Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 12, No. 3, 2024, 359-367.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Original paper

Anomaly Detection in Dynamic Graph Using Machine Learning Algorithms

Pouria Rabiei and Nosratali Ashrafi-Payaman*

Department of Electrical and Computer Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran.

Article Info

Article History:
Received 11 June 2024
Revised 06 November 2024
Accepted 11 November 2024

DOI:10.22044/jadm.2024.14476.2551

Keywords:
Anomaly Detection, Machine Learning, Dynamic Graph, Graph Neural Network, Graph-based Anomaly Detection.

**Corresponding author: ashrafi@khu.ac.ir (N. Ashrafi-Payaman).*

Abstract

Today, the amount of data with graph structure has increased dramatically. Detecting structural anomalies in the graph, such as nodes and edges whose behavior deviates from the expected behavior of the network, is important in real-world applications. Thus, in our research work, we extract the structural characteristics of the dynamic graph by using graph convolutional neural networks, then by using temporal neural network Like GRU, we extract the short-term temporal characteristics of the dynamic graph and by using the attention mechanism integrated with GRU, long-term temporal dependencies are considered. Finally, by using the neural network classifier, the abnormal edge is detected in each timestamp. Conducted experiments on the two datasets, UC Irvine messages and Digg with three baselines, including Goutlier, Netwalk and CMSketch illustrate our model outperform existing methods in a dynamic graph by 10 and 15% on average on the UCI and Digg datasets respectively. We also measured the model with AUC and confusion matrix for 1, 5, and 10 percent anomaly injection.

1. Introduction

In many areas, data are intrinsically interdependent and affect each other, such as people in social networks, the communication of computers in computer networks, or the communication of protein graphs in cells. Graph-based anomaly detection is done at four different levels: node level, edge level, subgraph level, and the whole graph level. One of the most challenging applications is anomaly detection, especially in heterogeneous graphs [1]. The concept of anomaly indicates rare observations that significantly deviate from other observations [1]. Graph neural networks achieved remarkable progress in various computing fields in recent years. The purpose of these networks is to use and generalize deep learning models on graph data.

Dynamic graph is defined in two ways: firstly, there is structural dynamics, that is, the structure of the graph, including the number of nodes and edges, changes, and secondly, the dynamic of features, in the sense that the values of network features, such as the features of nodes and edges, change over time. The terms like time-evolving graph, time-varying graph, temporal graph, graph stream, and dynamic graph are equivalent and used interchangeably. The main research on dynamic graphs is anomaly detection [2].

Research on anomaly detection in dynamic graphs using deep learning has started since 2018 [2]. For detecting anomalous data in the dynamic graphs, we face four major challenges: First, many anomaly detection models in dynamic graphs do not consider the features vector of nodes [3]. Second, many existing methods do not consider spatial dependencies and short- and long-term temporal dependencies simultaneously in dynamic graphs [3]. Third, Anomaly detection in dynamic graphs has a high time complexity, and fourth, there are a very few anomalous data samples in the dataset, so we are facing the problem of unbalanced data [4].

Our proposed model is supposed to solve these four challenges. The model includes four main parts. First, dividing the graph data set into timestamps, performing negative sampling in the training model, sampling the edges subgraph in a window, and determining the features vector for the nodes. Second, embedding vectors for all nodes of the dynamic graph at any timestamps while considering local and global structural features. Third, the time dependencies of the dynamic graph are considered by temporal models along with the attention mechanism. Fourth, with the multilayer neural network, the abnormality score of edges within the timestamp is calculated by the scoring function.

The rest of the paper is organized as follows: in Section 2 related works on detecting edge anomalies in dynamic graphs are reviewed. We introduce in Section 3 our proposed anomaly detection framework. Section 4 is dedicated to demonstration of experimental results and analysis. Section 5 concludes this paper with a summary and suggestions for future research.

2. Related Works

The existing methods of edges anomaly detection on dynamic graphs can be classified into three classes such as, non-machine learning, graph embedding, and end-to-end deep learning. Goutlier [5] utilizes a structural connectivity approach for anomaly detection in dynamic graphs. It employs reservoir sampling to maintain a summary of the graph's core structural properties. This method presents the following challenges. The practical implementation of this model for large-scale graph streams can be computationally expensive, especially if the graph is dynamic and constantly changing.

SpotLight [6] leverages random sketching to differentiate between normal and anomalous data in the sketch space. This approach guarantees a significant distance between these data points. This method presents the following challenges. Over summarizing the graph may lead to the loss of important details in its structure, resulting in the incorrect detection of some small or local anomalies. CM-Sketch [7] is a sketch-based method that incorporates both local and historical graph data for anomaly detection in edges. This method has the following challenges. The compression process may remove important details from the graph that are essential for accurate anomaly detection. StreamSpot [8] is a clustering-based anomaly detection method for dynamic graphs, using a similarity function for heterogeneous graphs and a centroid clustering approach. However, it struggles with scalability in large, complex graphs and may experience performance degradation when edge generation rates are high. CAD [9] detects anomalous edges by analyzing changes in graph structure and edge weights. However, it may struggle to identify macro anomalies in time-based graphs, as it's focused on local anomalies.

Traditional methods for analyzing graphs often struggle to capture the complex, non-linear relationships within the data. To address this limitation, recent research has explored leveraging graph embedding and deep learning techniques. Graph embedding is a powerful tool that transforms complex graph structures into lower-dimensional representations. a growing number of embedding-based methods are being developed specifically to handle dynamic graphs.

NetWalk [10] uses a random walk with an auto-encoder to learn embedding vectors and updates them incrementally. It then applies dynamic clustering for anomaly detection. However, its two-phase approach can't be trained end-to-end, as the embedding and anomaly detection targets are not jointly optimized. Therefore, it is important to use end-to-end deep learning approaches. TADDY [11] uses a transformer with spatial-temporal encoding and negative sampling to detect anomalies in dynamic graphs. However, it faces challenges with high computational costs and extensive parameter tuning due to its use of attention mechanisms. AddGraph [12] uses a graph convolution network and gated recurrent unit to score edges in dynamic graphs, preserving temporal features. However, it struggles with noisy data, leading to a higher false alarm rate by misidentifying noise as anomalies. HVGRAE [13] uses a hierarchical model with a variational graph autoencoder and recurrent neural network for edge reconstruction to detect anomalies. However, its hierarchical and stochastic design can lead to slow processing and high resource use, especially with big data or real-time applications. StrGNN [14] uses h-hop subgraphs, node labeling, and a graph convolution network with a gated recurrent unit to capture spatial-temporal information. However, it overlooks short-term time dependencies and node features.

In this paper, we propose a model based on the end-to-end deep learning approach that has main differences compared to the existing approaches mentioned above and it solves some of the problems mentioned in previous articles, which are introduced in the next section. In the following, the framework includes the problem statement and the proposed model is explained in detail.

3. Framework

In Section 3.1, the definitions of anomaly detection in dynamic graphs are provided, then four parts of the proposed model are explained in Section 3.2.

3.1. Problem statement

we represent dynamic graphs as a sequence of discrete snapshots captured at specific points in time. The formal definition of a dynamic graph is provided below:

A dynamic graph with a timestamp from $t=1$ to T, can be specified as $G=\{G^t\}_{t=1}^{T}$, where each graph $G^t=(V^t,E^t)$ is the timestamp graph at t, V^t and E^t are the node and edge sets at t, respectively. An edge $e_{i,j}^t=(V_i^t,V_j^t)\in E^t$ shows that there is a link between node V_i^t and V_j^t at time t. We use $n^t=|V^t|$ and $m^t=|E^t|$ to specify the number of nodes and edges at t, respectively. A adjacency matrix A^t is used to define G^t, where $A_{i,j}^t=1$ if there is a connection between nodes V^i and V^j at t, otherwise $A_{i,j}^t=0$. Anomalous edge detection in dynamic graphs is defined as a probability scoring problem. The goal is to learn a function $f(e_{i,j}^t)$ that assigns an anomaly score to each edge, with higher scores indicating greater abnormality.

We use a semi-supervised learning approach for anomaly detection in dynamic graphs, training exclusively on normal edges. Anomalous labels are added to the testing dataset, where 1 indicates an anomalous edge and 0 represents a normal edge.

3.2. Proposed model

Our proposed model consists of four steps, described in the following sections. The overall paradigm of our model is illustrated in Figure 1.

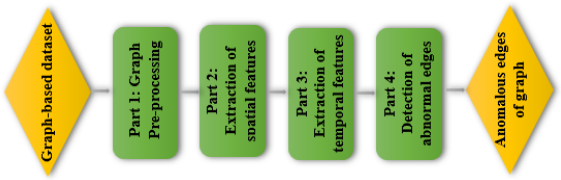

Figure 1. The overall four steps of our model.

A visual illustration of our proposed model is provided in Figure 2.

3.2.1. Part 1: Graph pre-processing

All the edges of the training and test set should be divided into timestamps. It should be ensured that the timestamps length is large enough so that the graph structure appears in each timestamp and the number of timestamps is reasonable so that the time complexity does not increase [15]. To create snapshots (timestamp), timestamps contain an equal number of edges.

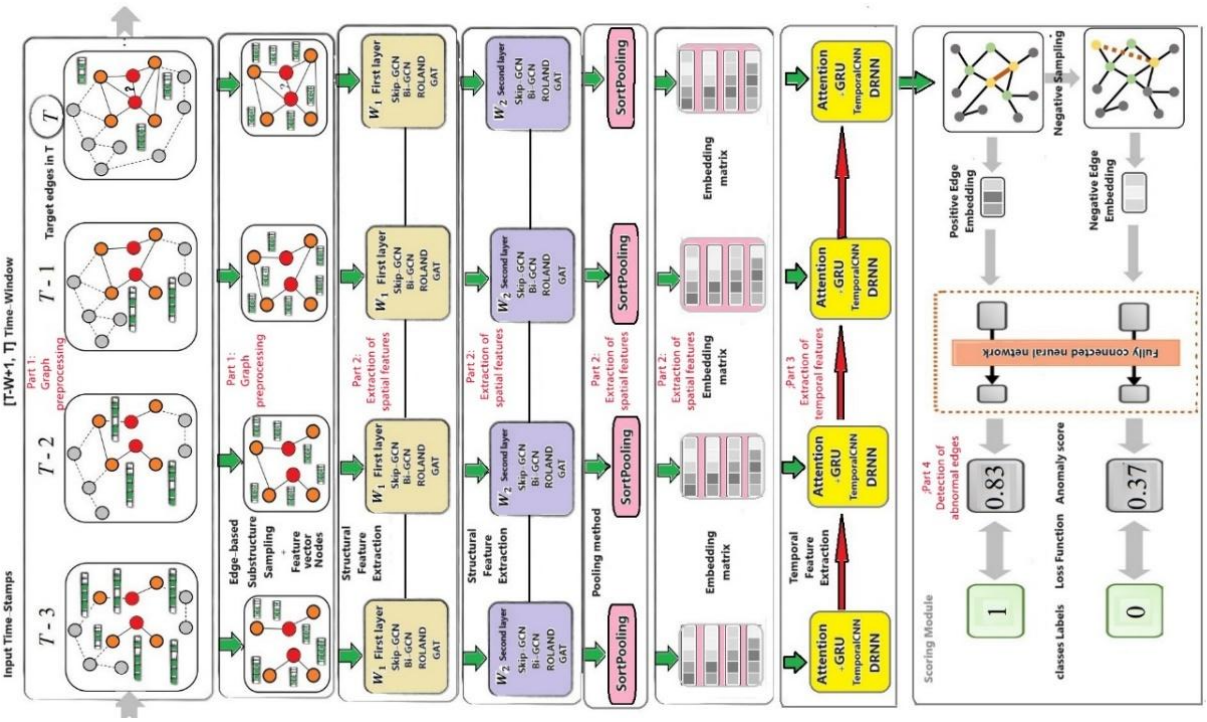

Figure 2. Proposed model for anomaly detection in the dynamic graph.

In this model, the connections (edges) between pairs of nodes (users) at any timestamp are defined as the weight of edges. The edge connected between the pair of nodes has a weight equal to the

number of messages sent by them. To specify labels of the edges at the current timestamp, instead of accessing all past edges, the model is limited to a fixed window of past interactions. The size of the time window (W) is a hyperparameter and determines the received field of the model in the time axis [16]. With the sliding window mechanism, dynamic changes between timestamps $t-w+1$ to t are preserved. The sliding window is used for the purpose that how many of the past timestamps are effective in the current timestamp.
Anomalies often occur in the local subgraphs of the graph, which indicates the receiving field of the proposed model should be enlarged to a suitable local scale [16]. Since in this research, we focus on the detection of anomalous edges, the sampling of the neighbors of the subgraph is done based on the edge. Each edge in the dynamic graph is considered as the center of the sampled subgraph. For each edge in dynamic graphs, the source and destination nodes are specified as target nodes. Other neighbor nodes in the sampled subgraph nodes are referred to as contextual nodes [17]. We extract the h -hop neighbors of the target edges in each timestamp. For a dynamic graph as $\{G(i)=\{V(i),E(i)\}\}_{i=t-w+1}^{t}$ with time window size W, target edge e^t, the $h-$hop enclosing subgraph, source node X and destination node Y, the subgraph associated with the target edge of the dynamic graph in a time window is shown as $\{G(i)_{x^t,y^t}^{h} \mid (t-w+1)\le i\le t\}$.

For popular nodes with high degrees, the number of H-hop neighbors is accompanied by explosive growth. To solve this problem, an upper bound can be used to consider the maximum number of nodes associated with the target nodes. H-hop neighbors sampling ignores the different roles and importance of nodes in the subgraph. To solve this problem, a feature function is used for nodes to assign a unique feature vector to each node. The main weakness of H-hop neighbors sampling is the different sizes of subgraph adjacency matrices, which can be solved using pooling methods.
It is difficult to find feature vectors for each node in the dynamic graph as input for a neural network. To solve this problem and distinguish the role of each node in each subgraph, a node labeling function should well represent the information of each node in each subgraph. We use the labeling function introduced in [14]. The label for each node in the sampled subgraph will be converted into a One-Hot vector as the attribute X for each node. The data samples in the training dataset do not include the GroundTruth label, so a random negative sampling approach is used to train the model with negative and positive pseudo-labels for positive (normal) and negative (abnormal) edges. This approach was used for the first time in the Word2Vec [18] method, where a negative edge is considered for each positive edge. For each timestamp of the graph, whose numbers of edges is m^t, an equal number of pairs of nodes are randomly sampled as candidates for negative pairs. Then, all these pairs of nodes are checked to ensure that they do not belong to the set of positive edges in all timestamps of the training dataset. A new pair is resampled and validation is performed for each node pair until the node pairs are valid.

3.2.2. Part 2: Extraction of spatial features

In this research, at first the spatial dependencies of the graph are extracted and then the temporal dependencies of the dynamic graph are preserved. The convolution layer in the graph neural network is implemented as follows:

$$H_t^{(K+1)}=\sigma(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H_t^{(K)}W_t^{(K)})+H_t^{(K-1)} \quad (1)$$

where W is the trainable weight matrix, W^0 is the initial random weight, $\hat{A}$ is the symmetric normalized adjacency matrix (the graph is assumed to be undirected by default), and $H^0=X$ is the feature matrix of the graph nodes. Each row of the embedding matrix represents the embedding vector for graph nodes. Batch normalization, attention mechanism, deletion rate, and skip connections (residual connections) are used in the proposed graph neural network model. In this research, graph convolution is formulated as aggregation and update function [19]. The trainable weight parameters of the graph convolution neural network are shared between the same level of GCNs with different timestamps.
The number of nodes in different subgraphs is different, which leads to different sizes of the embedding vector in different subgraphs. Training anomaly detection in dynamic graphs using neural networks is challenging due to the different sizes of the input vector. For this purpose, graph pooling methods are used to extract features with fixed sizes for each subgraph. The Sortpooling [20] layer implemented in the proposed model, could sort the nodes in each subgraph based on their importance and select K^{th} of the best nodes.

3.2.3. Part 3: Extraction of temporal features

In the attGRU implemented in the proposed model, temporal dependencies are modeled by GRU. GRU takes the time features in each timestamp as input and passes the output of the current timestamp to the next timestamp, which is considered a local

time feature [21]. The attention mechanism [22] is used to consider global temporal dependencies and assign importance scores to the embedding vectors of graph nodes in different timestamps, and by combining the attention mechanism with GRU, short long-term temporal dependencies can be maintained. The input of the attention layer is a sequence of embedding vectors in consecutive timestamps, and the output of the attention layer is the weighted sequence of embedding vectors of timestamps. A scoring function is used in the attention mechanism to score the embedding vectors obtained in the previous step based on their importance and similarity, and then, the attention function is used to calculate the contextual vector to model the global temporal changes of the data. In combining the attention mechanism with GRU, a contextual attention mechanism is used, which learns the embedding vector score for an arbitrary node. Contextual attention is implemented as follows:

$$\zeta_u^{(t)(\ell)} = \sum_{r=1}^{L} \alpha_u^{(\ell)} h_u^{(t)(\ell)} \tag{2}$$

Where $\zeta_u^{(t)(\ell)}$ is the hidden embedding vector of node *u* after the ℓth layer of attention in timestamp t, and $\alpha_u^{(\ell)}$ shows the importance score for node *u*. The output of the attention mechanism is a node embedding matrix at the graph level, and finally, the embedding vector $h_u^{(t)(\ell)}$ is aggregated with $\zeta_u^{(t)(\ell)}$ and entered into the GRU in the next timestamp.

3.2.4. Part 4: Detection of abnormal edges

In this research, the anomaly scoring function F is a probabilistic function that assigns a score or probability of anomaly to each edge. The timestamps of the test are entered into the proposed model immediately after the training data set in a time window, and the anomaly score is calculated for each edge in each test timestamp. To detect abnormal edges in a directed graph, the source and destination nodes of the target edge should be treated differently, for this purpose, the embedding vector of pairs of nodes are joined as $[h_v^t; h_u^t]$ and finally entered into a fully connected neural network.

The anomaly detection function should be distinguished between positive and negative edge embedding vectors, which use a fully connected neural network layer with a sigmoid activation function. When the model is well trained, the anomaly scores (probability of anomaly) are obtained for each edge of the test, and finally, are entered into the model's loss function and the label of each edge is specified. The anomaly scoring function and the entropy loss function of the proposed model are respectively implemented as follows:

$$f(e) = Sigmoid\left(H_e W_s + b_s\right) \tag{3}$$

$$L^t = -\sum_{i=1}^{m^t} log\left(1 - f\left(e_{pos}, i\right)\right) + log\left(f\left(e_{neg}, i\right)\right) \tag{4}$$

Where e is the target edge, H_e is an embedding matrix, W_s is a trainable weight matrix of the neural network, b_s is a bias, e_{pos} and e_{neg} are positive and negative edges, reopectively. Minimizing the loss function [23] of the proposed model makes the scoring function smaller for $f(e_{pos})$ and larger for $f(e_{neg})$. The loss function along with the regularization function is implemented as $L = L^t + \lambda L_{reg}$. The parameter λ is weight loss and the L_2 function is used to avoid over-fitting. The overall loss function of the model is the sum of the loss functions in timestamps and is implemented as $L = \sum_{1}^{t} L^t$.

4. Experiments

We evaluate our model's anomaly detection capabilities on the UCI and Digg datasets. The data is split 80/20 for training and testing, respectively. To assess performance, ground truth labels are introduced into the test set. These labels are binary (1: anomalous edge, 0: normal edge), and added as a new column. Anomaly injection is employed for further evaluation. During testing for each timestamp (G_t), synthetic anomalies are generated by connecting a specific number of non-adjacent node pairs. The anomaly ratio (p_A) controls the number of injected anomalies, representing the percentage of anomalous edges relative to the original positive edges (m^t) at that timestamp. Three anomaly ratios (1%, 5%, 10%) are introduced to evaluate the model's robustness under varying anomaly prevalence.

4.1. Datasets

The UCI social network dataset is a popular choice for evaluating methods in the field of dynamic graphs due to its practicality. The UCI represents a directed acyclic graph (DAG) of communication within an online student community at the University of California.

Each node corresponds to a user (identified by a unique user number), and a directed edge indicates a message sent from the source user (source node) to the recipient user (destination node). The

presence of multiple edges between users signifies the exchange of multiple messages. Notably, no edges are removed from the dataset. In the UCI dataset, there are no anomalous data samples. The UCI dataset is a dense graph, the ratio of the number of nodes to the available edges is a high number. The weight of each edge is determined by counting the number of edges between a pair of vertices. The growth rate of the graph in the UCI dataset is incremental and abrupt, and short-term time dependencies exist between its generated edges. The data set includes a total of 196 days. The UCI dataset consists of four columns, source node, destination node, edge weight, and Unix timestamp.

The Digg dataset is a graph based on users' responses to the news-social website Digg. Each node in the graph represents a user and each directed edge represents a user's response to another user. The original name of the dataset is munmun_digg_reply and its graph has a loop. The graph of the data set is unweighted multi-directed, each edge is labeled with a Timestamp. The total time of the dataset is 15 days. Digg behaves uniformly and in the Digg dataset, time dependencies are also short-term.

Table 1 indicates the statistical attributes of a datasets that has been used in this paper.

Table 1. Details of the datasets.

Dataset value	UCI	Digg
# of nodes	1899	30398
# of edges	59835	87627
# of unique edges	20296	86404
Maximum degree	1546	310

4.2. Model Assessment Criteria and Configuration metric and settings

We present the evaluation metrics used to compare our model's performance against baseline models.

AUC: AUC, or Area Under the Curve, is a metric used to evaluate the performance of classification models. It considers both the True Positive Rate (TPR) and False Positive Rate (FPR) and summarizes their relationship in a single value. A higher AUC score (closer to 1) generally indicates a more accurate model.

Confusion Matrix: The confusion matrix measures the efficiency of classification problems where the output can be two or more classes. This matrix is a combination of predicted and actual values. The confusion matrix is often used to describe the performance of a classifier on a test set whose ground truth values are known. Two types of error, including False Negative error (Y_{Real} = 1, but $Y_{predict}$ = 0) and False Positive error (Y_{real} = 0, but $Y_{predict}$ = 1) are represented by a confusion matrix. Note that one means abnormality and zero means normal.

4.3. Baselines

We assessed our model's performance by benchmarking its accuracy against baseline models.

GOutlier: It is the first paper published on the topic of anomalous edge detection in dynamic graphs by statistical and probabilistic models. It uses the structural connections approach to discover outliers or anomalous edges in a stream of edges. It uses the graph partitioning method to manage the volume and high speed of input edges to the model [5].

CM-Sketch: This model is proposed for detecting outlier data (abnormal edge) in a stream of edges by statistical and probabilistic models. This model uses the global and local structural dependencies of the stream graph to detect outliers. Sketch-based approximation methods and Count-Min sketch data structures have been used to detect anomalous edges in dynamic graphs [7].

NetWalk: This is the first paper that used deep learning approaches combined with graph embedding methods to detect anomalous edges in dynamic graphs. This model uses an encoder based on a random walk to generate the embedding vectors of the dynamic graph nodes and then models the dynamic graph by dynamic update reservoirs. Finally, an anomaly function based on dynamic clustering is used to score the anomalous edges [10].

The proposed model is implemented on a server with RTX 6000-16GB graphics processor, Intel-Xeon-4214R processor (2.50 GHz), and 32 GB RAM. We use PyCharm version 2024, Python version 3.7, and packages such as Numpy, Pandas, ScikitLearn, MatPlotLib, PyG, Pytorch, and Pytorch Geometric Temporal.

The parameters of the proposed model can be optimized by 5-fold cross-validation. By default, for neighbor sampling, we limit the search to 1- hop neighbors (immediate neighbors) within a time window of size 4. Both the embedding vectors and feature vectors are set to a dimensionality of 32. The graph neural network model utilizes two layers, with the outputs from each layer concatenated to form the final embedding vectors. The proposed model is trained by the Adam optimizer with a learning rate of 0.001 for 100 epochs in 32 batches. Batch normalization and a drop rate of 0.3 are used.

The GRU hidden vector size is set to 256. To create snapshots, each 1000 edges in the UCI dataset is divided into one timestamp. The parameters of the proposed model are updated with the backpropagation algorithm. The LeakyRelu activation function is used for all layers except the output layer of GCN. The aggregation function of GCN is implemented with attention and the update function is a multilayer perceptron.
The sort-polling rate is set to 0.6. The normal distribution has been used for the initial weights of the graph neural network and multilayer neural network.

4.4. Results and Discussion

Results are discussed and the outputs of model will be compared with baseline models. The results of model on the dataset compare with the Goutlier, CM-Sketch, and NetWalk in three cases of 1%, 5%, and 10% anomaly rate, and we use the 80% of datasets for train, and 20% of its for test.
As can be seen from the results (Figure 2 and Figure 3) for the UCI and Digg datasets, the model outperforms all baselines with different anomaly ratios. Compared to the best results for the UCI, the proposed model has performed 10% better on average.
For tes, the number of edges is equal to 4060, and no edge is removed. The number of timestamps is equal to 4 snapshots. The number of anomaly edges in the UCI test set is 40 (4020 normal edges), 203 (3857 normal edges), and 406 (3654 normal edges) in proportions of one percent, five percent, and ten percent, respectively.
Compared to the best results for the Digg, the proposed model has performed more than 10% better on average. For test, the number of edges is equal to 15675, and no edge is removed. The number of timestamps is equal to 3 snapshots.
The parameters and factors that have been changed and tested to reach the best accuracy and parameter value are: edge weight, node feature vector, number of time stamps, learning rate, the number of epochs, initial distribution of neural network weight, types of Graph neural networks, Attention mechanism model, polling methods, pooling rate, number of models layers, aggregation function and activity function.
Table 2 shows the comparison of AUC in the UCI and Digg datasets in three 1%, 5%, and 10% anomaly states.
Table 3 shows an average of all the real and predicted values of the test set confusion matrices with a 1% anomaly ratio for UCI and Digg datasets.

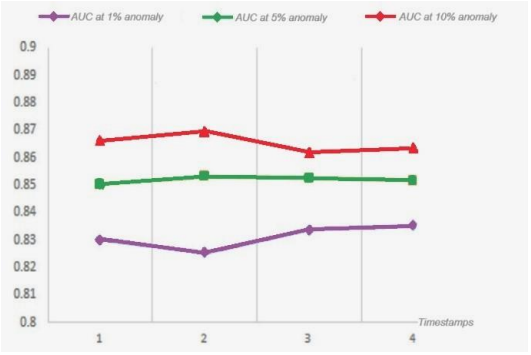

Figure 2. Evaluating AUC values for the proposed model in three cases on the UCI dataset.

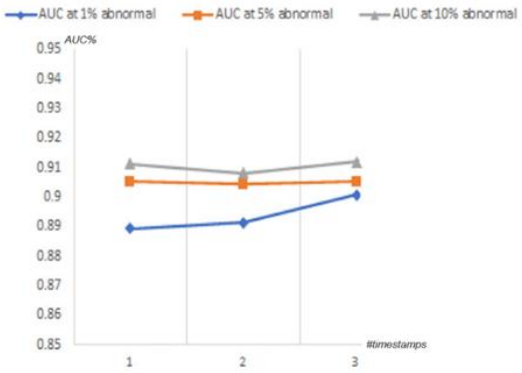

Figure 3. Evaluating AUC values for the proposed model in three cases on the Digg dataset.

Table 2. Comparing AUC values in baseline models.

Criterion	AUC (UCI)			AUC (Digg)		
Method	1%	5%	10%	1%	5%	10%
Goutlier	0.71	0.70	0.67	0.69	0.67	0.63
Sketch	0.72	0.70	0.67	0.68	0.65	0.61
NetWalk	0.77	0.76	0.68	0.75	0.71	0.68
Our model	0.83	0.85	0.72	0.88	0.89	0.89

Table 3. Confusion matrices with a 1% anomaly.

Criterion Dataset	TP	FP	TN	FN
UCI	35	20	4000	5
Digg	154	107	17001	19

In Table 3, TP, FP, TN, and FN are abbreviations of True Positive, False Positive, True Negative, and False Negative, respectively.
According to Table 3, in the UCI dataset, 20 edges with the norm label have been wrongly recognized as anomalous data samples. Out of 4020 available normal edges, 20 edges are wrongly detected as abnormal samples, in some applications, the number of false positives can lead to unnecessary warnings or actions that may have negative consequences. In the proposed model, only 0.5% of the total edges of the norm are wrongly identified as abnormal. Also, in the dataset, 5 edges with anomaly labels out of 40 anomalous edges are

wrongly identified as norm data class (false negative). In the proposed model, 12.5% of all abnormal edges are mistakenly recognized as normal. For UCI dataset, precision is 63%, Recall is 87%, and, F1-score is 73%.

Now let's see why our model is better. The training set is divided into timestamps in such a way that the number of timestamps is reasonable so that the time complexity does not increase and the timestamp length is large enough so that the graph structure appears in each time stamp. The proposed model learns the spatio-temporal dependencies of the dynamic graph without removing any information and by learning the embedding vectors simultaneously and end-to-end along with the feature vector of the nodes and preserves the local structural and global temporal dynamics. By considering a time window, the local temporal dynamics are learned in the dynamic graph. To predict the edge the next time, instead of accessing all the past edges, the model is limited to a fixed window w of past interactions, which reduces the time and memory computational complexity.

Because the proposed model is trained with an efficient negative sampling strategy, the robustness of the model is guaranteed under different anomaly ratios. The proposed model has a high true positive rate while keeping the false positive rate relatively low. Each node receives the most influence from its first-hop neighbors, and sampling the first-hop neighbors on each timestamps reduces the time and memory complexity of the model, and the training of the model is done more optimally than when the entire graph is entered into the model.

Due to the use of synthetic feature vectors for dynamic graph nodes, attributed graphs are supported in this model. By creating the initial feature vector for the graph nodes, using the node labeling function, the role of each node in the target edge is determined and richer vectors are obtained. In anomaly detection, the data samples of the normal and anomalous classes are unbalanced, and the challenge of the unbalanced class has been well overcome by the proposed method. It has been tried to make a trade-off between speed (running time of anomaly detection algorithm) and accuracy of the model. In the graph convolutional network used in the proposed model, skip connections have been used along with normalization with attention aggregation function to obtain richer features.

Due to the use of a global-contextual attention mechanism along with GRU, the accuracy of the proposed model has increased significantly compared to other previous research, because the attention mechanism scores the embedding vectors output from the graph neural network based on their similarity.

By using graph pooling and parameter sharing, model tuning is effectively performed, which focuses on model efficiency instead of focusing on the embedding vectors of nodes. Therefore, for graphs with unseen nodes, the model shows high performance. In this case, a more effective node representation can be learned and is not sensitive to the addition and deletion of nodes. The loss function of the proposed model is implemented with the regularization function to avoid overfitting and the direction of the dynamic graph is considered in this model.

5. Conclusion

A critical challenge in graph data analysis lies in effectively representing dynamic graphs. This representation should capture both the structural properties and the temporal evolution of the graph to the greatest extent possible.

In this research, an efficient graph convolution neural network with Attention-GRU was introduced to detect anomaly edges in dynamic graphs. In this research, by using negative sampling, 1-hop sampling neighbors of the target edge, multi-layer graph convolutional neural network with residual layer, and Attention-GRU, we achieved a higher accuracy than previous one. On average, on the UC Irvine messages dataset, with the AUC and confusion matrix evaluation metrics, the proposed model outperforms baseline methods in directed dynamic graphs by 10 percent on average. The proposed model has been evaluated on another dataset (Digg) and compared to previous research, higher accuracy has been obtained.

The following suggestions are proposed for future works: Generalizing abnormal edge detection in dynamic graphs to abnormal trend and event detection in dynamic graphs, using meta-heuristic methods to optimize objective function parameters, combining statistical and probabilistic learning models with graph neural networks to model uncertainty with the help of hidden random variables, using GAN with self-supervised learning for anomaly detection in continuous time dynamic graph.

References

[1] M. Khazaei, and N. Ashrafi-Payaman. "An Unsupervised Anomaly Detection Model for Weighted Heterogeneous Graph", *Journal of AI and Data Mining*, vlo. 11, no. 2, pp. 237-245, 2023.

[2] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly detection: A survey", *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1-58, 2009.

[3] V. Yepmo, G. Smits, and O. Pivert. "Anomaly explanation: A review", *Data & Knowledge Engineering*, vol. 137, p. 101946, 2022.

[4] A. Sgueglia, A. Di Sorbo, C. Aaron Visaggio, and G. Canfora", A systematic literature review of IoT time series anomaly detection solutions." *Future Generation Computer Systems*, vol. 134, pp. 170-186, 2022.

[5] C. Aggarwal, Y. Zhao, and S. Yu Philip. "Outlier detection in graph streams", In 2011 *IEEE 27th international conference on data engineering*, 2011, pp. 399-409.

[6] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra. "Spotlight: Detecting anomalies in streaming graphs", *In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1378-1386.

[7] S. Ranshous, S. Harenberg, K. Sharma, and N. F. Samatova. "A scalable approach for outlier detection in edge streams using sketch-based approximations", *In Proceedings of the 2016 SIAM international conference on data mining, Society for Industrial and Applied Mathematics*, 2016, pp. 189-197.

[8] E. Manzoor, M. Sadegh Milajerdi, and L. Akoglu. "Fast memory-efficient anomaly detection in streaming heterogeneous graphs", *In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1035-1044.

[9] K. Sricharan, and K. Das. "Localizing anomalous changes in time-evolving graphs", *In Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1347-1358.

[10] W. Yu, W. Cheng, C. Aggarwal, K. Zhang, H. Chen, and W. Wang. "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks", *In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2672-2681.

[11] Y. Liu, S. Pan, Y. Guang Wang, F. Xiong, L.Wang, Q. Chen, and V. Lee. "Anomaly detection in dynamic graphs via transformer", *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12081-12094, 2021.

[12] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao. "AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN", *In IJCAI*, vol. 3, p. 7, 2019.

[13] C. Yang, L. Zhou, H. Wen, Z. Zhou, and Y. Wu. "H-VGRAE: A hierarchical stochastic spatial-temporal embedding method for robust anomaly detection in dynamic networks", *arXiv preprint*, arXiv:2007.06903, 2020.

[14] L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen. "Structural temporal graph neural networks for anomaly detection in dynamic graphs", *In Proceedings of the 30th ACM international conference on Information & Knowledge Management*, 2021, pp. 3747-3756.

[15] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. "Graph neural networks: A review of methods and applications", *AI open Journal*, vol. 1, pp. 57-81, 2020.

[16] B. Jiandong, J. Zhu, Y. Song, L. Zhao, Z. Hou, R. Du, and H. Li. "A3t-gcn: Attention temporal graph convolutional network for traffic forecasting", *ISPRS International Journal of Geo-Information*, vol. 10, no. 7, p. 485, 2021.

[17] A. Longa, V. Lachi, G. Santin, M. Bianchini, B. Lepri, P. Lio, F. Scarselli, and A. Passerini. "Graph Neural Networks for temporal graphs: State of the art, open challenges, and opportunities", *arXiv preprint*, arXiv:2302.01018, 2023.

[18] Z. Yang, G. Zhang, J. Wu, J. Yang, Q. Sheng, S. Xue, C. Zhou et al. "A Comprehensive Survey of Graph-level Learning", *arXiv preprint*, arXiv: 2301.05860, 2023.

[19] J. Skarding, B. Gabrys, and K. Musial. "Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey", *IEEE Access*, vol. 9, no. 3, pp. 79143 - 79168, 2021.

[20] C. Li, Y. Liu, and L. Zou. "DynGCN: A dynamic graph convolutional network based on spatial-temporal modeling", *In Web Information Systems Engineering–WISE 2020: 21st International Conference, Amsterdam, The Netherlands*, October 20–24, 2020, Proceedings, Part I 21, 2020, pp. 83-95.

[21] W. Hamilton, Z. Ying, and J. Leskovec. "Inductive representation learning on large graphs", *Advances in neural information processing systems*, vol 30, 2017.

[22] A. Deng, and B. Hooi. "Graph neural network-based anomaly detection in multivariate time series", *In Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, pp. 4027-4035, 2021.

[23] R. Morshedi, S. M. Matinkhah, and M. T. Sadeghi, "Intrusion Detection for IoT Network Security with Deep learning", *Journal of AI and Data Mining*, vol. 12, no. 1, pp. 37-55, 2024.

تشخیص ناهنجاری در گراف پویا با استفاده از الگوریتم‌های یادگیری ماشین

پوریا ربیعی و نصرتعلی اشرفی پیامن*

[۱]گروه آموزشی مهندسی برق و کامپیوتر، دانشکده فنی و مهندسی، دانشگاه خوارزمی، تهران، ایران .

ارسال ۱۴۰۳/۰۳/۲۲؛ بازنگری ۱۴۰۳/۰۸/۱۶؛ پذیرش ۱۴۰۳/۰۸/۲۱

چکیده:

امروزه نرخ تولید داده‌های با ساختار گرافی با سرعت زیادی در حال افزایش است. تشخیص ساختارهای ناهنجار در گراف مثل گره‌ها و یال‌هایی که رفتار آنها با رفتار مورد انتظار شبکه انحراف دارد در کاربردهای واقعی دارای اهمیّت زیادی می‌باشد. در این کار تحقیقی، ابتدا ویژگی‌های ساختاری گراف پویا توسط شبکه‌های عصبی پیچشی استخراج شده و سپس توسط شبکه عصبی زمانی مثل GRU، ویژگی‌های زمانی کوتاه مدت گراف پویا استخراج می‌شوند و در ادامه با استفاده از مکانیسم توجه تجمیع شده با GRU، وابستگی‌های زمانی طولانی مدت در نظر گرفته می‌شوند. در نهایت با استفاده از یک طبقه‌بندی کننده مبتنی بر شبکه عصبی، یال ناهنجار در هر مهر زمانی تشخیص داده می‌شود. نتایج تجربی روی دو مـــجموعه داده UC Irvine messages و Digg و در مقایسه با سه روش مرجع Goutlier، Netwalk و CMSketch نشان می‌دهد که روش پیشنهادی در این مقاله دارای دقت بهتری در تشخیص یال‌های ناهنجار بوده و میزان بهبود دقت برای دو مجموعه داده UC Irvine messages و Digg به ترتیب ٪۱۰ و ٪۱۵ می‌باشد. در این پژوهش همچنین مدل پیشنهادی با معیارهای AUC و ماتریس درهمی برای ترزیق ناهنجاری به میزان ۱، ۵ و ۱۰ درصد ارزیابی شده است.

کلمات کلیدی: تشخیص ناهنجاری، یادگیری ماشین، گراف پویا، شبکه عصبی گرافی، تشیخص ناهنجاری مبتنی بر گراف.