Anomaly Detection for CyberSecurity Using Convolutional Graph Neural Network Compared to Graph Analysis and Graph Embedding

Amani Abou Rida[1], Rabih Amhaz[2], and Pierre Parrend[3]

University Strasbourg, CSTB Team ICUBE laboratory - UMR CNRS 7357, France
abou-rida.amani@etu.unistra.fr
amhaz@unistra.fr
parrend@unistra.fr

Abstract. In the face of continuous cyberattacks, many scientist have proposed machine learning-based network anomaly detection methods[1]. While deep learning effectively captures unseen patterns of Euclidean data, there is a huge number of applications where data are described in the form of graphs[2]. Graph analysis have improved detecting anomalies in non-Euclidean domains, but it suffered from high computational cost. Graph embeddings have solved this problem by going into low dimensional embeddings, but it lacks the ability of generalizing to unseen nodes. Graph convolution neural network GraphSAGE method have solved this problem by its inductive framework. In this paper we will see how graph convolutional neural network improved the performance of detecting anomalies over the traditional graph analysis and graph embedding methods.

Keywords: anomaly detection, graph convolutional neural network, graph embedding, graph analysis, GraphSAGE, FastRP, Node2Vec, SLLPA, KNN, Link Prediction, Neo4j

1 Introduction

Anomaly detection is any method for finding events that don't match a given expectation[3]. In the face of continuous cyberattacks, many scientist have proposed machine learning-based network anomaly detection methods such as one-class support vector machines (OSVM), autoencoder (AE), and isolation forest (IS)[1].

These methods typically consider that instances are separated and identically scattered. However, in many real-world applications, instances are usually explicitly linked with each other, they have graph structures. While deep learning definitely captures hidden patterns of Euclidean data, there is an increasing number of applications where data are expressed in the form of graphs[2]. Unlike the formal grid-like Euclidean space data (images, audio and text), the previous network data are from irregular non-Euclidean domains[4]. For that, a graph can be used as an effective tool to describe and model the complex structure of network data. Clearly, a graph G(V, E) is typically defined as a set of vertices indicated by V, and certain edges indicated by E between different vertices [2]. Currently, graph-structured data are progressively used to model complex systems, ranging from social media networks [5], traffic networks [6] to financial nets [7]. Concurrently, detecting anomalies from graph has become an important research problem [8].

Graph analytic is used for advanced quantitative considerations and control of complicated networks, but traditional methods suffer from high computational cost and excessive memory conditions [2]. Graph embedding methods can be effective in transforming from high-dimensional sparse graphs into low-dimensional, dense and continuous vector spaces, keeping maximally the graph structure properties[2]. However, Graph embedding covers various kinds of methods targeting the same task only. Therefore, graph convolutional neural network which are designed for various tasks can address the graph embedding problem through a graph auto encoder framework [4]. The main objective of this research is to further investigate an anomaly detection system that is suitable for detecting anomalies using graphs having the best performance. To accomplish this objective, the research addresses the following question:

- How can graph convolutional neural network improve the prediction performance in anomaly detection over the traditional graph analysis and graph embedding methods?

2 State of the art

In this section we will present how graph analytic and graph embedding were used for detecting anomalies. We will discuss the benefits and limits of the proposed methods, and how Graph Neural Network can address these limitations.

2.1 Non Euclidean Space

Definition and benefits: As we mentioned before there is an increasing number of applications where data are represented as a graph that means going from Euclidean to non-Euclidean space. However, in order to perform analysis on non-Euclidean space graph analysis methods have came into. Graph analytic (also known as network analysis) has become an exciting and impacting research area in recent years. Developing effective and efficient graph analytic can greatly help to better quantitative understanding and control of complex networks [2].

Methods: Graph analytic methods such as connectivity analysis, community detection analysis and centrality analysis, are largely based on extracting hand-crafted graph topological features of nodes and edges directly from the adjacency matrices. In our work we have used the Speaker-Listener Label Propagation Algorithm (SLLPA). SLLPA is an improvement of the Label Propagation algorithm that is capable of detecting multiple communities per node [9].

Limitations: When we apply these methods to large-scale network analysis in industrial systems, they may suffer from high computational cost and excessive memory conditions as a result of high-dimensionality and showing heterogeneous characteristics of the original networks [10]. In addition, the hand-engineered features are often task-specific and cannot bring identical performance while employing them for other tasks [2].

2.2 Low dimensionality

Definition and benefits: Graph embedding techniques have shown important role for the capacity of transforming high-dimensional sparse graphs into low-dimensional, dense and continuous vector spaces. The main purpose of graph embedding methods is to encode nodes into a latent vector space, pack every node's properties into a vector with a lower dimension.

Methods: Graph embedding methods used points to three main categories [11]: matrix factorization-based methods, random walk-based methods, and neural network-based methods [11]. In this paper we will focus on Random walk based method mainly Node2Vec, and fast random projection (FastRP). Node2Vec is a node embedding algorithm that measures a vector representation of a node where the neighborhood is sampled through random walks [12]. FastRP is a scalable and performance algorithm for learning distributed node representations in a graph [13].

Limitations:

1. They cannot easily scale up to large network embeddings and only consider local connections.
2. Do not articulate enough to take the diversity of connectivity patterns seen in networks.
3. Dynamic graph embedding mainly focus on the evolution of graphs and ignore the similarities among them [14].
4. Do not have node features that can be generalized to unseen nodes.

2.3 Graph Neural Network

Definition and benefits: Graph Neural Networks (GNNs) are a significant stride to operate precisely on graph-structured data, and a promising method to solve these above limitations. GNNs are actually a message passing in other words neighborhood node aggregation scheme, where every node aggregates feature information of its neighbors to measure its new feature vector. After various iterations of information aggregation, the feature vector of a node will arrest the structural information among the node's neighborhood [8]. GNNs have non-linear activation functions and parallelization skills, which can solve the data non-linearity and the computational complexity problems, respectively [15].

Methods: Graph neural networks (GNNs) are classified into recurrent graph neural networks (RecGNNs), convolutional graph neural networks (ConvGNNs), graph autoencoders (GAEs), and spatial-temporal graph neural networks (STGNNs) [4]. Although GNNs have established outstanding performance in many graph mining tasks [5][6][16], it remains unclear how to accomplish their potentiality for graph anomaly detection GAD problem [15].

Convolutional graph neural networks (ConvGNNs) ConvGNNs acquire the movement of convolution from grid data to graph data [4]. ConvGNNs play an important role in building up many other complex GNN models [4]. In recent years, some convolutional neural network methods for learning over graphs have been proposed. These methods do not scale to large graphs or are designed for whole-graph classification (or both) [17] [18] [19]. However, GraphSAGE method [19] adopts sampling to obtain a fixed number of neighbors for each node.

GraphSAGE Model GraphSAGE is graph convolution neural network that can be used for detecting anomalies. GraphSAGE is an inductive algorithm for computing node embeddings. GraphSAGE is applying node feature information to achieve node embeddings on unseen nodes or graphs [20]. Rather than training individual embeddings for every node, the algorithm learns a function that achieves embeddings by sampling and aggregating features from a node's regional neighborhood [19] as shown in Figure 1. GraphSAGE adopts sampling to obtain a fixed number of neighbors for each node [4]. It performs graph convolutions according to 1:

$$h_v^{(k)} = \sigma(W^{(k)} \cdot f_k(h_v^{(k-1)}, \{h_u^{(k-1)}, \forall u \in S_N(v)\})) \tag{1}$$

where $h_v^{(0)} = x_v$, $f_k(\cdot)$ is an aggregation function, $S_N(v)$is a random sample of the node v's neighbors [4].

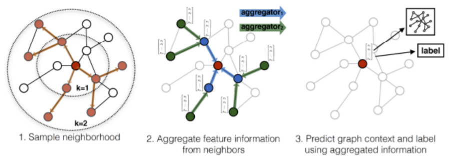

k=1
k=2
1. Sample neighborhood
aggregator
aggregator
2. Aggregate feature information from neighbors
label
3. Predict graph context and label using aggregated information

Fig. 1: Visual illustration of the GraphSAGE sample and aggregate approach [19].

3 Requirements

The first step for identifying the right model to detect anomalies using graph convolutional neural network is to define the requirements of such a model. The model aims to detect anomalies in graphs using graph convolutional neural network methods. These requirements are derived from the literature [4] and from our experience, and are summarised in this section. Detecting anomalies using ConvGNNs models should have the following requirements:

- *Dynamicity*: Graph is inherently dynamic, hence inductive frameworks are more appropriate since they can generalize on unseen data.
- *Scalability*: The dimensions of the graph play a fundamental role in driving the selection of the approach to be applied. Therefore, generation of the embeddings should be as fast and scalable as possible.

- *Inductively*: The model should be able to generate embeddings for some target nodes as soon as new information has been made available.
- *Transferable*: The nodes embeddings should be suitable to be fed into downstream Machine Learning applications.
- *Heterogeneity*: ConvGNNs works on homogeneous graphs. However, having different types and different forms of nodes and edges leads to heterogeneity of the graphs which is not yet handled by ConvGNNs.

4 Anomaly detection using ConvGNNs

In this section we will see how to use graph convolutional neural network for graph anomaly detection. In fact we have used GraphSAGE which is a graph convolution neural network method and it is used as a node embedding technique for non-Euclidean domain. In particular, we see how GraphSAGE works in detecting anomalies compared to graph analysis and graph embedding algorithms such as Fast-RP and Node2vec. The main challenge in this section is to prove that Graph neural network can detect anomalies better than using graph analysis and graph embedding.

4.1 Steps to detect anomalies

In this section we will show the steps needed to detect anomalies according to the above requirements.

1. Load a dataset as a csv, json, or pcap extension inside the neo4j. The standardization data format of the dataset should include the following properties: IP address, start timestamp, last timestamp, PKseqID, and a label attack to see if this node is normal or anomaly. These properties represent the node in the graph.
2. Create the graph composed of nodes and edges using Neo4j. In our work we represent a node as an event. The edges represent a relationship between two events and they are constructed according to the following conditions:
   (a) Events should have the same source IP address.
   (b) The difference between the last timestamp and start timestamp between each event should be less than 20 sec.

   When these two conditions are satisfied between two different events a relationship ”Connected-To” is created as an edge between them.
3. Apply different Graph Data Science methods on the graph created using the Neo4j Data Science Library.
   (a) Apply Graph Analysis: community detection, centrality, and similarity algorithms.
   (b) Apply Graph embedding algorithms: Node2Vec and FastRP.
   (c) Apply Graph Convolutional Neural Network: GraphSAGE.
4. To detect anomalies in the graph; Apply the output of GraphSAGE on the output of Graph Analysis and Graph Embedding to compare the results.

4.2 Model and schema

In this section we will show our schema as represented in figure 2. Our model consist of two parts; Methodology and Results. Our methodology shows that using Neo4j Graph Data Science library, we can apply graph analysis , graph embedding and graph convolutional neural network methods on our created graph. Results shows that using the output of ConvGNNs on graph Analysis and graph embedding can improve the performance of detecting anomalies.

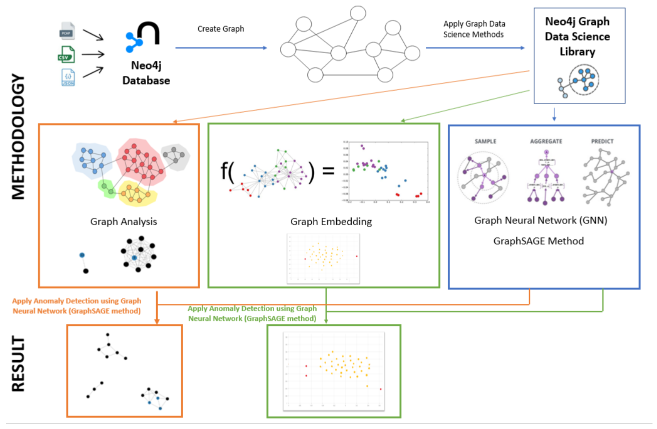

METHODOLOGY
PCAP
CSV
JSON
Neo4j Database
Create Graph
Apply Graph Data Science Methods
Neo4j Graph Data Science Library
f( ) =
Graph Analysis
Graph Embedding
SAMPLE
AGGREGATE
PREDICT
Graph Neural Network (GNN)
GraphSAGE Method
Apply Anomaly Detection using Graph Neural Network (GraphSAGE method)
Apply Anomaly Detection using Graph Neural Network (GraphSAGE method)
RESULT

Fig. 2: Model to detect anomalies

5 Implementation

In this section we will see the software and the dataset used for the evaluation.

5.1 Neo4j

The implementation of detecting anomalies using GraphSAGE is performed through cypher queries for graph data science in Neo4j. Neo4j is an in memory graph that offers an integrated graph database for graph persistence so there's no need to recreate your graph each time it changes [21]. Moreover, Neo4j graph data science library can be used using Neo4j graph data science playground that is called NEuler. All the evaluation are carried out on Neo4j cypher queries and NEuler.

5.2 DataSet

We have imported the IoT-BoT dataset in Neo4j. IoT-BoT dataset has a realistic testbed, a multiple tools being used to carry out several botnet scenarios, and by organizing packet capture files in directories, based on attack types[22]. In order to have some statistics on the IoT-BoT dataset we implemented a Python application that can show us the number of events in total and the number of events being attacked inside the dataset. The number of events is 3668522 and the number of attacked events is 3668045. For simplicity we have taken part of this dataset to do the evaluation.

6 Evaluation

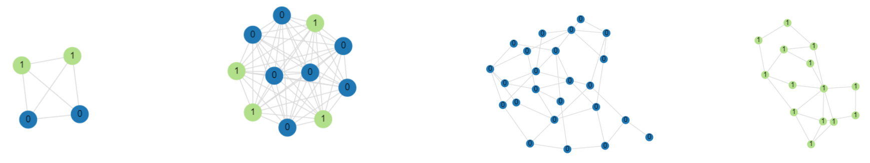
In this section, we perform a set of experiments to evaluate the performance of detecting anomalies using GraphSAGE model compared to different graph analysis and graph embedding algorithms. The evaluation of detecting anomalies using GraphSAGE model is performed for its four core target properties: Dynamicity, Scalability, Inductivity, and Transferability. We have used the IoT-BoT dataset for the evaluation part.

6.1 Dynamicity and Transferability

To evaluate the Transferability of detecting anomalies using GraphSAGE, we will challenge the model for its capability to be fed into downstream Machine Learning applications. We have applied the K-Nearest Neighbors (KNN) algorithm on the output training of GraphSAGE (graph convolution neural network). KNN computes a distance value for all node pairs in the graph and creates new relationships between each node and its k nearest neighbors [23]. Thus applying KNN on GraphSAGE will lead to a new relationship called "SIMILAR-GraphSAGE". This relationship will be used for comparing the results of using SLLPA (graph analysis method) and FastRP (graph embedding) on GraphSAGE output.

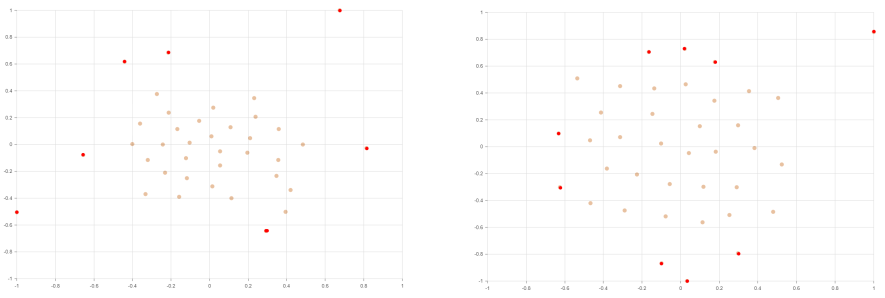
To evaluate the Dynamicity of detecting anomalies using GraphSAGE, we will challenge the model for its capability to detect unseen attacks. First we compared GraphSAGE to graph analysis (SLLPA method). Then we compared GraphSAGE to graph embedding (FastRP method). SLLPA is used to cluster the graphs according to attacked and normal events. Figures 3a and 3b shows the clustering using SLLPA method and clustering using SLLPA on GraphSAGE output respectively. Label 0 on the event means that the event is normal (blue event) and label 1 means that the event is attacked (green event). The result in Figure 3a shows two clusters containing 6 attacked events (2 in the frist cluster and 4 in the second one), while the result in Figure 3b shows two clusters containing 14 attacked events all in the same cluster. Moreover, Figures 4a and 4b shows the embedding using FastRP method and embedding using FastRP on GraphSAGE output respectively. The red events in the figures are the attacked events and the yellow events are the normal ones. The result in Figure 4a shows the detection of 7 attacked event, while the result in Figure 4b shows the detection of 9 attacked events.

This shows that using GraphSAGE we can get more anomalies detected. These anomalies were not seen in both clustering and embedding, in other words



(a) SLLPA (b) SLLPA-GraphSAGE

Fig. 3: Comparing the dynamicity of graph analysis (SLLPA) to graph neural network (GraphSAGE)



(a) FastRP (b) FastRP-GraphSAGE

Fig. 4: Comparing the dynamicity of graph embedding (FastRP) to graph neural network (GraphSAGE)

detecting unseen attacks is proved. The accuracy of clustering is also being improved, since by using GraphSAGE we are getting cluster for the normal events and different cluster for the attacked events. Moreover, although using SLLPA-GraphSAGE method is detecting more anomalies than FastRP-GraphSAGE, we can notice from figure 5 that computational time for FastRP-GraphSAGE is clearly less than the computational time using SLLPA-GraphSAGE method. Thus using GraphSAGE on embedding to detect anomalies have less complexity than using graph analysis.

6.2 Inductivity

The model should be able to generate embeddings for some target nodes as soon as new information has been made available. This means the ability to predict. Link prediction is a common machine learning task applied to graphs: training a model to learn, between pairs of nodes in a graph, where relationships should exist [24]. In our work we have applied Link prediction on graph analysis, graph embedding, and graph convolutional neural network. In addition, we have used two different types of aggregators for GraphSAGE to see their performance: mean and pool aggregators. The GraphSAGE aggregator function

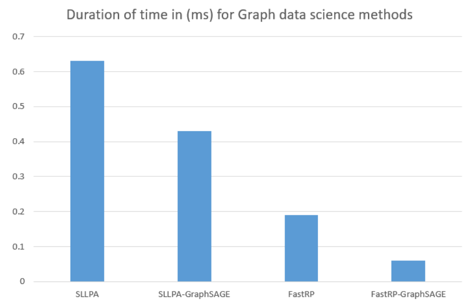


Fig. 5: Computational time for SLLPA, SLLPA-GraphSAGE, FastRP, and FastRP-GraphSAGE

can be customized at will. The simplest architecture consists in the mean operator which aggregates the neighbours' vectors by computing their element-wise mean.The pooling aggregator, instead, uses the neighbours' vectors as input to a fully connected layer before performing the concatenation, and then it applies an element wise max-pooling operation. Link prediction method in Neo4j is used to calculate the precision, recall, and F1-Score. The F1-Score is the harmonic mean of Precision and Recall, this score is widely exploited since it is a trade-off among the previous metrics and consent to have a better understanding of the predictive performance of the model. Table 1 shows the value of the precision, recall, F1-Score, and duration of time for applying link prediction on SLLPA, Node2Vec, FastRP, GraphSAGE-Mean, and GraphSAGE-Pool. It is possible to notice from the results presented in Table 1 that GraphSAGE-Pool is preferable since it has the higher F1-Score with less computational time than the other methods.

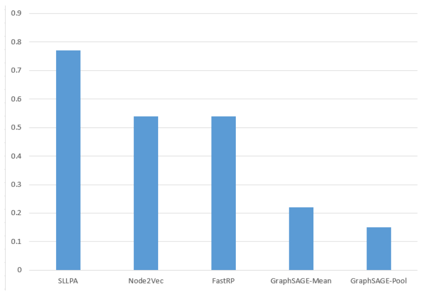
Link prediction	Precision	Recall	F1-Score	Time duration
SLLPA	0.748	0.757	0.752	0.77
Node2Vec	0.742	0.757	0.749	0.54
FastRP	0.749	0.755	0.752	0.54
GraphSAGE-Mean	0.755	0.76	0.757	0.22
GraphSAGE-Pool	0.798	0.817	0.807	0.15

Table 1: Link Prediction Performance on SLLPA, Node2Vec, FastRP, GraphSAGE-Mean, and GraphSAGE-Pool

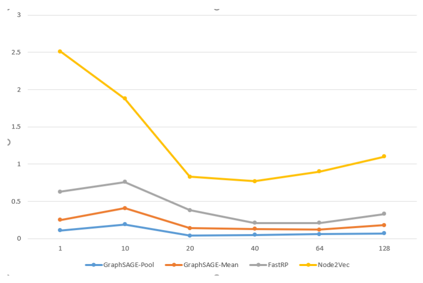
In figure 6a, we shows that applying Link prediction on GraphSAGE-Pool is having the best complexity score in terms of computational time. Thus having new information in our graph will give more accurate result and better performance when using GraphSAGE-Pool.

6.3 Scalability

To evaluate the Scalability of detecting anomalies using GraphSAGE, we will challenge the model for its capability to be faster in detecting anomalies according to different embedding dimensions. The embedding dimensions represent the dimension of the generated node embeddings as well as their hidden layer representations. We have chosen the following dimensions d = 1, d = 10, d = 20, d = 40, d = 64, and d = 128. Figure 6b shows the computational time with respect to different embedding dimensions for Node2Vec, FastRP, GraphSAGE-Mean, and GraphSAGE-Pool. A greater dimension offers a greater precision, but is more costly to operate over. However, when comparing the complexity of Node2Vec and FastRP to GraphSAGE we can notice that GraphSAGE is being more scalable and faster. GraphSAGE-Pool shows some slight improvement in its performance compared to GraphSAGE-Mean.



(a) Computational time for link prediction on SLLPA, Node2Vec, FastRP, GraphSAGE-Mean, and GraphSAGE-Pool



(b) Variation of time with respect to different dimensions for Node2Vec, FastRP, GraphSAGE-Mean and GraphSAGE-Pool

Fig. 6: Scalability of GraphSAGE (mean and pool) compared to SLLPA, Node2Vec, and FastRP

7 Conclusions and Perspectives

Detecting anomalies using GraphSAGE model prove to comply with the requirements for scalability, transferability, dynamicity, as well as inductivity for learning. Experiments showed that graph convolutional neural network GraphSAGE has improved the performance of detecting anomalies compared to graph analysis and graph embedding. This proposal opens a great challenge for the capability of detecting anomalies on heterogeneous graph. This graph have many different types of vertices and many types of edges. And they make the process of calculating embeddings more complicated. For this, a new algorithm in graph neural network should be proposed to study the ability of detecting anomalies on these graphs.

References

1. Q. Xiao, J. Liu, Q. Wang, Z. Jiang, X. Wang, and Y. Yao, “Towards network anomaly detection using graph embedding,” in *International Conference on Computational Science*. Springer, 2020, pp. 156–169.
2. M. Xu, “Understanding graph embedding methods and their applications,” *arXiv preprint arXiv:2012.08019*, 2020.
3. C. Chio and D. Freeman, *Machine learning and security: Protecting systems with data and algorithms.* ” O’Reilly Media, Inc.”, 2018.
4. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.
5. H. Peng, J. Li, Q. Gong, Y. Song, Y. Ning, K. Lai, and P. S. Yu, “Fine-grained event categorization with heterogeneous graph convolutional networks,” *arXiv preprint arXiv:1906.04580*, 2019.
6. B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting,” *arXiv preprint arXiv:1709.04875*, 2017.
7. M. Weber, J. Chen, T. Suzumura, A. Pareja, T. Ma, H. Kanezashi, T. Kaler, C. E. Leiserson, and T. B. Schardl, “Scalable graph learning for anti-money laundering: A first look,” *arXiv preprint arXiv:1812.00076*, 2018.
8. X. Wang, Y. Du, P. Cui, and Y. Yang, “Ocgnn: One-class classification with graph neural networks,” *arXiv preprint arXiv:2002.09594*, 2020.
9. J. Xie, B. K. Szymanski, and X. Liu, “Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process,” in *2011 ieee 11th international conference on data mining workshops*. IEEE, 2011, pp. 344–349.
10. C. Su, J. Tong, Y. Zhu, P. Cui, and F. Wang, “Network embedding in biomedical data science,” *Briefings in bioinformatics*, vol. 21, no. 1, pp. 182–197, 2020.
11. P. Cui, X. Wang, J. Pei, and W. Zhu, “A survey on network embedding,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.
12. 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*, 2016, pp. 855–864.
13. H. Chen, S. F. Sultan, Y. Tian, M. Chen, and S. Skiena, “Fast and accurate network embeddings via very sparse random projection,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 399–408.
14. G. Li and J. J. Jung, “Dynamic graph embedding for outlier detection on multiple meteorological time series,” *Plos one*, vol. 16, no. 2, p. e0247119, 2021.
15. X. Wang, B. Jin, Y. Du, P. Cui, and Y. Yang, “One-class graph neural networks for anomaly detection in attributed networks,” *arXiv preprint arXiv:2002.09594*, 2020.
16. D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi, “A semi-supervised graph attentive network for financial fraud detection,” in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 598–607.
17. J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” *arXiv preprint arXiv:1312.6203*, 2013.
18. D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, “Convolutional networks on graphs for learning molecular fingerprints,” *arXiv preprint arXiv:1509.09292*, 2015.

19. W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.
20. A. Pande, K. Ni, and V. Kini, "Swag: Item recommendations using convolutions on weighted graphs," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 2903–2912.
21. L. Akoglu and C. Faloutsos, "Anomaly, event, and fraud detection in large network datasets," in *Proceedings of the sixth ACM international conference on Web search and data mining*, 2013, pp. 773–774.
22. N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.
23. W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 577–586.
24. D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.