Hindawi
Security and Communication Networks
Volume 2022, Article ID 6945397, 11 pages
https://doi.org/10.1155/2022/6945397

Research Article

DEDGCN: Dual Evolving Dynamic Graph Convolutional Network

Fengzhe Zhong,[1] **Yan Liu**,[1] **Lian Liu**,[2] **Guangsheng Zhang**,[2] **and Shunran Duan**[1]

[1]*Henan Key Laboratory of Cyberspace Situation Awareness, Zhengzhou, Henan 450001, China*
[2]*Investigation Technology Center PLCMC, Beijing 100000, China*

Correspondence should be addressed to Yan Liu; ms_liuyan@aliyun.com

Received 25 November 2021; Accepted 18 April 2022; Published 10 May 2022

Academic Editor: Wei Wang

With the wide application of graph data in many fields, the research of graph representation learning technology has become the focus of scholars' attention. Especially, dynamic graph representation learning is an important part of solving the problem of change graph in reality. On the one hand, most dynamic graph representation methods focus either on graph structure changes or node embedding changes, ignoring the internal relationship. On the other hand, most dynamic graph neural networks require learn node embeddings from specific tasks, resulting in poor universality of node embeddings and cannot be used in unsupervised tasks. Hence, Dual Evolving Dynamic Graph Convolutional Network (DEDGCN) was proposed to solve the above problems. DEDGCN uses the recurrent neural network to push the evolvement of GCN and nodes, from which it can extract the structural features of dynamic graph and learns the stability features of nodes, respectively, forming an adaptive dynamic graph convolution network. DEDGCN can be classified as unsupervised graph convolutional network. Thus, it is capable of training the unlabeled dynamic graph, it has more extensive application scenarios, and the calculated node embedding has strong generality. We evaluate our proposed method on experimental data in three tasks which are node classification, edge classification, and link prediction. In the classification task, facing the graph with large scale, complex connection relationship, and uncertain change rule, the F1 value of node classification task obtained by DEDGCN reaches 77%, and the F1 value of edge classification task reaches more than 90%. The results show that DEDGCN is effective in capturing graph features, and the effect of DEDGCN is much higher than other baseline methods, which proves the importance of capturing node stability features in dynamic graph representation learning. At the same time, the ability of DEDGCN in unsupervised tasks is further verified by using clustering and anomaly detection tasks, which proves that DEDGCN learning network embedding is widely used.

1. Introduction

There are many graph data in complex practical systems [1], such as social platforms, financial investment platforms, e-commerce platforms, etc. Because of various shapes and complex connection relationships, the graph is more difficult to represent than other data types of data. This also makes graph representation the key of application graph data. Graph representation is also called graph embedding (GE). At present, mature methods mainly include DeepWalk [2] and Node2vec [3] based on a random walk, GraRep [4] and HOPE [5] based on matrix decomposition, Struc2vec [6] and LINE [7] based on graph structure characteristics, SDNE [8] and DRNE [9] based on neural network, graph neural network [10], etc.

These methods are mostly applied in static graph. In practical application, graph does not remain unchanged but constantly changes over time, for example, making new friends in the relationship network, increasing and decreasing devices in the topology network, and iterating commodities in the e-commerce network. These graphs have time attributes. Static network representation learning technology cannot capture the time characteristics of dynamic graphs, and cannot learn and update the changes of nodes and the relationship between nodes, which makes the learning results of graph representation lack authenticity and dynamics.

Graph Convolution Networks [11] (GCN) have a simple structure, low complexity, fewer training times, and a good learning effect. As the basis of graph representation, it is

applied by many models. GCN cannot capture the change features in dynamic graph, but the Recursive Neural Network (RNN) can extract these features by processing change sequence information with time relation. In other words, the combination of GCN and RNN can effectively process dynamic graph. GCRN [12], RgCNN [13], etc., input the structural features of graph data captured by GCN into RNN, learn the timing relationship of nodes, and obtain the time features. These methods separate the structural features of graph from the time features, ignoring the internal connection. Moreover, the single GCN model is unable to capture dynamic structural features adaptively. EvolveGCN [14] uses RNN to evolve the parameters of the GCN so that the GCN can adaptively adjust according to the shape of graph at different times to extract the structural features with time characteristics. In addition, the graph shows periodic changes in many cases, such as significant differences in the communication relationship between employees in enterprises on working days and rest days. Employees are closely connected on weekdays and sparsely connected on rest days. However, the essential attributes of employees have not changed, when mapped to the graph representation, although nodes do not appear, they still maintain their original properties and characteristics, which is the stability of nodes. EvolveGCN lacks cognition of node stability and learning of node inherent attributes. The essence of GCN is supervised learning, which requires labeled data and is difficult to apply to tasks such as clustering and community discovery, making the model poor in universality.

Through the above analysis, we summarized the challenges faced by dynamic graph representation learning: the first problem is the stability of nodes. When capturing the dynamically changing graph structure features, the embedding of nodes is entirely influenced by the graph structure. Still, the attributes of nodes themselves determine the nature of nodes, and this nature does not change significantly with time under normal circumstances, which is the stability feature of nodes. Maintaining the stability of nodes in the graph structure with dynamically changing protrusions is a significant challenge. The second problem is the problem of unlabeled graph data training. The node embedding features obtained by supervised learning are suitable for specific tasks, and the universality is poor. When faced with unsupervised tasks such as clustering and anomaly detection, node embedding cannot be directly applied. Given these two problems, in this paper, a dual evolution dynamic graph convolutional neural network, DEDGCN, is proposed. Firstly, the stability characteristics of nodes are learned through the node evolution module. Secondly, the loss function is calculated by using the stability characteristics of nodes, and the unsupervised dynamic graph representation model is constructed while modifying the network architecture.

The following are the main contributions of this paper:

(i) We propose a dual evolution model. Based on EvolveGCN, a node evolution module is added to form a dual evolution model. While capturing the structural characteristics of a dynamic graph, the stability characteristics of nodes are learned. The GCN network architecture is jointly revised from graph structure and nodes to build an adaptive dynamic graph convolution model.

(ii) We build an unsupervised GCN model. We use the node stability of the dynamic graph and the prediction ability of the node evolution module to construct the loss function independent of the data label. Without relying on specific data label and tasks, DEDGCN is an unsupervised dynamic graph convolutional network. We can apply DEDGCN to a broader range of tasks.

Architecture: Section 2 describes the related work, relevant concepts are given in Section 3, solutions to the problems are put forward in Section 4, an experimental evaluation of our methods is presented in Section 5, and Section 6 provides a summary.

2. Related Work

The purpose of graph representation learning technology is to obtain low-dimensional dense vectors of nodes and apply them to downstream tasks such as node classification, link prediction, network reconfiguration, community discovery, network data visualization, etc. [15]. At present, most graph representation learning methods are mainly aimed at static graph. When faced with dynamic graph data, these methods cannot capture the changing characteristics of the graph, resulting in unsatisfactory results in the application process. The essence of dynamic graph data and static graph is the same; so, many dynamic graph representations often evolve from the static model.

At the earliest stage, node embedding was generated by decomposing the Laplace and adjacency matrix of graph data, called Graph Factorization (GF). The most typical examples are GraRep [4], HOPE [5], etc. These algorithms use matrix decomposition to calculate eigenvalues and eigenvectors to represent node embedding. The key of graph decomposition technology is to obtain eigenvalues and eigenvectors which directly determine the quality of node embedding. In dynamic graph data, to reduce algorithm complexity, the evolution of eigenvalues and eigenvectors is carried out to update node embedding, the most typical of which is DHPE [16]. In addition, another direct factor that determines the quality of node embedding is the node itself. TIMERS [17] decomposes the adjacency matrix of the initial time graph and updates the node embedding in the subsequent time graph. After each update, the loss value represented by the graph is calculated. When the loss value exceeds the threshold, the time graph is matrix decomposed again. However, with the increase of graph scale, matrix decomposition becomes more and more difficult, and the node representation generated by matrix decomposition is challenging to explain. This problem makes applying the graph representation learning method based on matrix decomposition to large-scale graph challenging.

Dynamic graph has many structural characteristics, and many scholars use them to reconstruct the generation

probability of graphs to calculate node embedding. For example, DyREP [18] used the relational evolution and social evolution in dynamic graph to initially describe the characteristics of nodes, periodic changes of nodes, and the influence of external nodes on embedding from the perspectives of local embedding propagation, self-propagation, and external factor driving. Then, DyREP used the probability of node emergence to reconstruct data and calculate node embedding. Zhou et al. [19] adopted the triadic closure process, combined with social isomorphism and temporal smoothness, to construct the loss function and reconstruct the graph to obtain node embedding. These methods all belong to the transductive method. Whenever new graph data come, it is necessary to retrain the model to obtain node embeddings. The process is high in complexity, long in time, and inefficient.

With the appearance of Graph Neural Network (GNN), graph embedding technology has entered a new stage of development, especially GCN, which realizes the end-to-end learning of graphs of any size and shape. GCN has a good effect in node classification, link prediction, and other tasks, and has the advantages of simple structure, few parameters, strong ability to extract graph features, and suitability for large-scale graph. At present, many graph embedding technologies are based on GCN, learning the timing information of nodes themselves and capturing the time characteristics of graph data. For example, GCRN [12] combines GCN with RNN, captures the timing information of graph through input node embeddings into RNN. Similar ideas include WD-GCN/CD-GCN [20], RgCNN [13], and so on. However, these methods only start from the dynamic changes of the nodes themselves, ignoring the graph structure's constant changes. The learned structure features are fixed and single, and the adaptability to frequently changing graphs is poor. To make the model adapt to the shift of graph, Addgraph [21] uses an attention mechanism to aggregate GCN parameters in the past period to generate current GCN network parameters. EvolveGCN [14] also aims at the ever-changing graph structure problem. With the help of RNN to evolve GCN, a dynamic GCN model is built, which ensures that the model can learn the changing graph structure adaptively. However, the nature of GCN is still a kind of supervised learning, which cannot extract the structure and dynamic features of unlabeled data, leading to the failure of community division, clustering, anomaly detection, and other tasks.

The most common method for learning unsupervised graph is the dynamic autoencoder network, which utilizes the symmetry between encoder and decoder to generate highly nonlinear node embedding. For dynamic graph, DynGEM [22] increases the number and width of layers of encoder and decoder according to the graph size each time, which ensures the adaptive change of the model. At the same time, using the first-order approximation and second-order approximation of nodes, the loss values of local structure and global structure of graph are calculated to construct the loss function and train the network. Dyngraph2vec [23] replaces the neurons of the self-encoder with long-term and short-term memory cells (LSTM). It takes the historical neighbor information of the nodes multiple times as input so that the model can capture time characteristics. Dyngraph2vec constructs the loss function in the way of prediction, inputs multiple historical graph into the network, predicts the graph structure of the next time, compares it with the real graph structure, and forms the loss function. The autoencoder network includes multi-layer encoders and decoders, and there are many parameters in the training process, limiting its application in large-scale graph.

The dynamic network representation learning technologies involved are summarized in Table 1 according to the learning methods to facilitate further research in the future.

3. Relevant Concepts

In this section, we formally define the basic concepts and related issues of dynamic graph representation learning.

Definition 1. (Static Graph) $G = (V, E)$ is composed of a group of nodes $V = \{v_1, \ldots, v_N\}$ and the connection relationship (called edge) $E \in \{(v_i, v_j) | (v_i, v_j) \in V \times V\}$ between nodes, where N represents the number of nodes. A represents the adjacency matrix of a graph with the size of $N \times N$, where if $e_{ij} \in E$, then $A_{ij} = 1$, otherwise $A_{ij} = 0$. If the graph is undirected, A is a symmetric matrix.

Definition 2. (Dynamic Graph) A series of static graphs that change continuously constitute a dynamic graph, denoted by $\mathcal{G} = \{G_1, G_2, \cdots, G_t, \cdots\}$, where each $G_t = (V_t, E_t)$ is called a snapshot. t represents the serial number of snapshots, V_t represents the set of nodes under the t snapshot, E_t represents the set under the t snapshot, and A_t means the adjacency matrix of the graph of the t snapshot.

Definition 3. (Graph Representation Learning) For a given graph $G = (V, E)$, graph representation learning is defined as mapping the nodes into the vector space of $d\,(d \ll |V|)$ dimension by function $f: V \longrightarrow X \in \mathbb{R}^d$.

Generally speaking, we define the problems related to our work as follows.

Problem: . (Dynamic Graph Representation Learning) For a given $\mathcal{G} = \{G_1, G_2, \cdots, G_T\}$, the dynamic graph representation learning is defined as mapping nodes in snapshot t into d-dimensional vector space by function $f_t: V_t \longrightarrow X_t \in \mathbb{R}^d$, and the f_t can capture the following characteristics:

(i) Similarity characteristics of structure. The Eucliean distance between x_t^i and x_t^j is small if i and j are neighbors.

(ii) Stability characteristics of nodes. The Eucliean distance between V_t and V_{t+1} is small if graph evolves normally.

4. Methodology

This section puts forward a dual evolving dynamic graph convolution network, DEDGCN, whose framework is provided in Figure 1. DEDGCN mainly includes GCN

Table 1: A summary of dynamic network embedding methods.

Method	Learning techniques	Supervised	Unsupervised
DHPE [16]	Matrix decomposition, embedded update		√
TIMERS [17]	Matrix decomposition, embedded update		√
DyREP [18]	Dynamic network structure characteristics		√
DynamicTriad [19]	Dynamic network structure characteristics		√
GCRN [12]	Splicing GCN and RNN	√	
WD-GCN/CD-GCN [20]	Splicing GCN and RNN	√	
RgCNN [13]	Splicing GCN and RNN	√	
Addgraph [21]	Attentional mechanism evolves GCN	√	
EvolveGCN [14]	RNN evolves GCN	√	
DynGEM [22]	Scalable autoencoder network		√
Dyngraph2vec [23]	LSTM evolves autoencoder network		√

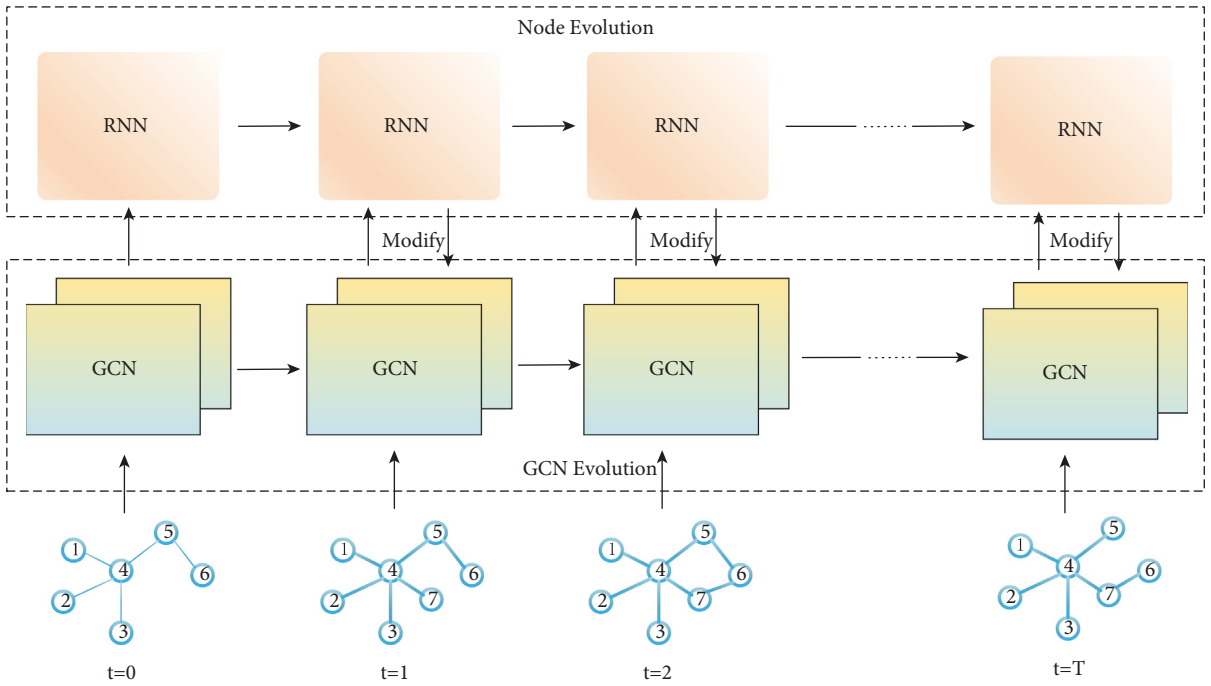

Figure 1: The framework of DEDGCN. DEDGCN consists of two parts: the upper Node Evolution learning the stability characteristics of nodes and the lower GCN evolution learning dynamic structure characteristics of graphs. They work together to modify GCN and build an unsupervised graph convolutional network framework.

evolution and node evolution. The graphs' dynamic structure characteristics and node stability characteristics are learned, respectively, to build an adaptive unsupervised dynamic graph convolutional network.

In the part of GCN evolution, we learn from the evolution process of GCN by EvolveGCN and use RNN to capture the morphological change rule of GCN in historical snapshots to generate the GCN network needed by the current snapshot. GCN parameters are the key to building a GCN model. Using RNN to evolve GCN parameters can quickly correct the GCN to capture changing graph structure features.

In node evolution, we mainly consider the stability characteristics of nodes. The attributes of nodes themselves are stable during the normal development of graph and will not change significantly. The change process with time contains a lot of temporal information, which can reflect the fluctuation of its attributes. By capturing the temporal information, we can predict the changes of nodes and get the stability characteristics of nodes. Similarly, we use RNN to learn the timing information of nodes, extract the attributes of nodes themselves, and ensure the stable generation of node embedding with time. We modify the GCN by constructing the loss function and feeding back the stability characteristics of nodes to the GCN.

The loss function mainly comprises node embedding generated by the GCN and node embedding predicted by RNN. RNN can predict the embedding of nodes at the next moment by learning the timing information of nodes. In vector space, the predicted value should be as close as possible to the generated value of GCN to ensure the authenticity of the learned timing information. In addition, the stability of nodes also makes the embedding of nodes in adjacent snapshots not change significantly, and the generated values calculated by GCN should be close to each other. Therefore, we use the gap

between the predicted value and the generated value of GCN and the distance of node embedding between adjacent snapshots to construct the loss function in order to provide constant feedback and correct the GCN. Ensure that the node embedding generated by the GCN can have the node stability characteristics in the predicted value. At the same time, the loss function of DEDGCN does not depend on data labels, forming an unsupervised dynamic graph convolution network. The number of nodes in each moment graph changes with time in the dynamic graph, and they are not precisely the same. To describe snapshots conveniently, we use N to indicate the number of nodes in each snapshot.

4.1. GCN Evolution. We use the GCN to extract the structural features of graph. GCN contains multiple graph convolution layers, which can aggregate multi-layer neighbor features and capture structure features. For $G_t = (V_t, E_t)$, the propagation rules among the graph convolution layers are as follows:

$$H_t^{(l+1)} = \sigma\left(\widetilde{A}_t \cdot H_t^{(l)} \cdot W_t^{(l)}\right),$$
$$\widetilde{A}_t = D_t^{-\frac{1}{2}} A_t' D_t^{-\frac{1}{2}}, A_t' = A_t + I. \tag{1}$$

Here, $H_t^{(l)}$ represents the calculation result of the snapshot t after the l-th graph convolution layer. $\widetilde{A}_t$ is defined as the regularized form of the adjacency matrix A_t of the snapshot t. A_t' is the self-connection matrix of snapshot t. D is the degree matrix, $D_{ii} = \sum_j A_{ij}'$, and $W_t^{(l)}$ represents the parameters of the l-th graph convolution layer at t. $\sigma(\cdot)$ refers to the activation function, such as ReLU, sigmoid, etc.

To enable GCN to capture the structural features of dynamic graphs adaptively, we use EvolveGCN as a reference for evolving the parameters of GCN, and modify the architecture of GCN to adaptively obtain to the changing graph. At present, the commonly RNN cell is Long-Short Term Memory [24] (called LSTM), which consists of an input gate, a forget gate, and an output gate, which can selectively capture the information of time series, save the key and forget the redundant content. For the parameter evolution of GCN, we choose LSTM as the memory cell, and the process is shown in Figure 2. We input the GCN parameter $W_{t-1}^{(l)}$ of l-th layer at $t-1$ into LSTM, and get GCN parameter $W_t^{(l)}$ of l -th layer at t.

The calculation method of GCN parameter evolution is as follows:

$$\begin{aligned}
W_t^{(l)} &= \mathrm{LSTM}\left(W_{t-1}^{(l)}\right),\\
i_t &= \mathrm{sigmoid}\left(U^{(i)} W_{t-1}^{(l)} + B^{(i)}\right),\\
f_t &= \mathrm{sigmoid}\left(U^{(f)} W_{t-1}^{(l)} + B^{(f)}\right),\\
o_t &= \mathrm{sigmoid}\left(U^{(o)} W_{t-1}^{(l)} + B^{(o)}\right),\\
\widetilde{c}_t &= \tanh\left(U^{(c)} W_{t-1}^{(l)} + B^{(c)}\right),\\
c_t &= f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t,\\
W_t^{(l)} &= o_t \circ \tanh\left(c_t\right).
\end{aligned} \tag{2}$$

In the process of GCN parameter evolution, LSTM constantly learns the changing rules of GCN parameters in continuous snapshots. Memory cells are updated continuously, and new GCN parameters are continually predicted so that the GCN can adaptively modify the shelf shape with the change of graph to capture the dynamic structural features effectively.

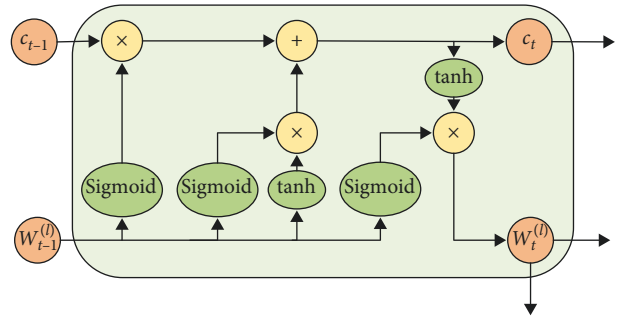

Figure 2: The process of GCN evolution.

4.2. Node Evolution. LSTM has an excellent ability to capture timing information. We use LSTM to learn the behavior of nodes in continuous snapshots, extract the stability characteristics of nodes, and predict the state of nodes in the next snapshot. Here, we use LSTM to evolve nodes, and the evolution process of nodes is shown in Figure 3.

We input the embedded $\{X_{t-w+1}, X_{t-w+2}, \cdots, X_t\}$ calculated by GCN in continuous snapshots into LSTM and get predicted embedding P_{t+1} at time $t+1$. w represents the window size of LSTM and P_t indicates node embedding at the predicted time t. The calculation method is shown in the following formula.

$$\begin{aligned}
P_{t+1} &= \mathrm{LSTM}\left(X_t, P_t\right),\\
i_{t+1} &= \mathrm{sigmoid}\left(W^{(i)} X_t + U^{(i)} P_t + B^{(i)}\right),\\
f_{t+1} &= \mathrm{sigmoid}\left(W^{(f)} X_t + U^{(f)} P_t + B^{(f)}\right),\\
o_{t+1} &= \mathrm{sigmoid}\left(W^{(o)} X_t + U^{(o)} P_t + B^{(o)}\right),\\
\widetilde{c}_{t+1} &= \tanh\left(W^{(c)} X_t + U^{(c)} P_t + B^{(c)}\right),\\
c_{t+1} &= f_{t+1} \circ c_t + i_{t+1} \circ \widetilde{c}_{t+1},\\
P_{t+1} &= o_{t+1} \circ \tanh\left(c_{t+1}\right).
\end{aligned} \tag{3}$$

By learning the node state during the window w, the predicted node embedding includes the attribute characteristics of the node itself and the stability characteristics in the changing process. We spread the attributes of nodes into the GCN in the form of the loss function and further modified the GCN network structure.

4.3. Construction of Loss Function. We use the node embedding generated by GCN and the node evolution results to construct the loss function and modify the GCN model. Let us go into the details below.

In the normal changes of dynamic networks, the characteristics of nodes will not change drastically. In dynamic graph

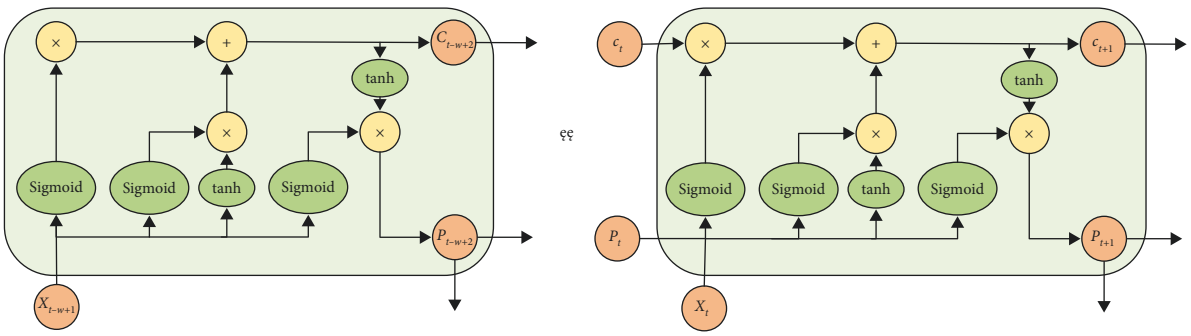

Figure 3: The process of Node Evolution.

networks with frequent increase or decrease of nodes, the inherent characteristics of nodes will not change enormously, which denotes the stability of the nodes. The property of nodes transformed into vector space shows that the embedding similarity between adjacent snapshots is higher. Therefore, we can express it by calculating the embedding similarity of adjacent snapshot nodes, and the calculation method is shown in the following formula:.

$$\mathscr{L}_1 = \sum_{i=1}^{N} 1 - \text{similarity}\left(x_{t-1}^i, x_t^i\right). \tag{4}$$

In formula (4), x_t^i indicates the node embedding generated by GCN by node i at time t, and *similarit y* represents cosine similarity function.

In the process of node evolution, LSTM is used to predict node embedding, and the expected embedding value of the node at the next moment is obtained. When the predicted value is consistent with the generated value of GCN, the GCN model can acquire the ability to extract the stability features of nodes. The same is true for the ability to extract the structural characteristics of nodes. We calculate the similarity between the predicted value and the GCN-generated value to keep their consistency. The calculation method is as shown in formula (5), where p_t^i represents the node embedding expected by LSTM for node i at time t.

$$\mathscr{L}_2 = \sum_{i=1}^{N} 1 - \text{similarity}\left(p_t^i, x_t^i\right). \tag{5}$$

In addition, to prevent the overfitting phenomenon in the training process, we add the weight attenuation function to the loss function to reduce the parameter weight as shown in formula (6), where w represents the GCN parameter.

$$\mathscr{L}_3 = \frac{\lambda \sum_w \|w\|_2^2}{2N}. \tag{6}$$

The final loss function is defined as follows:

$$\text{loss} = \mathscr{L}_1 + \mathscr{L}_2 + \mathscr{L}_3, \tag{7}$$

5. Experiments

5.1. Dataset. We experiment with our method DEDGCN on publicly available basic datasets. The datasets are described below.

Elliptic [25] is a bitcoin transaction graph in which nodes represent transactions and edges represent bitcoin flows between transactions. The types of nodes in this graph can be divided into two categories: legal and illegal, so we use this dataset for experiments of node classification and node clustering tasks.

Bitcoin alpha is a platform for trading in bitcoin, which uses the trading behaviors among users to form a graph. Members of the platform rate other members on a scale of F02D 10 to +10, which indicates the trust level of each member. According to the user's trust score, we divide the score into two categories: trustworthy and untrustworthy. The transaction behaviors among users are built into a dynamic network. The nodes represent users, and edges represent transactions between users. In scoring each user's transaction process, users are given the label of trust or distrust. For Bitcoin alpha, we carry out the edge classification task and link prediction task.

Reddit Hyperlink is a graph composed by extracting the link relationships in Reddit posts. Each link relationship contains the time and the source post's emotion (positive or negative) to the target post. We performed edge classification on this dataset to predict the emotional relationships that existed between unlinked posts.

UCI is a graph composed of private information sent by UC-Irvine on the campus social platforms. Users can search for other people on this social platform and then send conversations according to the profile information. The edge represents a piece of private information sent by user u to user v at time t. We predict the links of this dataset to capture the possible contacts of users at the next moment.

AS network is a graph composed of connections between autonomous systems. Nodes represent autonomous systems, while edges represent connectivity between autonomous systems. We make the link prediction task on this graph to predict the connection relationship of the network at the next moment.

Myanmar network refers to the network composed of autonomous systems applied by Myanmar. Nodes represent autonomous systems belonging to Myanmar, and edges represent the connectivity relationship between autonomous systems. Through the continuous monitoring of the network in the Myanmar autonomous system, the abnormal behavior of the network can be warned, and a reference can be provided for maintaining the regular operation of the network.

The basic information of these data is shown in Table 2. According to the characteristics of different datasets, we divide them into snapshots at different time intervals and apply them to different tasks.

5.2. Baseline. We will compare DEDGCN with the following five basic methods.

Methods 1: GCN is a static graph convolution neural network, which cannot extract dynamic features. We fuse all snapshots to form static graph, and compare the importance of time features. At the same time, GCN is a network structure feature extractor in DEDGCN, which is tested as a module unit of DEDGCN.

Method 2: GCN-GRU is a method of generating node embedding with fixed GCN and evolving node timing relationship with GRU. This method can be regarded as a node embedding module and tested as a unit module of DEDGCN.

Method 3: DynGEM is a dynamic unsupervised network representation learning method based on deep autoencoder model. DynGEM carries out adaptive learning for graph at different times through the evolution of the parameters of the deep autoencoder model, so as to ensure the learning of network structural features and spontaneously capture the relationship characteristics between graph at different times, so as to make the network embedding results at different times continuous.

Method 4: Dyngraph2vec is a dynamic unsupervised node representation method that integrates the deep autoencoder model, LSTM, and MLP networks. Neurons in the deep autoencoder model are replaced by short and long memory cells to construct a deep autocoding model with memory function. According to the different combination modes of the three networks, dyngraph2vec includes three versions, namely, dyngraph2vecAE, dyngraph2vecRNN, and dyngraph2vecAERNN. Dynagraph2vecAE and DynGEM are similar in architecture, so the second and third methods are used for comparison.

Method 5: EvolveGCN is similar to GCN-GRU, which uses RNN to evolve GCN parameters and learn the time series relationship of models. This method is regarded as a GCN parameter evolution module and tested as a DEDGCN unit module.

5.3. Metric. The evaluation methods used in this paper include F1 value and MAP. The calculation of these two evaluation methods is introduced in detail below.

First of all, we need to make the following definitions:

True positive (TP): the number of samples where the predicted and actual values are positive.

True negatives (TN): the number of samples where the predicted value is positive and the actual value is negative.

False positives (FP): the number of samples where the predicted value is negative and the actual value is positive.

False negatives (FN): the number of samples where the predicted and actual values are negative.

Through the above four definitions, we can calculate the precision and recall of the prediction results, and the formula is as follows:

$$\begin{aligned} \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}. \end{aligned} \tag{8}$$

F1 value is the harmonic average of accuracy rate and recall rate, which can objectively reflect the validity of prediction. The calculation formula is as follows:

$$F1 = 2\frac{\text{precision} \times \text{racall}}{\text{precision} + \text{recall}}. \tag{9}$$

AP refers to the integral of the PR (precision-recall) curve, the average precision of all recall values between 0 and 1. The formula is as follows:

$$\text{AP} = \sum_{i=1}^{N} \text{precision}(i)\Delta\text{recall}(i). \tag{10}$$

MAP refers to the average of all kinds of AP, and the calculation formula is as follows. K indicates the number of categories of AP.

$$\text{MAP} = \frac{1}{K}\sum_{i=1}^{K} \text{AP}_i. \tag{11}$$

5.4. Task. In this paper, we prove the effectiveness of our proposed DEDGCN through four tasks: node classification, edge classification, link prediction, and anomaly detection.

5.4.1. Node Classification. Predict the types of unlabeled nodes by learning the characteristics of labeled nodes. In this section, the probability of node embedding is calculated by the feedforward neural network and softmax function to judge the node type u at time t. For node classification, we use the F1 value to measure the effectiveness of the method.

5.4.2. Edge Classification. Predict the types of unlabeled edges by learning the features of labeled edges. In this paper, the probability of edge embedding is calculated by feedforward neural network and softmax function to judge the edge type at time t. The embedding is obtained by aggregating node u and node v representation. The aggregation method adopts the Hadamard product. The measurement method of edge classification adopts the F1 value.

5.4.3. Link Prediction. Whether the edge at time $t + 1$ exists or not is predicted by embedding of node u and node v before time $t + 1$. We aggregate node u and node v, and then use MLP to obtain the existence probability of edges. For link prediction, we use MAP to measure the effectiveness of the results.

Table 2: Basic information of experimental data.

Dataset	# Nodes	# Edges	# Snapshots	Tasks
Elliptic	230769	234355	49	Node classification and clustering
Bitcoin alpha	3783	24186	136	Link prediction, edge classification
Reddit hyperlink network	55863	858490	174	Edge classification
UCI	1899	59835	192	Link prediction
AS network	6474	13895	100	Link prediction
Myanmar	209	48857	75	Anomaly detection

5.4.4. Anomaly Detection. Through the representation of node embedding in the continuous snapshot, the normal state of the node at the next moment is predicted, and the expected value is compared with the actual value of the node to judge whether the node is abnormal.

5.5. Details

(1) For any dataset, we use a one-hot node-degree as the input feature of the model.

(2) For all GCN, we set the number of graph convolution layers to 2; For all MLP used in classification tasks, we set the depth to 2, and the softmax function calculates the classification probability. For classification tasks and link prediction tasks, the loss function of MLP adopts cross-entropy.

(3) We use fixed-size dimension to represent nodes for any dataset, and the value is 100.

(4) For the weight attenuation coefficient, we set its size to 0.01.

(5) We divide the data into training set and test set according to the ratio of 8 : 2

5.6. Results and Analysis

5.6.1. Node Classification. We apply Elliptic to the node classification task. For GCN parameters and node embedding evolution, we all adopt a time window size of 5. Meanwhile, GCN, GCN-GRU, EvloveGCN, and DEDGCN constitute unit test experiments to verify the effectiveness of different modules of DEDGCN in node classification tasks, respectively. The experimental results are shown in Figure 4.

As we can see from Figure 4, DEDGCN performs node classification tasks in Elliptic dataset, and its F1 value is much higher than that of GCN, GCN-GRU, and EvloveGCN. GCN is not effective in the dynamic graph representation learning, and the F1 value is only 47% in node classification tasks, which shows that the graph structure changes have a great impact on the generation of node embedding in the dynamic evolution process. The lack of time leads to a lot of old information in the graph, much noise in structural features, and poor effect in practical tasks of node embedding application. However, the two versions of EvolveGCN, version H and version O, are not effective in node classification of Elliptic dataset, especially in version H, the F1 value of node classification is 44%. To sum up, the attributes of nodes on Elliptic are relatively stable, and it is difficult to convert between legal nodes and illegal nodes. GCN-GRU model and DEDGCN model proposed in this section can evolve nodes, extract inherent features of nodes, and maintain the stability of nodes. F1 values are higher than EvolveGCN, which shows the importance of node stability features in the process of snapshot evolution. In addition, the F1 value of DEDGCN using MLP for supervised classification reaches 77%. Besides supervised classification, K-means clustering algorithm is used for unsupervised classification of node types, and the effect is only 1% lower than that of supervised method. This also proves that DEDGCN learning node embedding is suitable for unsupervised clustering tasks.

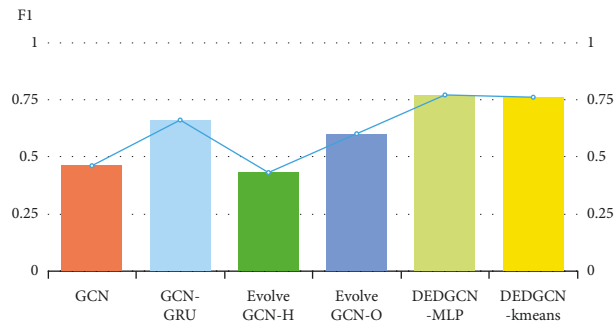

Figure 4: F1 value of Elliptic in the node classification task.

5.6.2. Edge Classification. We apply Bitcoin Alpha and Reddit Hyperlink Network datasets to edge classification task. For GCN parameters and node embedding evolution, we both adopt a time window size of 5. At the same time, GCN, GCN-GRU, EvloveGCN, and DEDGCN constitute unit test experiments to verify the effectiveness of different modules of DEDGCN in the edge classification task, respectively. The experimental results are shown in Figure 5.

As shown in Figure 5, in the edge classification task, the F1 value of DEDGCN classification reaches 93% on Bitcoin Alpha and 90% on Reddit Hyperlink Network dataset, and the classification effect is far better than that of module unit classification. The F1 value of EvolveGCN for edge classification is higher than that of GCN and GCN-GRU. In dynamic graph, nodes rely on edges to form a network, and edges are an important component of the graph, and the characteristics of edges depend on the network structure. In EvolveGCN and DEDGCN, relying on GCN parameter evolution module, dynamic network structure characteristics can be captured, while GCN and GCN-GRU can only

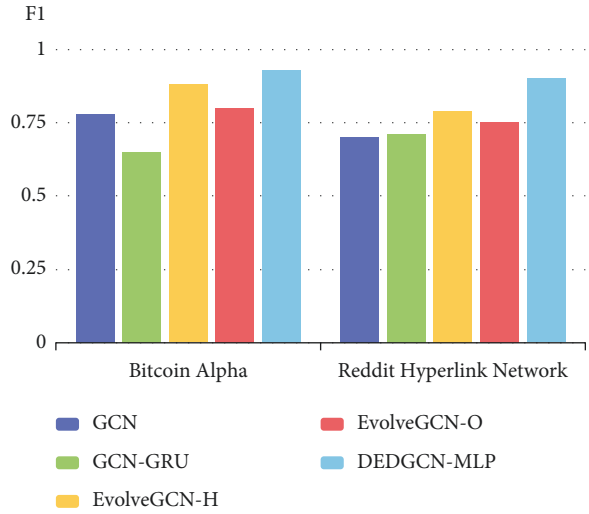

Figure 5: F1 value of Reddit hyperlink and Bitcoin Alpha in the edge classification task.

extract fixed network structure characteristics, but have no updating ability. The learned network embedding contains a lot of old information, which cannot reflect the current state of the graph, and has poor effect in edge classification tasks.

Combining the experimental results of node classification and edge classification tasks, DEDGCN has different advantages in different tasks. In node classification task, the stability feature captured by node embedding evolution module helps to improve the node classification effect. In edge classification task, the dynamic network structure feature captured by GCN parameter evolution module helps to improve the edge classification effect. Compared with GCN, GCN-GRU, and EvolveGCN, the F1 value of DEDGCN is higher than that of the unit module, which shows the importance of GCN parameter evolution module and node embedding evolution module in DEDGCN. DEDGCN is an integral part of dynamic graph representation learning.

5.6.3. Link Prediction. We conducted link prediction experiments on Bitcoin Alpha, UCI, and AS Network datasets, respectively, and the results are shown in Table 2.

From Table 3, we can see that the MAP values of DEDGCN are higher than those of other methods, and remain above 0.15, which has a good effect on the link prediction task. MAP value is the comprehensive embodiment of accuracy and recall rate and can reflect the global performance index of the algorithm. Bitcoin Alpha, UCI, and AS Network belong to three different types of graph data: transaction network, social network, and device network. DEDGCN has stable MAP values on different graph types, which also directly shows that DEDGCN can be applied to most graph data and has better robustness.

5.6.4. Anomaly Detection. We use the global BGP routing path information (including the AS relationship to which BGP belongs) collected by route views to build a global dynamic AS relationship network in two hours. Due to the huge number of AS globally, we selected Myanmar's AS network as the research object to verify DEDGCN's ability in anomaly detection tasks.

First of all, we learn the evolution law of the normal AS network through DEDGCN. Then, the AS network to be detected is input into the trained model to obtain the node embedding and node evolution results of the AS network. Finally, the similarity of node embedding and node evolution results is calculated to determine when the network anomaly occurs.

Here, we use the cosine similarity method to calculate the similarity between each node's embedding and evolution result. This similarity is also called the normal value in anomaly detection. To determine whether anomalies occur at time t, we take the average similarity of all nodes in the AS network at time t as the normal value of the network at time t. In the process of continuous monitoring, when the network is abnormal, the normal value of the network will appear to "steep drop." At this time, we can judge the time when the AS network is abnormal.

Taking Myanmar as an example, we collected Myanmar's AS network from January to February 2021, divided snapshots in two-hour units, and used DEDGCN for monitoring. The monitoring results are shown in Figure 6.

It can be clearly seen from Figure 6 that from 0 : 00 on February 1, 2021, the score of Myanmar's AS network has experienced a "steep drop," indicating that there is a problem with the Myanmar network at this time. Through Wikipedia verification, it can be known on February 1, 2021, the Myanmar military took over the government. To complete social control and control speech, it cut off the Internet and communications in major cities, resulting in large-scale network outages across the country. This also verified DEDGCN's ability in anomaly detection tasks.

Table 3: The MAP value on the link prediction task.

	DynGEM	Dyngraph 2vecAE	Dyngraph 2vecAERNN	GCN-GRU	Evolve GCN-H	Evolve GCN-O	DEDGCN
Bitcoin alpha	0.0525	0.0507	0.1100	0.0001	0.0049	0.0036	0.1668
UCI	0.0209	0.0044	0.0205	0.0114	0.0126	0.0270	0.1584
AS network	0.0529	0.0331	0.0711	0.0713	0.1534	0.1139	0.1691

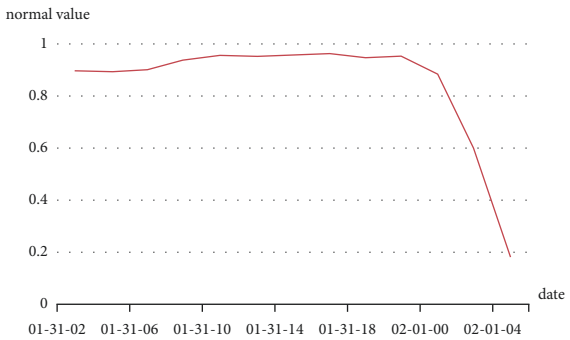

Figure 6: Monitoring of the AS network in Myanmar.

6. Conclusions

Through the performance of node classification, edge classification, link prediction, and anomaly detection tasks, the effect of our proposed method DEDGCN has been proved. Based on the experimental results, we discussed the applicable task scenarios of GCN-RNN based on node evolution and EvloveGCN based on parameter evolution. DEDGCN, which combines the advantages of node evolution and parameter evolution, makes the extracted node features dynamic while retaining the inherent stability characteristics of the node and has achieved good results in various tasks, which proves the broad application of the DEDGCN method. In addition, DEDGCN is an unsupervised graph representation model. We use unsupervised clustering methods to determine the type of nodes in the node classification task for experimental data. Its effect is only 1% lower than the effect of supervised classification. In the network anomaly detection task for realistic graph data, DEDGCN can perceive the time when the network anomaly occurs, which also shows the effectiveness of DEDGCN in unsupervised tasks.

Data Availability

Elliptic dataset can be obtained from https://www.kaggle.com/ellipticco/elliptic-data-set. Bitcoin alpha dataset can be obtained from http://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html. Reddit Hyperlink dataset can be obtained from http://snap.stanford.edu/data/soc-RedditHyperlinks.html. UCI dataset can be obtained from http://konect.cc/networks/opsahl-ucsocial/. AS network dataset can be obtained from http://snap.stanford.edu/data/as-733.html. Myanmar Network dataset is constructed by extracting global BGP path which can be obtained from http://archive.routeviews.org/. The wiki address for the February 1, 2021 events in Myanmar can be obtained from https://en.wikipedia.org/wiki/2021_Myanmar_coup_d%27%C3%A9tat.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.U1804263) and the Zhongyuan Science and Technology Innovation Leading Talent Project (No. 214200510019).

References

[1] X. Guotong, Z. Ming, and L. Jianxin, "Dynamic network embedding survey," p. 5, 2021, https://arxiv/abs.org/2103.15447.

[2] B. Perozzi, A. Rami, and S. Steven, "Deepwalk: online learning of social representations," in *Proceedings of the Twentyth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, New York NY USA, August 2014.

[3] A. Grover and J. Leskovec, "node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, San Francisco CA USA, August 2016.

[4] S. Cao, W. Lu, and Q. Xu, "Grarep: learning graph representations with global structural information," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, pp. 891–900, Melbourne Australia, October 2015.

[5] M. Ou, P. Cui, and J. Pei, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114, San Francisco CA USA, August 2016.

[6] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "struc2vec: learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 385–394, Halifax Canada, August 2017.

[7] T. Jian, Q. Meng, W. Mingzhe, Z. Ming, Y. Jun, and M. Qiaozhu, "Line: large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, Florence Italy, May 2015.

[8] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234, San Francisco CA USA, August 2016.

[9] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network emberegular equivalence," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2357–2366, London UK, August 2018.

[10] Z. Wu, S. Pan, F. Chen, and G. C. P. S. Long, "A comprehensive survey on graph neural networks," *IEEE Transactions*

on Neural Networks and Learning Systems, vol. 32, no. 1, pp. 4–24, 2021.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv/abs.org/1609.02907.

[12] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Proceedings of the International Conference on Neural Information Processing*, pp. 362–373, Springer, Siem Reap, Cambodia, December 2018.

[13] A. Narayan and P. H. ON Roe, "Learning graph dynamics using deep neural networks," *IFAC-PapersOnLine*, vol. 51, no. 2, pp. 433–438, 2018.

[14] A. Pareja, G. Domeniconi, J. Chen, and T. T. H. T. T. C. Ma, "EvolveGCN: evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 5363–5370, NewYork, NY, USA, February 2020.

[15] L. H. William, Y. Rex, and L. Jure, "Representation Learning on Graphs: Methods and Applications," 2017, https://arxiv/abs.org/1709.05584.

[16] D. Zhu, P. Cui, Z. Zhang, and J. W. Pei, "High-order proximity preserved embedding for dynamic networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 11, p. 1, 2018.

[17] Z. Zhang, P. Cui, and J. Pei, "Timers: error-bounded svd restart on dynamic networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, New Orleans, LA, USA, 2018.

[18] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: learning representations over dynamic graphs," in *Proceedings of the International Conference on Learning Representations*, p. 28, New Orleans, NO, USA, May 2019.

[19] L. Zhou, Y. Yang, and X. Ren, "Dynamic network embedding by modeling triadic closure process," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, New Orleans, LA, USA, 2018.

[20] F. Manessi, A. Rozza, and M. Manzo, "Dynamic Graph Convolutional Networks," Pattern Recognition, 2019, https://arxiv.org/abs/1704.06199.

[21] L. Zheng, Z. Li, and J. Li, "AddGraph: anomaly detection in dynamic graph using attention-based temporal GCN," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 4419–4425, Macao, August 2019.

[22] P. Goyal, "Dyngem: Deep Embedding Method for Dynamic Graphs," 2018, https://arxiv/abs.org/1805.11273.

[23] P. Goyal, S. R. Chhetri, and A. Canedo, "dyngraph2vec: Capturing Network Dynamics Using Dynamic Graph Representation Learning," *Knowledge-Based Systems*, vol. 187, 2020.

[24] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modelling," in *Proceedings of the Thirteenth annual conference of the international speech communication association*, Portland, OR, USA, September 2012.

[25] M. Weber, G. Domeniconi, and J. Chen, "Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics," in *Proceedings of the KDD '19 Workshop on Anomaly Detection in Finance*, Anchorage, AK, USA, 2019.