IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/44AI/2024/4440424/
Vol. 2024 (22), Article ID 4440424

Selected Graph Machine Learning Models and Methods with Applications in Cybersecurity*

Zbigniew TARAPATA and Jan ROMANCZUK

Military University of Technology, Faculty of Cybernetics, Warsaw, Poland

Correspondence should be addressed to: Zbigniew TARAPATA, zbigniew.tarapata@wat.edu.pl

* Presented at the 44th IBIMA International Conference, 27-28 November 2024 Granada, Spain

Abstract

The paper reviews and describes in detail selected graph machine learning methods as applied to modelling and solving selected cybersecurity problems. Special attention was paid to methods of graph mining (clustering, classification, similarity) and graph neural networks. The basic differences between an "ordinary" neural network and a graph neural network were described. Three types of graph neural networks are described: graph convolutional networks (CGN), graph attention networks (GAT), generative antagonistic networks (GAN). Applications of each group of methods in cybersecurity problems are presented. The presented examples of applications concern with: detection of DDoS attacks or Botnets, security incident management, analysis of user behaviour, detection of security risks, detecting fake news and countering disinformation, malware detection. In summary, the advantages and disadvantages of graph-based machine learning methods are also presented.

Keywords: graph machine learning, graph mining, graph neural networks, cybersecurity problems.

Introduction

Graph Machine Learning (GML) is one of the most rapidly growing branches of science related to Artificial Intelligence (AI). The essence of GML is the use of a graph as a representation (model) of a real-world object, on which a learning process takes place that allows various problems related to that real-world object to be solved. Graph modelling opens a new chapter in artificial intelligence research. By mathematically formulating a slice of reality using not only the raw data, but also the relationships between these data, it is possible to build learning techniques of unprecedented accuracy and generalizability. The techniques used in graph-based machine learning, analogous to classical machine learning, can be divided into (Blumenfeld (2023)):

- supervised;
- unsupervised.

Supervised learning techniques can be further subdivided in terms of the objective that the technique is intended to achieve and the level of analysis of the graph in question. Learning methods of this type can be used to solve problems:

- predicting the features of single nodes;
- predicting the existence of relations (edges) between given nodes;
- classifying the features of the graph as a whole (including, among others, the study of graph similarity).

Cite this Article as: Zbigniew TARAPATA and Jan ROMANCZUK, Vol. 2024 (22) "Selected Graph Machine Learning Models and Methods with Applications in Cybersecurity " Communications of International Proceedings, Vol. 2024 (22), Article ID 4440424, https://doi.org/10.5171/2024.4440424

Supervised methods include, among others, graph deep neural networks, graph spline networks and other supervised ML methods using graphs - almost all of which can be reduced by analogy to the graph problem. Unsupervised learning techniques include methods such as:

- clustering;
- path-finding in a network;
- determining measures of centrality and relevance of individual nodes, or edges, or entire subgraphs.

One of the unsupervised learning methods is the so-called unsupervised feature engineering (as a part of graph mining (Tarapata (2020)). This is a very important area of GML method development for supervised methods, as it focuses on discovering the appropriate selection of data representations for further processing with ML methods. This can be crucial for GML methods and the associated appropriate embedding (dipping) of features (embedding). Graph-based machine learning finds its application in a range of topics: tax fraud detection, recommendations, customer profiling, cracking down on drug trafficking, supply chains, image generation, cyber security (e.g. deep fake generation and detection) and others (Blumenfeld (2023)). In this article, we will focus on providing an overview of graph-based machine learning and the problems solved using it, with a particular focus on cybersecurity problems.

The paper is organized as follows. First, selected graph machine learning methods are described. Next, we describe examples of the use of graph machine learning methods in selected cybersecurity problems. Summary contains the advantages and disadvantages of graph machine learning methods.

Graph machine learning models and methods: overview

Graph Mining

Graph mining is an activity that uses a set of mathematically sophisticated models, methods and tools that often lead to the discovery of new knowledge about these objects. These activities can be used to:

- analysis of the properties of graphs representing real world objects, characterized by structure, i.e. a set of elements related to each other by relations;
- predicting how the structure and properties of a given graph might affect certain applications;
- creating models that can generate realistic graphs that correspond to patterns found in real world graphs of interest.

Graph mining methods are very often use as a computational procedures in graph neural networks. There are most commonly three basic types of problems and related methods for graph mining (sometimes also including link analysis, vertex feature analysis, analysis of parts of a graph (e.g. subgraphs), (Tarapata (2020)):

- cauterization (clustering): the vertices of a graph are grouped together to form clusters (groups) based on the structure of the links between the vertices of the graph (the edges of the graph) and a certain similarity of the substructures of the graphs. Clustering (grouping) is performed in such a way that the links between vertices belonging to different clusters are weaker than the links between vertices within a cluster. Clustering belongs to an unsupervised machine learning technique, because the classes are not known before clustering;
- classification: similar to clustering except that classes are known before classification and the membership of the vertices to the corresponding class must be determined. Classification belongs to a supervised machine learning technique (since the classes are known before classification));
- similarity: this amounts to comparing two or more graphs (one of which is the reference graph) with each other and determining the values of the similarity measures between them, in order to determine which graph is most similar to the reference graph or to create a ranking based on this. A variation of this approach is to search for the largest common part (subgraph) between two graphs. The study of graph similarity belongs to the unsupervised technique of machine learning.

One of the most popular clustering methods in graphs is the Louvain modularity method (Blondel et al (2008)). The essence of the method is to optimize so-called modularity as the algorithm progresses. Modularity is a value on a scale of -1 to 1, which measures the density of edges inside communities to edges outside communities.
When studying the similarity of graphs, one can speak of:

- exact matching: it is related to the notion of isomorphism of graphs (other so-called graph morphisms are also used: monomorphism, automorphism, homeomorphism);
- inexact matching: it allows to assess the level of similarity between graphs. Most often it amounts to mapping both graphs to some space and calculating the distance between them (e.g. based on the so-called edit distance (Gao et al (2010) or the so-called iterative measures (Tarapata et al (2009))).

Graph neural networks

A *Graph Neural Network* (GNN), as the name suggests, is a neural network that analyses data represented by a graph. The primary purpose of such a network is to incorporate vertex neighbourhood information during machine learning. A graph neural network is capable of three different types of prediction or classification:

- vertex-level (node-level prediction),
- edge-level (edge-level prediction),
- whole-graph or subgraph-level (graph/subgraph-level prediction).

In addition, GNNs are used not only for prediction, but e.g. for graph generation, community detection in graphs and others (Blumenfeld (2023)).

Graph Convolutional Networks (GCN) are one of the most relevant deep machine learning methods and are great for aggregating features of data that can be shed onto a certain plane, such as images. Their graphical generalization is graph spline networks, which allow the aggregation of data features that are multidimensional and non-Euclidean in nature. Convolution (spline) in the context of graph neural networks is realized by analogy with classical, convolutional networks. The convolutional layer realizes the aggregation of neighbouring values for a given node in the graph similarly to what happens to a pixel and its neighbouring other pixels in an image. If we consider the image to be an array of pixels placed on a plane, it can be concluded that it is a simple undirected graph. The image pixels are the vertices of this graph, and the neighbourhood relation of the pixels is represented by the edges of the graph. In both classical and graph convolution, a huge role is played by the aggregation method - the splicing of the original layer with the indicated filter. The choice of both size, value and aggregation function will determine with what intensity and in which directions the features of a given vertex or pixel will propagate. The difference between classical and graph convolution may be that the filter frame of classical convolution is determined arbitrarily, or perhaps even heuristically, whereas graph convolution benefits from the many advantages of the graph as a data structure. One of the most important characteristics of each vertex, which can be determined in a very simple way, is its degree. The degree is defined by the number of edges reaching a given vertex and can easily be used to identify the most important nodes and hierarchize them more generally. It is possible to use this parameter in defining a convolution graph aggregation method, as it is in fact precisely about hierarchizing the neighbours of a given vertex, through which it is possible to determine which relation to assign the greatest propagation of the feature of the vertex under analysis. It seems natural to define the weave aggregation method as a certain weighted average, in which the weights are the degrees of importance of the relations and the objects averaged, the features of the neighbouring vertices. Thus, a spline operation on a single node can be defined as the product of a vector of features of its neighbours with a vector of weights assigned to each relation (Wu et al (2021)). By interpreting the inverse of the degree of a vertex as the weight of a given neighbouring relation, we cause the feature of a vertex to be propagated most to those neighbours that are proportionally most influenced by the original vertex, and least to those for whom the neighbourhood is less important.

Graph Attention Networks (GATs) address the objections that can be had to graph convolutional networks, regarding a more complex and better mechanism to reflect the neighbourhood relations of vertices than their degree. A solution to this problem can be found by analysing message passing in a graph. By treating the feature vector of a given vertex as an information resource, one can interpret the propagation of these features as a message passing process in the graph. Following this line of reasoning, vertex neighbour feature convolution methods are an implementation of message propagation whose transition probabilities to each neighbour are expressed by a uniform distribution. This method, as it turns out, is quite unsophisticated, as it realises feature propagation in a purely random way. Thus, the aim of consideration is to find a transmission probability distribution that assigns the highest propagation chances according to the edges representing the most important relations. Such a distribution can, of course, be found heuristically, or on the basis of statistical analysis of the flows in the graph. A simple example would be a network representing a street layout, storing information about traffic volume on the edges, which in this case is a natural determinant of the attractiveness, or importance, of a vertex connection. Nevertheless, these methods have many disadvantages due to the very nature of heuristics. They are static - you

need historical data and expert knowledge to use them, while graphs work very well for representing dynamic data that can be analysed in real time. The possibilities for expert analysis are naturally also limited by the size of the graph. Therefore, a proposal for graph-based attentional networks GAT based on the attention mechanism was developed by Velickovic et al (2018).

Generative Adversarial Networks (GANs) are models proposed by Goodfellow et al (2014) and Goodfellow et al (2020). Yann LeCun, one of the fathers of convolutional neural networks (CNNs), described GANs as the most interesting idea of the decade in the field of machine learning. They belong to a group of generative models that take as input a simple random variable returning a random variable that follows a more complex target distribution. A GAN is composed of two deep neural networks:

- Generator - its purpose is to generate data with a distribution very similar to that of the training data. It must generate the data in such a way that the discriminator cannot distinguish it from the real data.
- Discriminator - gets two types of input data. One comes from the generator and the other from the training data set. Its purpose is to indicate from which of these two sets the input data comes.

Practical applications of the network include: e.g. generating high-resolution images from low-resolution images, encrypting messages, generating images from a verbal description, predicting the next frames of a video file, generating realistic images from sketches. From the point of view of cyber security problems, GANs can be used, among other things, to generate deep fakes in the form of realistic images and videos.

Basic differences between a graph-based and a traditional neural network

The biggest difference between a graph neural network and a traditional neural network is that in traditional neural networks the input data can be in any form, most often unstructured data, often vector data. In graph neural networks, the input data are graphs, so the networks are able to learn from how the data represented by the graph is connected to each other. In traditional neural networks, there are only individual data points, which do not contain any relationship information with other data points. All such connections and relationships must be manually created, through a process called feature engineering. In addition, other differences can be identified:

- graph neural networks are partially supervised. The network is learned on the whole graph, both on nodes with and without weights, so even if some nodes do not have their features and weights, information, for example about their neighbourhood and environment, can still be used in the learning process. In ordinary neural networks, on the other hand, data without weights is not used;
- graph neural networks usually contain fewer layers than classical networks. This is because graphs are a very complex structure. These networks also often have scalability problems;
- the operation of the convolution operation: in standard neural networks, it consists of moving filters of fixed dimensions, across the image. Since graphs do not have a fixed structure, fixed-size filters cannot be used here. The convolution operation is performed, as written earlier, as an aggregation of information from the neighbours of a vertex.

Examples of the use of GML methods in cybersecurity problems

Application of graph mining methods: clustering, classification, similarity

Graph clustering and classification algorithms are effective for threat detection and security analysis in ICT networks. They can help in responding quickly to attacks and in discovering vulnerabilities in systems so that they can be strengthened. Clustering algorithms also find application by clustering similar behaviour in a network to detect attacks and threats and anomalies. Several different applications of clustering algorithms in cyber security can be mentioned, such as:

- detection of DDoS attacks, Botnets (Chen et al (2017), Chowdhury et al (2017)): clustering algorithms can be used to identify the clustering of computers or devices used to launch a DDoS attack, allowing rapid detection and protection against this type of attack;
- analysis of user behaviour (Buczak et al (2016), Gamachchi et al (2017), Tarapata et al (2016)): the use of algorithms to identify similar user behaviour, which can help detect attempts to breach the system or attempts to attack individual accounts, impersonating other users;

- detection of security risks (Bhattacharyya et al (2013), Ghose et al (2019)): the use of clusters to analyse large volumes of data from different sources in order to detect anomalies and security threats such as hacking attempts, attacks, malware or phishing;
- security incident management (Studiawan et al (2019)): clustering algorithms can be used to automatically group security events, e.g. from system logs, and identify links between them, allowing rapid response to threats and minimising damage;
- detecting fake news and countering disinformation, hate speech (Mary et al (2023)): the use of clustering to identify active subgroups of user accounts on social media that may be spreading disinformation or hate speech.

In summary, clustering and classification methods in graphs, by appropriately characterising the relationships between nodes (representing, for example, individual servers or services), not only allow errors or attacks on the network to be identified, but also help to prioritise messages in a system overloaded with them. This allows the human controller overseeing such a system to respond to security incidents with greater efficiency.

Graph similarity analysis, in the context of cyber security, is used for such purposes as:

- detection of unknown attacks (Tarapata et al (2010)): graph similarity analysis can help identify previously unknown attacks that do not match known attack patterns. By comparing new patterns with existing reference data, potential hacking attacks can be detected that may be more sophisticated and more difficult to identify by other methods;
- anomaly detection (Tarapata et al (2009)): graph similarity analysis can help identify anomalies in computer networks, such as unauthorised connections, unexpected communication patterns or other anomalies. By comparing actual data with a reference model, abnormal behaviour can be detected that may indicate potential threats;
- Protection against data leakage, fraud, hardware trojans (Fyrbiak et al (2019)): graph similarity analysis can also be used to monitor network traffic for attempts at unauthorised data leakage. By comparing communication patterns with reference data, unauthorised information flows can be detected that may indicate attempts at data leakage or breaches of security policies.

Application of graph neural networks

Graph neural networks (GNNs) have a number of applications in cybersecurity. Some of these are listed below:

- detection of malicious accounts (Liu et al (2018), Wang et al (2019)),
- detection of deep fakes in images and video (El-Gayar et al (2024)),
- malware detection (Jiang et al (2018), Oliveira et al (2021)),
- fraud detection (Dou et al (2020), Liu et al (2020)),
- software vulnerability detection (Cao et al (2021), Cheng et al (2021), Zhou et al (2019)),
- forensic analysis of memory (Song et al (2018)),
- binary code analysis (Jafari et al (2021), Li et al (2019), Xu et al (2017)).

The problem of detecting deep fakes in images and video, as one of the most relevant in recent times, has received a solid literature review (Mary et al (2023), Rana et al (2022)).

We will give two examples concerning the use of GNNs: to detect of malware and to detect of deep fakes in video files. A suitable representation that can be used to characterize malware can be a graph or a behavioural network (see also the examples in the previous section). Such a graph can represent either low-level activities such as calls to system API functions of the operating system (Hassen et al (2017)) or high-level representations of communication between services in a computer network. Classical malware recognition methods rely on heuristic feature engineering based on expert knowledge and knowledge bases of already detected viruses, making them several steps behind the latest types of malware. The use of GML for automatic feature extraction from feature call graphs allows malware detection to be more effective in a dynamic way, already during the execution of the program under investigation itself. The first example concerns malware detection using autoencoders (Jiang et al (2018)). The use of autoencoder architecture in graph machine learning opens up a wide range of possibilities for advanced generation of embeddings (embeddings) of graph vertex features. Such an architecture consists in defining a symmetric neural network having output and input layers of the same size. The task of such a network

is to reproduce the input data as accurately as possible in the output. This problem is trivial when each layer has the same number of neurons. The situation becomes more interesting when the number of neurons in the hidden layers decreases from the input layer to the middle layer of the network. In such a situation, the network will try to generate a compressed form of the feature vector in the smallest layer. The embeddings generated by the middle layer of this network can serve as an efficient way of storing the features for the neighbourhood matrix of the graph. In the mentioned work (Jiang et al (2018)), a dual architecture of autoencoders generating embeddings for two different graphs of programme behaviour was used. The first is the Control Flow Graph (CFG) of the suspected programme control. Using the node2vec method described by Grover et al (2016), the dimension of its neighbourhood matrix was reduced to 500. The low-dimensional matrix thus embedded is given as input to the autoencoder SDA1. The second embedding is built from the 22000 system functions of the Windows API. It is a vector taking the value 1 in the cells corresponding to the functions called in the suspect program. The architecture of both networks is presented in Jiang et al (2018). The embeddings generated by both autoencoders are aggregated into a single output vector, which is passed through the activation function ReLU(*)=max{0,*} (Rectified Linear Unit), returning a value for classifying the software as harmful or harmless. The authors taught their model on a set containing both harmless and harmful software of various types and can boast of achieving an accuracy of greater than 99% on the test set.

The second example is based on the paper El-Gayar et al (2024) where authors describe a method for detecting deep fakes in movies (video files) using graph neural networks. The proposed method divides the detection process into two phases: a mini-batch stream of graph convolutional networks (called miniGNNs), a four-block stream of CNNs consisting of convolution, batch normalisation and activation functions. Comparable to CNNs, miniGNNs can effectively train a network to deep detect spurious videos on a down-sampled graph (or topological structure) in a mini-batch manner. The final step is the 'flattening' operation, which is necessary to merge the spliced (convolutional) layers with the dense layer. The fusion of these two phases is performed using three different fusion networks: FuNet-A (additive fusion), FuNet-M (multiplicative fusion) and FuNet-C (concatenative fusion). These fusion techniques aim to improve the performance of deep false-film detection by integrating the features extracted from both CNNs and the miniGNNs proposed in the paper into an end-to-end trainable network.
The new approach proposed by the Authors is intended, as they themselves write, to overcome some of the limitations and drawbacks of previous approaches:

- overfitting: the reduced complexity of GNNs compared to deep models, such as deep CNNs, makes it less prone to overfitting. Regularisation techniques are also used to improve the generalisability of the model;
- computational complexity: by exploiting the GNN's ability to handle relational data, the proposed method is able to achieve high detection accuracy without the need for highly complex models, thus reducing computational requirements;
- generalisability: the proposed model has been trained and tested on several different datasets ensuring that it works well under different conditions and deep fake techniques;
- adaptability: the flexible and scalable nature of GNN allows the model to adapt to new deep fake techniques, providing a future-proof solution for deep fake detection.

The paper El-Gayar et al (2024) also evaluates the accuracy of the proposed model on different datasets, where it achieved an impressive accuracy of 99.3% after 30 epochs of learning.

Summary

Graph-based machine learning (GML) methods have may applications in cybersecurity: detection of DDos attacks, Botnets; detection of network anomalies and security threats (intrusion attempts, malware or phishing); protection against data leakage, fraud, hardware Trojans; analysis of user behavior; detection of fake news (deep fakes in image and video) and countering disinformation, hate speech; security event management; detection of malicious accounts, detection of vulnerabilities in software, analysis of binary code.

GML methods are distinguished from classical methods not only by advanced feature propagation methods or solving problems, e.g. classification at multiple levels - vertex, graph, edge. The main feature that accounts for the superior performance of these methods, both unsupervised and supervised, is that they frame the problem in a

systematic way, using an advanced data structure - a graph or network. Conventional methods focused on representing the data using a Euclidean space, in which the features of individual occurrences were represented by decomposing them in relation to others on a plane. Both supervised (linear regression, k-neighbour method, etc.) and unsupervised methods focused on the appropriate positioning of the data so that, in a problem such as classification, clear boundaries were drawn between clusters of objects of one class. Graph methods take these representations to a higher level - they allow to directly determine the relations between objects by means of a graph vertex connected to its neighbours with its edges, and to characterise these relations with feature vectors, which makes it possible to build a network that is a detailed reflection of the modelled part of reality.

A major challenge in the development of GML methods is the so-called 'curse of dimensionality'. It refers to the exponential increase in the size of a computational problem as the number of vertices and features in a graph increases, for example in Bronstein et al (2021). This problem can be reduced at the level of network architecture design through various methods that allow operating on low-dimensional embeddings of features of vertices or entire graphs. This allows the use of a data structure with less complexity during, for example, learning a graph network, than a basic sparse matrix of vertex neighbourhoods. A group of methods that focus on solving the best embedding (embedding) problem is called Network Representation Learning (NRL), see Wu et al (2021). These methods are widely used in Natural Language Processing (NLP), among others. The use of methods such as DeepWalk (Perozzi et al (2014)), Node2Vec (Grover et al (2016)), or compression using a graph autoencoder allows the dimensionality of the data structures representing the graph to be dramatically reduced.

The problem of computational complexity in graph learning systems is also related to the need to retain total knowledge of the slice of reality represented by the graph. The very representation of this data structure for systems of, for example, the Big Data class is very troublesome, since the neighbourhood matrix, which is the basis for most graph algorithms, has a size that depends on the number of vertices in the graph. Each additional vertex in the graph is responsible for slowing down the learning of models due to cumbersome multiplication operations of large (often also sparse) matrices.

Acknowledgements

The study is supported from the funds of the University Research Grant UGB 701/2024 of the Faculty of Cybernetics at the Military University of Technology in Warsaw (WAT).

References

- Bhattacharyya, D. K. & Kalita, J. K. (2013) ‘Network Anomaly Detection: A Machine Learning Perspective’, CRC Press, Boca Raton.
- Blondel V.D, Guillaume J.-L., Lambiotte R., Lefebvre E. (2008) ‘Fast unfolding of communities in large networks’, *Journal of Statistical Mechanics: Theory and Experiment*, Vol. 2008, 10: P10008.
- Blumenfeld Z. (2023) ‘Graph Machine Learning: An Overview’, *Towards Data Science*.
- Bronstein M., Bruna J., Cohen T., Velickovic P. (2021) ‘Geometric Deep Learning, Grids, Groups, Graphs, Geodesics, and Gauges’, arXiv:2104.13478v2 [cs.LG].
- Buczak, A. L., Guven, E. (2016) ‘A survey of data mining and machine learning methods for cyber security intrusion detection’, *IEEE Communications Surveys & Tutorials*, 18(2), pp. 1153-1176.
- Cao S., Sun X., Bo L., Wei Y., Li B. (2021) ‘Bgnn4vd: Constructing bidirectional graph neural-network for vulnerability detection’, *Information and Software Technology*, vol. 136, p. 106576.
- Chen Y., Nadji Y., Kountouras A., Monrose F., Perdisci R., Antonakakis M., Vasiloglou N. (2017) ‘Practical atacks against graph-based clustering’, *Proceedings of the CCS’17*, October 30-November 3, Dallas, USA.
- Cheng X., Wang H., Hua J., Xu G., Sui Y. (2021) ‘Deepwukong: Statically detecting software vulnerabilities using deep graph neural network’, *ACM Transactions on Software Engineering and Methodology* (TOSEM), vol. 30, no. 3, pp. 1–33.
- Chowdhury S., Khanzadeh M., Akula R., Zhang F., Zhang S., Medal1 H., Marufuzzaman M., Bian L. (2017) ‘Botnet detection using graph-based feature clustering’, *Journal of Big Data*, 4:14, pp. 1-23.
- Dou Y., Liu Z., Sun L., Deng Y., Peng H., Yu P. (2020) ‘Enhancing graph neural network-based fraud detectors against camouflaged fraudsters’, *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 315– 324.

- El-Gayar M., Abouhawwash M., Askar S., Sweidan S. (2024) 'A novel approach for detecting deep fake videos using graph neural network', *Journal of Big Data*, 11:22, pp. 1-27.
- Fyrbiak M., Wallat S., Reinhard S., Bissantz N., Paar Ch. (2019) 'Graph Similarity and its Applications to Hardware Security', *IEEE Transactions on Computers*, pp. 505 - 519.
- Gamachchi A., Boztas S. (2017) 'Insider Threat Detection Through Attributed Graph Clustering', *Proceedings of the 16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*.
- Gao X., Xiao B., Tao D., Li X. (2010) 'A survey of graph edit distance', *Pattern Analysis and Applications*, vol. 13(1), pp. 113-129.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014) 'Generative adversarial nets', in: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger, eds. *Advances in Neural Information Processing Systems* 27, Curran Associates, Inc., Boston, 2672-2680.
- Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. (2020) 'Generative Adversarial Networks', *Communications of the ACM*, vol.63, Issue 11, pp.139-144.
- Grover A., Leskovec J. (2016) 'Node2vec: Scalable feature learning for networks', KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.855–864.
- Hassen M., Chan P. (2017) 'Scalable Function Call Graph-based Malware Classification', Proceedings of the 7th ACM Conference on Data and Application Security and Privacy, pp. 239-248.
- Jafari O., Maurya P., Nagarkar P., Islam K., Crushev Ch. (2021) 'A Survey on Locality Sensitive Hashing Algorithms and their Applications', arXiv:2102.08942.
- Jiang H., Turki T., Wang J. (2018) 'DLGraph: Malware Detection Using Deep Learning and Graph Embedding', Proceedings of the 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 17-20 December 2018, Orlando, USA.
- Li Y., Gu C., Dullien T., Vinyals O., Kohli P. (2019) 'Graph matching networks for learning the similarity of graph structured objects', International Conference on Machine Learning, pp. 3835–3845.
- Liu Z., Chen C., Yang X., Zhou J., Li X., Song L. (2018) 'Heterogeneous graph neural networks for malicious account detection', Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 2077–2085.
- Liu Z., Dou Y., Yu P., Deng Y., Peng H. (2020) 'Alleviating the inconsistency problem of applying graph neural network to fraud detection', Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1569–1572.
- Mary A., Edison A. (2023) 'Deep fake detection using deep learning techniques: a literature review', International Conference on Control, Communication and Computing, ICCC 2023.
- Oliveira A., Sassi R. (2021) 'Behavioral Malware Detection Using Deep Graph Convolutional Neural Networks', *International Journal of Computer Applications* 174/29, (2021).
- Perozzi B., Al-Rfou R., Skiena S. (2014) 'Deepwalk: Onlne learning of social representations', Proceedings of the 20th ACM SIGKDD International conference on knowledge discovery and data mining, pp. 701–710.
- Rana M., Nobi M., Murali B., Sung A. (2022) 'Deepfake detection: a systematic literature review', *IEEE Access*, vol.10, pp. 25494–513.
- Song W., Yin H., Liu C., Song D. (2018) 'Deepmem: Learning graph neural network models for fast and robust memory forensic analysis', Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 606–618.
- Studiawan H., Payne Ch., Sohel F. (2019) 'Automatic Graph-Based Clustering for Security Logs', Proceedings of the International Conference on Advanced Information Networking and Applications, pp. 914–926.
- Tarapata Z., Kasprzyk R. (2009) 'An application of multicriteria weighted graph similarity method to social networks analyzing', Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining, pp. 366-368, Athens (Greece), IEEE Computer Society.
- Tarapata Z., Kasprzyk R. (2010) 'Graph-Based Optimization Method for Information Diffusion and Attack Durability in Networks', RSCTC'2010, *Lecture Notes in Artificial Intelligence*, vol.6086, Springer, Heidelberg, pp. 698-709.
- Tarapata Z., Zabielski M., Kasprzyk R., Szkółka K. (2016) 'Profile Cloning Detection in Online Social Networks', *Computer Science and Mathematical Modelling*, Nr 3, 39-46.
- Tarapata Z. (2020) 'Application of Selected Models and Methods of Graph Mining on the Example of the Analysis of Publication Links', Proceedings of the 36th International Business Information

Management Association Conference (IBIMA), 4-5 November 2020 Granada, Spain ISBN: 978-0-9998551-5-7, pp. 10626-10637.

- Velicković P., Cucurull G., Casanova A., Romero A., Liò P., Bengio Y. (2018) 'Graph attention networks', Proceedings of the International Conference on Learning Representations (ICLR'2018).
- Wang J., Wen R., Wu C., Huang Y., Xion J. (2019) 'Fdgars: Fraudster detection via graph convolutional networks in online app review system', Companion Proceedings of The 2019 World Wide Web Conference, pp. 310–316.
- Wu Z., Pan S., Chen F., Long G., Zhang Ch., Yu P. (2021) 'A Comprehensive Survey on Graph Neural Networks', *IEEE Transactions on Neural Networks and Learning Systems* 32(1), 4–24.
- Xu X., Liu C., Feng Q., Yin H., Song L., Song D. (2017) 'Neural network-based graph embedding for cross-platform binary code similarity detection', Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 363–376.
- Zhou Y., Liu S., Siow J., Du X., Liu Y. (2019) 'Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks', NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp. 10197–10207.