Temporal Network Embedding for Link Prediction via VAE Joint Attention Mechanism

Pengfei Jiao, Xuan Guo, Xin Jing, Dongxiao He, Huaming Wu, *Member, IEEE*, Shirui Pan, *Member, IEEE*, Maoguo Gong, *Senior Member, IEEE*, and Wenjun Wang

***Abstract*— Network representation learning or embedding aims to project the network into a low-dimensional space that can be devoted to different network tasks. Temporal networks are an important type of network whose topological structure changes over time. Compared with methods on static networks, temporal network embedding (TNE) methods are facing three challenges: 1) it cannot describe the temporal dependence across network snapshots; 2) the node embedding in the latent space fails to indicate changes in the network topology; and 3) it cannot avoid a lot of redundant computation via parameter inheritance on a series of snapshots. To overcome these problems, we propose a novel TNE method named temporal network embedding method based on the VAE framework (TVAE), which is based on a variational autoencoder (VAE) to capture the evolution of temporal networks for link prediction. It not only generates low-dimensional embedding vectors for nodes but also preserves the dynamic nonlinear features of temporal networks. Through the combination of a self-attention mechanism and recurrent neural networks, TVAE can update node representations and keep the temporal dependence of vectors over time. We utilize parameter inheritance to keep the new embedding close to the previous one, rather than explicitly using regularization, and thus, it is effective for large-scale networks. We evaluate our model and several baselines on synthetic data sets and real-world networks. The experimental results demonstrate that TVAE has superior performance and lower time cost compared with the baselines.**

***Index Terms*— Link prediction, self-attention mechanism, temporal network embedding (TNE), variational autoencoder (VAE).**

Manuscript received May 15, 2020; revised November 23, 2020 and May 19, 2021; accepted May 24, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFC0809804; in part by the National Natural Science Foundation of China under Grant 61902278, Grant 62071327, and Grant 61876128; and in part by the Tianjin Municipal Science and Technology Project under Grant 19ZXZNGX00030. *(Corresponding author: Dongxiao He.)*

Pengfei Jiao is with the Center of Biosafety Research and Strategy, Law School, Tianjin University, Tianjin 300350, China (e-mail: pjiao@tju.edu.cn).

Xuan Guo, Xin Jing, Dongxiao He, and Wenjun Wang are with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, China (e-mail: jingxin1895@gmail.com; hedongxiao@tju.edu.cn; guoxuan@tju.edu.cn; wjwang@tju.edu.cn).

Huaming Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

Shirui Pan is with the Department of Data Science and AI, Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia (e-mail: shirui.pan@monash.edu).

Maoguo Gong is with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Xidian University, Xi'an 710071, China (e-mail: gong@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3084957.

Digital Object Identifier 10.1109/TNNLS.2021.3084957

I. Introduction

A VARIETY of complex systems can be represented as networks, e.g., protein–protein interaction networks [1], social networks [2], information networks [3], and co-authorship networks [4]. Generally speaking, complex networks have high-dimensional, isomorphism invariant, and nonlinear features [5]. Hence, directly using the original topological structure of the network in machine learning tasks is inefficient and difficult [6]. Learning latent low-dimensional representations (embeddings) of network nodes is an important way in data mining and complex network analysis. Then, network embeddings can be utilized in a variety of applications, including node classification [7], community detection [8], [9], link prediction [10], knowledge graphs [11], and recommendation system [6]. Thus, how to find low-dimensional embeddings that capture the essential features and properties of the network is an important and essential challenge.

Recently, various approaches for network embedding have been developed, including methods based on a random walk, matrix factorization, and deep learning methods. Random walk-based methods usually approximate many properties in the network by obtaining co-occurrence probabilities, including node centrality and similarity [12]–[14]. Matrix factorization-based methods usually decompose the adjacency or the higher order similarity matrix of the network into a low-dimensional space [15], [16]. Deep learning-based methods are adopted for capturing the nonlinear structure of the network [17], [18]. There are also some surveys on the network embedding in [6] and [19]–[21]. To sum up, these methods and models are only designed for static networks which do not undergo structural changes over time.

In the real world, however, the topological structure of the network is always varying with time, which is referred to as a temporal network [22]. Compared with static networks, temporal networks always present diverse evolutionary mechanisms [23], [24], which makes embeddings more complicated and difficult. For instance, in communication networks, complex interactions are ubiquitous, which gives rise to a dramatic change of the topology structure due to certain events. The temporal network is usually represented as a sequence of snapshots at different time steps [25], there are some complex and intrinsic associations and transformations across the snapshots [26]. Static embedding methods can only handle these network snapshots separately, and they cannot discover the dependencies between snapshots as they ignore historical information. Therefore, these methods fail to model the evolution of temporal networks and the

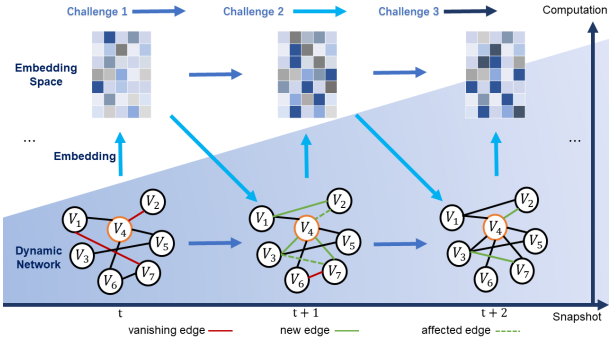


Fig. 1. Three continuous network snapshots and their representation in the latent space. Different arrows are used to express the three challenges for the temporal network embedding (TNE) task: 1) the dependence between consecutive snapshots should be captured into the embeddings; 2) the varying network structure needs to be reflected and predicted with the embeddings; and 3) with many consecutive snapshots might be input, the number of model parameters need be limited for scalability.

prediction performance of temporal networks is not good. In addition, they are difficult to deal with large-scale networks due to the independent and repetitive calculations for all the snapshots.

Although some embedding methods for the temporal or dynamic networks have been proposed [27]–[29], there are still several unfathomed challenges in existing researches: 1) Embeddings should effectively characterize and describe the dependence across the temporal networks. Since the formation of networks in the real world is usually sequential and organized, a TNE method should sniff out a ligament among the snapshots and reveal the evolution patterns. 2) Embeddings should be able to reflect and predict the changing network structure. In essence, the evolution of temporal networks is mainly reflected in the birth and death of links between nodes. Hence, the evolution of temporal networks can be considered as a generative process, and a temporal embedding method should appropriately reflect the underlying principle of network topology development. 3) Redundant calculations should be avoided in the embedding of temporal networks. There are usually stable and abrupt changes in temporal networks; therefore, the embedding vectors of some transitory snapshots are stable to a certain extent.

As an illustration, Fig. 1 shows a temporal social network, in which there are three consecutive snapshots. We consider that the embedding vector and topological structure between successive snapshots should have certain dependencies. For example, node V_4 has some familiar friends at time t. We have greater assurance that his friends will maintain a close relationship with him in the next snapshot. Since low-dimensional embedding has the ability to describe the original network, we consider that the evolution of embedding should reflect the evolution of the network. When one node in the low-dimensional space is sufficiently similar to some disconnected nodes, we can infer that they will establish connections in the next snapshot, e.g., the link between nodes V_3 and V_7. Similarly, when one node is far enough from some neighbors, they will lose connection. The continuous evolution process determines that the network evolution should remain stable at short notice. An incremental update is not only to keep embedding steady but also to reduce superfluous computation for efficiency. Therefore, one method should effectively and efficiently model dynamic information and predict changes in temporal networks.

To cope with these challenges, we try to come up with an end-to-end TNE method for link prediction. It can generate the node embeddings at each snapshot, which combines the generation of links and the dynamic changes of the network and can be used for link prediction.

To learn the essential features of the temporal network, we employ variational autoencoder (VAE) [30] to produce the representations of each snapshot for the prediction of the temporal network. In detail, we deem that the generation of links in the network follows a nonlinear rule. Accordingly, the node embedding in the latent space may follow a Gaussian or more complex distribution. We then reconstruct the topology of the next network snapshot. Therefore, the temporal embedding vectors can be generated which reflect the change of network topology. Furthermore, to overcome the fuzzification of VAE, we import a self-attention mechanism [31] to guide the embedding toward a more reasonable direction. This is because the attention mechanism can extract key structural information in networks [32]. In the meantime, we try to preserve the implicit temporal dependence of the network in the latent space. In the proposed model, we utilize a network embedding sequence with the long short term memory (LSTM) architecture [33] to preserve significant features of the temporal network. We take advantage of the forget gate and the output gate to conserve essential relationships and drop disturbance information in a network. In this way, we can learn the dependence on the nodes across the snapshots. These are solutions designed specifically for challenges 1 and 2. In addition, we employ parameter inheritance to maintain the stability of network embedding and avoid redundancy computing. More detailed, we formalize the current snapshot of the proposed model with the former parameters which are trained on the previous time step and it has the ability to reconstruct the current snapshot structure. With this trick, the model only needs to learn the changes between two snapshots and can be easily extended to different dynamic patterns of the temporal network. This is the solution designed for the last challenge.

In general, we have constructed a well-designed temporal network embedding method based on the VAE framework (TVAE) that can effectively solve the above-mentioned challenges. Besides, our proposed model shows an intuitive and interpretable mechanism to integrate the aforementioned means. Experimental results show that it has a convincing performance on the temporal link prediction task with a lower time cost than other methods. The key contributions of this article are summarized as follows.

1) We propose TVAE, a TNE method based on the VAE framework, which utilizes the representations in the latent space to describe the evolution of the network topology.
2) To handle the embedding orientation and snapshot sequence dependence, we combine a self-attention mechanism and LSTM, which stand for spatial dependence and temporal dependence of embedding, respectively. We also adopt a parameter inheritance mechanism to keep a time-friendly algorithm.

3) We compare our model on some synthetic and real-world networks with baselines. Experimental results demonstrate that our model has superior performance on link prediction and lower time cost than others.

II. Related Work

In this section, we will introduce the embedding methods for both the static network (one snapshot) and temporal network, as well as some related work for link prediction in dynamic or temporal networks.

A. Static Network Embedding

Network embedding methods usually represent the nodes of the network at a single point in time. In general, static embedding methods mainly fall into three categories: random walk, matrix factorization, and deep learning.

1) Random Walk-Based Methods: These methods are suitable for a partially observable network or large entirety. DeepWalk [13] considered a set of short truncated random walks as its own corpus while the network node had its own vocabulary. Furthermore, node2vec [12] defined a flexible notion of a node's network neighborhood whose embeddings organize nodes based on their network roles. On the whole, these methods can capture the context similarity of nodes in an unsupervised way. **Matrix decomposition-based methods:** these methods project the similarity matrix of the network to obtain the node embeddings. Grarep [15] concatenated the k-step representation as a global feature for each node, and hence, different local information was implied in each different step representation. Higher order proximity embedding [16] preserved asymmetric transitivity by approximating high-order proximity. Modularized nonnegative matrix factorization (M-NMF) [34] preserved the microscopic structure (first- and second-order proximities) and mesoscopic structure (community). In summary, this type of method mainly uses one certain similarity or a combination of several similarities or some combinations of kinds of proximity to represent the network structure. **Deep learning-based methods:** deep learning techniques have been recently exploited for learning network representations [19], [35], [36]. Structural deep network embedding (SDNE) [37] optimized first-order and second-order proximities to capture the highly nonlinear network structure by utilizing deep autoencoder architecture. Kipf and Welling [38] first proposed graph convolutional networks (GCNs) for network embedding and node classification and extended it to achieve network reconstruction in an unsupervised way [30]. GraphSAGE [25] stacked the aggregation layers to generate node embeddings, whose features can be sampled and aggregated from their local neighborhood.

B. Temporal Network Embedding

TNE methods need to consider a series of network snapshots and excavate the internal evolution mechanism. The embeddings should have the ability to generate the observed snapshot and predict the network changes. In recent years, the existing temporal embedding methods are generally divided into two categories: snapshot-based [39] and timestamped graph-based methods [40].

1) Snapshot-Based Methods: These methods treat the temporal network as a series of network sequences. Based on skip-gram, Du *et al.* [28] designed a decomposable objective function and node adjustment mechanism to update embedding selectively. Some deep autoencoder models reveal nice performance at the temporal scene. To generate stable embeddings, DynGEM [29] was proposed, which inherited the parameters of the last time step to initialize the model. It designed a heuristic to expand the layers automatically when the size of the ancient was unsuitable. NetWalk [41] can be used to detect anomalies in the dynamic network based on node embedding. DySAT [42] employed joint self-attention along with the two aspects of the structural neighborhood and temporal dynamics. DynGAN [43] leveraged generative adversarial networks and recurrent networks to capture temporal and structure information. The model utilized GAN to generate a raw embedding vector and push it into several stacked LSTM layers to reconstruct and predict original or unseen networks. Through multihead attention, they endowed their model with expressivity to capture dynamic graph evolution from different latent perspectives. Spatio-temporal attentive recurrent neural network model (STAR) [44] was designed for temporal attributed graphs based on the spatio–temporal gated recurrent unit (GRU) and a dual-attention module. Transformer-style relational reasoning network (TRRN) [45] took the transformer-style self-attention and policy network to update the node vector and further devoted it to node classification and link prediction.

2) Timestamped Graph-Based Methods: These methods endow the record for every event (the temporal network could be denoted as a series of events, including the varying nodes and edges [39]). CTDNE [46] considered the temporal network as a timestamped graph, which imposed the temporal sequence constraint on a random walk to improve the accuracy. M^2DNE [47] defined microdynamics and macrodynamics. Specifically, for microdynamics, the model designed a temporal attention point process for chronological events, while for macrodynamics, the inherent evolution pattern was captured by a parameterized dynamic equation. Unfortunately, most of these methods have observed changes in temporal networks, but have ignored the connotative factor of variation. Joint dynamic user-item embeddings (JODIE) [48] focused on modeling the sequential interaction patterns of users and items. Temporal graph attention (TGAT) [49] recognized the node embeddings with temporal feature and used for transductive and inductive tasks. TGNs [50] was an efficient framework with memory modules for dynamic events in the network. Temporal dependency interaction graph (TDIG)-message passing neural network (MPNN) [51] captured the fine-grained global and local information on the temporal dependence interaction graph.

Overall, these methods designed for temporal networks only focus on describing the changes of the network structure but ignore its prediction.

C. Temporal Link Prediction

The link prediction of the temporal network is incredibly challenging. Naturally, the link prediction task becomes a trustworthy metric for evaluating the embedding models

TABLE I
NOTATIONS AND THEIR DEFINITIONS

Notations	Definitions
$\mathcal{G}$	the temporal network
$\mathcal{A}$	the adjacency matrix of $\mathcal{G}$
$\mathcal{E}$	the series of edge set of the network
G_t, A_t	the network snapshot and its adjacency matrix of t
V, n	the set and the number of nodes of the network
d	the number of embedding dimensions
v_i	the i-th node of the network
N_i	the neighbor set of node v_i
Z^t	the $n \times d$ embedding matrix of nodes at snapshot t
z_i^t	the embedding vector of node v_i at snapshot t
h	the number of historical snapshots that Z^t depends on
$\bar{x}_i^t$	the output of node v_i for snapshot t of TVAE
$\bar{A}^t$	the reconstructed adjacency matrix by TVAE
$\mathcal{Q}\left(z^t \middle\| x^t\right)$	the approximated posterior distribution
$\mathcal{P}\left(z^t \middle\| x^t\right)$	the true posterior of the encoder
x_i^t	the input feature of node v_i at snapshot t of TVAE
$\dot{z}_i^t$	the latent variable of input x_i based on the encoder
$\bar{z}_i^t$	the latent embedding of x_i based on VAE encoder
W_e, b_e	the weight and bias of the encoder of VAE
ϵ_i	the random variable from a normal distribution
μ_i, σ_i	the normally distributed latent variables of $\bar{z}_i^t$
e_{ij}^{kt}, α_{ij}^{kt}	the k-th (normalized) attention coefficient of $(v_i, v_j) \in E^t$
a_k	the weight vector of the k-th head
$\hat{z}_i^t$	the latent variable of v_i at snapshot t of self-attention
W_a^k	the k-th shared weight parameters of self-attention
K	the number of multi-heads in the attention part
o_i^t	the output gate vector of node v_i at snapshot t in LSTM
W_o,b_o	the weight and bias of output gate in LSTM
u_i^t	the update gate value of node v_i at snapshot t in LSTM
W_u, b_u	the weight and bias of update gate in LSTM
f_i^t	the forget gate value of node v_i at snapshot t in LSTM
W_f, b_f	the weight and bias of forget gate in LSTM
$\widetilde{C}_i^t$	the new cell state of node v_i at snapshot t in LSTM
W_C, b_C	the weight and bias of new cell state in LSTM
C_i^t	the cell state of node v_i at snapshot t in LSTM
$x^\intercal$	the transpose of x
act_	the activation functions of the model

whether they reflect the intrinsic to the network itself [52]. Dunlavy *et al.* [53] introduced a weight-based algorithm for collapsing multiyear data into a single matrix. They extended the Katz method to bipartite graphs and used truncated singular value decomposition. E-LSTM-D [54] utilized LSTM layers to filtrate the embedding in latent space which was aggregated by GCN, although it was similar to the prediction part of our model, it could not deal with a slightly larger network. Hisano [55] employed both supervised and unsupervised losses to learn the embedding that reflects information from both the temporal and cross-sectional network structures. Zhu *et al.* [56] proposed a temporal latent space model, which preferred smoothly evolving by penalizing frequent changes in latent positions for each node. The model assumes that two nodes will establish links in the future if they have propinquity in the latent space. These methods are somewhat limited because they require more information, such as inherent topology structure or attributes.

III. PRELIMINARIES AND PROBLEM DEFINITION

In general, we represent a network as $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices representing the nodes in the network and E is a set of relationships between nodes, and n is the number of nodes in the network. Network embeddings are a set of low-dimensional and appropriate vectors for the nodes that can preserve the essential feature and properties of the original network. To get straight to the point, we discuss the temporal network, its embedding, and link predication in Section III-A. In Table I, we summarize the terms and notations in this article.

A. Temporal Network

A temporal network is usually considered a series of snapshots in a dynamic environment with time information. We denote it as $\mathcal{G} = (G^1, G^2, \ldots, G^T)$, where T is the number of snapshots. In general, we assume that the network is undirected and unweighted, the number of nodes in the network is constant n, and the links are varying over time (but our method can handle both directed and undirected networks, as well as adding and removing nodes). We denote the adjacency matrix of network $\mathcal{G}$ as $\mathcal{A} = (A^1, A^2, \ldots, A^T)$, and the edge set across the snapshots as $\mathcal{E} = (E^1, E^2, \ldots, E^T)$. $A_{ij}^t = 1$ means that nodes v_i and v_j have an edge at the snapshot t, otherwise $A_{ij}^t = 0$.

B. Temporal Network Embedding

Traditionally, network embedding is contraposing a single-autocephalous network. In a temporal scenario, we define this problem as follows. Given the sequence of snapshots $\mathcal{G}$ in the form of adjacency matrix sequence $A = (A^1, A^2, \ldots, A^T)$, we need to obtain the representations of every node v_i for each snapshot t in a low-dimensional vector space (d-dimensional) as $z_i^1, z_i^2, \ldots, z_i^T$, where z_i^t is the embedding of node v_i at time t. We also denote $Z^t \in R^{n \times d}$ as the embedding matrix of nodes in the snapshot t. The TNE is reflected by a mapping function: $Z^t = f(A^{\leq t})$, where $A^{\leq t}$ denotes some adjacency matrices at time less than t, such that Z^t is able to capture the temporal evolution of the network and be used to predict Z^{t+1}. More specifically, we wish to predict the structure of A^{t+1} based on the historical structure $A^{\leq t}$. In summary, the mapping function at every time step employs network evolution information to explore the network temporal development mechanism.

We need to restate, for the TNE, the design of f, the temporal prediction, and the optimization correspond to the previous three challenges.

C. Temporal Network Embedding for Link Prediction

For a temporal network, link prediction is aiming to predict the links of the network snapshot t based on the observed structure of snapshots 1 to $t-1$. Therefore, in this article, we consider training a model that receives a series of previous network snapshots $(A^1, \ldots, A^{t-1})$ as input and generates the network structure in the next snapshots $(A^2, \ldots, A^t)$ as output. It allows the model to predict the structure of the next snapshot because the latent embedding vectors can capture the temporal evolution of the network. For the temporal link prediction, we take the $A^{\leq t}$ as input and predict the links of snapshot $t+1$.

IV. PROPOSED MODEL

In this section, we introduce the proposed TNE framework TVAE in detail, which can effectively obtain the befitting representations of the temporal network. Moreover, it can preserve structural properties, analyze dynamic evolution, and predict future links of the network.

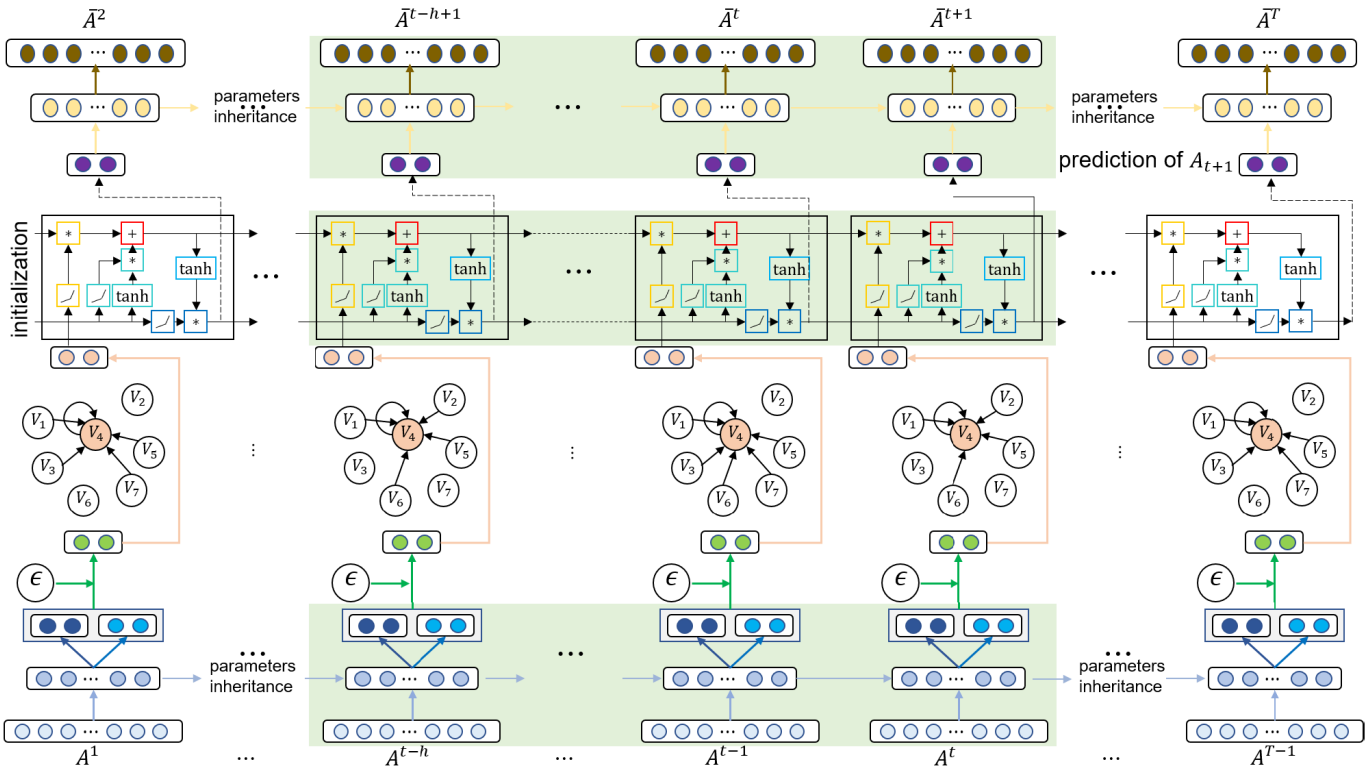


Fig. 2. Architecture of TVAE: a TNE model. At each snapshot t, it includes a VAE projecting each node into a latent embedding vector, a self-attention updating its embedding, an LSTM with h historical snapshots describing the dependencies, and a decoder generating the topological structure of snapshot $t+1$.

As shown in Fig. 2, the components of our model consist of four parts: 1) VAE, for each snapshot t, we take a conventional VAE with parameter-sharing as the encoder to obtain the initial node embeddings. 2) A self-attention mechanism to enhance the embeddings and maintain the diversity of each node. 3) An effective LSTM layer to model the dynamic evolution in the latent space and generate predictive embeddings. 4) Parameter inheritance to speed up the training of the model and deal with larger scale networks.

In the rest of this section, we will introduce our model particularly to demonstrate the feasibility of the proposed model.

A. VAE for Generating Embedding

To capture the network structure in snapshot t, according to the VAE framework, we take the adjacency matrix A^t as the input data and formalize it as X^t. We represent the part of input corresponding to a node as x^t. Use z^t (or Z^t) to denote the latent variable of x^t (or X^t). In this framework, we take $\mathcal{P}_\phi(x^t|z^t)$ to model the dependence between the latent variables and input data. The overall likelihood of X^t can be defined as follows:

$$\mathcal{P}(X^t) = \prod_{x^t} \mathcal{P}(x^t) = \prod_{x^t} \int \mathcal{P}_\phi(x^t|z^t)\mathcal{P}_\theta(z^t)dz^t \tag{1}$$

where ϕ and θ are the parameters of distributions $\mathcal{P}_\phi(x^t|z^t)$ and $\mathcal{P}_\theta(z^t)$. z^t is assumed to be distributed according to a standard normal distribution. The key intuition is that any complex distribution can be modeled as several arbitrary Gaussian distributions stacked. Thus, $\mathcal{P}_\phi(x^t|z^t)$ is parameterized by the function $f_\phi(z^t)$, which represents arbitrary unknown transformation expressing such a dependence. However, $\mathcal{P}(X^t)$ is usually too tanglesome to be computed directly. Thus, we also introduce variational inference to find an approximation. The theory devises a proposal distribution $\mathcal{Q}(z^t|x^t)$, which approximates the true posterior $\mathcal{P}(z^t|x^t)$, to handle this approximation. Therefore, we can obtain the relationship between the likelihood and the proposal distribution in (1)

$$\begin{aligned}&\log \mathcal{P}(x^t) - \mathrm{KL}\big[\mathcal{Q}(z^t|x^t) \parallel \mathcal{P}(z^t|x^t)\big] \\ &\quad = E_{z^t\sim\mathcal{Q}}\big[\log \mathcal{P}(x^t|z^t)\big] - \mathrm{KL}\big[\mathcal{Q}(z^t|x^t) \parallel \mathcal{P}(z^t)\big].\end{aligned} \tag{2}$$

For simplification, we introduce the ELBO L_b; therefore, the objective function can be represented as

$$L_b = \int_{z^t} \mathcal{Q}(z^t|x^t) \log \mathcal{P}(x^t|z^t)dz^t - \mathrm{KL}\big(\mathcal{Q}(z^t|x^t) \parallel \mathcal{P}(z^t)\big). \tag{3}$$

The two terms in the equation represent the reconstruction error and KL-divergence loss. From the perspective of neural networks, e.g., autoencoder, the modeling process can resort to two steps. The first is the encoder, which converts the input original network adjacencies into latent variables. The second is the decoder, which generates new representations of the network, which is referred to as the reconstruction process.

We take a two-layer encoder; however, we set this process for simplicity as follows:

$$\dot{z}_i^t = \mathrm{act}_e\big(W_e x_i^t + b_e\big) \tag{4}$$

where x_i^t is the transposed row of X^t corresponding to node v_i. act_e is the activation function, such as *ReLU*, W_e, and b_e are the weights and biases of the encoder network, respectively. Especially, $W_e \in \mathbb{R}^{d\times n}$ and $b_e \in \mathbb{R}^d$.

In an actual application scene, the use of reparametrization tricks [57] could strip the sampling process from stochastic gradient descent. To be specific, we sample a ministrant noise variable ϵ according to a random fixed distribution $\mathcal{P}(\epsilon)$, and then compute the new means and variances by ϵ. Typically, we sample Gaussian noise $\epsilon \sim \mathcal{N}(0,1)$ and compute the output of this encoder $\bar{Z}_i^t$ as the raw embedding

$$\bar{z}_i^t = \mu_i + \sigma_i \odot \epsilon_i \tag{5}$$

where μ_i and σ_i are the deriving parameters of node v_i, which are transformed from $\dot{z}_i^t$ via another two single layers, and $\odot$ is the elementwise product. Based on (3), we get the following objective:

$$\mathcal{L}_1 = \sum_{ij} \frac{1}{2}\Big(-\log\big(\sigma_{ij}\big)^2 + \big(\mu_{ij}\big)^2 + \big(\sigma_{ij}\big)^2 - 1\Big) \tag{6}$$

where μ_{ij} and σ_{ij} are the jth elements of mean and variance parameter vectors of node v_i, respectively.

Based on $\bar{z}_i^t$, we then propose a self-attention mechanism combined with LSTM to generate the final embedding z_i^t or Z^t containing the signatures of snapshots from $t-h$ to t and take a decoder network with two layers to reconstruct the next step structure A^{t+1} of the temporal network. In this way, we could achieve embeddings with good predictive ability.

B. Self-Attention on Topology Structure

The input of this self-attention layer is a raw embedding vector $\bar{z}_i^t$ generated by the encoder part. We devise structure self-attention layer to make it concern about the immediate neighbors of a node v_i [32]. The key procedure is computing attention coefficient from neighbors of each node in network. Then we filter the raw embeddings with attention layers and generate new embeddings with an aggregate function. To make our learning process of self-attention stable, we introduce the multihead attention mechanism. Then the operation of topology structural attention layer can be defined as follows:

$$e_{ij}^{kt} = \text{act}_s\big(a_k^{\mathsf{T}}\big[W_a^k \bar{z}_i^t \circ W_a^k \bar{z}_j^t\big]\big), \quad \forall\big(v_i, v_j\big) \in E^t \tag{7}$$

$$\alpha_{ij}^{kt} = \frac{\exp\Big(e_{ij}^{kt}\Big)}{\sum_{v_l \in N_i^t} \exp\big(e_{il}^{kt}\big)}, \quad \forall v_j \in N_i^t \tag{8}$$

$$\hat{z}_i^t = \text{act}_a\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N_i} \alpha_{ij}^{kt} W_a^k \bar{z}_i^t\right) \tag{9}$$

where $N_i^t = \{v_j \in V | (v_i, v_j) \in E^t\}$ is the set of immediate neighbors of node v_i in G_t, $W_a^k \in \mathbb{R}^{d\times d}$ is a shared weight matrix of the kth attention head, $a_k \in \mathbb{R}^{2d}$ is a weight vector of the kth head to parameterize the attention function which is implemented as a feed-forward layer, $\circ$ is concatenation operation, and $\text{act}_s(\cdot)$ and $\text{act}_a(\cdot)$ are nonlinear activation functions. α_{ij}^{kt} is the computed attention coefficient for edge $(v_i, v_j) \in E^t$ by the kth head. K is the number of attention heads and each head executes independent attention mechanisms. Inspired by [42], we utilize *LeakyReLU* and *ReLU* activation functions to normalized attention weight and output, respectively. Hence, we apply a structural attention layer on the original embedding and obtain a new embedding vector, which includes immediate neighbor information. We call it spatial dependence.

C. LSTM for Modeling Temporal Information

To capture abundant temporal evolutionary patterns in the temporal networks, we consider a recurrent neural network (RNN) to preserve historical information. LSTM is a preeminent RNN model of handling long-term dependence problems. In the temporal scene, take $\hat{z}_i^t$ in Section IV-B as input, we import the hidden state representation of a single-LSTM layer to fuse the information of previous continuous h snapshots in final embedding z_i^t. The updating rules of LSTM in our model are listed as follows:

$$o_i^t = \text{act}_l\Big(W_o\Big[\hat{z}_i^t \circ z_i^{(t-1)}\Big] + b_o\Big) \tag{10}$$

$$u_i^t = \text{act}_l\Big(W_u\Big[\hat{z}_i^t \circ z_i^{(t-1)}\Big] + b_u\Big) \tag{11}$$

$$f_i^t = \text{act}_l\Big(W_f\Big[\hat{z}_i^t \circ z_i^{(t-1)}\Big] + b_f\Big) \tag{12}$$

$$\widetilde{C}_i^t = \tanh\Big(W_c\Big[\hat{z}_i^t \circ z_i^{(t-1)}\Big] + b_c\Big) \tag{13}$$

$$C_i^t = f_i^t \odot C_i^{t-1} + u_i^t \odot \widetilde{C}_i^t \tag{14}$$

$$z_i^t = o_i^t \odot \tanh\big(C_i^t\big) \tag{15}$$

where C_i^t represents the cell states of LSTM layer, f_i^t is the forget gate vector, o_i^t is the output gate vector, u_i^t is update gate vector, weight matrices W_f, W_o, W_f, and W_c and biases b_f, b_o, b_f, and b_c are parameters for computing them. $\widetilde{C}_i^t$ is the new cell states, and the Sigmoid function is used for the activation function.

As of now, given the input sequences $(x_i^{t-h}, \ldots, x_i^t)$, we have got the embedding z_i^t which is used for reconstructing the adjacency relations $\bar{A}_i^{t+1}$ of v_i at time $t+1$. Then, we devise a fully connected decoder with two layers to generate reconstruction network with the hidden representation as follows:

$$\big(\bar{A}_i^{t+1}\big)^{\mathsf{T}} = \bar{x}_i^t = \text{act}_d\big(W_d z_i^t + b_d\big) \tag{16}$$

where act_d, W_d, and b_d are the activation function, the weights, and bias of the decoder part of our model, respectively. Written in matrix form, the reconstructed output is $\bar{A}^{t+1} = \bar{X}^t$.

D. Parameter Inheritance

A key intuition of maintaining the stability and low time cost of the TNE model is the parameter inheritance mechanism. We train our deep learning model on G_1 using random initialization of parameters. Then, for the remaining part of subsequent snapshots, we initialize the model parameters on time t by using the parameters on the previous time $t-1$ directly. Inspired by the time smoothness constraint, the model is only requested to learn the changes between two snapshots. We utilize specific parameter initialization procedures to endow our model more flexibility if the network snapshots at continuous time differ significantly, rather than appending the time

smoothness constraint to obliterate these changes. Although it is an intuitive and simple way, this parameter inheritance can achieve considerable computing speed in our model.

E. Loss and Optimization

From the overall framework, we have the loss function including two parts, one is $\mathcal{L}_1$ for VAE, and the other is reconstruction. Therefore, we provide an entirety loss function for the snapshot t as follows:

$$\mathcal{L}_t = \left\| \left(\bar{A}^{t+1} - A^t\right) \odot \mathcal{B} \right\|_F^2 + \alpha \sum_{ij} \frac{1}{2}\left(-\log\left(\sigma_{ij}^t\right)^2 + \left(\mu_{ij}^t\right)^2 + \left(\sigma_{ij}^t\right)^2 - 1 \right) \quad (17)$$

where $\mathcal{B}$ is a penalty matrix to impose more punishment to the reconstruction error of nonzero elements than the opposite one [37], such that $\mathcal{B}_{ij} = \beta > 1$ if $A_{ij}^t > 0$, else $\mathcal{B}_{ij} = 1$. μ and σ are the learned mean and variance of latent embedding. The first term is reconstruction loss which preserves the original network structure and temporal information. The second term is the KL-divergence loss, which makes it approximate to the real distribution as much as possible. α is a balance parameter (penalty factor), which controls the importance of VAE.

To be flexible and effective, we implement our model in Keras and use RMSProp for training. The RMSProp algorithm introduces the damping coefficient to increase the convergence rate. With the help of the proposed parameter inheritance mechanism, we devise an early stopping method to accelerate the training speed of (17).

V. Experiments

In this section, we employ real-world networks and synthetic data sets to verify the effectiveness of the model. In detail, we compare the performance of link prediction in dynamic networks with the baselines on network embedding, dynamic network embedding, and temporal link prediction. We also analyze the running time of representative methods and the ablation and parameter sensitively analysis of the TVAE.

A. Data Sets

1) Synthetic: We employ a stochastic block model (SBM) to generate a virtual network. We generate two synthetic data sets with 1000 and 5000 nodes in the network. We design two communities for one snapshot. The lock connectivity probability is set to 0.1 and 0.01, while the cross-block connectivity probability is 0.01 and 0.005, respectively. The probability difference between intrablock connectivity and cross-block connectivity controls the number of edges in the network. We migrate ten nodes to other communities randomly for each generation process. Then, we generate eight snapshots for this experiment.

2) Bitcoin OTC and Bitcoin Alpha [58]: Bitcoin over-the-counter (OTC) (BC-OTC)[1] and Bitcoin Alpha (BC-Alpha)[2] are data sets of bitcoin trades on different platforms. We build dynamic trust networks of bitcoin users with 12 snapshots.

[1]http://www.bitcoin-otc.com
[2]http://www.btc-alpha.com

TABLE II
Statistical Details of the Data Sets

Datasets	# node	# edge	# snapshot
SBM1000	1,000	56,000 $\sim$ 57,000	8
SBM5000	5,000	157,000 $\sim$ 158,000	8
Bitcoin OTC	5,881	35,500 $\sim$ 35,800	12
Bitcoin Alpha	3,777	24,100 $\sim$ 24,300	12
Hep-th	12,614 $\sim$ 14,446	37,930 $\sim$ 40,836	20
AS	7,716	8,931 $\sim$ 16,885	20

3) Hep-th [4]: This data set is the collaboration graph of authors in the High Energy Physics Theory conference. We get the abstracts of articles and then split data by month. Hence, every month's snapshot contains the coauthor network information.

4) Autonomous Systems [59]: The graph of routers comprising the Internet can be organized into subgraphs named autonomous systems (AS). We establish a communication network of who-talks-to-whom from the border gateway protocol logs.

The details of data sets are summarized in Table II. It contains the number of nodes, edges, and snapshots of each temporal network, respectively. The symbol $\sim$ indicates its value range of the number of nodes and edges. For the synthetic temporal networks, we construct two temporal networks of different sizes.

B. Baselines

We employ several static network embedding methods and TNE methods to compare with our model. For static network embedding methods, we provide access to the entire history of snapshots by constructing an aggregated graph up to time t.

1) **DeepWalk [13]:** A representative embedding method for static networks that performs random walk and skip-gram model.
2) **LINE [14]:** A network embedding method that considers first-order and second-order proximities and also designs an objective function that preserves both the local and global network structures.
3) **GAT-AE [60]:** It constructs an autoencoder network based on masked self-attentional layers for link prediction.

Furthermore, we compare several TNE methods with our model that operate on discrete snapshots.

1) **DynamicTriad [26]:** A dynamic graph embedding model that employs triadic closure theory on communication networks.
2) **DynAERNN [61]:** A deep neural network combined with autoencoders and RNN to handle temporal graph evolution.
3) **TNE [56]:** This is a dynamic network embedding method based on matrix factorization which presents a global optimization algorithm.
4) **DySAT [42]:** It employs both structure and temporal self-attention mechanisms to learn graph representations. We adopt a single-step link prediction experimental method and the node embedding of the last graph for evaluation.

TABLE III
AVERAGE LINK PREDICTION RESULTS OF TEMPORAL NETWORKS OF FOUR DATA SETS BASED ON THE MAP AND AUC, EACH VALUE IS THE AVERAGE OF ALL THE PREDICTED SNAPSHOTS

Performance	SBM(1000)		SBM(5000)		Hep-th		AS		BC-OTC		BC-Alpha	
	MAP	AUC	MAP	AUC	MAP	AUC	MAP	AUC	MAP	AUC	MAP	AUC
DeepWalk	0.5156	0.8728	0.5211	0.8728	0.3461	0.8432	0.1338	0.7994	0.0508	0.7067	0.2096	0.7417
LINE	0.4583	0.8375	0.5839	0.8845	0.3164	0.8379	0.1211	0.7604	0.0718	0.7501	0.2533	0.8163
GAT	0.6713	0.9223	0.6822	0.9125	0.4505	0.8729	0.1533	0.8055	0.0804	0.7197	0.2091	0.7507
DynamicTriad	0.4525	0.8583	0.4930	0.8770	0.4065	0.8353	0.1154	0.7634	0.2263	0.8184	0.4207	0.8441
DynAERNN	0.8593	0.9289	0.8492	0.9305	0.5614	0.8736	0.2696	0.8466	0.2394	0.8492	0.5260	0.9110
TNE	0.6405	0.8810	0.6930	0.9013	0.5673	0.8788	0.2065	0.7722	0.1825	0.8208	0.4807	0.8608
DySAT	0.8367	0.9174	0.8039	0.9315	0.5138	0.8741	0.1959	0.7822	0.2070	0.8461	0.5613	0.9284
Evolve-GCN	0.8659	0.9374	0.8125	0.9283	0.4822	0.8601	0.3102	0.8942	0.1418	0.7833	0.3106	0.8113
TVAE	**0.9553**	**0.9835**	**0.8763**	**0.9436**	**0.6636**	**0.9548**	**0.3207**	**0.8943**	**0.2773**	**0.8685**	**0.5779**	**0.9445**

5) **Evolve-GCN [58]:** A GCN for dynamic graphs, which uses RNN to regulate the GCN model at each time step. We consider two methods involved in the article: EvolveGCN-H and EvolveGCN-O and calculate the average result for evaluation.

For all these methods, we use the default parameter settings, as well as mean average precision (MAP) and area under the ROC curve (AUC) as our metrics. We train the models on networks from time $t-h$ to $t-1$, where h is the length of history and we predict links of the network at time t.

C. Overall Evaluation for Link Prediction

In this subsection, we use MAP and AUC to evaluate all methods on the aforementioned temporal networks. To be specific, we set the history value to 2, the value of the embedding dimension d to 128, and evaluate the models at all the snapshots. Then, we get results as shown in Table III. The link prediction results show that our model has significant performance advantages on both synthetic and real-world networks. The experimental results on different scale networks also prove the favorable extensibility of the proposed model.

First, all the methods for temporal networks have better performance on the link prediction than the static models. It shows that considering the varying patterns can help predict the structure. Second, in addition to our TVAE, DynAERNN and Evolve-GCN are the second-best methods among several candidates, the former learns the temporal transitions in the network with a deep neural network composed of recurrent layers, while the latter adopts the GCN along the temporal dimension; however, they all ignore the nonlinear evolution patterns. However, DynamicTriad and DySAT are slightly worse because they focus more on modeling dynamic networks. Third, although TNE is specifically designed for link prediction in temporal networks, it takes the nonnegative matrix factorization but ignores the nonlinear characteristics of the network. In addition, compared with the static methods, the methods used for dynamic networks achieve more than 30% and 3% gains of MAP and AUC metrics on average, respectively, and our method is superior to the best baseline on the whole.

D. Results on Synthetic Data Sets

The MAP and AUC values for different methods and multitime steps with SBM (1000) data set are shown in Fig. 3(a). Both MAP and AUC values display that our method TVAE has higher performance compared with the baselines, including three static and five dynamic embedding methods. For quantitative, our model TVAE has more than 10% precision gains, while 5% accuracy gains than the second-best method between these candidates on SBM (1000) data set. It demonstrates that our model can handle temporal networks with implicit feature sniffing. It is also significant that the curve tendency reveals the splendid stability of our model. To put it another way, TVAE is capable of excavating the essential evolution mechanism of temporal networks. It is relatively stable of temporal networks when there is over a short period of time. Therefore, the performance of the same model on continuous temporal network snapshots should show a glossy curve gradient on the line chart.

On the other hand, we also evaluate our model with baselines on the SBM (5000) data set to verify the scalability of these methods. Fig. 3(b) describes the performance across all concerned methods on a network scale of 5000 nodes. An obvious phenomenon is that large-scale information network embedding (LINE) tears the gap with other baselines due to its scalability on large-scale networks. TVAE still keeps nice precision and accuracy within the error range. One possible reason is that TVAE is based on a VAE model. The weight of KL-divergence in loss function leads to improving the performance of our model. By the way, TVAE displays excellent stability among the network snapshots yet. Compared with our TVAE, DynAERNN has achieved competitive results.

E. Results on Real Temporal Networks

In this section, we evaluate TVAE with all the baselines on four temporal networks on the link prediction task.

1) BC-OTC Data Set: The link prediction results at multitime steps on the BC-OTC data set are shown in Fig. 3(c). The MAP and AUC values of our TVAE are better than all the other baselines. The runner-ups on the MAP metric are DynAERNN, DynamicTriad, DySAT, and TNE, while DynAERNN and DySAT show closer results to ours on the AUC metric. The results of EvolveGCN are lower but still better than the static methods, including DeepWalk, LINE, and graph attention network (GAT).

2) BC-Alpha Data Set: Fig. 3(d) demonstrates the results of link prediction at multitime steps on the BC-Alpha data set. Our TVAE shows the best results at most snapshots and lower

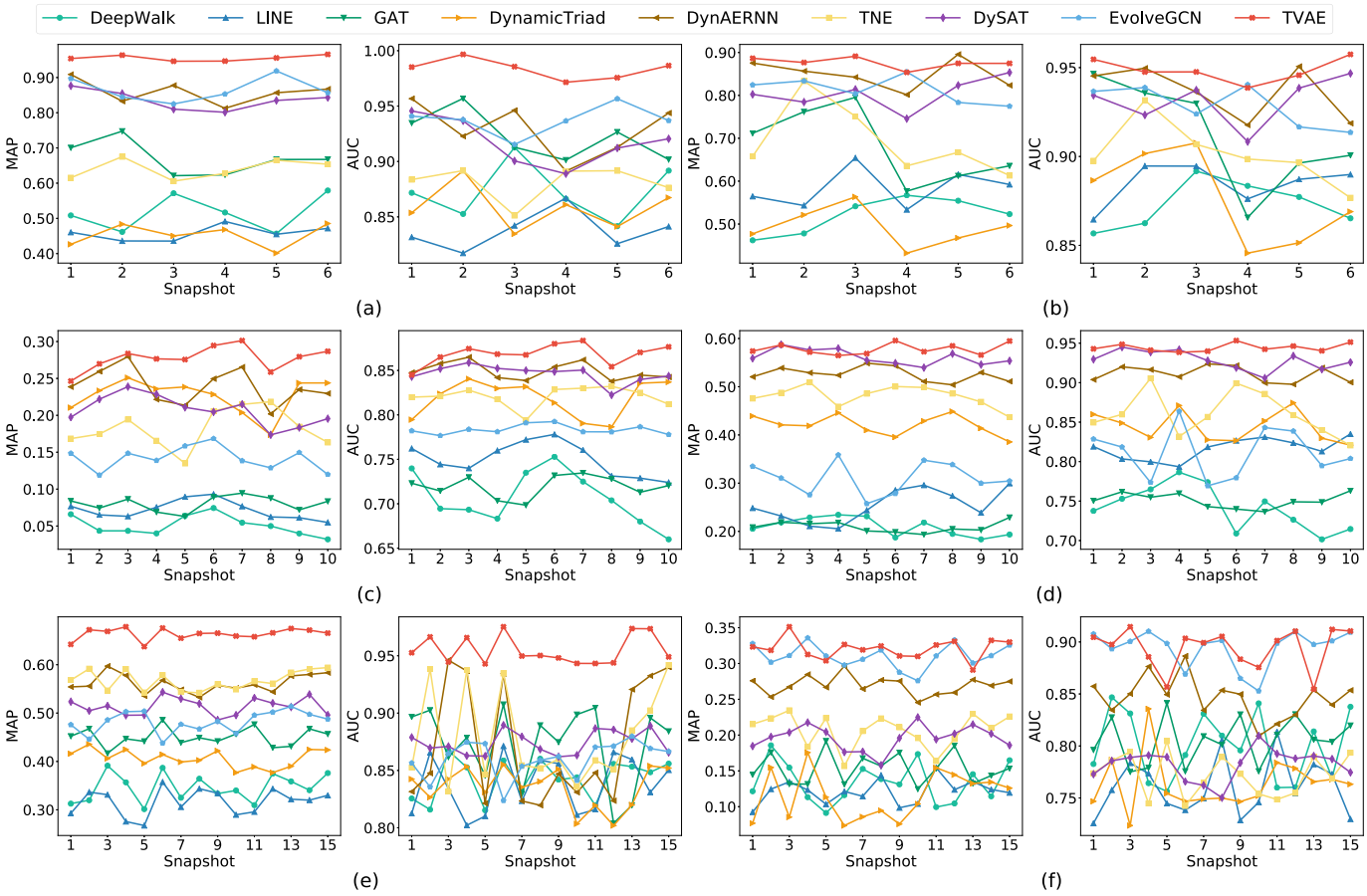


Fig. 3. Link prediction results based on the MAP and AUC values of our TVAE and the baseline methods on synthetic networks (a) SBM (1000). (b) SBM (5000). (c) BC-OTC. (d) BC-Alpha. (e) Hep-th. (h) AS.

results at a few snapshots than only DySAT. The ranks of the other baselines are basically like their results on BC-OTC.

3) Hep-th Data Set: The multitime steps link prediction results for the Hep-th data set are shown in Fig. 3(e) with MAP and AUC metrics. Our algorithm TVAE outperforms all other state-of-the-art models, whether static or TNE methods on a macroscopic promotion. As for MAP values, DynAERNN and TNE still have the second-best performance among these methods. Another TNE method DynamicTriad has a somewhat higher elevation than the previous manifestation. This algorithm is based on the closer triad theory, which may have more rationalization on coauthor network data sets. The triad closure process is able to describe the evolution of the relationship between social networks. DySAT has an acceptable result and GAT also has a better performance than the other static methods DeepWalk and LINE because different attention coefficients represent the distance among the global topology structure, which is the main governing factor of temporal network evolution.

4) AS Data Set: The MAP and AUC values for multitime steps link prediction with the results of various algorithms on the AS data set are displayed in Fig. 3(f). Although the multitime steps link prediction stability of all these candidates has decreased slightly, TVAE shows more smoothness on curve tendency than other baselines. Our model achieves consistent gains of 10%–20% in comparison to the state-of-the-art baselines. From the peaks and troughs of the curve in Fig. 3(f), we can discover that communication is more inconstant than social networks. In other words, traditional network embedding methods often utilize a single mechanism to generate the representations of networks which may lead to a lack of flexibility of models. As a result, we employ several constraints to improve our model and increase the accuracy of the link prediction task. It is worth mentioning that EvolveGCN has almost the same performance as our TVAE because its convolution neural network can capture the dynamic changes of the network.

As we have discussed earlier, for the four temporal networks, although one of the baselines has a comparable or competitive performance based on the MAP or AUC on a network, such as the DySAT on BC-Alpha, or the EvolveGCN on the AS network, our method has obvious advantages on the whole.

F. Ablation Analysis of TVAE

To verify the motivation of our proposed TVAE, we further conduct ablation experiments. We conduct a link prediction task at multitime steps on both a synthetic network SBM (1000) and a real-world network BC-OTC. Our TVAE is compared with three models degraded by removing the LSTM part (denoted as VAE+attention), the self-attention mechanism part (denoted as VAE+LSTM), or both of the two parts

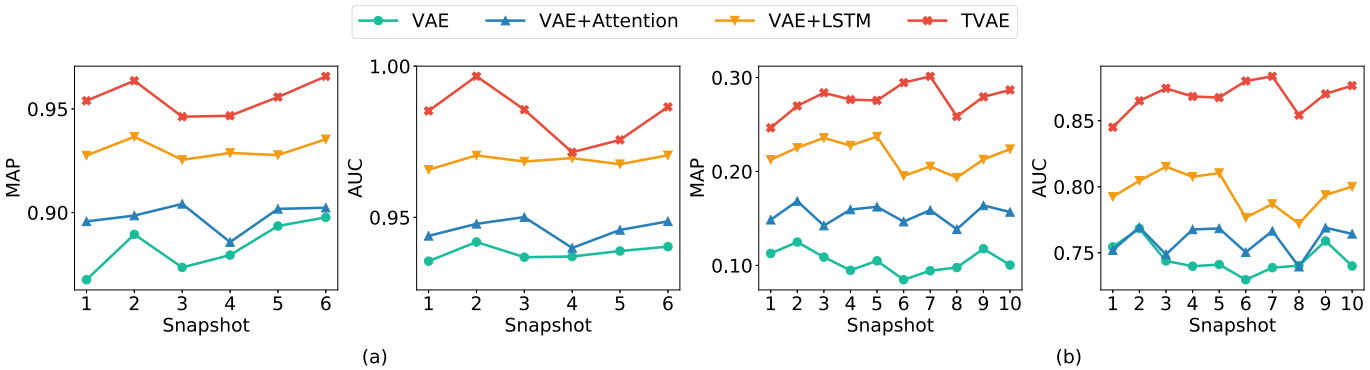


Fig. 4. Ablation experiments of our TVAE and its parts VAE, VAE+attention on two different networks (a) SBM(1000). (b) BC-OTC.

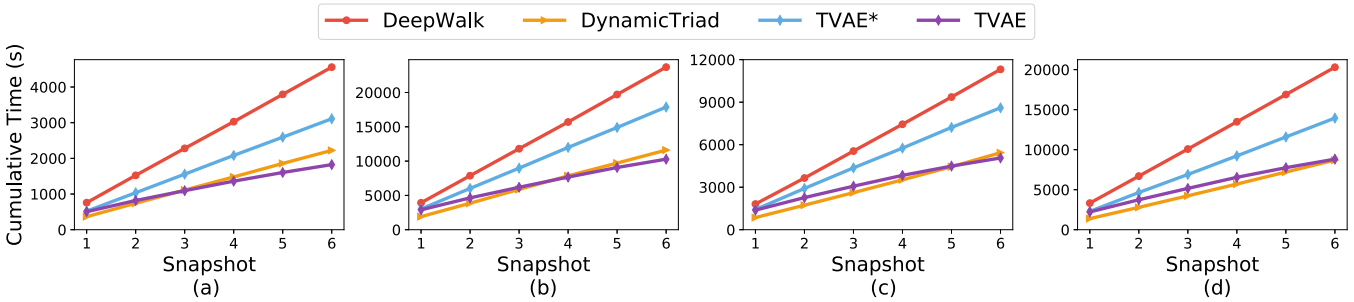


Fig. 5. Cumulative time cost of different algorithms. We compute the time cost of Deepwalk, DynamicTriad, and TVAE on different data sets. We choose six continuous snapshots for evaluation and obtain the cumulative time cost curves of different algorithms. TVAE* is our proposed model without applying the parameter inheritance mechanism. (a) SBM (1000). (b) SBM (5000). (c) Hep-th. (d) AS.

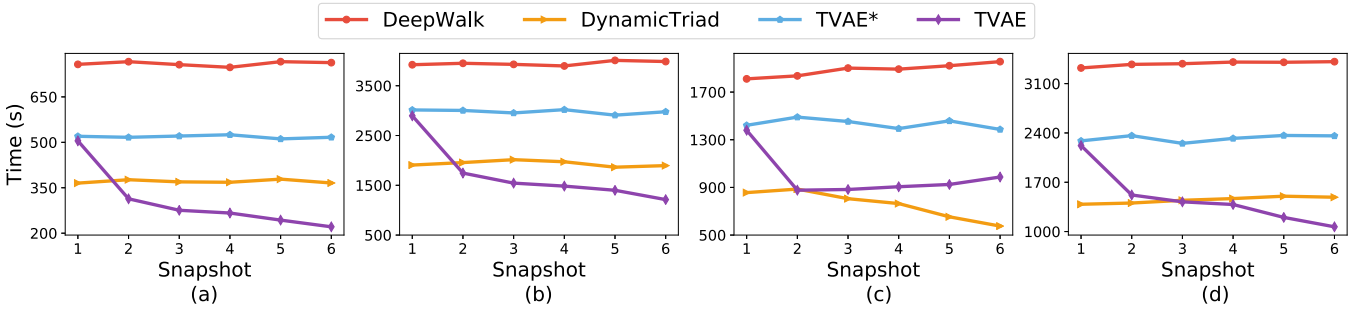


Fig. 6. Single-snapshot time cost of different algorithms. We compute the time cost of Deepwalk, DynamicTriad, and TVAE on different data sets. We choose six continuous snapshots for evaluation and obtain the time cost curves of every snapshot of different algorithms. TVAE* is our proposed model without applying the parameter inheritance mechanism. (a) SBM (1000). (b) SBM (5000). (c) Hep-th. (d) AS.

(denoted as VAE) from TVAE. The MAP and AUC values of these models are shown in Fig. 4. All the results support the following observations.

1) Compared with VAE, all other models achieve better results, indicating that it is feasible to use LSTM to model temporal dependencies and use the self-attention mechanism to capture structural information.
2) TVAE outperforms the other models, indicating that the self-attention mechanism and LSTM can jointly improve the information capture.
3) VAE+LSTM outperforms VAE+attention, which shows that the information of temporal dependencies has a greater impact on the results than static structures.

G. Time Cost

In this section, we discuss the time cost among our model and some baselines. Different from static network embedding methods, with the complex evolution and the serried data sets of temporal networks, the model is requested more time efficiency in the temporal scenario. As mentioned earlier, we design the experiment about embedding learning efficiency on two baselines, DeepWalk and DynamicTriad, which represent static embedding methods and temporal embedding methods, respectively. Besides, to analyze the effectiveness of the parameter inheritance mechanism, we calculate its time of each snapshot separately and denote it as the TVAE*. The experimental results of the time cost of our model and the mentioned baselines are shown in Figs. 5 and 6. We record ten times the training cost on the same snapshot with every algorithm and compute the average of them. Then, we conduct six continuous snapshots with the same procedure and generate the experimental result. All the methods run on a Dell workstation. The operating system is a 64 bit Ubuntu 16.04 with Intel Xeon CPU E5-2680 V3@2.5-GHz processor and 128 Gbytes of main memory.

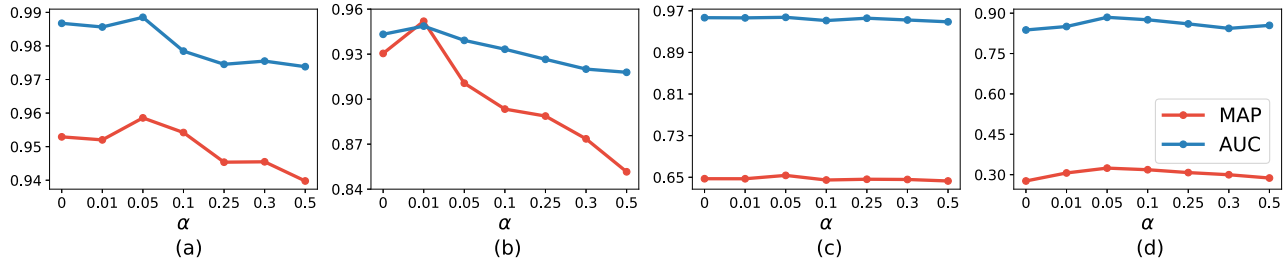


Fig. 7. Influence of balance factor α on MAP and AUC values. For every data set, we set the balance factor α from 0 to 0.5 to explore whether different α can control the precision of experimental results. (a) SBM (1000). (b) SBM (5000). (c) Hep-th. (d) AS.

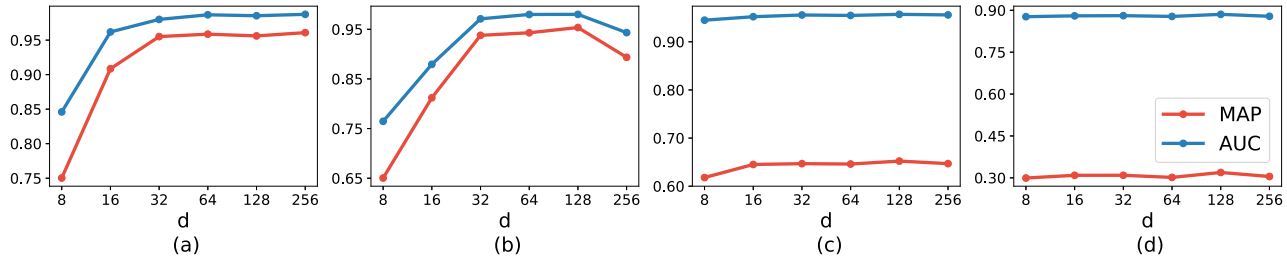


Fig. 8. Influence of embedding dimension for MAP and AUC values. For every data set, we set the number of embedding dimensions as 8, 16, 32, 64, 128, and 256, respectively, to explore the significance of hidden features learned by TVAE. (a) SBM (1000). (b) SBM (5000). (c) Hep-th. (d) AS.

From Fig. 5, we can discover the distinct superiority of TNE methods in multiple time steps data sets at a first impression. To be specific, we focus on the DeepWalk and DynamicTriad algorithms. The former oversteps about 2–2.5 times than the latter of time cost. On the one hand, the experimental result demonstrates the tremendous discrepancy between traditional static network embedding methods and TNE methods explicitly in evolutionary link prediction tasks. On the other hand, our model TVAE has a little bit of inefficiency compared with DynamicTriad at the first time step. This is because DynamicTriad utilizes all of the temporal edges to generate one embedding, while our model generates a series of embeddings. TVAE also has a lower time cost rate of increase at the rest of the time steps. The curve tendency shows that DynamicTriad is increasing as a linear growth process, while TVAE is likely a parabola. With the experimental results of a single-snapshot time cost in Fig. 6, we can see that as time goes by, our model will bring lower time costs. Compared with the TVAE*, the parameter inheritance mechanism of our model not only endows the model more stability but also extends the scalability when network snapshots are growing along with time.

H. Parameter Sensitivity Analysis

Here, we analyze the sensitivity of the balance factor α, the number of embedding dimensions, and historical snapshots on the performance of link prediction in the TVAE model.

1) Balance Factor α: In this section, we consider the parameter sensitivity of our model. The most important parameter of TVAE is the KL-divergence factor α. This balancing factor controls that the embeddings in latent space may represent different properties of original afferent networks. We employ α to constrain the randomness of the generative module. The parameter balance experimental result is shown in Fig. 7. And for every value of α, we conduct ten times experiments and compute the average for justification.

We discover that the MAP and AUC values are maximal above this curve when factor α is set to 0.01–0.1. And their tendencies are both descending whether factor α increases or decreases except for the Hep-th data set. From two points of view for this experimental result, we have different opinions on it. When we set factor α to 0, our VAE will cast off KL-divergence loss and kick over the traces. The model is too lazy to learn the exclusive Gaussian distribution and chooses the onefold means. In other words, the model will not have the flexibility to generate polyphyletic embedding vectors. When we heighten factor α from zero, we can discover the performance of the model for MAP and AUC values rising to some extent at first. However, when the curve crosses the wave crest, it is difficult for the model to learn plenty of varied Gaussian distribution for every representation of the original network in the latent space. We adjust the factor α to make TVAE nimble and be capable of miscellaneous data sets. As for the Hep-th data set, we consider that reconstruction and KL loss happen to keep a balance which results in the robustness of experimental results.

2) Embedding Dimension: We compare the performance of TVAE at different embedding dimensions. The experimental result is shown in Fig. 8. Be consistent with our intuition, the better precision and accuracy performance is often appearing at higher embedding dimensions. This is because a lower dimensional vector can have more information loss than a higher one. From Fig. 8, we can see that AS data set displays a little bit smoother than other data sets. This is because AS is a communication network and has better compression radio at different embedding dimensions. In the meantime, our model TVAE shows excellent performance since the 32 embedding size. This phenomenon demonstrates that TVAE has been captured the essential features of the original network so that it can predict unknown networks commendably with highly compressed embeddings.

3) Number of Historical Snapshots: To analyze the dependence of the embedding on historical snapshots, we design this

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

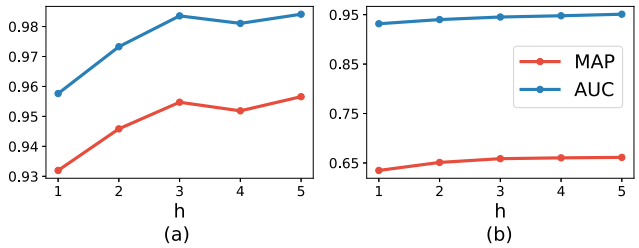


Fig. 9. Influence of history based on MAP and AUC values on two different networks, and we set the number of history snapshots from 1 to 6. (a) SBM (1000). (b) Hep-th.

experiment on the SBM (1000) and Hep-th dynamic networks. In detail, we set h, i.e., the number of historical snapshots Z_t depends on, from 1 to 6, we train the model and compute the MAP and AUC on the last snapshot T. From the results shown in Fig. 9, it is easy to know that h is important for the temporal link prediction, as it increases, the result tends to be better and stable. However, the larger h means more parameters and calculation time. In other experiments, we all set $h = 2$ as a compromise, and it also shows that dependence is very important in dynamic networks and TVAE can capture it.

VI. Conclusion and Future Work

The complex network has a lot of evolution tendencies in the real world. It can be hardly described by the static network embedding method because of various changes and amount of time costs. In this article, we propose TVAE to capture the evolution of the network topology with the representation in latent space. The model learns not only the low-dimensional embedding vectors and nonlinearity but the time dependencies between continuous network snapshots. To be specific, we design a deep autoencoder framework as a basis. Then, we impose the structure and temporal restriction with a self-attention mechanism and LSTM, respectively. We also utilize the parameter inheritance technique to keep embedding stable and scalable. To verify the effectiveness of our model, we conduct several experiments on both artificial and real data sets. The experimental results show that our model is capable to exploit the temporal network patterns and outperforms other baselines on link prediction precision and accuracy, and has a lower time cost.

In the future, we have several research directions on this basis: 1) in temporal networks, there are many important tasks, e.g., node classification and community detection. We will assess the model on these metrics. 2) We take two synthetic and two real data sets for experiments. We will attempt more kinds of data sets, and simultaneously increase the effectiveness and efficiency of large-scale data sets ulteriorly. 3) We have found some effects of different hyperparameters in the experiment. We will explore more essential reasons for the impact of hyperparameters.

References

[1] A. Theocharidis, S. Van Dongen, A. J. Enright, and T. C. Freeman, "Network visualization and analysis of gene expression data using biolayout express 3D," *Nature Protocols*, vol. 4, no. 10, p. 1535, 2009.

[2] L. C. Freeman, "Visualizing social networks," *J. Social Struct.*, vol. 1, no. 1, p. 4, 2000.

[3] I. Herman, G. Melancon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," *IEEE Trans. Vis. Comput. Graphics*, vol. 6, no. 1, pp. 24–43, Jan. 2000.

[4] J. Gehrke, P. Ginsparg, and J. Kleinberg, "Overview of the 2003 KDD cup," *ACM SIGKDD Explor. Newslett.*, vol. 5, no. 2, pp. 149–151, Dec. 2003.

[5] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, p. 52—74, Sep. 2017.

[6] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, May 2019.

[7] C. C. Aggarwal and N. Li, "On node classification in dynamic content-based networks," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2011, pp. 355–366.

[8] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community evolution in dynamic multi-mode networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, pp. 677–685, 2008.

[9] S. Cavallari, E. Cambria, H. Cai, K. C.-C. Chang, and V. W. Zheng, "Embedding both finite and infinite communities on graphs [application notes]," *IEEE Comput. Intell. Mag.*, vol. 14, no. 3, pp. 39–50, Aug. 2019.

[10] N. Barbieri, F. Bonchi, and G. Manco, "Who to follow and why: Link prediction with explanations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 1266–1275.

[11] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 26, 2021, doi: 10.1109/TNNLS.2021.3070843.

[12] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 855–864.

[13] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 701–710.

[14] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[15] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 891–900.

[16] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1105–1114.

[17] C. Zheng, L. Pan, and P. Wu, "Multimodal deep network embedding with integrated structure and attribute information," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 5, pp. 1437–1449, May 2020.

[18] M. Shi, Y. Tang, and X. Zhu, "MLNE: Multi-label network embedding," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3682–3695, Sep. 2020.

[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[20] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 3–28, Mar. 2020.

[21] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 17, 2020, doi: 10.1109/TKDE.2020.2981333.

[22] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1407–1418, May 2019.

[23] S. Mehran Kazemi *et al.*, "Representation learning for dynamic graphs: A survey," 2019, *arXiv:1905.11485*. [Online]. Available: http://arxiv.org/abs/1905.11485

[24] S. M. Kazemi *et al.*, "Representation learning for dynamic graphs: A survey," *J. Mach. Learn. Res.*, vol. 21, no. 70, pp. 1–73, 2020.

[25] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[26] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 1–8.

[27] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 387–396.

[28] L. Du, Y. Wang, G. Song, Z. Lu, and J. Wang, "Dynamic network embedding : An extended approach for skip-gram based network embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2086–2092.

[29] P. Goyal, N. Kamra, X. He, and Y. Liu, "DynGEM: Deep embedding method for dynamic graphs," 2018, *arXiv:1805.11273*. [Online]. Available: https://arxiv.org/abs/1805.11273

[30] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS Workshop Bayesian Deep Learn.*, 2016.

[31] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed, and E. Koh, "Attention models in graphs: A survey," *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 6, pp. 1–25, Nov. 2019.

[32] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[33] G. Zhu *et al.*, "Redundancy and attention in convolutional LSTM for gesture recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1323–1335, Apr. 2020.

[34] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 203–209.

[35] H.-N. Tran and E. Cambria, "A survey of graph processing on graphics processing units," *J. Supercomput.*, vol. 74, no. 5, pp. 2086–2115, May 2018.

[36] D. Jin, C. Huo, C. Liang, and L. Yang, "Heterogeneous graph neural network via attribute completion," in *Proc. 30th Int. Conf. World Wide Web (WWW)*, ACM, 2021.

[37] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1225–1234.

[38] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[39] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, p. 2, Mar. 2007.

[40] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," in *Proc. 34th Int. Conf. Mach. Learn. (JMLR)*, vol. 70, 2017, pp. 3462–3471.

[41] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "NetWalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2672–2681.

[42] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "DySAT: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proc. 13th Int. Conf. Web Search Data Mining*, Jan. 2020, pp. 519–527.

[43] A. Maheshwari, A. Goyal, M. K. Hanawal, and G. Ramakrishnan, "DynGAN: Generative adversarial networks for dynamic network embedding," in *Proc. Graph Represent. Learn. Workshop (NeurIPS)*, 2019, p. 1.

[44] D. Xu, W. Cheng, D. Luo, X. Liu, and X. Zhang, "Spatio-temporal attentive RNN for node classification in temporal attributed graphs," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 3947–3953.

[45] D. Xu, J. Liang, W. Cheng, H. Wei, H. Chen, and X. Zhang, "Transformer-style relational reasoning with dynamic memory updating for temporal network modeling," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 5, 2021, pp. 4546–4554.

[46] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Proc. Companion Web Conf.*, 2048, pp. 969–976.

[47] Y. Lu, X. Wang, C. Shi, P. S. Yu, and Y. Ye, "Temporal network embedding with micro- and macro-dynamics," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 469–478.

[48] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1269–1278.

[49] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–19.

[50] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[51] X. Chang *et al.*, "Continuous-time dynamic graph learning via neural interaction processes," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 145–154.

[52] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 58, no. 7, pp. 1019–1031, 2007.

[53] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Trans. Knowl. Discovery Data*, vol. 5, no. 2, pp. 1–27, Feb. 2011.

[54] J. Chen, X. Xu, Y. Wu, and H. Zheng, "GC-LSTM: Graph convolution embedded lstm for dynamic link prediction," 2018, *arXiv:1812.04206*. [Online]. Available: https://arxiv.org/abs/1812.04206

[55] R. Hisano, "Semi-supervised graph embedding approach to dynamic link prediction," in *Proc. Int. Workshop Complex Netw.* Boston, MA, USA: Springer, 2018, pp. 109–121.

[56] L. Zhu, D. Guo, J. Yin, G. V. Steeg, and A. Galstyan, "Scalable temporal latent space inference for link prediction in dynamic social networks," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2765–2777, Oct. 2016.

[57] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Represent.*, 2014.

[58] A. Pareja *et al.*, "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 5363–5370.

[59] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2005, pp. 177–187.

[60] Z. Marinka, A. Monica, and L. Jure, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[61] P. Goyal, S. R. Chhetri, and A. Canedo, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowl.-Based Syst.*, vol. 187, Jan. 2020, Art. no. 104816.

Pengfei Jiao received the Ph.D. degree in computer science from Tianjin University, Tianjin, China, in 2018.

He is currently a Lecturer with the Center of Biosafety Research and Strategy, Tianjin University. He is currently working on temporal community detection, link prediction, network embedding, recommender systems, and applications of the statistical network model. He has authored or coauthored more than 50 international journals and conference papers. His current research interests include complex network analysis, data mining, and graph neural networks.

Xuan Guo received the bachelor's degree in computer science from Tianjin University, Tianjin, China, in 2018, where he is currently pursuing the master's degree with the School of Computer Science and Technology.

His current research interests include complex network analysis and network representation learning.

Xin Jing received the bachelor's degree from Tianjin University, Tianjin, China, in 2018, where he is currently pursuing the master's degree with the College of Intelligence and Computing.

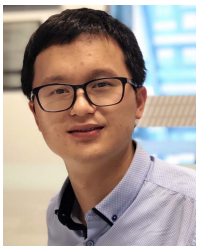
His current research interests include complex network analysis, temporal network embedding, and community detection.

Dongxiao He received the Ph.D. degree in computer science from Jilin University, Changchun, China, in 2014.

She is currently an Associate Professor with the College of Intelligence and Computing, Tianjin University, Tianjin, China. She has authored or coauthored more than 50 international journals and conference papers. Her current research interests include data mining and analysis of complex networks.

Huaming Wu (Member, IEEE) received the B.E. and M.S. degrees in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2009 and 2011, respectively, and the Ph.D. degree (Hons.) in computer science from the Freie Universität Berlin, Berlin, Germany, in 2015.

He is currently an Associate Professor with the Center for Applied Mathematics, Tianjin University, Tianjin, China. His research interests include model-based evaluation, wireless networks, edge/cloud computing, deep learning, and complex networks.



Shirui Pan (Member, IEEE) received the Ph.D. degree in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia, in 2015.

He was a Lecturer with the School of Software, UTS. He is currently a Senior Lecturer with the Faculty of Information Technology, Monash University, Clayton, VIC, Australia. He has authored or coauthored over 60 research articles in top-tier journals and conferences, including the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE International Conference on Data Engineering, the AAAI Conference on Artificial Intelligence, the International Joint Conferences on Artificial Intelligence, and the IEEE International Conference on Data Mining. His research interests include data mining and machine learning.

Maoguo Gong (Senior Member, IEEE) received the B.S. degree (Hons.) in electronic engineering and the Ph.D. degree in electronic science and technology from Xidian University, Xi'an, China, in 2003 and 2009, respectively.

Since 2006, he has been a Teacher with Xidian University, where he was promoted as an Associate Professor and a Full Professor in 2008 and 2010, respectively, with exceptive admission. His research interests are in the areas of computational intelligence with applications to optimization, learning, data mining, and image understanding.

Dr. Gong received the Prestigious National Program for the support of Top-Notch Young Professionals from the Central Organization Department of China, the Excellent Young Scientist Foundation from the National Natural Science Foundation of China, and the New Century Excellent Talent in University from the Ministry of Education of China. He is also an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Wenjun Wang is currently a Professor with the School of College of Intelligence and Computing, Tianjin University, Tianjin, China, the Chief Expert of major projects of the National Social Science Foundation, the Big Data Specially-Invited Expert of the Tianjin Public Security Bureau, Tianjin, and the Director of the Tianjin Engineering Research Center of Big Data on Public Security, Tianjin. His research interests include computational social science, large-scale data mining, intelligence analysis, and multilayer complex network modeling. He was the Principal Investigator or was responsible for more than 50 research projects, including the Major Project of the National Social Science Fund, the Major Research Plan of the National Natural Science Foundation, and the National Science-Technology Support Plan Project of China. He has authored or coauthored more than 50 papers on main international journals and conferences.