Evaluation of Anomaly Detection for CyberSecurity Using Inductive Node Embedding with Convolutional Graph Neural Networks

Amani Abou Rida, Rabih Amhaz, Pierre Parrend

Evaluation of Anomaly Detection for Cybersecurity Using Inductive Node Embedding with Convolutional Graph Neural Networks

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

[1] Université de Strasbourg, CNRS, ICube (Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie), UMR 7357, 67000 Strasbourg, France
[2] ECAM Strasbourg-Europe, 67300 Schiltigheim, France
[3] EPITA, 5, Rue Gustave Adolphe Hirn, 67000 Strasbourg, France
abou-rida.amani@etu.unistra.fr, amhaz@unistra.fr, parrend@unistra.fr

Abstract. In the face of continuous cyberattacks, many scientists have proposed machine learning-based network anomaly detection methods. 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. Graph analysis have improved detecting anomalies in non-Euclidean domains, but it suffered from high computational cost. Graph embeddings have solved this problem by converting each node in the network into low dimensional representation, but it lacks the ability to generalize to unseen nodes. Graph convolution neural network methods solve this problem through inductive node embedding (inductive GNN). Inductive GNN shows better performance in detecting anomalies with less complexity than graph analysis and graph embedding methods.

Keywords: anomaly detection, convolutional graph neural network, inductive graph learning, graph embedding, graph analysis, link prediction

1 Introduction

Anomaly detection is any method for finding events that don't match a given expectation. In the face of continuous cyberattacks, many scientist have proposed machine learning-based network anomaly detection methods [1] such as one-class support vector machines (OSVM), autoencoders (AE), and isolation forests (IS). These methods have proven to be very effective in detecting anomalies [2] and they discover hidden patterns in Euclidean data [3] where they are being plotted in n-dimensional linear space. However, there is an increasing number of applications where data are expressed in the form of graphs [4]. Unlike the n-dimensional grid-like Euclidean space data (images, audio and text), network data represent irregular non-Euclidean domains [5]. A graph can be used as an effective tool to describe and model the complex structure of network data. A graph G(V, E) is typically defined as a set of vertices indicated by V, and edges indicated by E between different vertices [4]. Graph-structured data are used to model complex systems, ranging from social media networks [6], traffic networks [7] to financial

nets [8]. Consequently, detecting anomalies from graphs has become an important research problem [9]. We obtain rich information from the graph relation structures which provide a natural way to understand links between entities [5]. Graph analysis [4] is used for advanced quantitative characterisation and control of large-scale networks, but traditional methods suffer from high computational cost and excessive memory requirements. Graph embedding [4] methods are effective in transforming high-dimensional graphs into low-dimensional representation of dense and continuous vector spaces, keeping the graph structure properties. However, generating new representations using graph embedding at each analysis and updating all node embeddings is costly. Therefore, scientists proposed to update the embeddings according to the changes incrementally instead of relearning the whole embeddings again. Graph convolutional neural networks can address the graph embedding problem by dealing with graphs incrementally using an inductive node embedding model [10]. This incremental computation ensures its efficiency since each new node only needs to sample and aggregate its neighbor's features. The main objective of this research is to further investigate an anomaly detection system that is suitable for detecting anomalies incrementally having the best performance. To accomplish this objective, the research addresses the following question:

• How can inductive node embedding graph convolutional neural networks (in short Inductive GNNs) improve the prediction performance in anomaly detection over the traditional graph analysis and graph embedding methods?

The paper is organised as follows. Section 2 introduces the state of the art, Section 3 identifies the requirements for the schemes under evaluation. Section 4 defines the process for anomaly detection using Inductive GNNs, Section 5 presents its implementation. Section 6 provides the evaluations and Section 7 discusses the implications of these evaluation. Section 8 concludes this work.

2 State of the art

In this section we present the use of graph analysis and embedding for detecting anomalies. We discuss the benefits and limits of the methods, and how Inductive Node Embedding Graph Convolutional Neural Networks can address them.

Graph analysis [4] is a process for analyzing data in graph structures, in a **Non-Euclidean Space** using data points as nodes and relationships as edges. There is an increasing number of applications where data are represented as a graph with complex relationships and inter-dependency between objects. In other words, analyses are increasingly going from Euclidean (e.g., images, audio and text) to non-Euclidean space with higher number of dimensions. However, to perform analysis on non-Euclidean space, graph analysis methods have came into that can help improving the quantitative understanding and the control of complex networks. Graph analysis methods, such as connectivity analysis, community detection analysis and centrality analysis, are largely based on extracting handcrafted graph topological features of nodes and edges directly from the adjacency matrices. In our work, we use the Speaker-Listener Label Propa-

gation Algorithm (SLLPA) [11]. SLLPA is an improvement of the Label Propagation algorithm that is capable of detecting multiple communities per node. When applied to large-scale network analysis, these methods may suffer from high computational cost and excessive memory requirements as a result of high-dimensionality [12]. In addition, hand-engineered features are often task-specific and cannot bring identical performance when re-used for other tasks [4].

Graph embedding [4] techniques have shown an important role for the capacity of transforming high-dimensional sparse graphs into **low-dimensional representations**, dense and continuous vector spaces. The main purpose of graph embedding methods is to encode nodes into a latent vector space and to pack every node's properties into a vector with a lower dimension. Graph embedding methods integrate three complementary domains [13]: matrix factorization, random walk, and neural network methods [13]. In this paper we focus on Random walk-based methods, in particular Node2Vec [14] and fast random projection (FastRP) [15]. Node2Vec is a node embedding method that measures a vector illustration of a node where the neighborhood is sampled using random walks [14]. FastRP is a scalable and performant algorithm for preserving similarity between nodes and their neighbors. This means that two nodes that have similar neighborhoods should be assigned similar embedding vectors [15]. The limitations of graph embedding methods are:

1. They cannot easily scale up to large network embeddings and only consider local connections.
2. The model should be able to generate embeddings for some target nodes as soon as new information has been made available.
3. The generation of a representation using graph embedding at each time and updating all the node embeddings is costly.

Graph Neural Networks (GNNs) [5] are a significant stride to operate precisely on graph-structured data, and a promising method for solving above limitations. The features of the neighbours of a graph node are aggregated and passed as a message to that node. After different aggregation iterations, the feature vector of a node extracts the structural information from the node's neighborhood [9]. The final result, i.e., the aggregated information obtained at each node, is referred to as the node embedding. GNNs have non-linear activation functions and parallelization skills, which can solve the data non-linearity and the computational complexity problems, respectively [16]. Graph neural networks (GNNs) are classified into 4 categories: recurrent graph neural networks (RecGNNs), convolutional graph neural networks (ConvGNNs), graph autoencoders (GAEs), and spatial-temporal graph neural networks (STGNNs) [5]. Although GNNs have established outstanding performance in many graph mining tasks [6, 7, 17], it remains unclear how to accomplish their potentiality for Graph Anomaly Detection [16].

Convolutional Graph Neural Networks (ConvGNNs) acquire the movement of convolution from grid data (Euclidean structure) to graph data (non-Euclidean structure) [5]. ConvGNNs play an important role in building up many other complex GNN models [5]. They fall into two categories, spectral-based and spatial-

based. Spectral based approaches specify graph convolutions by proposing filters from graph signal processing [18]. Spatial-based approaches perform graph convolutions locally on each node where weights can be easily shared across different locations and structures [19]. In recent years, some ConvGNNs methods for learning over graphs have been proposed. These methods do not scale to large graphs or are designed for whole-graph classification [20–22]. However, an inductive node embedding method is needed to adopt sampling so as to provide a fix number of neighbors for each node [22]. In the next section we introduce and define Inductive GNNs. Inductive GNNs are an inductive way to iteratively update the node embeddings. It follows an embedding propagation schema where the embedding of a node is recursively updated by aggregating values propagated from its neighboring nodes. Inductive GNN apply node feature information to achieve node embeddings on unseen nodes or graphs [23]. 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 [22]. This means that inductive GNNs are able to update the embeddings according to the changes incrementally instead of relearning the embeddings whole again. 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 [5].

3 Requirements

The first step for identifying the right model to detect anomalies using Inductive GNNs is to define the requirements of such a model. The model aims at detecting anomalies in graphs using graph convolutional neural network methods. These requirements are derived from the literature [22, 5, 10, 24, 25] and from our experience, Detecting anomalies using Inductive GNNs model should have the following requirements:

- *Transferability*: The nodes embeddings can be fed into downstream Machine Learning methods [24].
- *Scalability*: Generation of the embeddings should be as fast and scalable as possible. This is measured according to two properties: 1) Dimensionality [5]: Dimensions of the embedding play a fundamental role in the selection of the approach to be applied. 2) Inductive embedding generation [22]: Iterative generation of node embedding as soon as new information have been made available without retraining the node embedding of all the graph again.

4 Anomaly detection using Inductive GNNs

In this section we see how to use Inductive GNNs for graph anomaly detection.In particular, Inductive GNNs based methods for detecting anomalies can update

the embeddings according to the changes incrementally instead of relearning the embeddings whole again. Moreover, we compare Inductive GNNs with graph analysis and graph embedding algorithms such as FastRP and Node2vec to evaluate their relative anomaly detection capabilities.

The steps needed to detect anomalies according to the above requirements are the following ones:

1. Load a dataset as a csv, json, or pcap extension inside the graph database. The standardized data format of the dataset should include the following properties: IP address, start timestamp, end timestamp, PKseqID, and a label attack to determine if this node is normal or anomalous. These properties represent the node in the graph.
2. Create the graph composed of nodes and edges. 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 end 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.
   (a) Apply Graph Analysis: community detection, centrality, and similarity algorithms.
   (b) Apply Graph embedding algorithms: Node2Vec and FastRP.
   (c) Apply Graph Convolutional Neural Network: Inductive GNNs.
4. To detect anomalies in the graph, apply the output of Inductive GNNs on the output of Graph Analysis and Graph Embedding to compare the results. Anomaly detection should there exhibit better performance and more accurate results than the traditional methods.

Our model consists of three parts, as shown in Figure 1: Input, process, and output. The input of our model is the dataset that is converted into graph and stored in a graph database. The process consists in applying graph analysis, graph embedding and Inductive GNNs methods on the graph. The output shows that the performance of detecting anomalies using Inductive GNNs on graph analysis and graph embedding is improved according to the performance and accuracy.

5 Implementation

We consider 2 datasets for the evaluation: BoT-IoT [26] and UNSW-NB15 [27]. They are created by designing a realistic network environment in the Cyber Range Lab of UNSW Canberra. In order to have some statistics on these datasets we implemented "Data Summary" (`data_sum`), a Python application that extracts the total number of records and the number of attack records inside a dataset. The number of records of BoT-IoT is 72,388,626 and the number of attack records is 30,668,045. The types of attacks are: DDoS, DoS, OS and Service

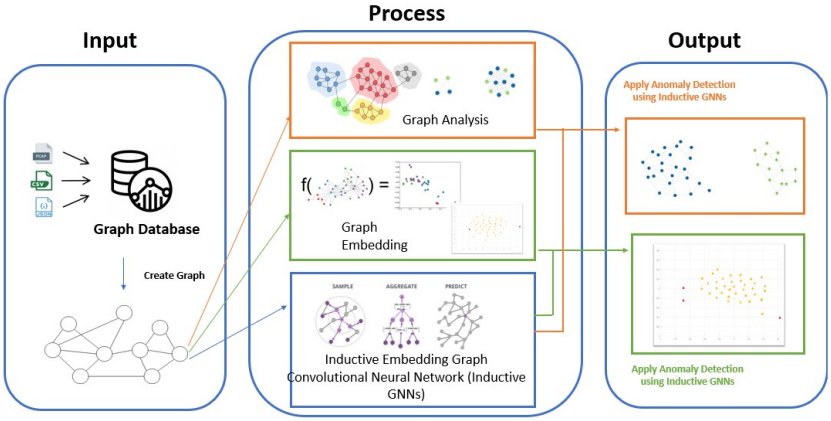


Fig. 1: Process for detecting anomalies using Inductive GNNs

Scan, Keylogging and Data exfiltration. The number of records of UNSW-NB15 dataset is 540,044 and the number of attack records is 164,673. The types of attacks are: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.

Neo4j[4] is an in memory graph handling environment that offers an integrated graph database for persistence so there is no need to recreate the graph each time it changes [28]. Moreover, Neo4j offers a graph data science library that can be used for our process to apply graph analysis, graph embedding, and Inductive GNNs methods, in particular the GraphSAGE library which is the reference implementation. Detecting anomalies using GraphSAGE is performed through cypher graph query language [29] which is used to query the Neo4j graph database for creating, storing, performing graph data science libraries.

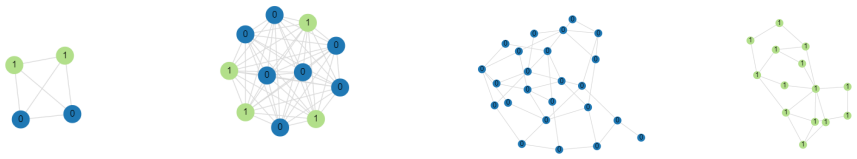
The hyperparameter used in Neo4J GraphSAGE implementation are: dimension d = 64, sample normalization vector S = [25,10], and aggregator function. GraphSAGE provides in particular GraphSAGE-Mean and GraphSAGE-Pool aggregation strategies. The mean operator aggregates the neighbours' vectors by computing 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 elementwise max-pooling operation.

6 Evaluation

In this section, we perform a set of experiments to evaluate the performance of anomaly detection using Inductive GNNs model compared to different graph analysis and graph embedding algorithms. The evaluation of anomaly detection is performed for the two core target properties: Transferability, and Scalability.

[4] https://neo4j.com/

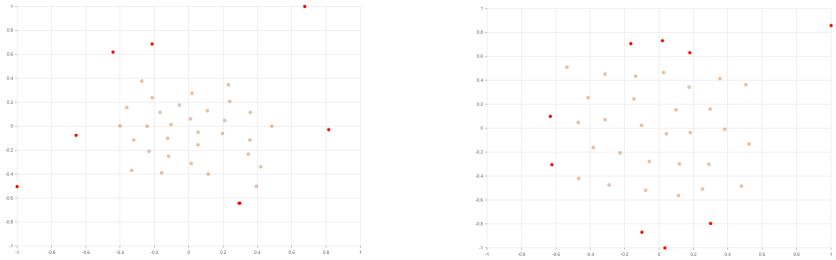
To evaluate the **Transferability** of detecting anomalies using Inductive GNNs, we challenge the model for its capability to be fed into downstream Machine Learning applications. We apply the K-Nearest Neighbors (KNN) algorithm on the output training of GraphSAGE. KNN computes a distance value for all node pairs in the graph and creates new relationships between each node and its k nearest neighbors [30]. This operation leads to a new relationship called "SIMILAR-GraphSAGE". This relationship is used as an output of GraphSAGE and it is used for comparing the results of using SLLPA (graph analysis method) and FastRP (graph embedding) to SLLPA-GraphSAGE and FastRP-GraphSAGE methods as shown in Figures 2 and 3. First we compare GraphSAGE to graph analysis (SLLPA method). Then we compare GraphSAGE to graph embedding (FastRP method). SLLPA is used to cluster the graphs according to attack and normal events. Figures 2a and 2b 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) and label 1 means that the event is an attack (green). The result in Figure 2a shows two clusters containing 6 attack events (2 in the first cluster and 4 in the second one), while the results in Figure 2b shows two clusters containing 14 attack events all in the same cluster. Moreover, Figures 3a and 3b show the embedding using FastRP method and embedding using FastRP on GraphSAGE output respectively. The red events in the figures are the attack events and the yellow events are the normal ones. The result in Figure 3a show the detection of 7 attack event, while the result in Figure 3b show the detection of 9 attack events.



(a) SLLPA (b) SLLPA-GraphSAGE

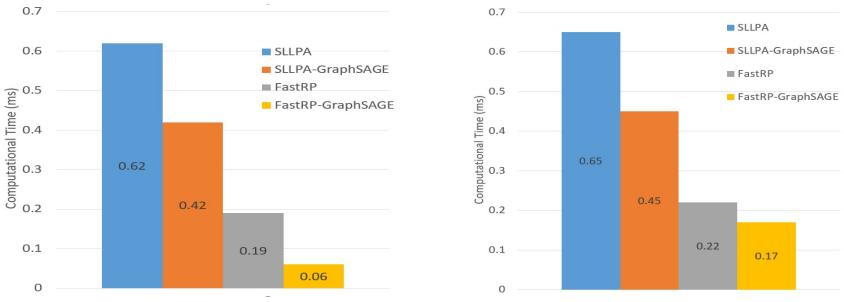
Fig. 2: Comparing the dynamicity of graph analysis (SLLPA) to graph neural network (GraphSAGE implementation)

The generation of the embeddings should be as fast and scalable as possible. **Scalability** is measured according to two main properties: 1) Inductive embedding generation [22] and 1) Embedding Dimensions [5]. The model should be able to generate embeddings in an iterative way as soon as new information has been made available without re-running the embedding all over again. To evaluate the scalability according to inductivity we use Link prediction method to compute the time and F1 score for graph analysis, graph embedding, and graph convolutional neural network methods. Link prediction is a common machine learning task applied to graphs [31]: training a model to learn, between pairs



(a) FastRP (b) FastRP-GraphSAGE

Fig. 3: Comparing the dynamicity of graph embedding (FastRP) to graph neural network (GraphSAGE implementation)



(a) Computational time for graph analysis (SLLPA, SLLPA-GraphSAGE) and graph embedding (FastRP, FastRP-GraphSAGE) for BoT-IoT dataset

(b) Computational time for graph analysis (SLLPA, SLLPA-GraphSAGE) and graph embedding (FastRP, FastRP-GraphSAGE) for UNSW-NB15 dataset

Fig. 4: Computational time for graph analysis (SLLPA, SLLPA-GraphSAGE) and graph embedding (FastRP, FastRP-GraphSAGE) for BoT-IoT and UNSW-NB15 datasets

of nodes in a graph, where relationships should exist. Moreover, this method 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 and Table 2 show 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 for BoT-IoT and UNSW-NB15 respectively. GraphSAGE-Pool has the highest F1-score 0.807, 0.751, and the less time duration 0.15, 0.08 ms in both BoT-IoT and UNSW-NB15 datasets respectively.

Dimensions of the embedding play a fundamental role in the selection of the approach to be applied. To evaluate the Scalability according to dimensionality we use different embedding dimensions. The embedding dimensions represent

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 for BoT-IoT dataset

Link prediction	Precision	Recall	F1-Score	Time duration
SLLPA	0.743	0.738	0.74	0.65
Node2Vec	0.742	0.751	0.7464	0.26
FastRP	0.751	0.743	0.7469	0.16
GraphSAGE-Mean	0.7516	0.745	0.748	0.17
GraphSAGE-Pool	0.7516	0.7517	0.751	0.08

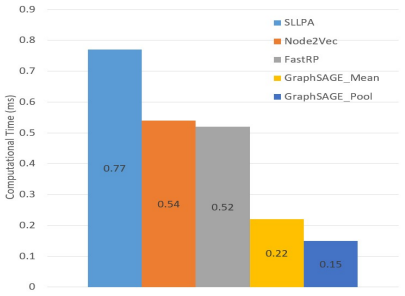
Table 2: Link Prediction Performance on SLLPA, Node2Vec, FastRP, GraphSAGE-Mean, and GraphSAGE-Pool for UNSW-NB15 dataset

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. Figures 5c and 5d shows the computational time with respect to different embedding dimensions for Node2Vec, FastRP, GraphSAGE-Mean, and GraphSAGE-Pool. GraphSAGE-Pool has the best time 0.04 ms and 0.02 ms in both BoT-IoT and UNSW-NB15 datasets respectively.

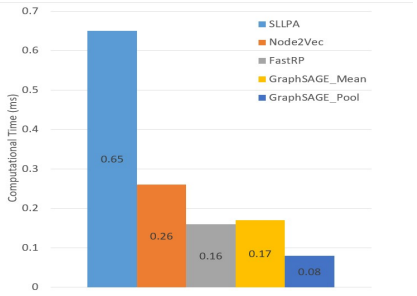
7 Discussion

The evaluation of **Transferability** shows that using Inductive GNNs we can get more anomalies detected. The accuracy of clustering is also being improved, since we are getting one cluster for the normal events and one different cluster for the attack events. Moreover, although using SLLPA-GraphSAGE method is detecting more anomalies than FastRP-GraphSAGE, Figures 4a and 4b show that computational time for FastRP-GraphSAGE is 0.06 ms, 0.17 ms less than the computational time using SLLPA-GraphSAGE 0.42 ms, 0.45 ms in both BoT-IoT and UNSW-NB15 datasets respectively. Thus using inductive Node Embedding to detect anomalies have less complexity than using graph analysis.

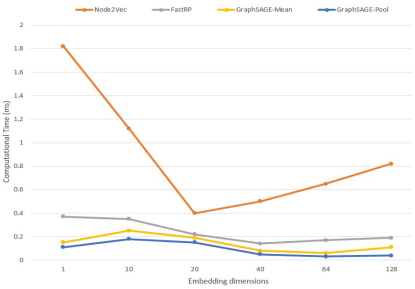
The evaluation of **Scalability** from the Table 1 and Table 2 shows that GraphSAGE-Pool is preferable since it has the higher F1-Score with less computational time than the other methods. Figures 5a and 5b show that Link prediction on GraphSAGE-Pool provides the best complexity score in terms of computational time. Thus having new information in the graph gives more accurate



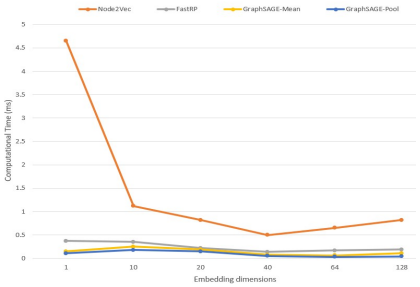
(a) Computational time for link prediction on SLLPA, Node2Vec, FastRP, GraphSAGE-Mean, and GraphSAGE-Pool for BoT-IoT dataset



(b) Computational time for link prediction on SLLPA, Node2Vec, FastRP, GraphSAGE-Mean, and GraphSAGE-Pool for UNSW-NB15 dataset



(c) Variation of time with respect to different dimensions for Node2Vec, FastRP, GraphSAGE-Mean and GraphSAGE-Pool for BoT-ToT dataset



(d) Variation of time with respect to different dimensions for Node2Vec, FastRP, GraphSAGE-Mean and GraphSAGE-Pool for UNSW-NB15 dataset

Fig. 5: Scalability of GraphSAGE (mean and pool) compared to SLLPA, Node2Vec, and FastRP

result and better performance when using GraphSAGE-Pool. The evaluation of scalability according to different dimensions shows that having a greater dimension offers a greater precision, but is more costly to operate over. However, when comparing the complexity of Node2Vec and FastRP to Inductive GNNs one can notice that Inductive GNNs is more scalable and faster. GraphSAGE-Pool shows some slight improvement in its performance compared to GraphSAGE-Mean.

8 Conclusions and Perspectives

Detecting anomalies using inductive Node Embedding with Convolutional Graph Neural Networks proves to comply with the requirements for scalability and transferability. Experiments shows that inductive embedding graph convolutional neural networks improve 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 and dynamic graphs. Heterogeneous graphs have many different types of vertices and many types of edges, and they make the process of calculating embeddings more complicated. Dynamic graphs are the graphs where their nodes/edges may change over time. For this, we should propose new Inductive Node Embedding GNN that is able to detect anomalies on heterogeneous and dynamic 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. J. Tonejc, S. Güttes, A. Kobekova, and J. Kaur, "Machine learning methods for anomaly detection in bacnet networks." *J. Univers. Comput. Sci.*, vol. 22, no. 9, pp. 1203–1224, 2016.
3. P. Barrett, "Euclidean distance: Raw, normalised, and double-scaled coefficients," *Tech. Rep.*, 2005.
4. M. Xu, "Understanding graph embedding methods and their applications," *arXiv preprint arXiv:2012.08019*, 2020.
5. 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.
6. 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.
7. 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.
8. 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.
9. X. Wang, Y. Du, P. Cui, and Y. Yang, "Ocgnn: One-class classification with graph neural networks," *arXiv preprint arXiv:2002.09594*, 2020.
10. Z. Cui, Z. Li, S. Wu, X. Zhang, Q. Liu, L. Wang, and M. Ai, "Dygcn: Dynamic graph embedding with graph convolutional network," *arXiv preprint arXiv:2104.02962*, 2021.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.

16. 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.
17. 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.
18. D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.
19. S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 922–929.
20. J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.
21. 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.
22. 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.
23. 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.
24. A. Martignano, "Real-time anomaly detection on financial data," 2020.
25. D. Yu, Y. Yang, R. Zhang, and Y. Wu, "Knowledge embedding based graph convolutional network," in *Proceedings of the Web Conference 2021*, 2021, pp. 1619–1628.
26. 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.
27. N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.
28. 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.
29. F. Holzschuher and R. Peinl, "Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j," in *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, 2013, pp. 195–204.
30. 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.
31. 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.