{\tau+1}\right)_{ij} > 0$;
(b) $(\mathbf{A}_{\tau+1})_{ij} > 0$ but $\left(\tilde{\mathbf{A}}_{\tau+1}\right)_{ij} = 0$.
Particularly, such two cases mean the prediction results improperly determine the existence of the edge (i, j), which should be considered as serious mistakes for the temporal link prediction of weighted dynamic networks. Hence, we use the mismatch rate, which represents the proportion of such mismatched edges in a certain graph snapshot, as an additional evaluation metric.

C. Performance Evaluation

We evaluate the effectiveness of our model by comparing it with six baseline methods on the four datasets, including *ED* [12], *SVD* [12], *NMF* [8], *GrNMF* [6], *AM-NMF* [17] and *LSTM* [10]. Among the baselines, *ED*, *SVD* and *NMF* are conventional approaches based on the collapsed network [14], [15], while *GrNMF* and *AM-NMF* are state-of-the-art matrix factorization-based methods without collapsing the dynamic network. *LSTM* represents the non-linear model that directly utilize LSTM [10] to perform the temporal link prediction.

In the experiment, we uniformly set $l = 10$ for all the methods to be evaluated. For the matrix factorization-based methods (i.e., *ED*, *SVD*, *NMF*, *GrNMF* and *AM-NMF*), we set the hidden space size to be 16, 64, 128 and 64 for *UCSB*, *KAIST*, *BJ-Taxi* and *NumFabric*, respectively. For *ED*, *SVD* as well as *NMF*, we select the parameters with the best performance, while we utilize the recommended parameter settings for *GrNMF* and *AM-NMF*.

With regard to the non-linear methods (i.e., *LSTM* and *GCN-GAN*), we use the Xavier approach [26] to initialize the parameters. For a dataset with N nodes, we set the noise input $\mathbf{Z}$ and the adjacency matrix input $\mathbf{A}_t$ of the GCN-GAN model to have the same size of $N \times N$. Note that the input of a GCN unit is in the form of matrix, and its output should be reshaped into a row-wise vector before being fed into the LSTM layer. The layer configurations of the four datasets with the format of m_i-m_h-m_o are illustrated in Table III, where m_i is the size of the input in each time step, m_h represents the hidden size of LSTM and m_o is the size of the output. Especially, we use $N \times N$ to represent the matrix input (with the size of $N \times N$) and utilize N^2 to represent the size of a (row-wise) long vector which can be reshaped into an $N \times N$ matrix.

TABLE III
THE LAYER CONFIGURATIONS OF GCN-GAN AND LSTM.

| Datasets | GCN-GAN | | LSTM |
	G	D	
UCSB	(38×38)-38-38^2	38^2-512-1	38^2-128-38^2
KAIST	(92×92)-92-92^2	92^2-512-1	92^2-128-92^2
BJ-Taxi	(256×256)-256-256^2	256^2-1024-1	256^2-512-256^2
NumFabric	(128×128)-128-128^2	128^2-1024-1	128^2-512-128^2

The parameter settings of the four datasets are shown in Table IV, where λ is the parameter controls the effect of the L_2-regularization term in (12); ε is the threshold used to refine the prediction result in (17); α_0 is the learning rate of the pre-training process; α_D and α_G are the learning rate for training D and G; c is the clipping bound for D's parameters θ_D.

TABLE IV
THE PARAMETER SETTINGS OF GCN-GAN.

| Datasets | Parameters | | | | | |
	λ	ε	α_0	α_D	α_G	c
UCSB	0	0.01	0.005	0.001	0.001	
KAIST	1e-5	0.01	0.01	0.0005	0.0005	0.01
BJ-Taxi	1e-5	0.01	0.005	0.001	0.001	
NumFabric	0	0.5	0.001	0.001	0.001	

For each dataset, we use the first $(l+2)$ graph snapshots as the original training data to conduct the initial training process. Then, we continuously slide the window with size $(l+1)$ for the rest snapshots, in which we alternatively conduct the training and predicting process. For each evaluation metric, we record the average performance value of all the predicted snapshots. The evaluation results in terms of MSE, edge-wise KL-divergence and mismatch rate are shown in Table V, Table VI and Table VII respectively, where the best performance value is in **bold** and the second-best is with underline.

TABLE V
PERFORMANCE EVALUATION RESULTS IN TERMS OF MSE.

| Methods | Datasets | | | |
	UCSB	KAIST	BJ-Taxi	NumFabric
ED	8.1118	0.5067	0.3134	33.6076
SVD	5.4185	0.2043	0.2789	3.9849
NMF	5.8586	0.2683	0.2825	4.0368
GrNMF	5.6767	0.1381	0.2342	2.6210
AM-NMF	5.6716	0.1380	0.2343	2.6364
LSTM	5.2518	0.1196	**0.2283**	0.6867
GCN-GAN	**5.1154**	**0.1189**	0.2284	**0.6570**

Note that the precision differences of the evaluation metrics among the comparative methods in different datasets are different, since different network systems may have diverse edge weight ranges and temporal characteristics. In Table V, the non-linear models (i.e., *LSTM* and *GCN-GAN*) achieve much better MSE scores than the linear approaches, indicating the powerful feature learning ability of the non-linear deep neural network. Particularly, our method has the best MSE scores on three datasets (i.e., *UCSB*, *KAIST* and *NumFabric*)

TABLE VI
PERFORMANCE EVALUATION RESULTS IN TERMS OF EDGE-WISE
KL-DIVERGENCE.

| Method | Datasets | | | |
	UCSB	KAIST	BJ-Taxi	NumFabric
ED	0.5960	0.3619	1.0541	0.0681
SVD	0.4726	0.1351	1.0085	0.0085
NMF	0.6980	0.1021	1.2519	0.0094
GrNMF	0.7696	0.0653	0.4674	0.0073
AM-NMF	0.7981	0.0649	0.4752	0.0073
LSTM	0.6120	0.0578	1.3892	0.0012
GCN-GAN	**0.3247**	**0.0262**	**0.2831**	**0.0009**

TABLE VII
PERFORMANCE EVALUATION RESULTS IN TERMS OF MISMATCH RATE.

| Method | Datasets | | | |
	UCSB	KAIST	BJ-Taxi	NumFabric
ED	0.2227	0.1907	0.1923	0.4806
SVD	0.1801	0.2466	0.2425	0.0003
NMF	0.1651	0.1638	0.1932	0.0001
GrNMF	0.1264	0.0912	0.0258	0.0001
AM-NMF	0.1305	0.0918	0.0283	0.0001
LSTM	0.2905	0.1175	0.9118	0.4923
GCN-GAN	**0.0133**	**0.0122**	**0.0173**	**3e-5**

and still obtains the competitive second-best MSE with the best performer (i.e., *LSTM*) in *BJ-Taxi*. For edge-wise KL-divergence and mismatch rate, the proposed method outperforms all the baselines on the four datasets, which further verifies that GCN-GAN is able to alleviate the sparsity and wide-value-range problem of weighted dynamic networks.

D. Case Study

We use an exemplary case, which is selected from the prediction results of *UCSB*, to demonstrate our model's capacity to generate high-quality weighted links. For a certain graph snapshot, we compare the adjacency matrices generated by different methods with the ground-truth. Heat maps are used to visualize the results. Typically, in an adjacency matrix, the zero elements and those with small values have entirely different physical meanings, thus we specially set all the zero values in the matrix to -200 to emphasize such difference.

The results are reported in Fig. 3, in which subfigure (a) presents the ground-truth while subfigures (b) and (c) show the prediction results of *GCN-GAN* and *LSTM*, respectively. In the heat map, black represents zero value of the adjacency matrix (note that we set values of all the zero elements to be -200). Moreover, the color depth indicates the weights of the edges, in which the color close to dark red indicates relatively small edge weight while that close to white means large value.

As illustrated in Fig. 3, both *GCN-GAN* and *LSTM* can fit the large edge weight (with color close to white) well. However, the *LSTM* fails to distinguish the zero values (with color of black) and small edge weights (with color close to dark red). On the other hand, our *GCN-GAN* model can effectively cope with such challenging problem, reflecting the sparsity of edge weights of the network snapshot.

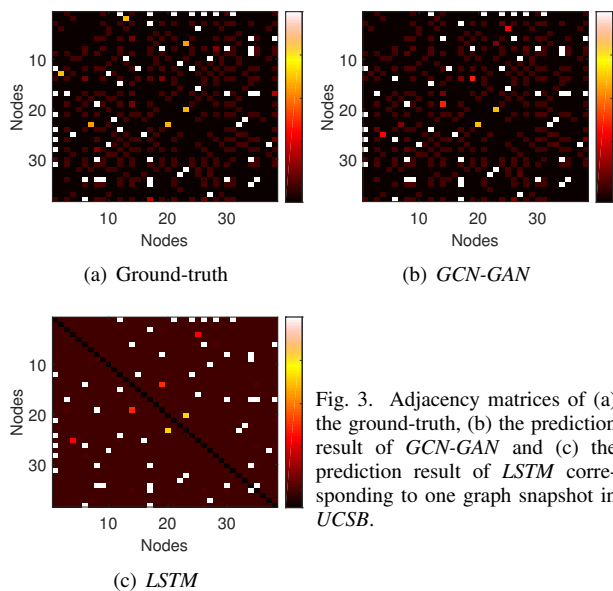(a) Ground-truth

(b) *GCN-GAN*

(c) *LSTM*

Fig. 3. Adjacency matrices of (a) the ground-truth, (b) the prediction result of *GCN-GAN* and (c) the prediction result of *LSTM* corresponding to one graph snapshot in *UCSB*.

VI. Conclusion

In this paper, we proposed a novel temporal link prediction model GCN-GAN to generally tackle the dynamics prediction problem in network systems (e.g., the prediction of mobility, traffic and topology). Our model can effectively deal with challenging prediction task of weighted dynamic networks, because it combined the strengths of the deep neural networks (i.e., GCN and LSTM) in learning the comprehensive distributed representations of networks as well as GAN in generating high-quality weighted links. In addition, we applied the proposed model to four datasets of different network systems and specially analyzed the sparsity as well as the wide-value-range properties of the edge weights in real-life network systems. The performance evaluation results demonstrated that the proposed model outperformed other six competitors while having the powerful capacity to tackle the sparsity and wide-value-range problem of weighted dynamic networks.

In our future work, we will consider the concrete deployment scenario of real network systems and use the measures of communication networks to evaluate the performance improvement for different systems. More importantly, how to cope with the challenging temporal link prediction problem with an unfixed node set is also our next research focus.

VII. Acknowledgment

References

[1] L. Ghouti, T. R. Sheltami, and K. S. Alutaibi, "Mobility prediction in mobile ad hoc networks using extreme learning machines," *Procedia Computer Science*, vol. 19, pp. 305–312, 2013.

[2] A. Mozo, B. Ordozgoiti, and S. Gomez, "Forecasting short-term data center network traffic load with convolutional neural networks," *Plos One*, vol. 13, no. 2, p. e0191939, 2018.

[3] N. B. Prajapati and D. R. Kathiriya, "Mobility prediction for dynamic location area in cellular network using hidden markov model," in *Industry Interactive Innovations in Science, Engineering and Technology*. Springer Singapore, 2018, pp. 349–355.

[4] L. Nie, X. Wang, L. Wan, S. Yu, H. Song, and D. Jiang, "Network traffic prediction based on deep belief network and spatiotemporal compressive sensing in wireless mesh backbone networks," *Wireless Communications & Mobile Computing*, vol. 2018, pp. 1–10, 2018.

[5] S. Verma, A. Narayanan, and Z. L. Zhang, "Multi-low-rank approximation for traffic matrices," in *Teletraffic Congress*, 2017, pp. 72–80.

[6] X. Ma, P. Sun, and Y. Wang, "Graph regularized nonnegative matrix factorization for temporal link prediction in dynamic networks," *Physica A Statistical Mechanics & Its Applications*, vol. 496, 2018.

[7] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge & Data Engineering*, vol. PP, no. 99, pp. 1–1, 2017.

[8] D. Lee, "Learning the parts of objects with nonnegative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016.

[10] F. A. Gers, J. A. Schmidhuber, and F. A. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2014.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *International Conference on Neural Information Processing Systems*, 2014, pp. 2672–2680.

[12] X. Ma, P. Sun, and G. Qin, "Nonnegative matrix factorization algorithms for link prediction in temporal networks using graph communicability," *Pattern Recognition*, vol. 71, 2017.

[13] Z. Huang and D. K. J. Lin, *The Time-Series Link Prediction Problem with Applications in Communication Surveillance*. INFORMS, 2009.

[14] D. Liben-Nowell and J. Kleinberg, *The link-prediction problem for social networks*. John Wiley & Sons, Inc., 2007.

[15] U. Sharan and J. Neville, "Temporal-relational classifiers for prediction in evolving domains," in *Eighth IEEE International Conference on Data Mining*, 2008, pp. 540–549.

[16] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *Acm Transactions on Knowledge Discovery from Data*, vol. 5, no. 2, pp. 1–27, 2011.

[17] K. Lei, M. Qin, B. Bai, and G. Zhang, "Adaptive multiple non-negative matrix factorization for temporal link prediction in dynamic networks," in *ACM SIGCOMM 2018 Workshop on Network Meets AI and ML*, 2018, pp. 1–27.

[18] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang, "A deep learning approach to link prediction in dynamic networks," in *SIAM International Conference on Data Mining 2014, SDM 2014*, vol. 1, 2014, pp. 289–297.

[19] R. Hisano, "Semi-supervised graph embedding approach to dynamic link prediction," in *International Workshop on Complex Networks*, 2018, pp. 109–121.

[20] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," 2017.

[21] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[22] K. Ramachandran, I. Sheriff, E. Belding, and K. Almeroth, "Routing stability in static wireless mesh networks." in *Passive and Active Measurement Conference, Louvain-La-Neuve, Belgium, April*, 2007, pp. 73–82.

[23] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti, "Numfabric: Fast and flexible bandwidth allocation in datacenters," in *Conference on ACM SIGCOMM 2016 Conference*, 2016, pp. 188–201.

[24] K. Lee, S. Hong, S. J. Kim, and I. Rhee, "Slaw: A new mobility model for human walks," in *INFOCOM*, 2009, pp. 855–863.

[25] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 316–324.

[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research*, vol. 9, pp. 249–256, 2010.