Displays 68 (2021) 102023

Contents lists available at ScienceDirect

Displays

journal homepage: www.elsevier.com/locate/displa

T-GAN: A deep learning framework for prediction of temporal complex networks with adaptive graph convolution and attention mechanism☆

Ru Huang [a,*], Lei Ma [a], Jianhua He [b], Xiaoli Chu [c]

[a] School of Information Science & Engineering, East China University of Science and Technology, Meilong Road 130, Shanghai 200237, China
[b] School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK
[c] Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield S1 3JD, UK

ARTICLE INFO

Keywords:
Complex networks
Temporal graph embedding
Graph data mining
Graph neural network

ABSTRACT

Complex network is graph network with non-trivial topological features often occurring in real systems, such as video monitoring networks, social networks and sensor networks. While there is growing research study on complex networks, the main focus has been on the analysis and modeling of large networks with static topology. Predicting and control of temporal complex networks with evolving patterns are urgently needed but have been rarely studied. In view of the research gaps we are motivated to propose a novel end-to-end deep learning based network model, which is called temporal graph convolution and attention (T-GAN) for prediction of temporal complex networks. To joint extract both spatial and temporal features of complex networks, we design new adaptive graph convolution and integrate it with Long Short-Term Memory (LSTM) cells. An encoder-decoder framework is applied to achieve the objectives of predicting properties and trends of complex networks. And we proposed a dual attention block to improve the sensitivity of the model to different time slices. Our proposed T-GAN architecture is general and scalable, which can be used for a wide range of real applications. We demonstrate the applications of T-GAN to three prediction tasks for evolving complex networks, namely, node classification, feature forecasting and topology prediction over 6 open datasets. Our T-GAN based approach significantly outperforms the existing models, achieving improvement of more than 4.7% in recall and 25.1% in precision. Additional experiments are also conducted to show the generalization of the proposed model on learning the characteristic of time-series images. Extensive experiments demonstrate the effectiveness of T-GAN in learning spatial and temporal feature and predicting properties for complex networks.

1. Introduction

Complex network is a graph network with non-trivial topological features, which often occurs in real systems, such as Internet, social networks, sensor networks, transportation networks and utility networks. Study of complex networks is still a relatively new but active research area. In the early research on complex networks, the main focus is on the analysis and modeling the features of complex networks. More specifically, the small world [1] and scale-free [2] characteristics of complex networks have been deeply explored and proved theoretically. In addition, the research problems of community detection [3] and link prediction [4] are widely investigated for complex networks. With more extreme weathers and advanced large scale cyber-security attacks, the vulnerability of real complex networks has never been exposed as now and the cost of complex network breaking down could be very high. There is increasing demand on the prediction of complex network states and events, which can provide important support for intelligent decision making on the control and management of complex networks. However, prediction of complex networks is very challenging due to the large scale and complex interaction among the network nodes. The research problem of prediction of complex networks has been rarely studied in the literature.

Many real applications on complex network can be modeled as a dual problem of network topology prediction or network node feature forecasting. For example, as shown in Fig. 1, e-commerce recommendation can be modeled as a topology prediction problem in a user-commodity bipartite graph [5], in which the purchasing behaviors of consumers will be recorded at different time step (t_1 and t_2) that constitute the time

☆ This paper was recommended for publication by Prof G Guangtao Zhai.

* Corresponding author.
E-mail address: huangrabbit@ecust.edu.cn (R. Huang).

https://doi.org/10.1016/j.displa.2021.102023
Received 28 March 2021; Received in revised form 8 May 2021; Accepted 9 May 2021
Available online 13 May 2021
0141-9382/© 2021 Elsevier B.V. All rights reserved.

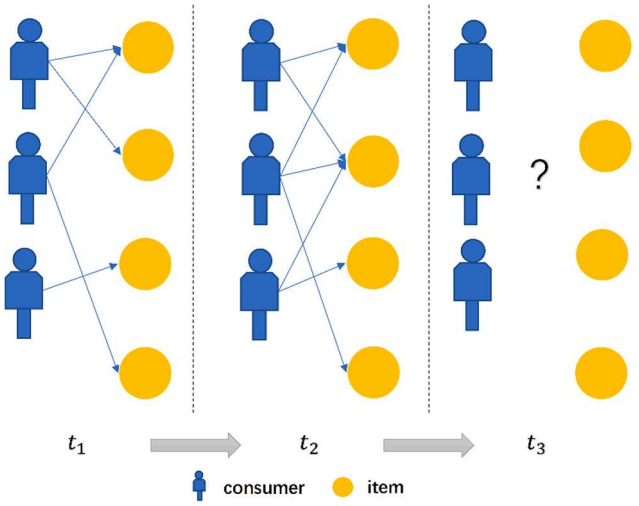

Fig. 1. An evolutionary pattern illustration of a consumer-item bipartite network.

sequence of the bipartite graph, and taking advantage of the historical information from the time sequence to make prediction for the future orders (connections between consumers and items at time step t_3) is the core objective in recommendation systems. Malicious node detection in wireless sensor networks (WSN) [6] can be applied to the network lifetime assessment for sensor networks [7], and risk forecasting for finance companies [8] can be considered as the node classification task in complex networks, as is shown in Fig. 2. In Fig. 2, there was only one malicious node in the network at time step t_1, with another 2 nodes infected soon at t_2. Hence, there is an urgent need to use the time series to study the evolution pattern of the infection; making prediction for the next time step t_3 is of vital importance.

Topology evolution in social networks can be formulated as the link prediction problem for undirected or directed weighted graphs, as shown in Fig. 3, where the original snapshot of the dynamic network is presented as time step t_1, the blue dotted line in the graph at $t_2$indicates that the link between node 1 and node 2 is broken, and the solid black line denotes that a new link presents between node 2 and node 5. The topology prediction task for this temporal network shown in Fig. 3 includes predictions for weights or links of each node pair of the graph at time step t_3

For the real complex networks, along with complex network topology, another prominent feature of temporal evolution makes the network prediction and control problems more challenging. The network topology and the properties of the networks nodes and links could all change significantly with time. In most of the existing research works on complex network, static complex networks have been studied, while temporally evolving complex networks received much less study.

In view of the above research gaps and challenges on the study of prediction of complex networks properties and events, we are motivated to propose a novel end-to-end deep learning based network model, which is called temporal graph convolution and attention (T-GAN) for predicting property and events of temporal complex networks. An encoder-decoder framework is applied to achieve the objectives. As long short-term memory (LSTM) has demonstrated the ability in extracting temporal features and has achieved huge successes in deep learning based computer vision and natural language processing, we apply it to our proposed model for joint extraction of both spatial and temporal features of complex networks. As GCN adopts symmetric Laplacian matrix as the graph shift operator in frequency domain, it has been proved that GCN has the property of low-pass filtering [9], which directly affects the depth of GCN based deep learning model [10].

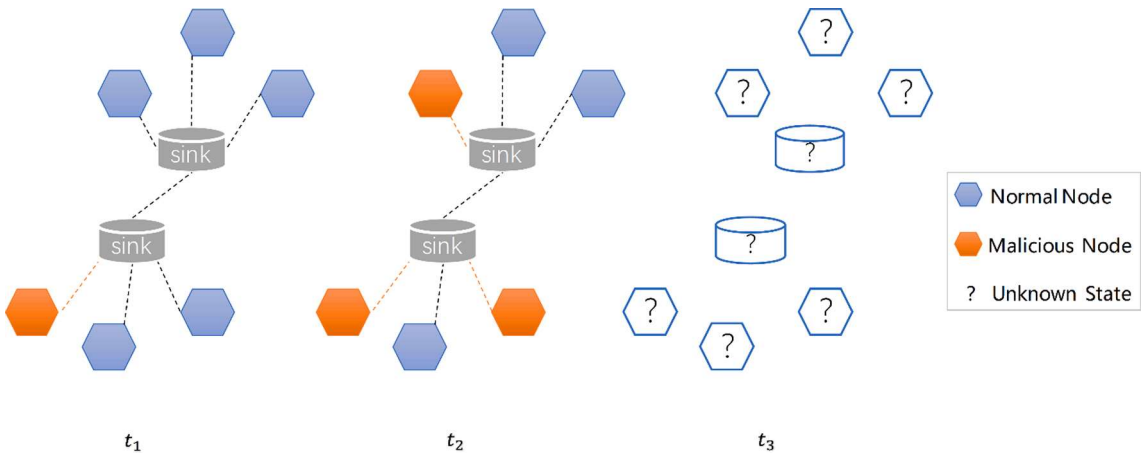

Fig. 2. An example of the virus infection process in WSNs.

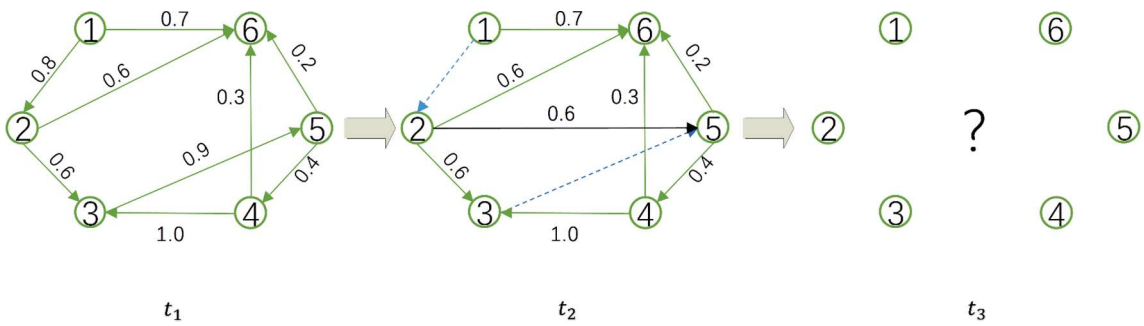

Fig. 3. An example of the evolution of a temporal network.

Therefore, we propose Adaptive Graph Convolution Network (AGCN) and integrated it with LSTM cells, which provides T-GAN stronger ability in extracting features of complex networks. Furthermore, a dual attention mechanism is introduced to improve the sensitivity of the model to each time step of temporal networks. After generating the embedded representation vectors for the nodes in dynamic complex networks, a neural network with single layer serves as the decoder, in which nonlinear affine transformation is used as the mapping function. Compared with the fully connected neural network, the number of the parameters of the decoder designed in T-GAN can be greatly reduced due to parameter sharing, which plays a significant role in reducing training time and avoiding over-fitting.

The main contributions of this paper are summarized as follows:

- We proposed a novel deep learning based network model (T-GAN) with an encoder-decoder framework for the purpose of prediction of properties and events of complex networks. The model can be used for the control and management of complex networks, with many real applications.
- For the encoder of the model T-GAN, we propose new adaptive graph convolution network and integrate it with LSTM cells, which provides T-GAN strong ability in extracting features of complex network and greatly improves the scalability of the model. And we applied attention blocks to help capture the importance and uniqueness of different time steps in complex networks and enhance the sensitivity of the model.
- Our proposed T-GAN architecture is general and scalable, which can be used for a wide range of real applications. We demonstrate the applications of T-GAN to three prediction tasks for evolving complex networks, namely, node classification, feature forecasting and topology prediction over 6 open datasets. Our T-GAN based approach significantly outperforms the existing, achieving improvement of more than 4.7% in recall and 25.1% in precision.
- We also conducted additional experiments to evaluate the generalization of the proposed model on learning the characteristic of time-series images. Extensive experiments demonstrate the effectiveness of T-GAN in learning spatial and temporal feature and predicting properties for complex networks.

The remainder of the paper is organized as follows. Section 2 presents the related works. Section 3 presents the problem definition and notations. Section 4 introduces the proposed architecture in detail. Extensive experimental results and analysis are shown in Section 5, and additional experiments are presented in Section 6. Finally, conclusions are drawn in Section 7.

2. Literature review

Complex network is a relatively new research area. The main focus of complex network in the early stage is on the analysis and modeling of complex networks. Graph embedding (graph representation learning) is an important research branch of deep learning applied in complex networks [11], which takes advantage of the powerful non-linear fitting capability of the deep neural network that maps high-dimensional sparse networks to dense vectors in a lower-dimensional space. It is convenient to perform downstream tasks such as topology prediction, node classification and so on through learning the low-dimensional representation vector of nodes (problems from in complex networks. It can be seen that graph embedding method is a powerful tool to study complex networks. Deep Walk [12] refers to Skip2gram [13] which is widely used in the natural language processing (NLP), uses the truncated random walk [14] to sample the nodes in the network to obtain the context (semantic information) of the nodes in the network, and adopts the training approach similar to Word2vec [15] to train the neural network. Deep Walk achieves good results in handling the problem of analyzing webpage relationship and performs well in the webpage classification task. However Deep Walk is not suitable for weighted graphs. LINE [16] optimizes an objective function to learn the node embedding which can reveal both the local and global structures of a network by introducing the first-order and second-order proximity. Node2vec [17] adopts both breadth first search (BFS) and DFS strategies during the sampling period, which can be regarded as a variant and extension of Deep Walk. SDNE [18] introduces the encoder-decoder architecture into graph embedding for the first time, which is an end-to-end deep learning model.

However, the graph embedding methods proposed above have common defects. They are only suitable for static graphs, and most of them merely utilize the topological connection information of complex networks, ignoring the extensive features of nodes. The complex network in the real world is dynamic with changing topology. There are increasing research interests on dynamic complex networks. Inspired by the restricted Boltzmann machine, Li et al. [19] proposed a generative model named conditional temporal restricted Boltzmann machine, which not only overcomes the high computational cost of traditional probability models such as Gaussian process model [20], but also has the ability of feature extraction and transformation which linear models [21] cannot provide. DynamicTriad [22] was proposed to capture the triadic closure process in dynamic networks to learn its evolution pattern and representation vectors for each vertex at different time steps, which achieves good gains in topology prediction and node classification tasks. Ibrahim et al. [23] aggregates the sequence of adjacency matrices by adding corresponding attenuation coefficients to different time steps, and utilizes topological indicators to perform the topology prediction task. On the basis of SDNE, DDNE [24] uses GRU (Gated Recurrent Unit) as the encoder to replace the fully connected neural network in the original SDNE model for graph representation learning, and Goyal et al. [25] proposed the DynGem model inspired by SDNE, in which the encoder and decoder of the model are trained simultaneously by optimizing the local and global losses. Chen et al. [26] adopted fully connected neural network as the encoder to learn the spatial features with stacked LSTM layers capturing the time characteristics of dynamic networks. TNE [27] learns the hidden state of dynamic graphs by non-negative matrix factorization and maps temporal networks into vector space which can be used to predict future topological state of the network. CANDECOMP/PARAFAC(CP) [28] decomposes a three-dimensional time-series tensor, which can be considered as a similar form of SVD who decomposes the target tensor into the product of three tensor multipliers, just as SVD decomposes a matrix into orthogonal matrices, but the difference is that the CP method no longer requires orthogonal conditions. However, the data with graph structure is a kind of typical data in non-Euclidean space, i.e., unstructured data, and most of the above models are based on traditional neural networks, having not achieved satisfactory results when adopted to perform the embedding task for the non-Euclidean data. For example, convolution operations in Convolutional Neural Networks (CNNs) are designed for image pixels in Euclidean space, resulting that the traditional CNN cannot be simply extended to graphs. In [29], a variant of CNN suitable for the data with graph structure was proposed to extract node features, which extends the application of CNN to the graph field. Apart from the specific CNN model mentioned above, Monti et al. [30] proposed the MoNet that introduces high-dimensional Gaussian distribution into the feed-forward process of CNN based on manifold theory, making the model suitable for non-Euclidean data. GNN (Graph Neural Network) was first elaborated by Scarselli et al. [31], but was not widely used at that time due to its large computation overheads on its parameters. Bruna et al. [32] presented the polynomial form of Laplacian matrix which transform the original parameters into a new form that can be trained. GCN (Graph Convolution Network) [33] is proposed to simplify the parameters in [32], and a trainable form of single-layer GCN was born. GraphSage [34] aggregates and propagates the features of nodes by spatial graph convolution and achieves good results in the task of node classification. Velickovic et al. [35] proposed GAT (Graph Attention

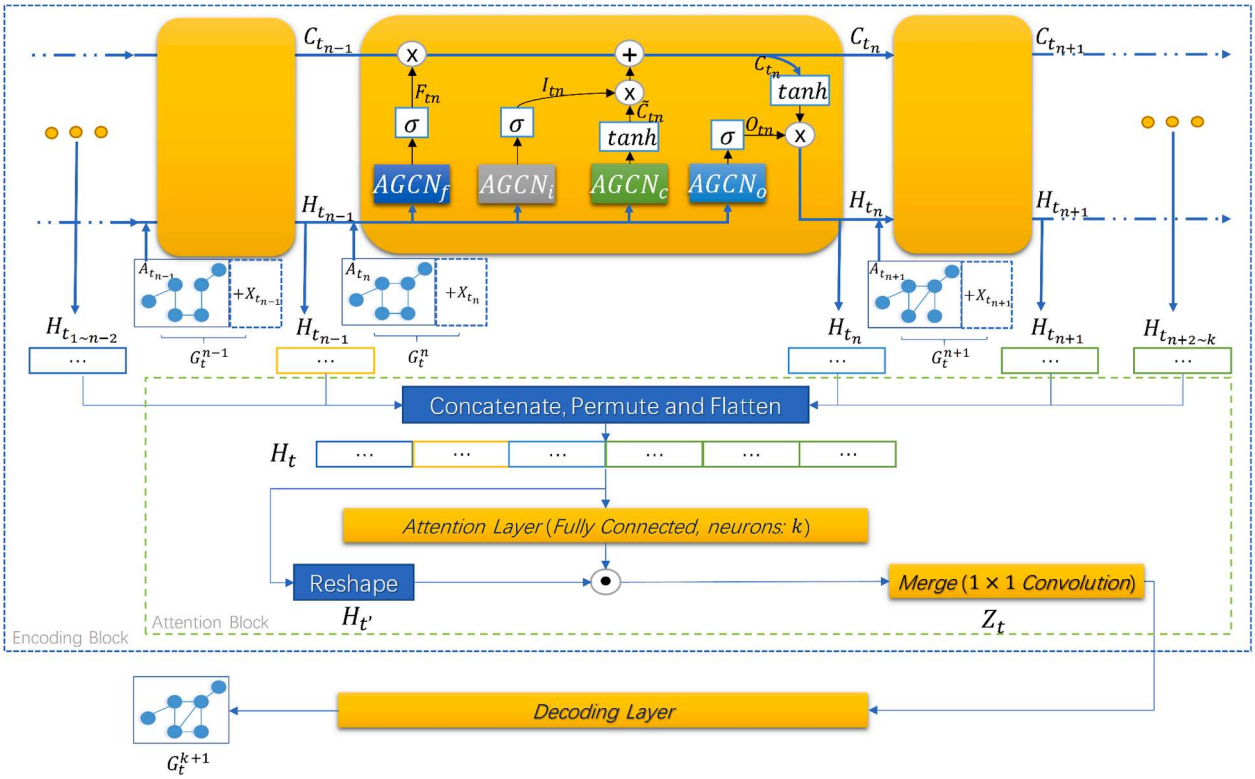

Fig. 4. The system framework of T-GAN. The rounded rectangles and circles in golden refer to the LSTM cells, rectangles in blue indicate tensor manipulations, and rectangles in golden denote neural network layers. $AGCN_f$, $AGCN_i$, $AGCN_c$, and $AGCN_o$ represent the four gates (input gate, forget gate, cell state, and output gate) in each cell respectively, in which adaptive graph convolution is adopted in the process of feature extraction. σ (sigmoid) and $tanh$ are the activation functions for each cell. $\odot$ is the symbol of Hadamard Product. H_{t_i} is the hidden state tensor of the ith time step, generated by the corresponding T-GAN cell.

Network), which is also a spatial convolution model, to solve the problem that the GCN based deep learning models treat all neighbors of each node equally. GAE (Graph Auto Encoder) [36] is an end-to-end deep learning model, which autonomously learns the hidden state representation of networks in an unsupervised manner. TAGCN (Topology Adaptive Graph Convolutional Networks) [37] extends the Laplacian Chebyshev polynomials of reference [33] to the k-order form, improving the predictive effect accordingly. Referring to the concept of PageRank and residual connection, Klicpera et al. [38] designed a model with graph residual convolution neural network to maintain the non-degeneracy of features, especially when the model needs to combine the features of the previous layers with the current layer via identity mapping. In reference [39], it is proved that the multi-layer GCN is actually equivalent to a single-layer with multi-order kernel that aggregates the original node features by cascading convolution operations, which simplifies the calculation process of multi-layer GCN and provides a new perspective for designing graph neural networks. Zhao et al. [40] built a traffic flow prediction model by combining LSTM and GCN. Lei et al. [41] adopt the generative adversarial neural network as the discriminator to predict the topology of temporal networks coupled with GCN. It is noted that the existing research works on temporal graph networks are relatively few, and the temporally evolving features of complex networks has not been well studied.

3. Problem definition and notations

Temporal complex network can be denoted by a time sequence$G_T = \{G_1, G_2 \cdots, G_t\}$, where G_t is a sampling snapshot at time step t which is a triple comprised of vertices (nodes), edge set and node features, i.e., $G_t = (V, E_t, X_t)$. $V = \{v_1, v_2, \cdots\}$ denotes the set of vertices in the network. E_t denotes the set of edges connecting the network nodes at time step t. $X_t \in \mathbb{R}^{|V| \times F_t}$ is the node feature matrix at the sampling time t, where $|V|$ is the number of the nodes in the network and F_t is the dimension of node features. An adjacency matrix $A_t \in \mathbb{R}^{|V| \times |V|}$ of the network at time t can be constructed by V and E_t. For a general element $A_{i,j}$ of A_t, if $A_{i,j} > 0$, it represents the weight of the link between node i and node j, and if $A_{i,j} = 0$, it means there is no link between nodes i and j.A_t will degenerate into a binary matrix if the network is an undirected and unweighted graph, where $A_{i,j} = 1$ if there is an edge linking nodes i and j, and $A_{i,j} = 0$ otherwise

Temporal network embedding can defined as mapping the sequence of triples $G_T = \{G_1, G_2 \cdots, G_t\}$ into dense vectors in Euclid space for each vertex. The process of embedding can de denoted by$Z_t = f(G_t)$, $Z_t \in \mathbb{R}^{|V| \times F'}$, where f is the mapping function and F' is embedding dimension. Embedding the time sequence of temporal network is instrumental for the downstream classification or prediction task, which makes it possible to study and predict the evolutionary properties of dynamic complex networks.

Next we introduce the three prediction problems for complex networks studied in this paper.

Definition 1: Topology Prediction

The topology prediction task is that, given a sequence of topology denoted by $\{A_{t_1}, A_{t_2}, \cdots, A_{t_k}\}$, predicting the topological state of the network at the following time step. The problem can be defined as reconstructing the adjacency matrix at the next step by using the historical topology information of the network. The problem is defined mathematically as (1), where $\widehat{A}_{t_{k+1}}$ is the prediction result and $A_{t_{k+1}}$ is the ground truth (i.e., the actual topological state of the network at the next time step).

$$\widehat{A}_{t_{k+1}} = argmaxP\left(A_{t_{k+1}} \middle| A_{t_1}, A_{t_2}, \cdots, A_{t_k}\right) \tag{1}$$

Definition 2: Node Features Forecasting

Node feature forecasting problem is that, given a sequence of triples

$G_T = \{G_1, G_2 \cdots, G_t\}$, forecast node feature forecasting as shown (2). $f: G_T \rightarrow Z_T$ is a mapping function that embeds the temporal network into low-dimensional vectors, where $Z_T \in \mathbb{R}^{|V| \times F}$ is the embedding result for G_T. *decoding* denotes the function of the neural networks, which depends on the specific applications.

$$X_{t+1} = decoding(f(G_T)) \tag{2}$$

Definition 3: Node Classification

The problem of node classification is that, given the sequence of triples $G_T = \{G_1, G_2 \cdots, G_t\}$, predict the categories of all the nodes in the dynamic network at the next sampling time as shown in (3), where $f: G_T \rightarrow Z_T$ is the mapping function, and$Z_T \in \mathbb{R}^{|V| \times F}$ is the embedding result. In particular, if it is a binary classification problem, Support Vector Machine or Logistic Regression can be adopted to perform the classification task.

$$class(Node_i) = Softmax(f(G_T)) \tag{3}$$

4. Methodology

4.1. System framework of T-GAN

In order to capture the evolution pattern of topology and node features of temporal networks, we propose a deep learning network model T-GAN with adaptive graph convolution and attention mechanism. T-GAN is a deep neural network model consisting of an encoder and decoder, which integrates the topological structure of complex networks with the extensive feature information of vertices for learning and modeling the evolutionary property of temporal networks. Stacked-LSTM layers are used as the backbone of the encoder in T-GAN, and the feature transformation of data stream in LSTM cells is changed by the proposed adaptive graph convolution. The overall structure of T-GAN is shown in Fig. 4.

In T-GAN network model tensors consisted of hidden states of each cell are output to the next parallel LSTM layer with identical structure to learn the deeper and abstract spatial and temporal characteristics of temporal complex networks. The depth of LSTM layer is 2 for the sake of memory limitation. Finally, the model spontaneously learns the weights of each time step of the temporal network by adopting the dual attention mechanism to integrate the outputs of each cell of the last stacked-LSTM layer into the final embedding vectors of the temporal network. The output of T-GAN from the decoding layer that is designed according to subsequent tasks, such as node classification, topology prediction, and feature forecasting.

In order to improve the sensitivity of the model to each time step of the input, we design the attention module to integrate the extracted features of each time step. The detailed data flow and calculation process of the tensors generated by LSTM cells equipped with AGCN are depicted in Fig. 6. There are two stages for the operation of the attention mechanism. The first stage is to train a fully connected neural network to get the coefficients for each time step, and in the second stage a CNN layer with $1 \times 1 \times k$ convolution kernel is introduced to aggregate all the weighted tensors of different time step and output the final embedding results for the temporal network G_T. The decoding layer at the end of T-GAN is a neural network layer in which nonlinear affine transformation is adopted. The kernel size and activation function of it can be selected according to the different tasks. As the data that the model handles are large-scale 2D or even 3D tensors, compared with fully connected neural network, the decoder of T-GAN can not only exponentially reduce the number of parameters that needs to be trained, but also do not need to flatten or reshape the tensors, so as not to affect the original spatial structure.

4.2. Adaptive graph convolution network

Among the existing literatures, there are two main graph convolution methods. One is based on Laplacian spectrum, and the other refers to the spatial convolution. The convolution form based on spectrum theory proposed by Kipf et al. [33] has been widely used, and the representative works of spatial graph convolution include GraphSage [34] and GAT [35]. The graph convolution based on spectral method is only suitable for undirected graph because it depends on the symmetric Laplacian matrix of the adjacency matrix, and it has been proved to have low-pass filtering characteristics, which leads to a sharp decline in the performance of the model when the convolution layer depth is greater than three. GraphSage needs to do time-consuming sampling operation for each node and updates the feature of the node by label propagation mechanism through aggregation function. However, the number of sampling nodes in GraphSage is fixed, so that it can't completely integrate all neighbor nodes, resulting in information loss. GAT assumes that nodes with similar inner product will have greater weight by introducing multi-head attention mechanism. But in heterogeneous networks, such operation cannot reflect the actual relationship among nodes well.

In the field of graph signal processing, the operations of suppressing, reserving or enhancing the intensity of the spectrum components in the graph signals are defined as the graph signal filtering. The graph filter applied is also called the graph shift operator, which is defined as formula (4), where V is the orthogonal matrix after eigen-decomposition of the filter H.

$$H \in \mathbb{R}^{N \times N} = V \Lambda_h V^T = V \begin{bmatrix} h(\lambda_1) & & \\ & \ddots & \\ & & h(\lambda_N) \end{bmatrix} V^T \tag{4}$$

Designing the function $H(\lambda_k)$ means filtering the eigenvalues of the graph shift operator, and this process is regarded as the frequency response function. The graph filter H acts on the graph signal X, and its output is $y = HX$. The graph shift operator characterizes its transformation operation on the first-order subgraph of the whole graph, which is also the property of spatial graph convolution. Given that the kth iteration result of graph signal x filtered by the graph shift operator L is $x^{(k)} = Lx^{(k-1)}$, the $(k+1)^{\text{th}}$ convolution result can be obtained by (5).

$$y = Hx = \sum_{k=0}^{K} h_k L^k x = \sum_{k=0}^{K} h^k x^{(k)} \tag{5}$$

As shown in (5), from the kth iteration to the $k+1$th iteration, we only needs use the graph shift operator L to filter the sinal x for only once, i.e., the next feature update process of all nodes only needs their 1-st order neighbors to participate in such a filtering calculation iteration. In other words, the final filtering operation can be completed by calculating the inner product of the graph shift operator H and the signal x iteratively.

In this paper, we propose the adaptive graph convolution network (AGCN) in spatial domain, whose feature extraction and update process is defined as follows:

$$H^{l+1} = \sigma\left(\left(M \odot \widehat{A}\right) H^l W + b\right)$$

$$\widehat{A} \in \mathbb{R}^{|V| \times |V|} = A + I + \delta$$

$$\delta_{i,j} = \frac{1}{|V|}, \forall i, j \in [1, |V|] \tag{6}$$

where $\widehat{A} \in \mathbb{R}^{|V| \times |V|}$ is the revised adjacency matrix (δ is introduced to avoid the failure of M, and identity matrix I is introduced to keep self-loop), and $M \in \mathbb{R}^{|V| \times |V|}$ is the self-adaptive weight matrix that can be trained. In the graph signal processing theory, $M \odot \widehat{A}$ is also defined as the graph shift operator, which can filter the graph signal adaptively. In (6),$H \in \mathbb{R}^{|V| \times F}$ is the feature matrix, which is consisted of the feature vectors of each node. Specifically, $H^0 = X$ is the initial features of each node, and W, b denote the transformation kernel and the bias to be

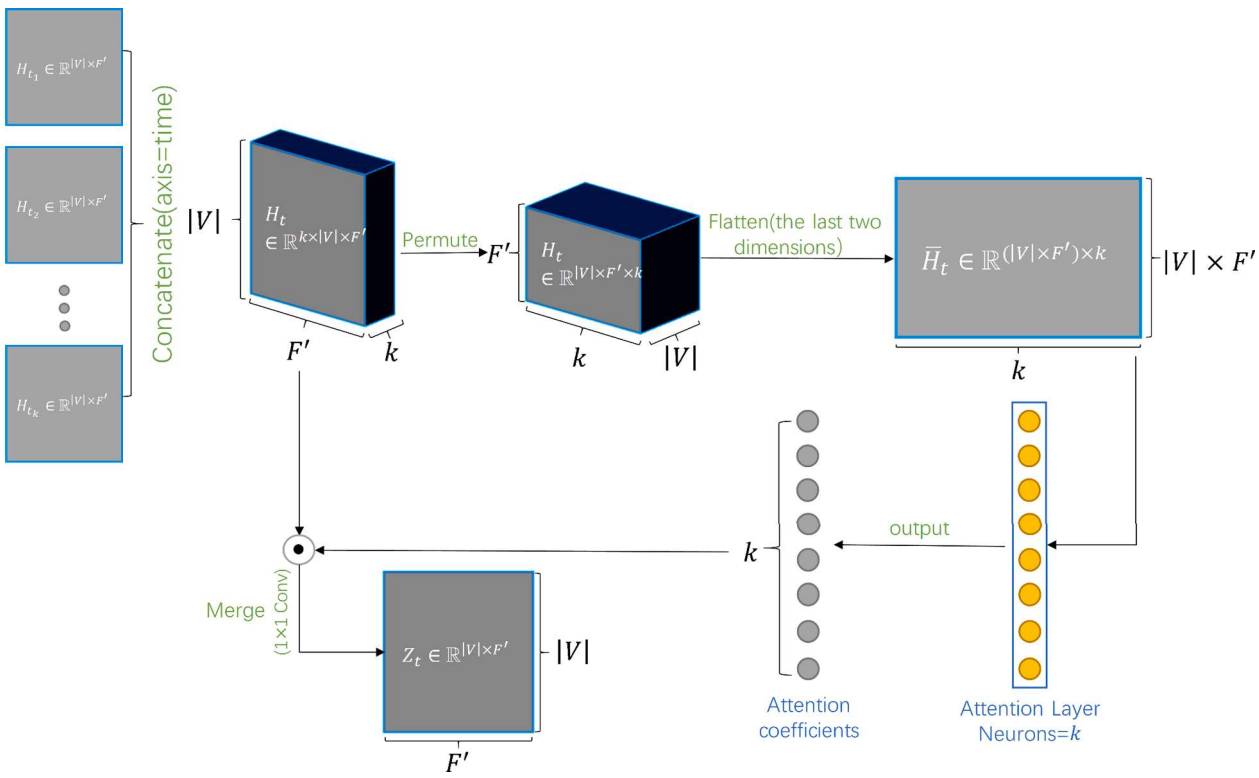

Fig. 5. A detailed illustration of the attention block of T-GAN. Squares in gray indicate the tensors throughout the model, and circles in golden refer to the neurons of the attention layer. $\boldsymbol{H}_{t_i}$ is the hidden state embedding of the ith time step, and $\boldsymbol{H}_t$ is the combination of all the $\boldsymbol{H}_{t_i}$, $\overline{\boldsymbol{H}}_t$ is the tensor generated by reshaping and permuting $\boldsymbol{H}_t$, which is also the input of the attention layer. $\boldsymbol{F}'$ is the dimension of the embedding vectors of nodes, $\boldsymbol{k}$ denotes the number of time steps, and $|\boldsymbol{V}|$ is the number of nodes in the temporal network. $\boldsymbol{Z}_t$ is the final embedding matrix of the dynamic network $\boldsymbol{G}_T = \{\boldsymbol{G}_t^1, \boldsymbol{G}_t^2 \cdots, \boldsymbol{G}_t^k\}$.

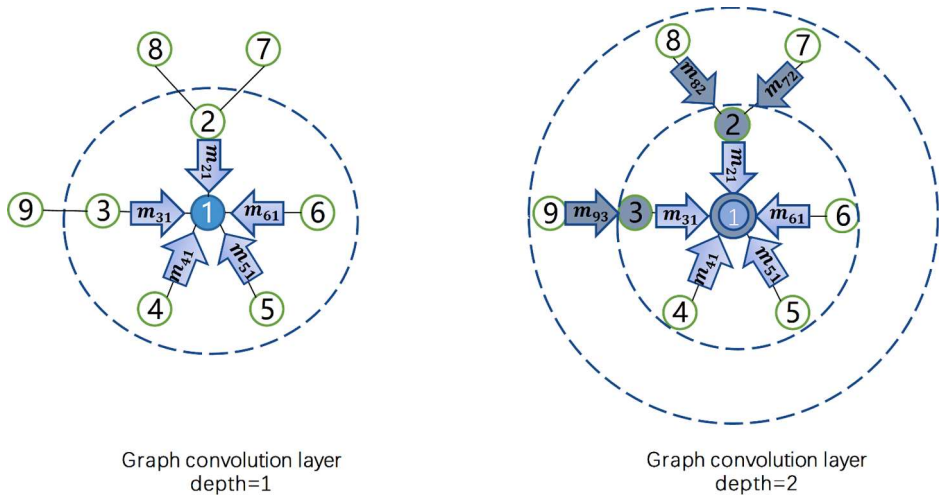

Fig. 6. Visual illustration of the adaptive graph convolution operation. The figure left denotes one complete cycle of adaptive graph convolution for node 1, in which $\boldsymbol{m}_{ij} \in \boldsymbol{M}$ are the adaptive coefficients for node feature aggregation. After the operation, each node can aggregate the features of its neighbor nodes adaptively. As is shown in the right-hand picture, the 2nd convolution can render node 1 the ability to extract the features of its second-order neighbors (node 7,8 and 9).

trained respectively. $\odot$ indicates Hadamard Product, and σ is the activation function. The weight of neighbors of each vertex can be automatically updated by back propagation after introducing self-learning weight matrix M. When calculating the inner product of the weighted topology matrix$M \odot \widehat{A}$ and the node feature matrix H, the feature update process of each node will be affected by all their neighbors. The feature aggregation process of $(M \odot \widehat{A})H^l$ is shown in (7).

$$f\left(v_i \middle| \widehat{A}, H\right) = \bigcup_{f=1}^{F} \sum_{j \in \Gamma(v_i)} M_{i,j} \times \widehat{A}_{i,j} \times H_{j,f} \tag{7}$$

where $\Gamma(v_i)$ is the set of the neighbors of node v_i, and the outmost operation of (7) denotes concatenating all the dimensions of node feature vectors, in which F is the number of dimensions. Using adaptive spatial graph convolution can avoid the sampling operation, which is computationally expensive. The whole graph including adjacency

Table 1
Terms and notations used in T-GAN.

Symbol	Description
G_T	The temporal network feeded into the model
A_{t_i}, X_{t_i}	The adjacency matrix and feature matrix of G_T at time step i
$F_{t_i}, I_{t_i}, O_{t_i}$	The output of the 3 gates of LSTM cell
C_{t_i}, C_{t_i}	The cell state of each LSTM cell
H_{t_i}	The hidden state extracted by the model at i^{th} time step
$H_t, \overline{H}_t$	Hidden state matrix composed by H_{t_i},another form of H_t after reshaping
$Z_t, F^{'}$	Embedding of G_T, dimension of embedding
$M, W_{i,f,c,o}$, $b_{i,f,c,o}$	Weights and biases of adaptive graph convolution
α	Attention coefficients

matrix and feature matrix can be inputted into AGCN at the same time. W, b, and the activation function σ in AGCN can cooperatively extract deeper and more abstract features for each node. Fig. 5 gives a visual explanation for equation (7).

- Computational Complexity Analysis: as is shown in (6), no eigen-decomposition is required and the operation of the self-adaptive layer can be parallelized across the whole graph, with the computation of output features being also parallelized, which indicates that AGCN is computationally efficient. The time complexity of a single AGCN layer can be expressed as $O(|V|^2FF^{'})$, where $F^{'}$ is the embedding dimension.
- Difference between AGCN and GAT (Graph Attention Networks): AGCN introduces $\delta = 1/|V|$ and M to allow every node having the chance to aggregate the features of the nodes that are not their direct neighbors, which has a positive effect on performing the learning task for heterogeneous graphs. Besides, In a densely connected network, AGCN will be more computationally effective than GAT whose time complexity is $O(k(|V|FF^{'} + |E|F^{'}))$, where $|E|$ is the number of edges in the graph.

Proof
$|V|^2FF^{'} = (k - k + |V|)|V|FF^{'}$
$= k|V|FF^{'} + (|V| - k)|V|FF^{'}$
$k[|V|FF^{'} + |E|F^{'}] = k|V|FF^{'} + k|E|F^{'}$
scaling $k|E|F^{'}$: $k|E|F^{'} \le k\frac{|V|}{2}[|V| - 1]F^{'}$
If$(|V| - k)|V|FF^{'} < k\frac{|V|}{2}[|V| - 1]F^{'}$
Then$(|V| - k)F < k\frac{(|V| - 1)}{2}$
scaling both sides of the above equation
$2(|V| - k)F < k(|V| - k)$
If$(|V| - k) > 0$ and $2F < k$,
We can conclude that AGCN is more computationally effective.

4.3. Encoder model of T-GAN

Before explaining the detailed propagation process of the proposed deep neural network model, we summarize the notations and terms used in our model in Table 1. Given the sequence of triples $G_T = \{G_t^1, G_t^2 \cdots, G_t^k\}$, where $G_t^i = (A_{t_i}, X_{t_i})$ and $A_{t_i} = (V, E_{t_i})$, the first step is to calculate the outputs of the three gates for LSTM cells equipped with AGCN. The forget gate and input gate decide what should be forgotten from the previous cell state and what should be injected into the current state respectively as shown in (8), where || denotes concatenating operation.

$$F_{t_i} = \sigma\big((M \odot A_{t_i})[H_{t_{i-1}}||X_{t_i}]W_f + b_f\big)$$

$$I_{t_i} = \sigma((M \odot A_{t_i})[H_{t_{i-1}}||X_{t_i}]W_i + b_i)) \tag{8}$$

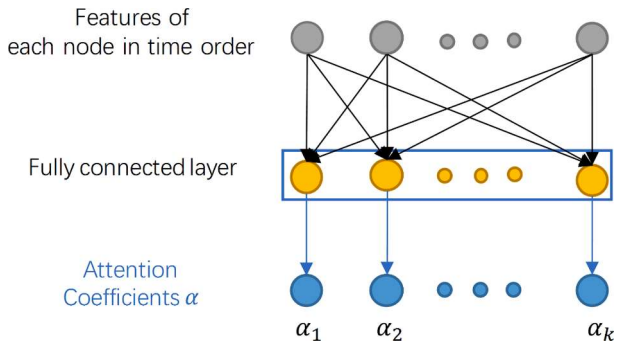

Fig. 7. Depiction of the Softmax Attention layer.

The update process of cell state is shown in Eq. (9).

$$C_{t_i} = tanh((M \odot A_{t_i})[H_{t_{i-1}}||X_{t_i}]W_c + b_c)$$

$$C_{t_i} = F_{t_i} \odot C_{t_{i-1}} + I_{t_i} \odot C_{t_i} \tag{9}$$

The output gate and the final hidden state are calculated by the following formula:

$$O_{t_i} = \sigma((M \odot A_{t_i})[H_{t_{i-1}}||X_{t_i}]W_o + b_o)$$

$$H_{t_i} = O_{t_i} \odot \tanh(C_{t_i}) \tag{10}$$

The hidden state of a time step in the input time series can be obtained via Eq. (10), and the hidden states of all time steps can be obtained by repeating the process mentioned above. The hidden states of k time steps are then merged into a three-dimensional tensor$H_t = \{H_{t_1}, H_{t_2}, \cdots, H_{t_k}\} \in \mathbb{R}^{k \times |V| \times F^{'}}$. H_t will be transformed into a two-dimensional tensor $\overline{H}_t \in \mathbb{R}^{(|V| \times F^{'}) \times k}$ by axis rotation and shape transformation. The purpose of axis rotation is that the subsequent neural network can extract the evolution pattern of the temporal network and distinguish the influence of different time steps on the evolution of the whole network. The two-dimensional tensor $\overline{H}_t$ is used as the input of the first attention layer. The first attention layer is composed of a fully connected neural network, and its detailed structure is shown in Fig. 7, where the number of neurons is the same as the sampling time steps k of the dynamic network. The activation function of the first attention layer is *Softmax*, and the attention coefficients can be obtained by Eq. (11), where W_{att} and b are the weights and bias of the first attention layer. The importance of each time step will be automatically determined by back propagation.

$$\alpha_i = \frac{\exp\left(W_{att}\overline{H}_{ti} + b\right)}{\sum_{j=1}^{k}\exp\left(W_{att}\overline{H}_{tj} + b\right)} \tag{11}$$

The weighted hidden states $H_t^{'} \in \mathbb{R}^{k \times |V| \times F^{'}}$ can be obtained by calculating the Hadamard product of the attention coefficients α_i and hidden state H_{t_i}, as shown in (12), where the Sigma operation denotes the concatenation.

$$H_t^{'} = \sum_{i=1}^{k} \alpha_i H_{t_i} \tag{12}$$

$H_t^{'}$ is still a three-dimensional tensor and can be regarded as a two-dimensional tensor with k channels. We need to merge $H_t^{'}$ from the time dimension, which can be achieved by means of calculating the mean, sum or concatenation of $H_t^{'}$. In this paper, the convolutional neural network (CNN) is exploited to constitute the second attention layer, and the CNN contains only one convolution kernel ($1 \times 1 \times k$) with linear activation function. The 1×1 convolution operation serves as the second attention layer of the attention module which is designed to improve the sensitivity of the model to each time step. Finally, we can

get the embedding vectors $Z_t \in \mathbb{R}^{|V|\times F'}$ of each node, as shown in (13).

$$Z_t = Conv_{1\times 1\times k}\left(H_t^{'}\right) \tag{13}$$

4.4. Decoder model of T-GAN

Section4.3 introduces the embedding method in detail. In this section, we introduce the decoder block of T-GAN. In the scenario of embedding temporal networks, the data that we operate are usually high-dimensional tensors, and the embedding result $Z_t \in \mathbb{R}^{|V|\times F'}$ is a matrix whose elements will increase exponentially with the increase of nodes. If a fully connected neural network is used as the decoder, the scale of training parameters will be very large. For example, considering that a dynamic graph with 100 vertices is embedded into $Z_t \in \mathbb{R}^{100\times 32}$, if we need to perform the topology prediction task, the objective is to obtain the adjacency matrix $A_{t_{k+1}} \in \mathbb{R}^{100\times 100}$ at the next time step, which means a mapping function $f : Z_t \to A_{t_{k+1}}$ is needed to perform the task. If a fully connected neural network is used to fit the mapping function, there will be more than 32 million parameters to be trained, which easily leads to the over-fitting problem. Moreover, the fully connected neural network can only deal with one-dimensional vector, implying flattening Z_t into one-dimensional vector, which breaks the spatial structure of the original data. However, if a neural network which takes advantage of nonlinear affine transformation kernel is used as the decoder, the two-dimensional transformation kernel W_d is set to 32×100 for the same topology prediction task mentioned above, and we only need to train 3300 parameters (including bias). Because of the parameter sharing property, the amount of training parameters of the latter is only 1/10000 of that of the former, and this gap will become more obvious with the increase of the number of nodes.

The decoder of T-GAN can be defined mathematically as $\sigma(Z_t W_d + b_d)$, where W_d is the transformation kernel, b_d is the bias, Z_t is the embedding matrix for the specific temporal network, and σ is the activation function. The fitting ability of the decoder can be enhanced by using nonlinear activation function. According to different downstream tasks, the activation function of decoder and the size of the kernel W_d can be flexibly set.

4.5. Loss functions and training

The proposed model is trained to capture the spatial and temporal evolution patterns of the input temporal network. This section will list the loss functions of the proposed model in different application scenarios.

Since there are many kernels in T-GAN, the regularization item which is designed to avoid over-fitting and constitute the loss function is defined as Eq. (14), where $W_{i,f,c,o}$ is the weights of LSTM cell equipped with AGCN, W_{att} is the weight of the neurons of the 1st attention layer, W_{conv} is the convolution kernel weight, W_d is the weight of the decoder layer, and M is the weight of the kernel in AGCN layer.

$$\mathscr{L}_{reg} = \|W_i\|_F^2 + \|W_f\|_F^2 + \|W_c\|_F^2 + \|W_o\|_F^2 + \|W_{att}\|_F^2 + \|W_{conv}\|_F^2 + \|W_d\|_F^2 + \|M\|_F^2 \tag{14}$$

The Mean Square Error (MSE) can be adopted as the loss function to perform the edge weight prediction task or non-discrete feature forecasting, as shown in Eq. (15), in which $\widehat{A}/\widehat{X}$ is the prediction results of T-GAN, A/X (adjacency matrix/feature matrix) is the groud truth, and λ is the trade-off coefficient which will be determined in the training process. It should be noted that using sigmoid as the activation function is not recommended when the objective function of the model is MSE, or the training process will be hampered due to the gradient vanishing problem.

$$\mathscr{L} = \lambda\mathscr{L}_{reg} + \frac{1}{2}\|\widehat{A} - A\|_F^2 = \lambda\mathscr{L}_{reg} + \frac{1}{2}\sum_i\sum_j\left(\widehat{A}_{i,j} - A_{i,j}\right)^2$$

$$\mathscr{L} = \lambda\mathscr{L}_{reg} + \frac{1}{2}\|\widehat{X} - X\|_F^2 = \lambda\mathscr{L}_{reg} + \frac{1}{2}\sum_i\sum_j\left(\widehat{X}_{i,j} - X_{i,j}\right)^2 \tag{15}$$

For binary topology or feature prediction task, the goal is to output a matrix as close as possible to the ground truth. Considering the sparsity of network connections, the penalty matrix P is introduced into the focal loss function which is defined as (16), where α is the balance factor and γ is designed to make the model pay more attention the samples which are hard to distinguish.

$$\mathscr{L} = \lambda\mathscr{L}_{reg} + P \odot focal\ loss\left(\widehat{A}\right)$$

$$focal\ loss\left(\widehat{A}_{i,j}\right) = \begin{cases} -\alpha\left(1 - \widehat{A}_{i,j}\right)\log\widehat{A}_{i,j}, A_{i,j} = 1 \\ -(1-\alpha)\widehat{A}_{i,j}^{\ \gamma}\log\left(1 - \widehat{A}_{i,j}\right), A_{i,j} = 0 \end{cases}$$

$$P_{i,j} = \begin{cases} \rho, A_{i,j} = 1 \\ \varepsilon, A_{i,j} = 0 \end{cases} \tag{16}$$

The purpose of introducing penalty matrix is to further alleviate the situation that the model always tends to output 0 due to the sparsity of topological connections and be skewed towards fitting the negative samples. In order to make the model pay more attention to the sample whose label is 1 in the training period, ε can be fixed to 1, then ρ is set to a larger number. In the case of binary discrete feature prediction, the definition of loss function is the same as (16), where $A_{i,j}$ will be replaced by the feature matrix $X_{i,j}$ accordingly, illustrated by (17). We use RMSProp [42] optimizer to train the proposed model to optimize the objective function.

$$\mathscr{L} = \lambda\mathscr{L}_{reg} + P \odot focal\ loss\left(\widehat{X}\right)$$

$$focal\ loss\left(\widehat{X}_{i,j}\right) = \begin{cases} -\alpha\left(1 - \widehat{X}_{i,j}\right)\log\widehat{X}_{i,j}, X_{i,j} = 1 \\ -(1-\alpha)\widehat{A}_{i,j}^{\ \gamma}\log\left(1 - \widehat{X}_{i,j}\right), X_{i,j} = 0 \end{cases}$$

$$P_{i,j} = \begin{cases} \rho, X_{i,j} = 1 \\ \varepsilon, X_{i,j} = 0 \end{cases} \tag{17}$$

5. Experiments and analysis

In order to verify the effectiveness of the proposed model, we conduct various experiments and analysis of the model for 3 applications over 6 open temporal complex network datasets. We also compare our approach with the current baseline methods. All experiments are run on the server equipped with two Nvida RTX 3060. The deep learning model is built by Keras 2.1.4. The version of TensorFlow on the server is TensorFlow-GPU 1.14.0, and the CUDA version is 10.1. The code of the T-GAN deep learning architecture proposed in this paper has been uploaded to our personal GitHub homepage [43].

5.1. Datasets and preprocessing

The 6 investigated datasets can be divided into three categories, including the user-commodity bipartite graph network, the transaction records of a financial company and dynamic social networks, which are used for three different learning tasks (feature forecasting, node classification and topology prediction).

Table 2
Statistics of datasets.

Dataset	$\|V\|$	Featureless	Timespan	Time interval	Timesteps
TaoBao	1000	No	31 days	1 day	31
Loan	2000	No	13 months	1 month	13
Facebook	1899	Yes	193 days	7 days	27
Eu-Core	986	Yes	803 days	14 days	57
Enron	151	Yes	11 weeks	1 week	11
Hypertext	113	Yes	2.46 days	1 h	46

- Samples in TaoBao dataset [44] is composed of user details, commodity information and purchase history. 1000 users' purchase records of 5755 kinds of goods within 31 days (2016 / 11 / 18–2016 / 12 / 18) were randomly selected as the experimental data from the original data set. A user-commodity bipartite graph is comprised of the consumers' purchasing behaviors towards all kinds of products. Each user can be abstracted as a node, then the user's purchasing behavior towards all categories of goods constitutes the user's temporal feature matrix X_t. Because the topological relationship between users is not explicit, in order to simplify the process and facilitate the research, we use the address information of users and the buying habit (shopping time) to construct a static similarity matrix S through algorithm 1, which is then used as the adjacency matrix A for T-GAN. The time step is defined as consecutive none-overlapping one day periods. Feature forecasting for TaoBao data set indicates predicting the purchase behavior for the 1000 users at the next time step by taking advantage of the history sequence of X_t and the similarity matrix S.
- Dataset of Loan [45] is provided by PPD (a finance company in China), which consists of detailed loan records of 20,000 users over 13 months. Each record contains rich user characteristics, labeled with the repayment status of the loan. The topology information between users is not shown in the original data set, and therefore we construct the corresponding adjacency matrix by utilizing the Social Network field in the record. Each vertex has a time-sensitive label, indicating if the user has ever failed to repay the loan. The length of each time step is set to one month. The purpose of the experiment on Loan is to perform the node classification task, predicting whether a user will repay his loan on schedule. We reduced the number of users to 2000 for the sake of memory limitation.
- Dataset of Facebook [46], is an instant messaging dataset. This dataset records the contacts between 1899 Facebook users within 193 days. The contacts between users can be modeled as a directed adjacency matrix A_t with vertices representing the users, which denotes the topological relationship between all users at the fixed time interval. The time step is defined as consecutive 7 days, and the experiment on this data set uses the historical contact information to predict the topology state of all users at the next time step.
- Dataset Eu-Core [47] is created from a dynamic email network. This data set records the contact information of 986 users within 803 days. We construct the temporal snapshot of the network in the same way as Facebook except that the time interval is 14 days.
- Dataset Enron [48] is also created from a dynamic email network, which contains the e-mail tracing records of 151 employees over three years. In the experiment, only logs from week 147 to 157 of the original data are selected to construct the temporal network snapshots, with time step defined as 1 week. The reason for choosing the time span is that the email network is the most active in this period, and the snapshots of topology are generated with one week as the time step.
- Dataset Hypertext [49] is created a human contact dynamic network of face-to-face proximity, recorded by wearable sensors carried by people who attended a conference. Only the most active part of the dataset is selected (records in night has been removed). Logs per hour will be transformed into one snapshot of the dynamic network.

Table 3
A summary of sample composition.

Dataset	Timesteps	k	Samples
TaoBao	31	3	28
Loan	13	2	11
Facebook	27	2	25
Eu-Core	57	4	53
Enron	11	2	9
Hypertext	46	5	41

Detailed statistics of all data sets are recorded in Table 2.

Since there is no explicit topological relationship between users in TaoBao dataset, Algorithm 1 is proposed to construct the adjacency matrix, where $\boldsymbol{\delta}$ is introduced to avoid the failure of the adaptive matrix M, In addition, shopping habit vectors of each node are formed by calculating the proportion of the quantity of transactions happened in different periods (morning to noon, afternoon, evening, and early morning) to the total volume of purchases of the whole day for each consumer, and the vectors are used to calculate the cosine similarity between users. Finally, the address similarity between nodes is presented by the location code of each node. α and β are the weights of shopping habits and the location respectively, which are set to 0.8 and 0.2 respectively. Our model assumes that users with similar shopping habits and geographical environment tend to buy the same category of items, and shopping habits have a greater impact on the purchase results which accounts for a larger proportion in the similarity calculation. For datasets without attribute information (Facebook, EU core, Enron, and hypertext), the identity matrix is adopted to replace the feature matrix X_t that needs to be fed into T-GAN [33].

Algorithm 1 Constructing the Similarity Matrix $S_{i,j}$

Input: address code l and shopping hours H of each consumer
Output: $S_{i,j}$

1. Initialize $S_{i,j} = \frac{1}{|V|} = \boldsymbol{\delta}, \forall i,j \in V_K$, where V_K is the set of vertices.
2. Calculate the vector p_i using the proportion of entries during different period $\boldsymbol{H}$ (morning, afternoon, evening and midnight) for each consumer.
3. $S_{i,j} = \boldsymbol{\delta} + \alpha \frac{p_i \times p_j}{|p_i||p_j|} + \beta \boldsymbol{\theta}_{i,j}$, where $\boldsymbol{\theta}_{i,j}$ is similarity between l_i and l_j(the ratio of the number of identical characters to the sum of all characters of the 2 address codes)

The method of generating training samples is as follows: for each data set, the consecutive $timestep_{1\ k}$ constitute a training sample, and the label of the sample is the snapshot of the complex network at $timestep_{k+1}$. The statistics of samples generated by the 6 dataset is summarized in Table 3. Considering that the number of samples generated by each data set is not particularly large, the verification set is ignored in our experiments, and Leave-One-Out cross validation is used to divide the samples into the training set and test set to make full use of these samples.

5.2. Evaluation metrics, baseline methods, and parameter setting

5.2.1. Evaluation metrics

Considering the sparsity and the binary feature of the data in our experiments, the area under the receiver operating characteristic curve (AUROC) [50], Precision, Recall and F1 score are selected as the evaluation metrics to assess the overall performance of the proposed T-GAN model. Moreover, since the feature forecasting task for Taobao data set equals to predicting the purchase behavior of consumers, the Normalized Discounted Cumulative Gain (NDCG) is also included in evaluating the model. NDCG is an information retrieval metric, which is widely used in recommendation system and link prediction (defined as whether the most important edges rank higher among all the links). NDCG is mathematically defined as formula (18), where $r(i)$ denotes as the

relevance score of items after sorting the probability of all links in descending order and IDCG is the DCG value calculated ideally (in our experiments, the correlation score is either 0 or 1, and p is set to 50). The value of NDCG ranges from 0 to 1, and 1 means that it is completely consistent with the expectation.

$$nDCG = \frac{DCG}{iDCG}, DCG = \sum_{i=1}^{p} \frac{2^{r(i)} - 1}{\log_2(i+1)} \tag{18}$$

5.2.2. Baseline methods

T-GAN is a deep learning model designed for graph embedding. In order to verify the effectiveness of our model, 5 benchmark methods of graph representation learning are chosen for performance comparison. Considering that TaoBao and Loan are both attributed graphs, thus Node2vec [17], Graph Auto Encoder (GAE) [36] and CANDECOMP / PARAFAC [28] based on matrix factorization, are selected for comparison. Noting that Node2vec and GAE are designed for the static attributed graph, we need to aggregate the temporal network into a static graph. The way to aggregate the network is shown in formula (19). In [28], the best results can be obtained when θ = 0.2.

$$X = \sum_{t=1}^{T} \left((1-\theta)^{T-t} X_t \right) \tag{19}$$

For the other four datasets, Node2vec is no longer applicable because only the topology information is available in these data sets. Hence, E-LSTM-D [26] and TNE [27] are used for comparison. Among the tested methods for our experiments, CP, E-LSTM-D and TNE are the models originally designed for temporal networks.

- Node2vec: The model learns a mapping function that maps the nodes into a low-dimensional vector space which can maximizes the likelihood of preserving network neighborhoods of nodes. Besides, the author designed a biased random walk procedure that using p and q to control the random walk process and generate the sampling sequence of vertices. Negative sampling is adopted to simplify the sampling process.
- GAE: GAE uses spectral graph convolutional neural network as the encoder and maps graphs into hidden vectors in an unsupervised manner.
- CP: The CP method decomposes a three-dimensional time-series tensor, which can be considered as a similar form of SVD. It decomposes the target tensor into the product of three tensor multipliers, just as SVD decomposes a matrix into orthogonal matrices, but the difference is that the CP method no longer requires orthogonal conditions. The hidden vector can be easily utilized for the subsequent learning task.
- E-LSTM-D(E-L-D): The model is an end-to-end deep neural network model. It trains two fully connected neural networks as the encoder and decoder respectively. The sequence of temporal topology is inputted into the encoder to generate the hidden vector. Then, the hidden vector will be fed into a stacked-LSTM model, and finally mapped to an adjacency matrix by the decoder.
- TNE: Time-series tensors are mapped to vector space via Markov process and non-negative matrix factorization.

5.2.3. Parameter setting

In our experiment, Node2vec, CP, and TNE need to be pre-trained, the trained models will be used as an encoder, and the corresponding neural network will be added as a decoder at the end of each model according to different prediction tasks. As for E-LSTM-D, GAE and the proposed model T-GAN, they are all end-to-end deep learning architectures, which means the encoder and decoder in the model can be trained synchronously. Since the labels of the 6 data sets are binary, we select sigmoid as the activation function for all decoders, and focal loss with penalty matrix is the objective function that needs to be optimized.

Table 4
Parameter setting.

Dataset	Model	Parameter setting
TaoBao	Node2vec	$p = 0.25, q = 4$, Decoder: fully connected layer
	GAE	Hidden units = 256, Decoder: fully connected layer
	CP	L = times steps of dataset TaoBao; weighting sliceθ = 0.2
	T-GAN	$\alpha = 0.8, \beta = 0.2, \rho = 100\lambda = 0.001$; Embedding dimension = 256, Depth of LSTM:2
Loan	Node2vec	$p = 0.25, q = 4$, Decoder: fully connected layer
	GAE	Hidden units = 256, Decoder: fully connected layer
	CP	L = times steps of dataset Loan; weighting slice $\theta = 0.2$
	T-GAN	$\rho = 100, \lambda = 0.001$; Embedding dimension = 256; Depth of LSTM:2
Facebook Eu-Core Enron Hypertext	GAE	Hidden units:64, Decoder: GCN
	CP	L = times steps of corresponding dataset; weighting slice$\theta = 0.2$
	E-LSTM-D	Encoder: 2 fully connected layers; depth of LSTM:2; Decoder: fully connected
	TNE	$BCGD_G$, Dimension $k = 20$, $\lambda = 0.0001$, Decoder: fully connected layer
	T-GAN	$\rho = 50, \lambda = 0.001$; Embedding dimension = 64; Depth of LSTM is 2.

More detailed information is shown in Table 4.

5.3. Experimental results and analysis

Leave-one-out cross validation strategy will split each data set into many subsets, and the number of the subsets is identical to the number of the samples in each data set. Each subset will be used for training the corresponding independent T-GAN model. The training results of each model will be recorded, including the loss function value and the epoch at which the model stops training. For each model, the number of training iterations is initialized to 1000. In order to avoid overfitting, we set the condition of the early termination of training: the difference of the loss function value between 2 consecutive iterations is less than 0.0001 or the loss function tends to deteriorate for 10 consecutive training epochs. The test set in each subset will be used to evaluate the performance of the proposed model. In order to simplify the whole training process and save time, for the baseline approaches selected for comparison, the method of splitting data set is as follows: 90% of the samples are randomly selected to constitute the training set, and the remaining 10% is used as the test set. The statistical results of the training history for each data set are shown in Fig. 8.

As shown in Fig. 8, due to the randomness of the parameter initialization process of the neural network, the final epoch at which the model achieved convergence is slightly different for the same data set. As for different data sets, the training results of the models are quite different for the factors of time step k, the number of nodes in the network and the initial dimension of node features. Using leave-out-out splitting method indicates training multiple identical models. Hence, each subset is trained in parallel on the server equipped with 2 Nvida RTX3060 by using multi-threading technology for the purpose of saving time. In order to integrate the training results of these independent models and evaluate the effectiveness of the proposed deep learning model for the graph evolution learning task, boxplot is used to visually illustrate the results of the proposed T-GAN on different evaluation metrics. Moreover, the results for the baseline methods have been transformed into a scatter plot which is integrated into the boxplot results of T-GAN according to the corresponding dataset, where the advantage and difference between our model and other methods can become obvious and easy to tell.

The experimental results and the visualization of the improvement over the 6 data sets on all evaluation metrics are shown in Table 5 and

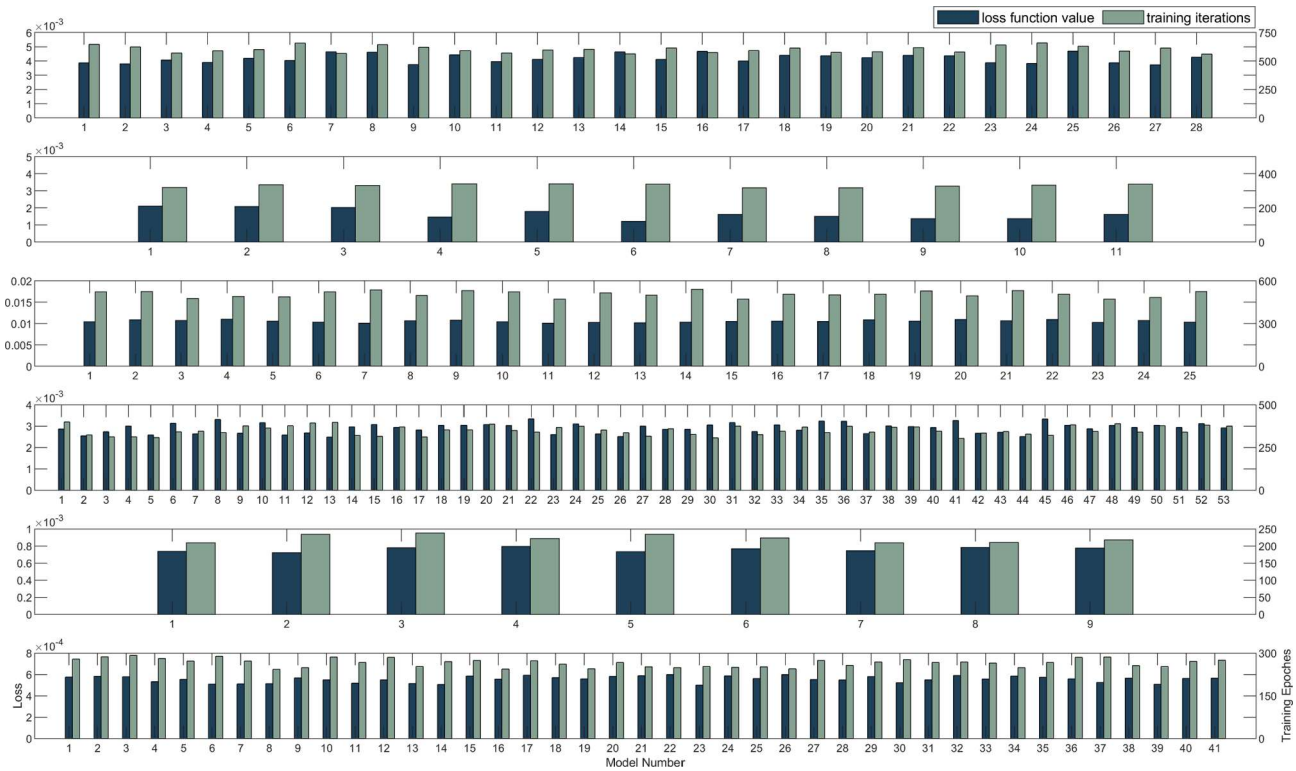

Fig. 8. Training results for the 6 datasets. From top to bottom, the 6 subfigures illustrate the training outcomes (loss function value and training epochs) for each model over the 6 datasets (TaoBao, Loan, Facebook, Eu-Core, Enron, Hypertext) respectively.

Table 5
Prediction performance on the evaluation metrics for Dataset TaoBao and Loan.

Model	Dataset TaoBao					Dataset loan			
	AUROC	Precision	Recall	F1	NDCG	AUROC	Precision	Recall	F1
T-GAN	**0.931 ± 0.0049**	**0.183 ± 0.01**	**0.64 ± 0.012**	**0.283 ± 0.0125**	**0.892 ± 0.003**	**0.76 ± 0.005**	**0.135 ± 0.003**	**0.559 ± 0.005**	**0.217 ± 0.0035**
Node2vec	0.760	0.129	0.498	0.205	0.682	0.712	0.112	0.259	0.156
GAE	0.798	0.137	0.513	0.216	0.711	0.734	0.109	0.473	0.177
CP	0.646	0.114	0.417	0.179	0.573	0.595	0.007	0.534	0.014
Gain*	14.52%	23.50%	19.50%	22.70%	20.80%	4.18%	17.13%	4.58%	18.62%
Gain**	21.16%	29.2%	25.6%	28.4%	27.14%	11.19%	43.77%	24.60%	46.82%

Gain*: compared with the best result. Gain**: compared with the worst.

Table 6
Prediction performance on the evaluation metrics for the datasets Facebook, Eu-core, Enron, Hypertext respectively.

Dataset		T-GAN	GAE	CP	E-L-D	TNE	Gain*	Gain**
Facebook	AUROC	**0.786 ± 0.0024**	0.460	0.160	0.532	0.761	3.32%	39.25%
	Precision	**0.185 ± 0.0021**	0.079	0.008	0.101	0.117	36.90%	58.80%
	Recall	0.611 ± 0.0032	0.365	0.279	0.483	**0.621**	−1.58%	28.50%
	F1	**0.284 ± 0.0026**	0.129	0.016	0.167	0.197	3.08%	55.24%
Eu-core	AUROC	**0.856 ± 0.0029**	0.54	0.21	0.711	0.813	5.33%	31.92%
	Precision	**0.168 ± 0.0019**	0.112	0.09	0.121	0.102	28.13%	36.89%
	Recall	0.587 ± **0.0008**	0.501	0.368	**0.629**	0.595	−7.42%	10.67%
	F1	**0.262 ± 0.0027**	0.183	0.145	0.203	0.174	22.40%	32.63%
Enron	AUROC	**0.916 ± 0.0017**	0.807	0.320	0.864	0.878	4.03%	21.68%
	Precision	**0.250 ± 0.009**	0.211	0.106	0.191	0.232	7.36%	25.83%
	Recall	**0.699 ± 0.0006**	0.603	0.341	0.616	0.622	11.12%	21.96%
	F1	**0.368 ± 0.021**	0.311	0.165	0.292	0.338	8.30%	24.85%
Hypertext	AUROC	**0.965 ± 0.0031**	0.821	0.473	0.957	0.920	0.81%	17.93%
	Precision	**0.160 ± 0.0247**	0.079	0.010	0.083	0.092	42.57%	58.81%
	Recall	0.485 ± 0.0025	0.472	0.290	0.479	**0.498**	−2.22%	10.51%
	F1	**0.240 ± 0.0281**	0.135	0.019	0.142	0.155	35.26%	52.94%

Gain*: compared with the best result. Gain**: compared with the worst.

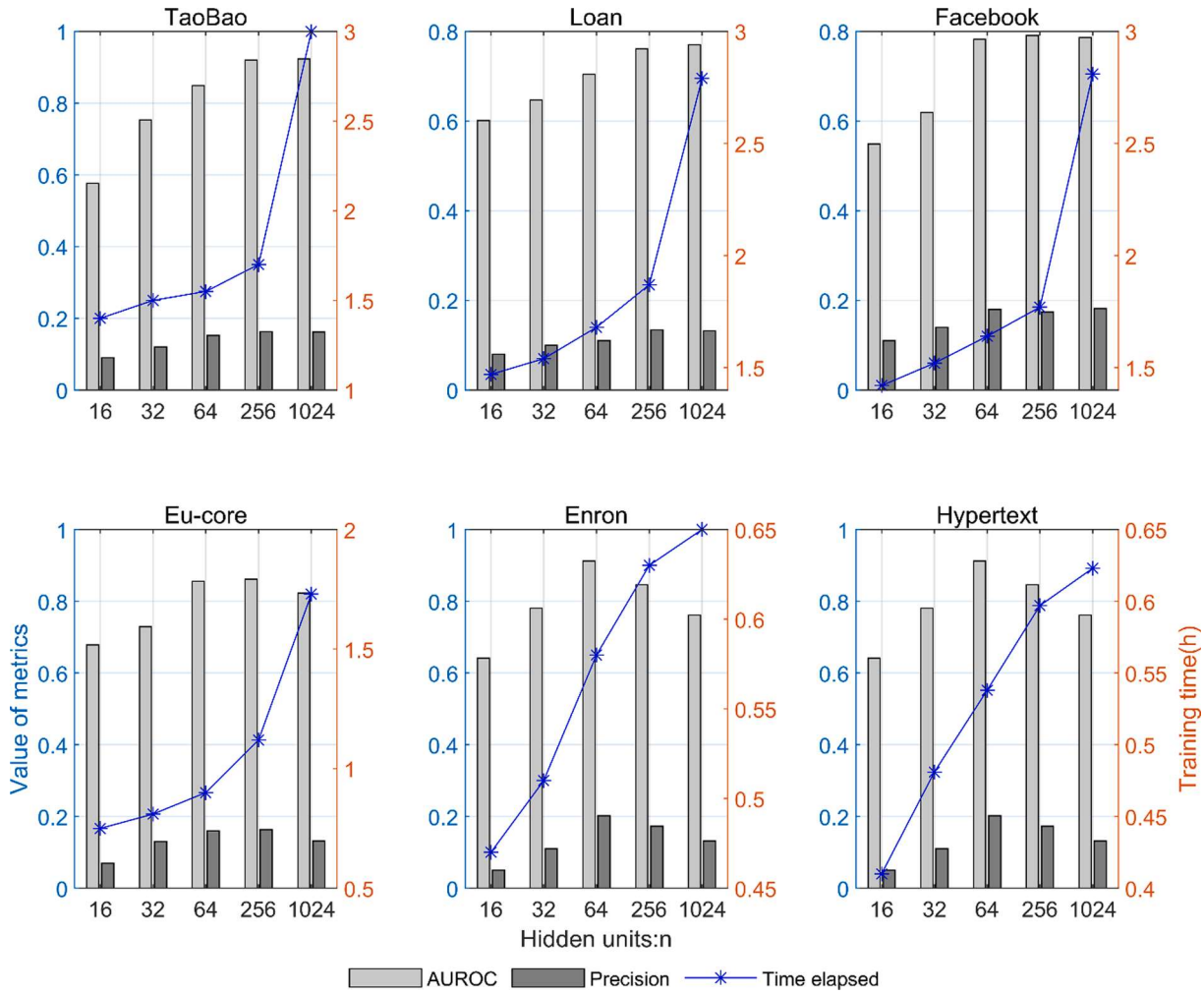

Fig. 9. The variation trend of AUROC, Precision, and Training time as a function of hidden units ***n***.

Table 6, where the best results are marked in bold.

As can be seen from Table 5, the CP method performs poorly over attributed graphs. The reason is that CP just considers the feature information of nodes and ignores the topological relationship between nodes, which leads to unsatisfactory results. Table 6 manifests that the CP method still performs the worst on non-attribute graphs which merely offer the topology information. The reason is that the CP method first aggregates time-series tensors and then decomposes them into matrices, which leads to the destruction of time series networks. It shows that the CP method is not ideal for representation learning for the tested temporal complex networks. Both Node2vec and GAE model are designed for static graphs, which use the aggregated time steps as the input, but the experimental results of GAE are marginally better than Node2vec, which can be ascribed to the superiority of GCN. According to Table 5 and Table 6, compared with the optimal benchmark algorithm, the AUROC result of T-GAN on all data sets is improved by 5%, and 4.7% improvement of Recall is obtained. The improvement effect of AUROC and Recall is marginal, whereas the improvement on Precision is significant which is improved by 25.1%. Since data set Taobao does not provide the connection information between users, in our method, we generate feature vectors for all the nodes by utilizing the user's shopping habit and location information that are used to construct the similarity matrix by calculating the cosine similarity between the user's feature vectors coupled with their address similarity. The purpose of constructing the similarity matrix for TaoBao is to assume that users with similar shopping habits and geographical location tend to buy the same category of commodities, and Table 5 also verifies the superiority of our method. As for Taobao, Loan and Enron, T-GAN outperforms other methods in all indicators. E-LSTM-D model adopts fully connected neural network as the encoder and the decoder trained with LSTM layer simultaneously, leading to that the two-dimensional topological structure of the network must be flattened into one-dimensional tensor at the input end and output end of the model. This flattening operation will result in the loss of spatial information of nodes and over-fitting. Therefore, in Table 6, the performance on Precision and F1 scores of E-LSTM-D is even slightly worse than that of GAE model that makes use of aggregated feature information. Due to the assumption of temporal smoothness in TNE model, TNE is not suitable for complex networks with frequent and dramatic changes. Compared with the deep neural network models, the node feature representation extracted by TNE is relatively shallow, so that its performance in subsequent tasks is inferior to that of T-GAN. It is worth noting that the Recall of the proposed T-GAN on dataset Facebook, EU-core, and hypertext is not as good as the optimal benchmark algorithms, whereas the corresponding Precision is improved significantly, which demonstrates that the penalty matrix in the loss function of T-GAN makes the model dare not easily misclassify positive samples.

It can be concluded from Table 5 and Table 6 that the overall performance of the proposed T-GAN is the best, which can be ascribed to that T-GAN combines the topological information of complex network with node features through adaptive graph convolution which extracts features by aggregating the characteristics of all neighbors adaptively to learn the deeper spatial feature representation for each node, and that the attention module enhances the recognition ability of the model to

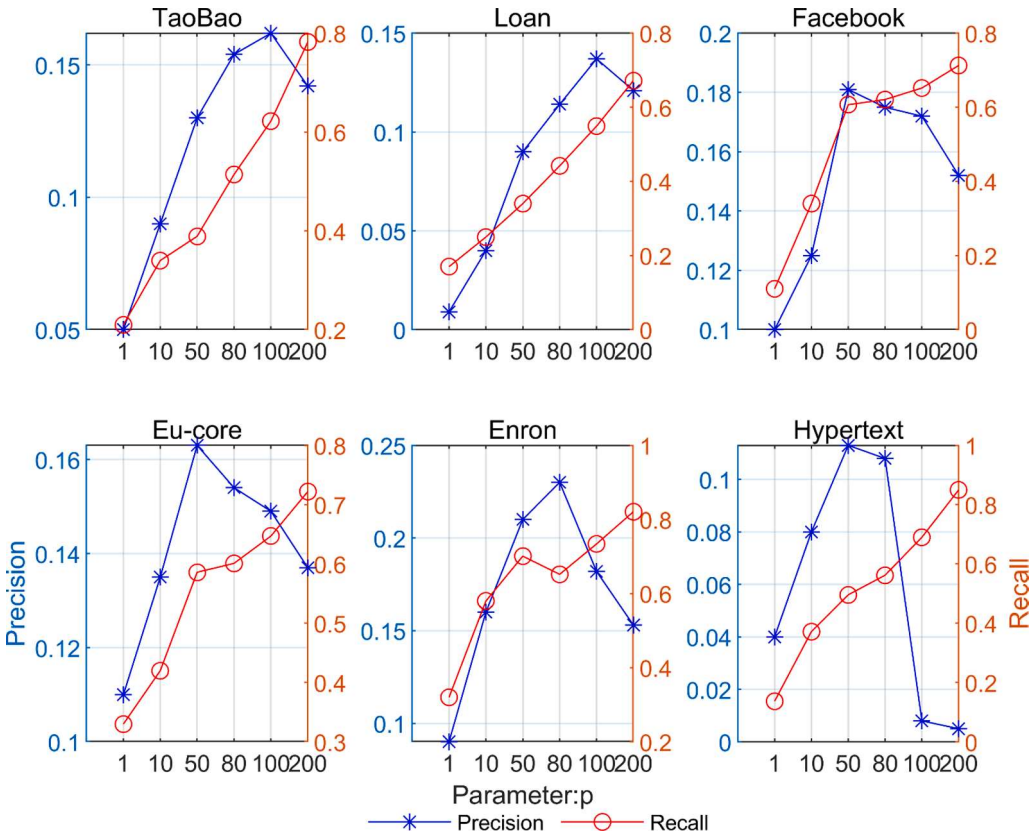

Fig. 10. Comparison of evaluation performance on Precision and Recall of the proposed model under different p.

capture the temporal evolution pattern of complex networks.

5.4. Parameter sensitivity

There are three key parameters of the T-GAN model: the embedding dimension n, the penalty coefficient ρ of the penalty matrix for positive samples, and the time step k of samples. We need to do the grid search for the 3 key parameters. Therefore, K-fold cross validation strategy is no longer used in this section. For the selection of embedded dimension n, the purpose is to achieve the balance of model effect and training time. If the dimension is set too low the network model may suffer from under-fitting, whereas the time cost will become very high and the network may risk over-fitting if the dimension is too high. The impact of n on T-GAN is also discussed in this section, and Fig. 9 shows the trend of the performance of our model as a function of n.

It can be seen from Fig. 9 that, for Enron and Hypertext, when the embedded dimension is higher than the number of nodes, the performance of the model starts to the become poor when applied to the test set as a result of over-fitting to the training set and the poor generalization ability, for the reason that the number of nodes of these two complex networks is small. For Facebook and EU-core datasets, when the embedding dimension is higher than 64, there is no significant improvement on evaluation metrics, whereas the training time shows an exponential growth, and the training time of the model shows the same trend for dataset Taobao and Loan. The reason is that increasing the embedded dimension means we need take the memory limitation into consideration and we have to reduce the batch size, resulting in the GPU losing the advantage of parallel computing and the training time increasing significantly. When embedding dimension is set to 1024, batch size can only be set to 2 for Taobao and Loan, otherwise it will lead to memory overflow. In order to balance the training time and the effect of the model, the final embedding dimensions selected for the 6 datasets (TaoBao, Loan, Facebook, Eu-core, Enron, and Hypertext) are as follows: 256, 256, 64, 64, 64, 64.

Due to the sparsity of network connections and the unbalanced samples in the node classification task, we adopt the focal loss as the loss function for the proposed model. In addition, in order to further improve the accuracy of the model, the penalty matrix is introduced to alleviate the problem. We pay more attention to the recognition of the minority, i.e., it is more important to identify the newly-formed edges in the task of topology reconstruction, and it is also important to identify who are more likely to fail to repay his loan in the node classification task. Fig. 10 shows the influence of parameter p on the Precision and Recall performance of the model. As shown in Fig. 10, with the increase of punishment, T-GAN model tend to become afraid to misclassify the minority class. However, if the punishment factor continues to expand, the model will be skewed towards classifying all samples as the minority class, because the punishment for the samples with misclassification is relatively small. For data set Taobao, we need predict users' purchasing behavior towards 5755 categories of goods, which is very sparse after the transactions being transformed into the user feature matrix. Similarly, for data set Loan, the proportion of malicious node is very small. Hence, the experimental results show that the best value of the penalty factor p is 100 for Taobao and Loan. The other four datasets are topology networks, which have more links, and therefore the sparsity problem is alleviated compared with the first two data sets. When the penalty factor is 50, the best results can be obtained for Facebook, Eu-core, Enron, and Hypertext, verifying that the optimal value of the penalty factor is closely related to the sparsity of the dataset. When $p = 1$, the loss function will degenerate into the original focal loss, whereas the prediction performance is not satisfactory, as is shown in Fig. 11, which demonstrates the effectiveness of the penalty matrix.

The time step k of each sample is the key parameter to describe the temporal feature extracted by the proposed model. Fig. 11 gives the sensitivity of T-GAN to k. It can be seen from Fig. 11 that the proposed deep neural network shows fluctuations for dataset Facebook and

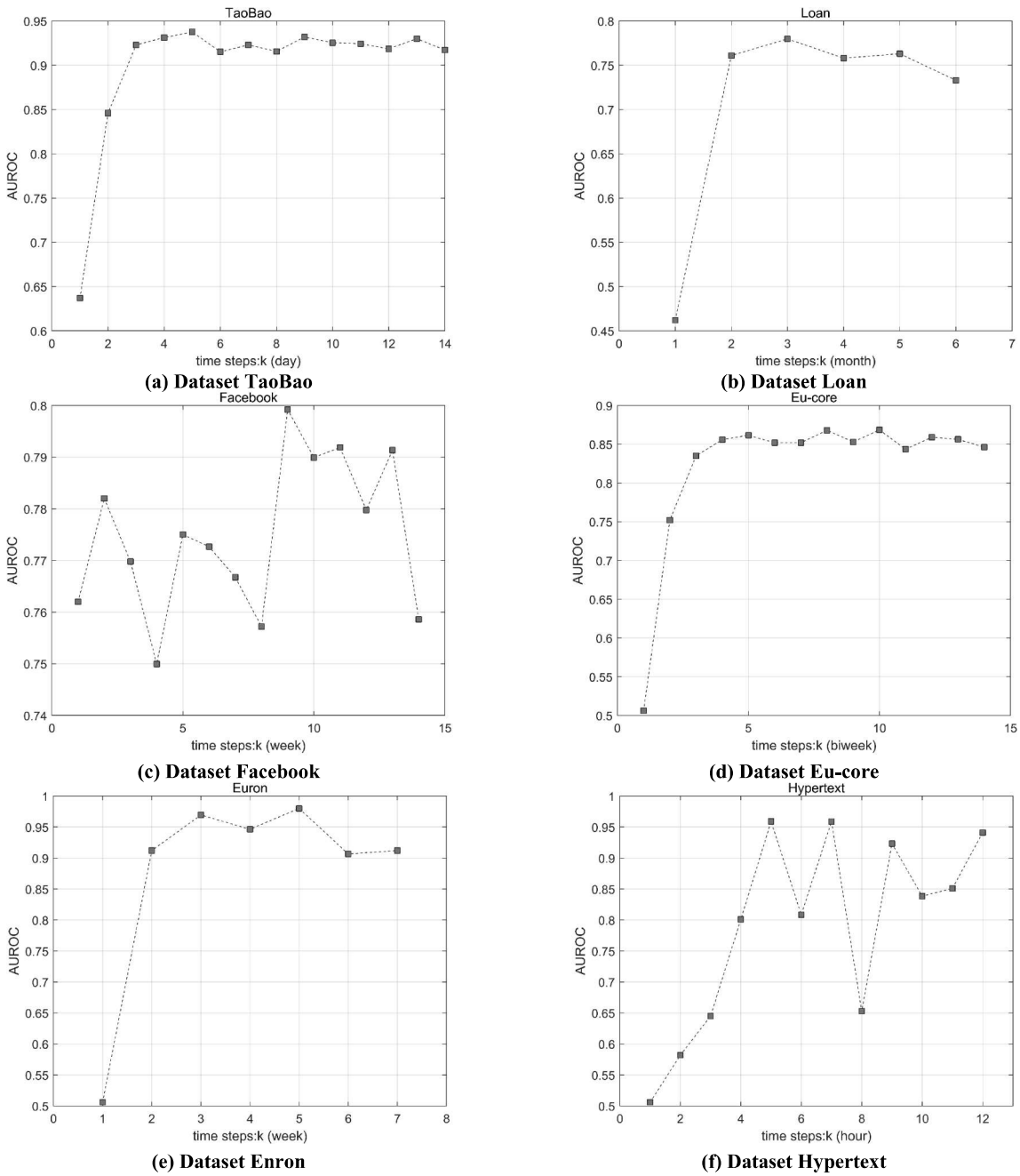

Fig. 11. The impact of k on the prediction performance of the proposed T-GAN for each dataset.

hypertext, which are complex social networks. Part of the reason is that the connections in social networks changes dramatically and randomly, and the model will be affected when extracting the features for these time-series networks. But for the other four datasets, T-GAN model shows strong robustness to the change of time step k. It can be noted that when the time step $k = 1$, the AUROC results for all data sets are very poor, because the recurrent neural network of T-GAN not only does not work, but also injects the initial state into the intermediate state, which influenced the final prediction effect. With the increase of k, the model shows strong predictive and reasoning ability for the 4 datasets (TaoBao, Loan, Eu-core, and Enron), which can capture the time evolution pattern and spatial topology features of the network.

α and β are the parameters introduced by **Algorithm 1** to determine the importance of the shopping habits of users and their geographical similarity, and we needs to discuss the impact that they have on the prediction performance of the proposed model. If $= \beta = 0$, it means that the similarity matrix A is invalid and the graph shift operator M in T-GAN will be removed, i.e., only the binary features of nodes will be the input of T-GAN. Fig. 12 illustrates the impact of the different combinations of α and β on the final predictive results of T-GAN.

As can be seen from Fig. 12, when the topological information of the complex network is ignored and we merely utilize the temporal feature matrix composed by all the nodes, i.e., $\alpha = \beta = 0$, the result of the model is not even as good as the GAE model who exploits aggregated information shown in Table 5. Although T-GAN can still make use of LSTM and nonlinear transformation to model and learn the time-series pattern of nodes after removing the topology structure, the result manifests that the absence of topology information still greatly affects the feature extraction process which in turn results in the poor performance. As the weight of α increases, the prediction results of T-GAN can be improved. When giving all attention to the shopping habits, i.e., $\alpha = 1$ and $\beta = 0$, the performance of T-GAN is better than that the one that only considers

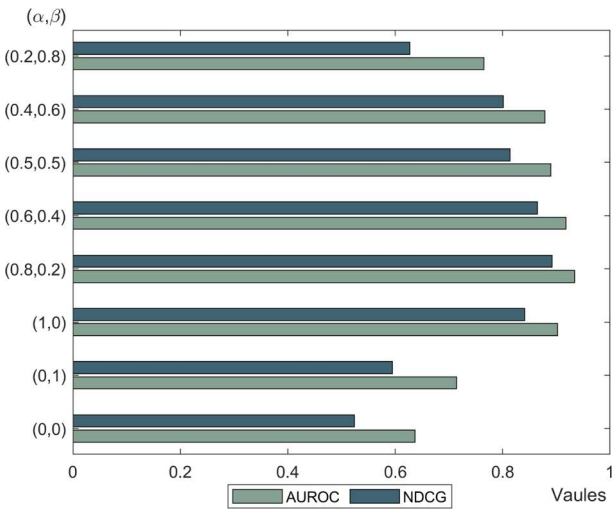

Fig. 12. Evaluation performance on AUROC and NDCG over Dataset TaoBao under different combinations of α and β.

the geographical information, which demonstrates that shopping habits makes the association among nodes stronger. However, T-GAN may gain more benefits when combining the two factors as illustrated by Fig. 12. When α is 0.8 and β is 0.2, we can get the best results. The above experimental results demonstrate that the topology information constructed by **Algorithm1** can make the adaptive graph convolution layer in T-GAN work well, which plays a significant role in extracting and aggregating the semantic features for each node in the network so as to improve the accuracy of downstream prediction tasks.

6. Additional experiments beyond graph

In this section, we will visualize the beneficial effect that the attention module shows on the proposed model through additional experiments, and evaluate the feasibility of applying the revised T-GAN to the image reconstruction and prediction task.

6.1. Visualization for the dual attention module

In order to visualize the importance of the attention block in T-GAN, and to evaluate the design of the decoder in our model, we conduct additional experiments. In this section, we do not use complex network data (because it is difficult to visualize the triples of temporal networks), and we consider the image dataset for this specific task. The developments of image processing and deep learning keep pace with each other [55,56]. Therefore, we adopt Minist [51] (handwritten digit images) for the additional experiments. We can illustrate the function of the attention mechanism and the superiority of the proposed model through the output images and visualization results for the attention coefficients of T-GAN. Because the deep learning architecture proposed in this paper is designed for graph data, in order to extend it to the reconstruction task for Euclidean data such as images, we revise the model by removing the adjacency matrix input and the adaptive graph operator M, as well as replacing the original loss function of the model with the Mean Square Error. Specifically, the remainder of T-GAN and all parameters are not altered except for the feature dimension. The model after the adjustment is named T-GAN2. Then, a certain number of images are combined into time series and will be the input for T-GAN2, and our goal is to reconstruct one of these images. For example, when a sequence of handwritten images (3, 7, 5, 9, 1) is inputted into the network, if the first image, et., the handwritten image of digit 3, is used as the label for the sequence, the purpose is to make the deep learning network focus on the first time slice and learn how to reconstruct the first image of the sequence.

Table 7
An overview of the parameter setup for each model.

Model name	Decoder	Parameters	The Number of Weights
$T-GAN2$	Nonlinear Affine Transformation Activation: ReLu	Embedding Dimension:64	**25,677**
$T-GAN2^*$			**25,628**
FullyConnectedLayer	Dense Layer Activation: ReLu	64 hidden units	352,080
LSTM		64 hidden units	268,304
RNN		64 hidden units	105,296
Conv −LSTM		Kernel $(8 \times 3 \times 3 \times 1)$+stride$(1 \times 1)$ + Valid Padding	4,243,280

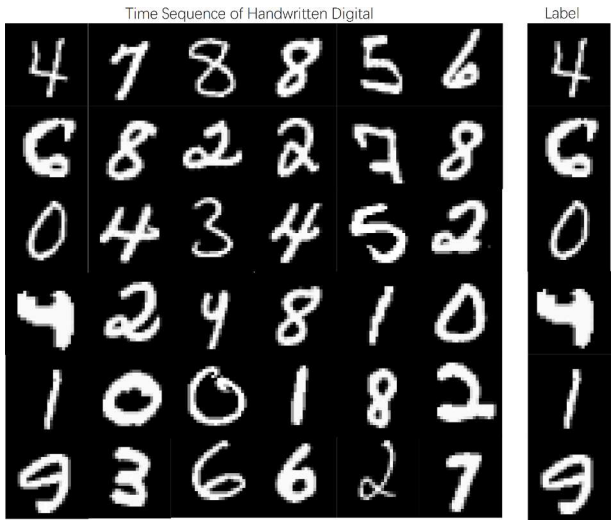

Fig. 13. Visualization of the Datasets. Each row is a sample, and the label of each sample is the first image of the sequence.

In order to provide a better comparison, a variation of T-GAN2 (named T-GAN2*) is used for comparison whose attention block has been dropped. T-GAN2* only outputs the hidden state of the last time step as the embedding for the input image sequence. Moreover, the fully connected neural network model, LSTM, Recurrent Neural Network (RNN), and Conv-LSTM [52] are the compared. The experimental parameters of the six neural network models are listed in Table 7.

The sample composition of the experiment in this section is as follows: 60,000 handwritten digital images of dataset Minist are transformed into 10,000 training samples and each sample is a sequence which contains 6 images,

one of which will be used as the label of the training sample. Among the 10,000 samples, 6000 samples are used for training, 2000 samples constitute the verification set, and the remaining 2000 samples will be used as the test set.

We train every model for only 5 iterations. For LSTM, RNN and the fully connected neural network model which accept one-dimensional temporal tensors as their input and output, it is necessary for us to flatten each image (28×28) into a one-dimensional tensor (1×784) for these models. However, we do not need flatten the image for T-GAN2, T-GAN2* and Conv-LSTM, because these models are meant to perform the learning task for high-dimensional tensors. The input samples of the fully connected neural network model need to be concatenated in the time axis, and the dimension of each sample after concatenating is 1×4704.

Fig. 13 gives an example for the sample composition of our test set, where each sample (each row) is a time series, and the first picture of the sequence is the corresponding label (in our experiment, the first image

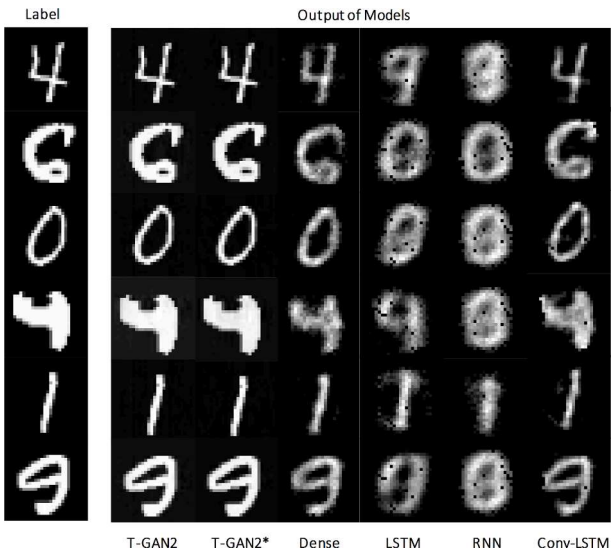

Fig. 14. Visualization of the image reconstruction task carried out by the 6 models.

of each sample is selected as the label).

After 5 training iterations, the reconstruction images outputted by the 6 models are presented in Fig. 14. From the visualization results, RNN and LSTM performed the worst, and they almost did not capture the evolution pattern of the input time series after 5 training epochs. The reason is that the original data is flattened into one-dimensional vector, which destroyed the spatial structure of data, and the problem of gradient disappearance in RNN leads to the poor effect of the model. The reconstructed image of the full connected neural network model looks fuzzy, although the contour is clearer than that generated by RNN and LSTM, accompanied by the problem of destroying the spatial structure of data. Conv-LSTM is the method that needs the most parameters and the longest training time among the six models, but the results are still unsatisfactory. In fact, the result obtained by Conv-LSTM get even worse than that of the fully connected neural network model. As shown in Fig. 14, T-GAN2 and T-GAN2* outperform all the other comparable approaches, and there is nearly no visual difference between the outputs of T-GAN2 (T-GAN2*) and the corresponding labels of samples. The visualization results of the attention coefficients and corresponding loss function value are given in Fig. 15. It can be seen from Fig. 15 (a) that the attention block in our model plays a significant role in capturing the evolutionary pattern for time series, where the normalized attention coefficient of the first time step is much larger than that of other time steps, which is consistent with the composition of samples and the expectation of the experiment. Fig. 15 (b) shows that the mean square error loss of T-GAN2 is only one third of that of T-GAN2* whose attention block was dropped out for comparison, which verifies the effectiveness of the attention mechanism of the proposed model. Table 4 and Fig. 15 (b) collaboratively demonstrate that compared with other models, our method needs the least amount of training parameters and T-GAN2 achieves the minimum error under the condition of the least parameters. The visualization results in Fig. 14 provide a convincing evidence that supports the superiority of the proposed deep learning architecture. This supplementary experiment also shows the advantages of the decoder in our model: it can not only reduce the amount of training parameters greatly, but also has prominent decoding capability. The experimental results of T-GAN2* are still better than the other four models even if the attention block is removed, manifesting that the proposed model has excellent capability of capturing the evolutionary pattern of time series.

6.2. Visualization for situation assessment in sensor networks

In this section, we apply the proposed model to analyze sensor network traffic to assess network security situation. The number of packets sent and received by nodes in the network that does not suffer from any attack should be in a state of dynamic equilibrium [53]. Hence, the traffic flows of nodes can present a balanced trend. In order to simulate the real transmission process of sensor networks, in this experiment, we use KDD99 network attack dataset [54]. Specifically, we define the security feature map of a network as a matrix M_t composed by the Fourier amplitude spectrums of the traffic flows in both directions (sending and receiving) of all nodes in a fixed sampling interval, for the sake of visual presentation. The Fourier transform results of the traffic flows in both directions (sending and receiving) of each node will be distributed symmetrically in the corresponding row of M_t, with the center of M_t as the symmetry axis. In this way, areas which are closer to the center of the feature map M_t corresponds to higher frequency components of the data flow spectrum, and thus the left edge/right edge of M_t indicate DC components. M_t can be visualized as a gray image (the amplitude values of all spectral components are normalized to 0–255), to intuitively display the experimental results.

Because the KDD99 dataset is too large, we only selected the one-hour logs among 200 nodes. For each node, the sampling interval is 0.05 s, and the frame length is 5 s. The traffic data recorded in each frame will be used to perform 100-point FFT (Fast Fourier Transform, with the Discrete Fourier Transform shown in (20)), and thus 720

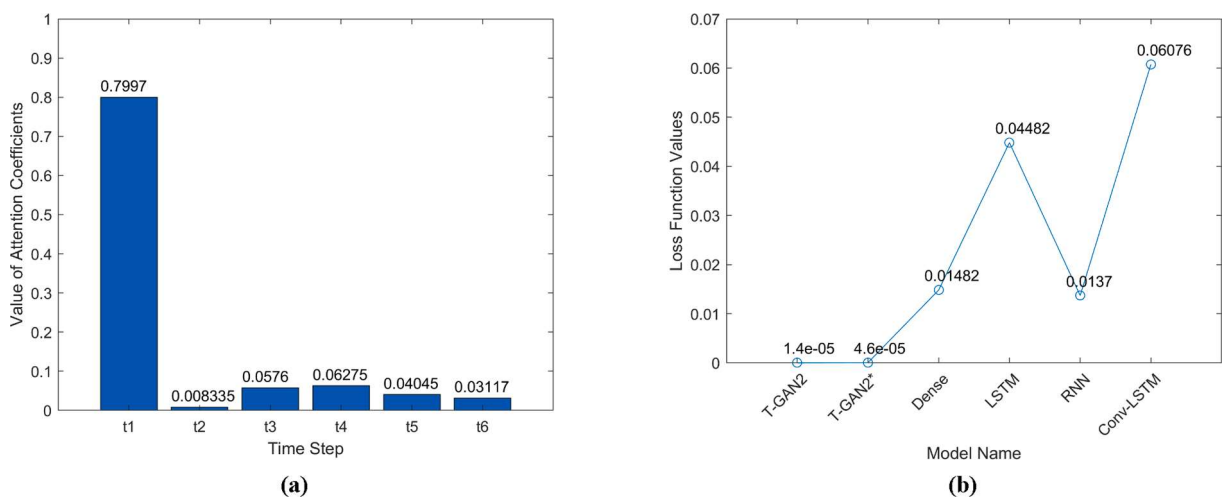

Fig. 15. (a) Normalized attention coefficient value of each time step in T-GAN. (b) Loss values of each model after being trained for 5 iterations.

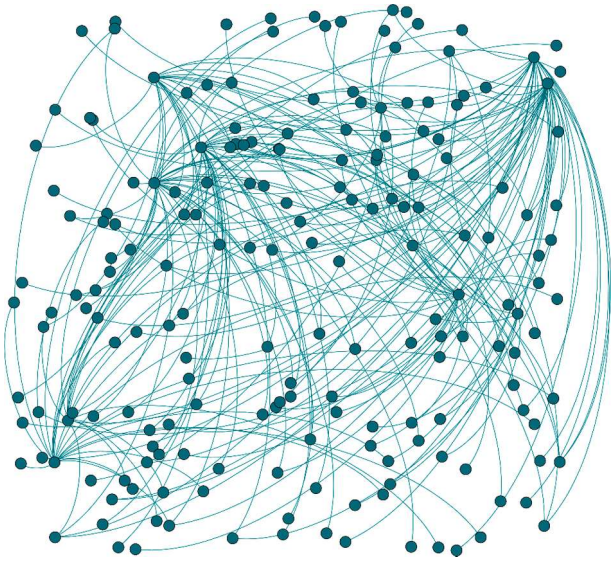

Fig. 16. Visualization of the network topology.

security feature maps are constructed. The reason that we perform 100-point FFT when preprocessing the original data is that the sampling frequency is 20 Hz and the data frame length is 100.

$$X(k) = DFT(x(n)) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi}{N} nk} \tag{20}$$

The adjacency matrix of the network and feature maps are used to train the proposed model, aiming at predicting the future security situation of the network. Time step k is set to 10, and the size of each feature map is 200×200. The topology information of nodes selected for the experiment is shown in Fig. 16, and an example of the comparison of security feature maps between 2 different networks is illustrated in Fig. 17.

It can be seen from Fig. 17(a) that the security feature map of the network that has not been attacked shows a significant bilateral symmetry, and the phenomenon that DC component of traffic flows accounts for large proportion implies steady data flows in healthy networks. On the contrary, the symmetry of the feature map of the network which suffered from attacks is obviously destroyed, and the high frequency regions of the feature map become relatively active, as shown in Fig. 17 (b).

720 feature maps can constitute 710 time-series samples, and each sample contains 10 consecutive feature maps and a static adjacency matrix. 70% of the total samples are used for training, and the remaining samples are used for evaluating the performance of T-GAN. Fig. 18 and Fig. 19 show the comparison between the prediction results of T-GAN

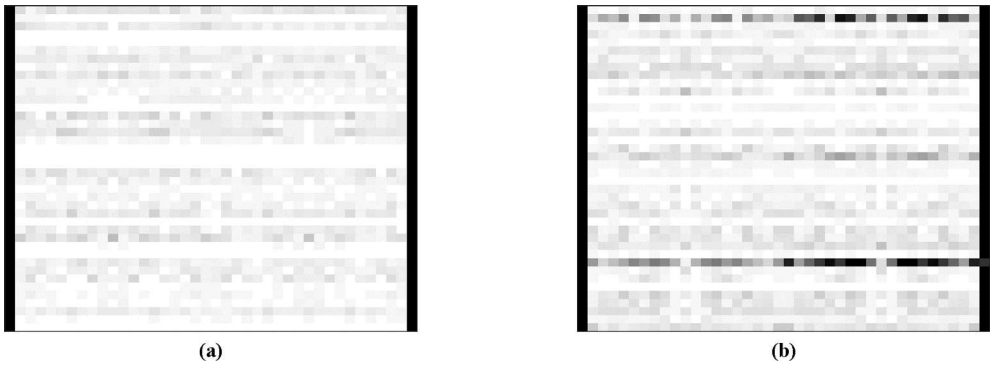

Fig. 17. Visualization for the comparison of the magnitude spectrums of traffic flows between (a) the normal network and (b) the network under attack. Pixels with deeper color refer to larger values, i.e., the white pixel indicates 0 and the black 255.

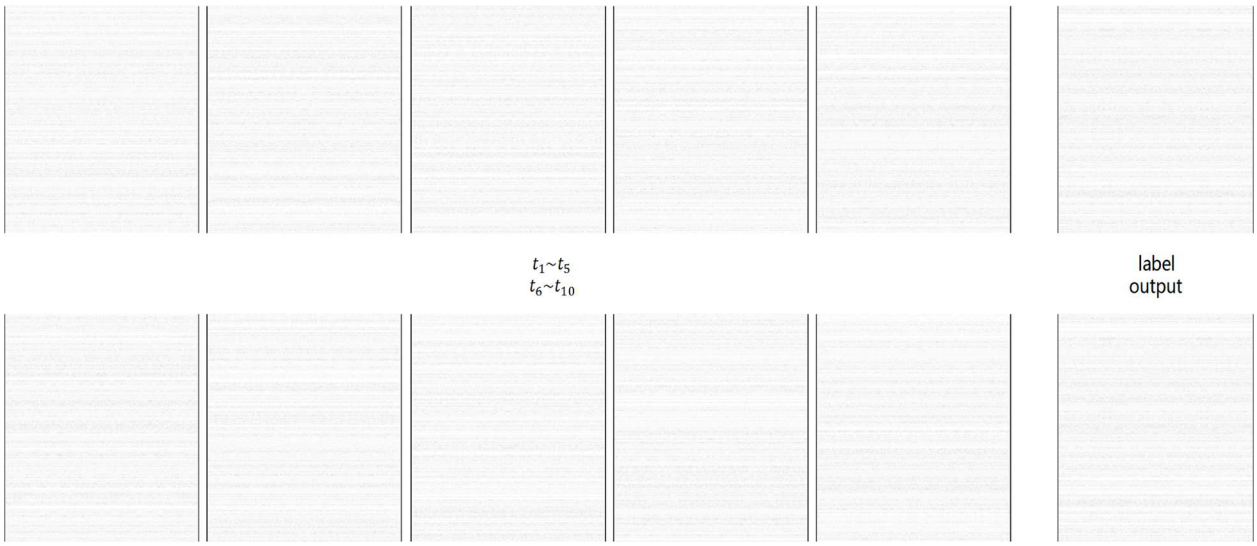

Fig. 18. The comparison between the predictive result of T-GAN and ground truth of a time-series feature map, in which no attack was deployed.

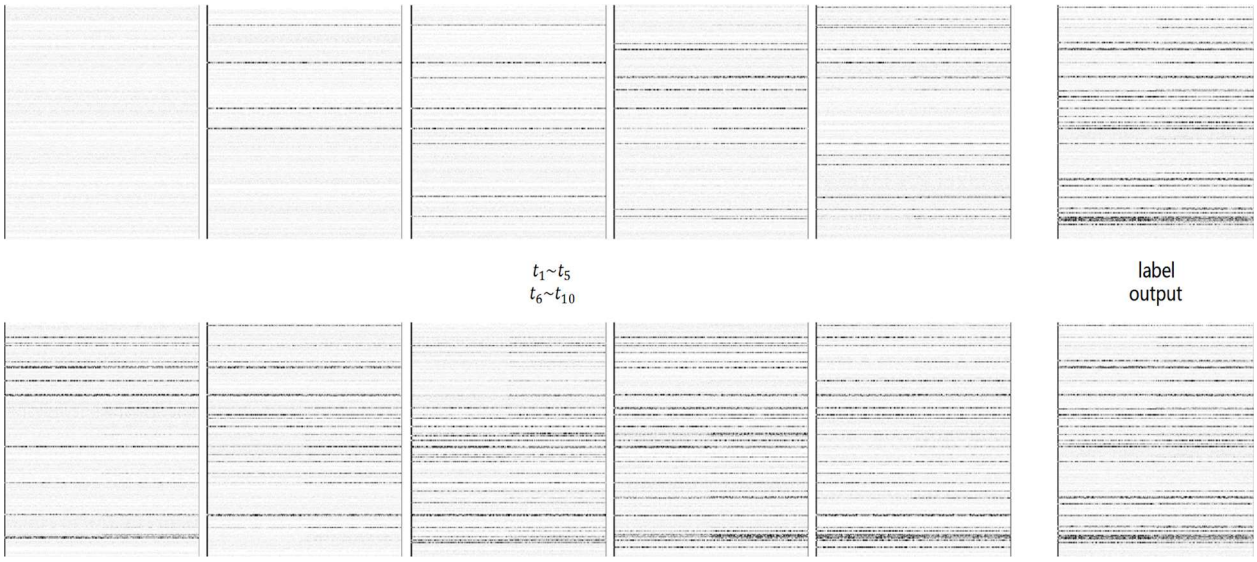

Fig. 19. Visualization for the security feature map prediction in a temporal network (a sample in the test set), which is under attack.

and the actual security feature map (2 samples in the test set).

In Fig. 18, because the nodes of the sample are in a safe state in 10 consecutive sampling periods, we can see that the security feature map present a symmetrical distribution trend of spectrum components. Besides, there is almost no visual difference between the output of T-GAN model and the label of the sample with the loss function value of T-GAN being 0.0013. In Fig. 19, the nodes of the network had not been attacked initially, but they were attacked later in subsequent sampling periods, resulting in the destruction of the symmetry of the security feature map and the abnormal activity of the high-frequency area. It can be concluded from the security situation maps in Fig. 19 that the network is under attack, and the situation is getting more and more severe. The output image of T-GAN is nearly identical to the ground truth, which accurately evaluates the security situation of the temporal network at the next moment, with the loss function value being 0.0021.

The experimental results manifest that the proposed deep learning model T-GAN has excellent abilities of temporal feature extraction and prediction for data with graph structures, and can accurately capture the evolution pattern of network security situation when applied to network collapse assessment. Visualization results in this section demonstrate the effectiveness of T-GAN and also verify its high scalability.

7. Conclusion

Prediction of properties and events of complex networks is of increasing importance but has not been well studied. In this paper, we investigated the prediction of properties and events of complex networks with temporally changing features, and proposed a novel deep learning based model T-GAN for the challenging prediction task. In the T-GAN network model, adaptive graph convolution network was proposed and integrated with LSTM to extract the spatial characteristics of nodes and learn the temporal evolution pattern of the complex networks. A dual attention block was used in T-GAN to embed the sequence of network graph triples into vector space. With efficient use of both spatial topology information, node properties and temporal features of complex networks, the proposed network model showed excellent performance over existing methods in three prediction tasks for real applications, including node classification, feature forecasting and topology prediction. Additional experiments also demonstrated the applicability of the proposed model to image fields and network survivability assessment.

As T-GAN was proposed to embed temporal complex networks and perform the evolution prediction task, and complex networks usually have high-dimensional topology information and features, there could be high memory demand and dramatically increase of training time. In our future work we will investigate matrix factorization, more efficient and optimization of models and training methods.

CRediT authorship contribution statement

Ru Huang: Conceptualization, Methodology. **Lei Ma:** Software, Data curation, Writing - original draft. **Jianhua He:** Visualization, Investigation. **Xiaoli Chu:** Supervision, Validation, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by National Natural Science Foundation of China, grant number 61673178 and 61922063, Natural Science Foundation of Shanghai, No.20ZR1413800. Our work has also received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 824019 (project COSAFE) and the Marie Sklodowska-Curie grant agreement No 101022280 (project VESAFE). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this paper.

References

[1] Y. Gandica, I. Bonalde, J.L. Cabrera, Complex propagation on directed small world networks, Phys. A Stat. Mech. Its Appl. 389 (2010) 4172–4176, https://doi.org/10.1016/j.physa.2010.05.043.

[2] Z. Xu, K. Li, M. Sun, X. Fu, Interaction between epidemic spread and collective behavior in scale-free networks with community structure, J. Theor. Biol. 462 (2019) 122–133, https://doi.org/10.1016/j.jtbi.2018.11.003.

[3] M. Li, J. Liu, P. Wu, X. Teng, Evolutionary network embedding preserving both local proximity and community structure, IEEE Trans. Evol. Comput. 24 (2020) 523–535, https://doi.org/10.1109/TEVC.2019.2937455.

[4] H. Jiang, Z. Liu, C. Liu, Y. Su, X. Zhang, Community detection in complex networks with an ambiguous structure using central node based link prediction, Knowledge-Based Syst. 195 (2020), 105626, https://doi.org/10.1016/j.knosys.2020.105626.

[5] M. Gao, L. Chen, B. Li, Y. Li, W. Liu, Y. Xu, Projection-based link prediction in a bipartite network, Inf. Sci. (Ny) 376 (2017) 158–171, https://doi.org/10.1016/j.ins.2016.10.015.

[6] A. Yessembayev, D. Sarkar, F. Sikder, Detection of good and bad sensor nodes in the presence of malicious attacks and its application to data aggregation, IEEE Trans. Signal Inf. Process. over Networks. 4 (2018) 549–563, https://doi.org/10.1109/TSIPN.2018.2790164.

[7] I. Khalil, S. Bagchi, Stealthy attacks in wireless ad hoc networks: detection and countermeasure, IEEE Trans. Mob. Comput. 10 (2011) 1096–1112, https://doi.org/10.1109/TMC.2010.249.

[8] K. Pongaliur, L. Xiao, A.X. Liu, CENDA: Camouflage event based malicious node detection architecture, IEEE 6th Int Conf. Mob. Adhoc Sens. Syst. 2009 (2009) 848–855, https://doi.org/10.1109/MOBHOC.2009.5337045.

[9] Q. Li, X.-M. Wu, H. Liu, X. Zhang, Z. Guan, Label efficient semi-supervised learning via graph filtering, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2019: pp. 9574–9583. https://doi.org/10.1109/CVPR.2019.00981.

[10] Q. Li, Z. Han, X.-M. Wu, Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning., in: Proc. Thirty-Second AAAI Conf. Artif. Intell. (AAAI-18), 30th Innov. Appl. Artif. Intell. (IAAI-18), 8th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI-18), New Orleans, Louisiana, USA, Febr. 2-7, 2018, 2018: pp. 3538–3545. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16098.

[11] S. Pan, R. Hu, S.-F. Fung, G. Long, J. Jiang, C. Zhang, Learning graph embedding with adversarial training methods, IEEE Trans. Cybern. 50 (2020) 2475–2487, https://doi.org/10.1109/TCYB.2019.2932096.

[12] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2014: pp. 701–710. https://doi.org/10.1145/2623330.2623732.

[13] S. Ji, N. Satish, S. Li, P.K. Dubey, Parallelizing Word2Vec in shared and distributed memory, IEEE Trans. Parallel Distrib. Syst. 30 (2019) 2090–2100, https://doi.org/10.1109/TPDS.2019.2904058.

[14] F. Fouss, A. Pirotte, J. Renders, M. Saerens, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE Trans. Knowl. Data Eng. 19 (2007) 355–369, https://doi.org/10.1109/TKDE.2007.46.

[15] K.W. Church, Word2Vec, Nat. Lang. Eng. 23 (2017) 155–162, https://doi.org/10.1017/S1351324916000334.

[16] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, LINE: Large-scale Information Network Embedding. Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015, (2015) 1067–1077. https://doi.org/10.1145/2736277.2741093.

[17] A. Grover, J. Leskovec, node2vec: Scalable Feature Learning for Networks., in: Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, San Fr. CA, USA, August 13-17, 2016, 2016: pp. 855–864. https://doi.org/10.1145/2939672.2939754.

[18] J. Wu, Y. Sha, B. Jiang, J. Tan, DSINE: Deep Structural Influence Learning via Network Embedding., in: Thirty-Third AAAI Conf. Artif. Intell. AAAI 2019, Thirty-First Innov. Appl. Artif. Intell. Conf. IAAI 2019, Ninth AAAI Symp. Educ. Adv. Artif. Intell. EAAI 2019, Honolulu, Hawaii, USA, January 27 - Febr. 1, 2019., 2019: pp. 10065–10066. https://doi.org/10.1609/aaai.v33i01.330110065.

[19] X. Li, N. Du, H. Li, K. Li, J. Gao, A. Zhang, A Deep Learning Approach to Link Prediction in Dynamic Networks., in: Proc. 2014 SIAM Int. Conf. Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014, 2014: pp. 289–297. https://doi.org/10.1137/1.9781611973440.33.

[20] Z. Zhu, J. Zhang, J. Zou, C. Deng, Multi-kernel Gaussian process latent variable regression model for high-dimensional sequential data modeling, Neurocomputing. 348 (2019) 3–15, https://doi.org/10.1016/j.neucom.2018.07.082.

[21] Z. Ghahramani, G.E. Hinton, Variational learning for switching state-space models, Neural Comput. 12 (2000) 831–864, https://doi.org/10.1162/089976600300015619.

[22] L. Zhou, Y. Yang, X. Ren, F. Wu, Y. Zhuang, Dynamic Network Embedding by Modeling Triadic Closure Process., in: Proc. Thirty-Second AAAI Conf. Artif. Intell. (AAAI-18), 30th Innov. Appl. Artif. Intell. (IAAI-18), 8th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI-18), New Orleans, Louisiana, USA, Febr. 2-7, 2018, 2018: pp. 571–578. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16572.

[23] N.M.A. Ibrahim, L. Chen, Link prediction in dynamic social networks by integrating different types of information, Appl. Intell. 42 (2015) 738–750, https://doi.org/10.1007/s10489-014-0631-0.

[24] X. Li, Y. Li, Y. Shang, L. Tong, F. Fang, P. Yin, J. Cheng, J. Li, DDNE: Discriminative Distance Metric Learning for Network Embedding., in: Comput. Sci. - ICCS 2020 - 20th Int. Conf. Amsterdam, Netherlands, June 3-5, 2020, Proceedings, Part I, 2020: pp. 568–581. https://doi.org/10.1007/978-3-030-50371-0_42.

[25] N. Kamra, P. Goyal, X. He, Y. Liu, DynGEM: deep embedding method for dynamic graphs, IJCAI Int. Work. Represent. Learn. Graphs (2017).

[26] J. Chen, J. Zhang, X. Xu, C. Fu, D. Zhang, Q. Zhang, Q. Xuan, E-LSTM-D: A deep learning framework for dynamic network link prediction, IEEE Trans. Syst. Man Cybern. Syst. (2019) 1–14, https://doi.org/10.1109/TSMC.2019.2932913.

[27] L. Zhu, D. Guo, J. Yin, G.V. Steeg, A. Galstyan, Scalable temporal latent space inference for link prediction in dynamic social networks, IEEE Trans. Knowl. Data Eng. 28 (2016) 2765–2777, https://doi.org/10.1109/TKDE.2016.2591009.

[28] D.M. Dunlavy, T.G. Kolda, E. Acar, Temporal Link Prediction Using Matrix and Tensor Factorizations., ACM Trans. Knowl. Discov. Data. 5 (2011) 10:1-10:27. https://doi.org/10.1145/1921632.1921636.

[29] M. Niepert, M. Ahmed, K. Kutzkov, Learning Convolutional Neural Networks for Graphs., in: Proc. 33nd Int. Conf. Mach. Learn. ICML 2016, New York City, NY, USA, June 19-24, 2016, 2016: pp. 2014–2023. http://proceedings.mlr.press/v48/niepert16.html.

[30] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, M.M. Bronstein, Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs., in: 2017 IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, 2017: pp. 5425–5434. https://doi.org/10.1109/CVPR.2017.576.

[31] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Networks. 20 (2009) 61–80, https://doi.org/10.1109/TNN.2008.2005605.

[32] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral Networks and Locally Connected Networks on Graphs., in: 2nd Int. Conf. Learn. Represent. ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conf. Track Proc., 2014. http://arxiv.org/abs/1312.6203.

[33] T.N. Kipf, M. Welling, Semi-Supervised Classification with Graph Convolutional Networks., in: 5th Int. Conf. Learn. Represent. ICLR 2017, Toulon, Fr. April 24-26, 2017, Conf. Track Proc., 2017. https://openreview.net/forum?id=SJU4ayYgl.

[34] W.L. Hamilton, Z. Ying, J. Leskovec, Inductive Representation Learning on Large Graphs., in: Adv. Neural Inf. Process. Syst. 30 Annu. Conf. Neural Inf. Process. Syst. 2017, December 4-9, 2017, Long Beach, CA, USA, 2017: pp. 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

[35] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph Attention Networks., in: 6th Int. Conf. Learn. Represent. ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conf. Track Proc., 2018. https://openreview.net/forum?id=rJXMpikCZ.

[36] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, R. Zemel, Neural relational inference for Interacting systems, in: 35th Int. Conf. Mach. Learn. ICML 2018, 2018: pp. 4209–4225. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85057233216&partnerID=40&md5=7fbcaff24ae683b92ca857a18646e0ac.

[37] J. Du, S. Zhang, G. Wu, J.M.F. Moura, S. Kar, Topology adaptive graph convolutional networks., in: 6th Int. Conf. Learn. Represent. ICLR 2018, 2018.

[38] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then Propagate: Graph Neural Networks meet Personalized PageRank., in: 7th Int. Conf. Learn. Represent. ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019. https://openreview.net/forum?id=H1gL-2A9Ym.

[39] F. Wu, A.H. Souza Jr., T. Zhang, C. Fifty, T. Yu, K.Q. Weinberger, Simplifying Graph Convolutional Networks., in: Proc. 36th Int. Conf. Mach. Learn. ICML 2019, 9-15 June 2019, Long Beach, California, USA, 2019: pp. 6861–6871. http://proceedings.mlr.press/v97/wu19e.html.

[40] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, H. Li, T-GCN: A temporal graph convolutional network for traffic prediction, IEEE Trans. Intell. Transp. Syst. 21 (2020) 3848–3858, https://doi.org/10.1109/TITS.2019.2935152.

[41] K. Lei, M. Qin, B. Bai, G. Zhang, M. Yang, GCN-GAN: A Non-linear Temporal Link Prediction Model for Weighted Dynamic Networks., in: 2019 IEEE Conf. Comput. Commun. INFOCOM 2019, Paris, Fr. April 29 - May 2, 2019, 2019: pp. 388–396. https://doi.org/10.1109/INFOCOM.2019.8737631.

[42] F. Zou, L. Shen, Z. Jie, W. Zhang, W. Liu, A sufficient condition for convergences of adam and rmsprop, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2019: pp. 11119–11127. https://doi.org/10.1109/CVPR.2019.01138.

[43] Github, Code repository. https://github.com/malei666666/T_GAN, 2021 (accessed 31 March).

[44] Alibaba, TaoBao, 2020. https://tianchi.aliyun.com/competition/entrance/231522/information, 2021 (accessed 31 March).

[45] PPD, Loan, 2015. https://ai.ppdai.com/mirror/showCompetitionRisk, 2021 (accessed 31 March).

[46] A. Rossi, Nesreen K. Ahmed, Facebook, 2015. http://networkrepository.com/socfb.php, 2021 (accessed 31 March).

[47] A. Paranjape, A.R. Benson, J. Leskovec, Motifs in Temporal Networks., in: Proc. Tenth ACM Int. Conf. Web Search Data Mining, WSDM 2017, Cambridge, United Kingdom, Febr. 6-10, 2017, 2017: pp. 601–610. https://doi.org/10.1145/3018661.3018731.

[48] A.R. Benson, R. Abebe, M.T. Schaub, A. Jadbabaie, J.M. Kleinberg, Simplicial closure and higher-order link prediction, Proc. Natl. Acad. Sci. USA 115 (2018) E11221–E11230, https://doi.org/10.1073/pnas.1800683115.

[49] SocioPatterns, Hypertext, 2011. http://networkrepository.com/ia-infect-hyper.php, 2021 (accessed 31 March).

[50] J. Davis, M. Goadrich, The relationship between Precision-Recall and ROC curves., in: Mach. Learn. Proc. Twenty-Third Int. Conf. (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006, 2006: pp. 233–240. https://doi.org/10.1145/1143844.1143874.

[51] Y. LeCun, Minist, 1998. http://yann.lecun.com/exdb/mnist, 2021 (accessed 31 March).

[52] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W. Woo, Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting., in: Adv. Neural Inf. Process. Syst. 28 Annu. Conf. Neural Inf. Process. Syst. 2015, December 7-12, 2015, Montr. Quebec, Canada, 2015: pp. 802–810. https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html.

[53] X. Chen, K. Makki, K. Yen, N. Pissinou, Sensor network security: a survey, IEEE Commun. Surv. Tutorials. 11 (2009) 52–73, https://doi.org/10.1109/SURV.2009.090205.

[54] S. Hettich, S.D. Bay, The UCI KDD Archive, 1999. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html , 2021 (accessed 31 March).
[55] Z. Gao, G. Zhai, H. Deng, X. Yang, Extended geometric models for stereoscopic 3D with vertical screen disparity, Displays 65 (2020), 101972, https://doi.org/10.1016/j.displa.2020.101972.
[56] J. You, U. Reiter, M.M. Hannuksela, M. Gabbouj, A. Perkis, Perceptual-based quality assessment for audio-visual services: A survey, Signal Process. Image Commun 25 (7) (2010) 482–501, https://doi.org/10.1016/j.image.2010.02.002.

Ru Huang received his B.S. degree from Nanjing University, Nanjing, China, in 1999 and his Ph.D. degree in circuit and system from Shanghai Jiao Tong University, Shanghai, China, in 2008. He was a visiting scholar in the University of Wisconsin-Madison, WI, USA, from March 31, 2015 to March 30, 2016. He is currently an associate professor of electronics and communication engineering at East China University of Science and Technology, Shanghai, China. His current research interests include wireless sensor networks, complex networks and software-defined networks. (huangrabbit@ecust.edu.cn)

Lei Ma received his B.S. degree in information engineering from the University of East China University of Science and Technology, in 2018.And he is now pursing his M.S. degree, majoring in Information and Communication Engineering at East China University of Science and Technology. And his Research directions comprises complex networks, deep learning, and data mining, specializing at building deep learning model for data mining tasks in complex networks. (y30180622@mail.ecust.edu.cn)

Jianhua He received the Ph.D. degree from Nanyang Technological University, Singapore, in 2002. He is a Reader at University of Essex, U.K. His research interests include 5G and beyond technologies, ADAS, Internet of Things, machine learning and big data analytics. He has authored or co-authored over 100 technical papers in major international journals and conferences. Dr. He is Editor or Guest Editor of several international journals, including Wireless Communications and Mobile Computing, IEEE Access, International Journals of Communication Systems, International Journal of Distributed Sensor Networks, and KSII Transactions on Internet and Information Systems. He acted as TPC chair for international conference ICAIT'2010 and ICONI'2010. He is a TPC member of many international conferences including IEEE Globecom and IEEE ICC. (j.he@essex.ac.uk)

Xiaoli Chu received her B.Eng. degree in electronic and information engineering from Xi'an Jiao Tong University in 2001 and her Ph.D. degree in electrical and electronic engineering from Hong Kong University of Science and Technology in 2005. From September 2005 to April 2012, she was with the Centre for Telecommunications Research at King's College London. Her research interests include modelling, analysis, and algorithm design for improving the performance and efficiency of wireless communication systems. (x.chu@sheffield.ac.uk)