Performance Evaluation of Adversarial Attacks on Whole-Graph Embedding Models

Mario Manzo[1], Maurizio Giordano[2], Lucia Maddalena[2], and Mario R. Guarracino[3,4(✉)]

[1] University of Naples "L'Orientale", Naples, Italy
[2] National Research Council, Rome, Italy
[3] University of Cassino and Southern Lazio, Cassino, Italy
mario.guarracino@unicas.it
[4] National Research University Higher School of Economics, Moscow, Russia

Abstract. Graph embedding techniques are becoming increasingly common in many fields ranging from scientific computing to biomedical applications and finance. These techniques aim to automatically learn low-dimensional representations for a variety of network analysis tasks. In literature, several methods (e.g., random walk-based, factorization-based, and neural network-based) show very promising results in terms of their usability and potential. Despite their spreading diffusion, little is known about their reliability and robustness, particularly when applied to the real world of data, where adversaries or malfunctioning/noisy data sources may supply deceptive data. The vulnerability emerges mainly by inserting limited perturbations in the input data when these lead to a dramatic deterioration in performance. In this work, we propose an analysis of different adversarial attacks in the context of whole-graph embedding. The attack strategies involve a limited number of nodes based on the role they play in the graph. The study aims to measure the robustness of different whole-graph embedding approaches to those types of attacks, when the network analysis task consists in the supervised classification of whole-graphs. Extensive experiments carried out on synthetic and real data provide empirical insights on the vulnerability of whole-graph embedding models to node-level attacks in supervised classification tasks.

Keywords: Whole-graph embedding · Adversarial attacks · Graph classification

1 Introduction

Graph structure plays an important role in many real-world applications. Representation learning on structured data with machine and deep learning methods has shown promising results in various applications, including drug screening [46], protein analysis [41], and knowledge graph completion [27].

D. E. Simos et al. (Eds.): LION 2021, LNCS 12931, pp. 219–236, 2021.
https://doi.org/10.1007/978-3-030-92121-7_19

Many graph embedding methods have been developed aiming at mapping graph data into a vector space [5]. The result is a low-dimensional feature representation for each node in the graph where the distance between the nodes in the destination space is preserved as much as possible. Actually, working on embedded data turns out to be easier and faster than on the original data. Furthermore, the resulting vectors in the transformed space can be adopted for downstream analysis, either by analyzing the target space or by applying machine learning (ML) techniques to the vector space. Indeed, by maintaining the topological information of the graph, low-dimensional representations can be adopted as features for different tasks such as graphs/nodes classification or clustering.

Despite the remarkable success, the lack of interpretability and robustness of these models makes them highly risky in fields like biomedicine, finance, and security, to name a few. Typically, sensitive information concerns the user-user relationship within the graph. A user who connects with many users with sensitive information may have sensitive information as well. As heuristics learned from graph-based methods often produce good predictions, they could also jeopardize the model. For example, an ill-intentioned person could disguise himself by connecting to other people on a social network. Such an "attack" on the model is simple enough but could lead to severe consequences [11]. Due to a large number of daily interactions, even if only a few of them are fraudulent, the ill-intentioned could gain enormous benefits.

The concept of graph robustness was first introduced in the 1970s [10] and is certainly interdisciplinary. This aspect has generated a variety of points of view, opening up to challenging and implicit problems with the aim of providing fundamental knowledge.

Robustness in networked systems is commonly defined as a measure of their ability to continue operating when one or more of their parts are naturally damaged or targeted by an attack [4]. The study of network robustness concerns the understanding of interconnected complex systems. For example, consider a network that is prone to natural failures or targeted attacks. A natural failure occurs when a single element fails due to natural causes such as obsolescence. The consequence is an additional load of the whole remaining network, causing a series of cascading faults. Not all failures come from natural causes; some may be induced by targeted attacks, penetrating the network and sabotaging an important part of it. The antonym of network robustness is *vulnerability* [42], defined as a measure of a network's susceptibility to the spread of perturbations across the network. The concepts of *robustness* and *vulnerability* can be extended to different types of networks, such as biological ones. Also in this case, they are two important indicators to verify the possible fault of a part of the network or any criticalities that can compromise the general functions with irreversible impact.

Robustness and vulnerability analysis is a crucial problem for today's research focusing on machine learning, deep learning, and AI algorithms operating on networked data in several domains, from cybersecurity to online financial trading, from social media to big-data analytics. In these contexts, while the networked systems (i.e., the graph-structured data) are the target of the attacks or perturbations, the real goal is to cause either the malfunctioning (intentionally or not)

or an induced fraudulent behavior of the algorithms which operate on the modified data.

According to this interpretation, adversarial machine learning [23] is the area of research in which ML models vulnerability is studied under adversarial manipulation of their input intended to cause incorrect classification [12]. Neural networks and many other machine learning models are highly vulnerable to adversarial perturbations of the input to the model either at train or at test time, or both.

Several works on adversarial machine learning in the literature focus on the computer vision domain [2,34] with application to image recognition. Specifically, they address the problem of studying and improving the robustness of classification methods when adversarial images are present in the training and/or testing stages. More recently, adversarial ML has been increasingly utilized in other domains, such as natural language processing (NLP) [16] and cybersecurity [36]. Examples of applications in computer vision and NLP domains include handling autonomous cars' systems vulnerability, fake news, and financial fraud detection algorithms. In the cybersecurity domain, adversaries can be terrorists and fraudulent attackers. Examples of AI cyber systems that can be vulnerable to adversarial attacks are detection algorithms of malware stealing user information and/or collecting money and network worms causing network damages and malicious functionality.

In this work, we focus on adversarial ML techniques and approaches in the domain of machine learning models applied to the classification of biological networks. In this domain, we do not think of a scenario in which a "real adversary" intentionally introduces malicious perturbations in the input of learning models. In our interpretation of "adversarial attacks" within the realm of biological networks, we mean any type of perturbation to the graph structure, either due to noise introduced by the experimental environment from where the biological data is extracted or to the lack of information due to corrupted sources or incomplete pre-processing of raw data.

We propose a broad experimentation phase to address the various aspects mentioned above, using several methods and datasets. To the best of our knowledge, a performance analysis of whole-graph embedding methods under conditions of adversarial attacks has never been carried out.

The paper is structured as follows. Section 2 provides an overview of the state-of-art about adversarial attacks on whole-graph embedding models. Section 3 gives details about the problem statement. Section 4 provides a comprehensive experimental phase, while Sect. 5 concludes the paper.

2 Related Work

The literature concerning adversarial attacks for graph data is very recent and often aimed at node-level or link-level applications [8,39]. Here, we focus on graph-level applications, and specifically on adversarial attacks on whole-graph embedding methods, for which few recent papers (mainly preprints) are available.

In [40], Tang et al. design a surrogate model that consists of convolutional and pooling operators to generate adversarial samples to fool the hierarchical Graph Neural Networks (GNN)-based graph classification models. Nodes preserved by the pooling operator are set as attack targets. Then the attack targets are perturbed slightly to trick the pooling operator in hierarchical GNNs into selecting the wrong nodes to preserve. Furthermore, a robust training on the target models is performed to demonstrate that the retrained graph classification models can better defend against the attack from the adversarial samples.

Chen et al. [7] propose a graph attack framework named GraphAttacker that works to adjust the structures and to provide the attack strategies according to the graph analysis tasks. It generates adversarial samples based on the Generative Adversarial Network (GAN) through alternate training on three key components: the Multi-strategy Attack Generator, the Similarity Discriminator, and the Attack Discriminator. Furthermore, to achieve attacks within perturbation budget, a novel Similarity Modification Rate to quantify the similarity between nodes and thus to constrain the attack budget is introduced.

Another graph attack framework, named Graph Backdoor, is presented by Xi et al. [48]. It can be applied readily without knowing data models or tuning strategies to optimize both attack effectiveness and evasiveness. It works in different ways: i) graph-oriented – it defines triggers as specific subgraphs, including topological structures and descriptive features, entailing a large design spectrum for the adversary; ii) input-tailored – it dynamically adapts triggers to individual graphs; and iii) attack-extensible – it can be instantiated for both transductive and inductive tasks.

The vulnerability of Graph Convolutional Networks (GCNs) to adversarial attacks has been debated in the literature. In [24], Jin et al. introduce a robustness certificate for graph classification using GCNs under structural attacks. The method is based on Lagrange dualization and convex envelope, which result in tight approximation bounds computable by dynamic programming. Applied in conjunction with robust training, it allows an increased number of graphs to be certified as robust.

Faber et al. [14] discuss the particularities of explaining GNN predictions. In graphs, the structure is fundamental, and a slight modification can lead to little knowledge of the data. Therefore, the explanation is reflected in adversarial attacks. The authors argue that the explanation methods should stay with the training data distribution and produce Distribution Compliant Explanation (DCE). To this end, they propose a novel explanation method, Contrastive GNN Explanation, for graph classification that adheres to DCE.

You et al. [49] propose a graph contrastive learning (GraphCL) framework for learning unsupervised representations of graph data. The impact of various combinations of graph augmentations in different tasks (semi-supervised, unsupervised, transfer learning, and adversarial attacks) is explored. The proposed framework can produce graph representations of similar or better generalizability, transferability, and robustness than state-of-the-art methods.

In [9], Chung et al. present a framework named Graph Augmentations with Bi-level Optimization (GABO). It is built to provide a graph augmentation approach based on bi-level optimization to investigate the effects on graph classification performance. The augmentation procedure can be applied without a priori domain knowledge about the task. Indeed, the framework combines a Graph Isomorphism Network (GIN) layer augmentation generator with a bias transformation.

All the above described approaches propose different types of adversarial attacks. However, none of them shares our aim, i.e., to compare the robustness to adversarial attacks of different whole-graph embedding methods.

3 Background

In this section, we introduce the formalization of a graph adversarial attack for the graph classification task. We will first give some preliminary notions about graphs and the whole-graph embedding problem. Then, we introduce the graph adversarial attack and related strategies for graph classification.

3.1 Whole-Graph Embedding

A graph $G = (V, E)$ is represented by a pair of sets: $V = \{v_i\}_{i=1}^{N}$ is the set of nodes, and $E \subseteq V \times V$ is the set of edges, each one represented by a pair of nodes (v_i, v_j), where v_i is the source node and v_j the target node. This definition holds for unweighted graphs, which means graphs whose vertices relation is simply represented by a connection between them. Let W be a set of real numbers, called weights, such that for each $(v_i, v_j) \in E$ there exists a weight $w_{i,j} \in W$ associated to the edge; then $G(V, E, W)$ is called a weighted graph. An alternative representation of a weighted graph is through its adjacency matrix $A = \{A_{i,j}\}$, whose elements are:

$$A_{i,j} = \begin{cases} w_{i,j} & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

For unweighted graphs, a unitary weight is considered for each edge to obtain the adjacency matrix. In general, the adjacency matrix A is not symmetric, since the occurrence of an edge from node v to node u does not imply the existence of the edge (u, v). This is only the case of undirected graphs, in which the connection between two nodes u and v has no direction, thus both $(u, v) \in E$ and $(v, u) \in E$ and A is symmetric. In the following, we will refer to generic graphs $G = (V, E)$, specifying their weights or their directionality only if needed.

In a very general definition, graph embedding learns a mapping from a graph to a vector space with the purpose of preserving main graph properties.

Definition 1. *Given a graph $G = (V, E)$, a* graph embedding *(or node-level graph embedding) is a mapping ϕ: $v_i \in V \rightarrow \mathbf{y}_i \in \mathbb{R}^d, i = 1, \ldots, N$, $d \in \mathbb{N}$, such that the function ϕ preserves some proximity measure defined on graph G.*

Specifically, it is a space reduction that maps the nodes of a graph into a d-dimensional feature vector space, also known as *latent space*, trying to maintain structural information in terms of connections between vertices. The goal of keeping as much information as possible about the graph space in the transformation is linked to the choice of node/edge properties for the initial representation of the graph. The criticality concerns the final latent space that expresses valuable information, for applications such as classification or grouping, despite being in a lower-dimensional search space.

The concept of graph embedding refers to node-level since it maps each node in a graph into a vector, preserving node-node proximity (similarity/distance).

Definition 2. *Given a set of graphs* $\mathcal{G} = \{G_i\}_{i=1}^{M}$ *with the same set of vertices* V, *a* whole-graph embedding *is a mapping* $\psi : G_i \rightarrow \mathbf{y}_i \in \mathbb{R}^d, i = 1, \ldots, M$, $d \in \mathbb{N}$, *such that the function* ψ *preserves some proximity measure defined on* $\mathcal{G}$.

In this context, the fundamental condition is that the nodes of the graphs represent the same information. This requires an alignment procedure that verifies this property to provide compliant embedding.

Unlike graph embedding, which is adopted in applications such as link prediction and node label predictions, whole-graph embedding is more suited to graph classification, graph similarity ranking, and graph visualization.

3.2 Graph Adversarial Attacks

Generally, a network can become damaged through two primary ways: natural failure and targeted attack. Natural failures typically occur when a part fails due to natural causes. This results in the malfunction or elimination of a node or edge in the graph. Despite random network failures are much more common, they are less harmful than targeted attacks. This phenomenon has been verified across a range of graph structures [4]. Otherwise, targeted attacks carefully and through precise rules select the nodes and edges of the network for removal to maximally disrupt network functionality.

Our attention is focused on the modifications to the discrete structures and different attack strategies. Generally, the attacker tries to add or delete edges from G to create the new graph. These kinds of actions are varied since adding or deleting nodes can be performed by a series of modifications to the edges. Editing edges requires more effort than editing nodes. Indeed choosing a node only requires $O(|V|)$ complexity, while choosing an edge requires $O(|V|^2)$. In our experiments, we consider two attack strategies

– Degree-based Attack (DA): a percentage p of graph nodes having the highest degree is removed. The degree (or connectivity) δ_{v_i} of node v_i is the number of edges connected to it and can be computed using the graph adjacency matrix $A = \{A_{i,j}\}$ as

$$\delta_{v_i} = \sum_{j \neq i} A_{i,j}.$$

The effect of a DA is to reduce the total number of edges in the network as fast as possible [22]. It only takes into account the neighbors of the target node v when making a decision and can be considered a local attack. It is performed with a low computational overhead.

– Betweenness-based Attack (BA): a percentage p of graph nodes having the highest betweenness centrality is removed. The betweenness centrality for a node v_i is defined as

$$b_{v_i} = \sum_{j,k \neq i} \frac{\sigma_{j,k}(v_i)}{\sigma_{j,k}},$$

where $\sigma_{j,k}$ is the total number of shortest paths from node v_j to node v_k and $\sigma_{j,k}(v_i)$ is the number of those paths that pass through the target node v_i. The effect of a BA is to destroy as many paths as possible [22]. It is considered a global attack strategy due to the path information is aggregated from the whole network. Clearly, global information carries significant computational overhead compared to local attacks.

The robustness of the whole-graph embedding methods to adversarial attacks will be evaluated in terms of their performance for the task of graph classification on the attacked data.

4 Experiments

In our experiments, we analyze and compare the behavior of some whole-graph embedding methods under attack conditions for the task of graph classification. There are different challenges in this topic [40]

– *Selection of the target nodes and edges for the attack.* Suppose one or a few nodes or edges are perturbed at random. In that case, the graph classification results may not change because such a perturbation may not affect or destroy the intrinsic characteristics of graphs discriminating for the classification. In this regard, node selection strategies have been chosen as illustrated in Sect. 3.2.
– *Parameters setting to generate effective results.* The choice is undoubtedly difficult as the starting graphs are perturbed. A consequence could also fall on the computational costs during classification. As is well known, optimizing the parameters is a crucial aspect for obtaining the best performance. Concerning this point, we explored the parameter space to choose those that lead to the best results.
– *Robustness* is always an essential factor in evaluating the performance of the models. In the scenario of adversarial attacks, how to improve the robustness of the classification models? This is one of the two crucial points on which the paper was founded. In fact, as it is possible to observe through provided results, it is not certain that, by weakening the structure of the graphs, the transformation into a vector space, through the embedding phase, necessarily

produces an unrepresentative features vector, affecting the classification. We will see how some methods adapt even when the graph structures are less dense and informative.
- *Vulnerability*, in the same way, is always an essential factor in evaluating the performance of the models. In the scenario of adversarial attacks, how to identify the vulnerability of the classification models? It is the second crucial node on which the paper was founded. As it is possible to observe through the provided results, also in this case, by weakening the structure of the graphs, the transformation into a vector space, through the embedding phase, could produce an unrepresentative feature vector, affecting the classification. We will see how some methods do not fit when the graph structures are less dense and informative.
- *Data-driven selection.* The choice of data is driven by the characteristics of the graphs. In this way, models can show robustness or highlight critical issues when a variation of the data occurs. We decided to stress the various methods chosen for the evaluation based on different characteristics related to data. As we can see from Table 1, for example, three of the five datasets are unweighted. This detail is fundamental for calculating the centrality measures and, therefore, for selecting the nodes to be attacked.

4.1 Datasets

Table 1 illustrates the main properties of the datasets adopted in the experiments and includes synthetic and real network datasets, concerning some case studies of our current research on graph classification and clustering [17,29,30].

LFR is a synthetic dataset introduced in [21] based on the Lancichinetti–Fortunato–Radicchi (LFR) method [25]. As described in [29], we generated two classes of graphs containing 81 nodes, constructed using two different values of the parameter μ (expected proportion of edges having a vertex in one community and the other vertex in another community): 600 graphs with 0.1 μ and 1000 with 0.5 μ. Therefore, this dataset includes many small and unweighted graphs, subdivided into classes differing by well-defined community properties.

The MREG model [47] is adopted to generate the synthetic Multiple Random Eigen Graphs (MREG) dataset. Settings for MREG parameters, chosen based on the authors suggestions and our previous choices [29], are: $d = 2$ (model dimension), $n = 100$ (number of nodes), $h_1, h_2 \in \mathbb{R}^n$, where $h_1(i) = 0.1, \forall i$, $h_2(i) = -0.1, i = 1, \ldots, 50$, $h_2(i) = 0.1, i = 51, \ldots, 100$. The total number of unweighted graphs is 300, each composed of 100 nodes each, equally subdivided into 3 classes using $\lambda = [24.5, 4.75]$ for class c1, $\lambda = [20.75, 2.25]$ for class c2, and $\lambda = [24.5, 2.25]$ for class c3. In [47], further parameters' details are given.

The Brain fMRI dataset contains real networks built in [3] from functional magnetic resonance imaging (fMRI) time-series data [1] from the Center for Biomedical Research Excellence (COBRE) dataset. It is composed of 54 graphs from Schizophrenia subjects and 70 graphs from healthy controls. Each graph includes 263 nodes corresponding to different brain regions. The edges weights represent the Fisher-transformed correlation between the fMRI time-series of the

nodes after ranking [3], and we only kept the weights of the positively correlated edges. The dataset ends up including dense weighted graphs with a high average degree but a small diameter.

The Kidney dataset describes real metabolic networks created for validating related research [18,20,30]. It contains networks derived from data of 299 patients divided into three classes: 159 clear cell Renal Cell Carcinoma (KIRC), 90 Papillary Renal Cell Carcinoma (KIRP), and 50 Solid Tissue Normal samples (STN). We obtained the networks by mapping gene expression data coming from the Genomic Data Commons (GDC, https://portal.gdc.cancer.gov) portal (Projects TCGA-KIRC and TCGA-KIRP) on the biochemical reactions extracted from the kidney tissue metabolic model [44] (https://metabolicatlas.org). Specifically, given the stoichiometric matrix of the metabolic model, the graph nodes represent the metabolites, and the edges connect reagent and product metabolites in the same reaction, weighted by the average of the expression values of the genes/enzymes catalyzing that reaction [20]. Different reactions represented by multiple edges connecting two metabolites were fused in a single edge, where the weight includes the sum of the weights of the fused edges. Disconnected nodes, due to reactions not catalyzed by an enzyme, and recurrent metabolites, were not included [20]. The simplification procedure described in [18] is applied to reduce the complexity of the network, leading to reduce the number of nodes from 4022 to 1034. Overall, the dataset includes sparse weighted graphs with a small average degree but wide diameter.

MUTAG [13] is a popular benchmark dataset and is composed of networks of 188 mutagenic aromatic and heteroaromatic nitro compounds. The nodes represent atoms, while the edges represent chemical bonds between them. The graphs contain both vertex and edge labels. The two classes indicate whether or not the compound has mutagenic effects on a bacterium. Contrary to the other datasets, the nodes are not perfectly aligned. Indeed, the MUTAG networks have an average of eighteen vertices, but the labels are only seven.

4.2 Compared Methods

In the experiments, we compared the classification results obtained using the network embeddings produced by seven whole-graph embedding methods, briefly described in the following

- GL2vec [6]. It is an extended version of Graph2vec. The method is named Graph and Line graph to vector (GL2vec) because it concatenates the embedding of an original graph to that of the corresponding line graph. The line graph is an edge-to-vertex dual graph of the original graph. Specifically, GL2vec integrates either the edge label information or the structural information, which Graph2vec misses with the embeddings of the line graph.

Table 1. Main properties of the adopted datasets

	LFR	MREG	Kidney	Brain fMRI	MUTAG
Graphs	1600	300	299	124	188
Classes	2	3	3	2	2
Samples per class	600/1000	100/100/100	159/90/50	70/54	125/63
Vertices	82	100	1034	263	17.93
Average edges	844.45	1151.71	3226.00	19748.88	39.59
Average edge density	0.13	0.23	0.01	0.57	0.138454
Distinct vertex labels	82	100	1034	263	7
Edge weights	✗	✗	✓	✓	✗
Minimum diameter	3	2	126	0.03	5
Maximum diameter	7	3	455.36	0.07	15
Average degree	20.60	23.03	6.24	150.18	2.19

- Graph2vec [33]. It provides a Skip-Gram neural network model, typically adopted in the NLP domain. It learns data-driven distributed representations of arbitrarily sized graphs. The resulting embeddings are learned in an unsupervised manner and are task-unaware.
- IGE [15]. It extracts handcrafted invariant features based on graph spectral decomposition. These features are easy to compute, permutation-invariant, and include sufficient information on the graph's structure.
- NetLSD [43]. It computes a compact graph signature derived from the solution of the heat equation involving the normalized Laplacian matrix. It is permutation and size-invariant, scale-adaptive, and computationally efficient.
- FGSD [45]. It provides a graph representation based on a family of graph spectral distances with uniqueness, stability, sparsity, and computational efficiency properties.
- FeatherGraph [38]. It adopts characteristic functions defined on graph vertices to describe the distribution of node features at multiple scales. The probability weights of the characteristic function are defined as the transition probabilities of random walks. The node-level features are combined by mean pooling to create graph-level statistics.
- Netpro2vec [31]. It is a neural-network method that produces embedding of whole-graphs which are independent from the task and nature of the data. It first represents graphs as textual documents whose words are formed by processing probability distributions of graph node paths/distances (e.g., the Transition Matrix, TM, or the Node Distance Distribution, NDD). Then, it embeds graph documents by using the doc2vec method [26].

4.3 Implementation Details

For the first six whole-graph embedding methods (GL2vec, Graph2vec, IGE, NetLSD, FGSD, and FeatherGraph), we used their implementation provided

in the Karate Club software [37]. Our Netpro2vec framework, implemented in Python, is publicly available (https://github.com/cds-group/Netpro2vec). It also includes the code for extracting the NDD and TM distribution matrices, based on the GraphDistances R package [19], and the doc2vec embedding is performed using the gensim Natural Language Processing (NLP) library [35]. Even though the method can exploit different distribution distances as well their combinations, in the experimental results, we only report the two obtained using NDD (Netpro2vec^{ndd}) and the combination of NDD with TM1 (Netpro2vec$^{ndd+tm1}$), which lead to the best performance results.

The dimension d of the latent feature space for GL2Vec, Graph2Vec, FGSD, and Netpro2vec was set to 512; this value has been experimentally chosen so as to maximize accuracy. Instead, for IGE, FeatherGraph, and NetLSD, the output dimension cannot be specified as an input parameter.

For classification, we adopted an SVM model with a linear kernel (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html). To avoid hiding the effect of adversarial attacks, we did not apply any feature selection, even though it could certainly provide higher performance results for any of the considered methods. We validated the developed models through ten-fold stratified cross-validation iterated ten times, measuring the mean and standard deviation of classification accuracy and Matthews correlation coefficient (MCC) [32].

All the experiments were run on Google Colab Machine, which provides by default a virtual machine based on a bi-processor with two CPUs @ 2.30 GHz Intel(R) Xeon(R), 13 GB RAM and 33 GB HDD.

4.4 Performance Evaluation

Performance results obtained using the seven whole-graph embedding methods described in Sect. 4.2 on the five datasets detailed in Sect. 4.1 are reported in the bar plots of Fig. 1. Detailed numerical results are given in Tables 2 and 3. Here, we consider the results achieved using the original network data (Unattacked), as well as those using data that underwent the removal of the 30% and 50% of the nodes having the highest betweenness centrality (BA) or the highest degree (DA), respectively. The choice of these percentages p of nodes to be removed aims at investigating the effects of both *moderate* (30%) and *strong* (50%) adversarial attacks. The performance is evaluated in terms of the Accuracy and MCC values, defined as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}},$$

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}},$$

respectively. Here, TP, TN, FP, and FN indicate the number of true positives, true negatives, false positives, and false negatives. While the Accuracy provides the percentage of correctly classified samples, MCC gives the correlation coefficient between observed and predicted binary classifications.

Table 2. Accuracy (%) and MCC (mean and std over ten iterations) of adversarial attacks on whole-graph embedding models (30% of attacked nodes). In boldface the best results for each dataset and each attack.

Dataset	Method	Accuracy			MCC		
		Unattacked	BA	DA	Unattacked	BA	DA
LFR	GL2Vec	94.59 ± 1.75	84.66 ± 2.45	87.66 ± 2.12	0.88 ± 0.03	0.66 ± 0.05	0.74 ± 0.04
	Graph2vec	91.94 ± 2.04	84.41 ± 2.74	89.44 ± 2.13	0.82 ± 0.04	0.66 ± 0.05	0.77 ± 0.04
	IGE	$\mathbf{100.00 \pm 0.00}$	97.06 ± 1.34	97.17 ± 1.17	$\mathbf{1.00 \pm 0.00}$	0.93 ± 0.02	0.93 ± 0.02
	NetLSD	$\mathbf{100.00 \pm 0.00}$	$\mathbf{99.09 \pm 0.73}$	99.04 ± 0.71	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.98 \pm 0.01}$	0.97 ± 0.01
	FGSD	$\mathbf{100.00 \pm 0.00}$	97.97 ± 0.99	$\mathbf{99.15 \pm 0.68}$	$\mathbf{1.00 \pm 0.00}$	0.95 ± 0.02	$\mathbf{0.98 \pm 0.01}$
	FeatherGraph	$\mathbf{100.00 \pm 0.00}$	98.99 ± 0.69	99.00 ± 0.74	$\mathbf{1.00 \pm 0.00}$	0.97 ± 0.01	0.97 ± 0.01
	Netpro2vec^{ndd}	$\mathbf{100.00 \pm 0.00}$	98.41 ± 0.96	97.40 ± 1.16	$\mathbf{1.00 \pm 0.00}$	0.96 ± 0.01	0.94 ± 0.02
	Netpro2vec$^{ndd+tm1}$	$\mathbf{100.00 \pm 0.00}$	95.26 ± 1.16	72.13 ± 2.99	$\mathbf{1.00 \pm 0.00}$	0.89 ± 0.03	0.38 ± 0.07
MREG	GL2Vec	66.83 ± 7.14	$\mathbf{65.23 \pm 7.63}$	64.23 ± 8.09	0.44 ± 0.10	$\mathbf{0.48 \pm 0.11}$	0.47 ± 0.12
	Graph2vec	38.70 ± 8.46	40.60 ± 9.46	46.27 ± 9.30	0.08 ± 0.13	0.11 ± 0.14	0.19 ± 0.14
	IGE	65.80 ± 8.60	62.67 ± 7.53	65.70 ± 7.55	0.49 ± 0.12	0.44 ± 0.11	0.49 ± 0.11
	NetLSD	$\mathbf{71.57 \pm 7.05}$	58.03 ± 8.38	58.30 ± 8.29	$\mathbf{0.58 \pm 0.10}$	0.37 ± 0.12	0.38 ± 0.12
	FGSD	59.60 ± 6.08	59.73 ± 7.71	65.93 ± 7.97	0.40 ± 0.09	0.40 ± 0.11	$\mathbf{0.49 \pm 0.12}$
	FeatherGraph	59.60 ± 6.08	62.80 ± 7.77	$\mathbf{66.10 \pm 6.77}$	0.40 ± 0.09	0.44 ± 0.11	0.49 ± 0.10
	Netpro2vec^{ndd}	63.07 ± 7.60	42.70 ± 8.73	55.33 ± 9.44	0.45 ± 0.11	0.14 ± 0.13	0.33 ± 0.14
	Netpro2vec$^{ndd+tm1}$	36.80 ± 8.48	34.77 ± 7.95	30.87 ± 8.47	0.05 ± 0.12	0.02 ± 0.12	0.03 ± 0.13
Brain fMRI COBRE	GL2Vec	No conv	No conv	No conv	No conv	No conv	No conv
	Graph2vec	43.85 ± 11.27	46.29 ± 13.43	42.58 ± 12.38	-0.18 ± 0.24	-0.09 ± 0.27	-0.18 ± 0.25
	IGE	44.88 ± 14.70	48.99 ± 13.33	53.69 ± 14.64	-0.11 ± 0.30	-0.03 ± 0.28	0.05 ± 0.30
	NetLSD	56.12 ± 6.59	55.98 ± 12.10	$\mathbf{56.01 \pm 8.70}$	0.01 ± 0.18	0.09 ± 0.26	0.03 ± 0.21
	FGSD	56.54 ± 2.20	54.68 ± 12.75	48.31 ± 13.90	0.00 ± 0.00	0.07 ± 0.26	-0.06 ± 0.29
	FeatherGraph	53.77 ± 5.89	52.65 ± 7.77	53.77 ± 12.30	-0.06 ± 0.16	-0.08 ± 0.17	0.01 ± 0.28
	Netpro2vec^{ndd}	$\mathbf{56.58 \pm 12.74}$	58.58 ± 10.20	52.97 ± 11.36	$\mathbf{0.11 \pm 0.27}$	0.14 ± 0.23	-0.00 ± 0.27
	Netpro2vec$^{ndd+tm1}$	$\mathbf{56.58 \pm 12.74}$	$\mathbf{59.18 \pm 13.32}$	53.30 ± 12.70	$\mathbf{0.11 \pm 0.27}$	$\mathbf{0.17 \pm 0.28}$	$\mathbf{0.05 \pm 0.27}$
Kidney RNASeq	GL2Vec	90.09 ± 4.74	82.58 ± 6.73	59.83 ± 6.05	0.83 ± 0.08	0.71 ± 0.11	0.25 ± 0.16
	Graph2vec	90.79 ± 5.11	79.87 ± 7.05	58.08 ± 5.94	0.83 ± 0.08	0.66 ± 0.12	0.21 ± 0.17
	IGE	No conv	No conv	No conv	No conv	No conv	No conv
	NetLSD	53.46 ± 7.02	59.07 ± 7.14	62.23 ± 8.68	0.11 ± 0.16	0.25 ± 0.15	0.36 ± 0.15
	FGSD	No conv	No conv	No conv	No conv	No conv	No conv
	FeatherGraph	81.51 ± 7.96	81.67 ± 6.44	84.36 ± 6.64	0.68 ± 0.13	0.69 ± 0.10	0.74 ± 0.11
	Netpro2vec^{ndd}	83.53 ± 6.42	87.22 ± 6.17	85.83 ± 6.19	0.71 ± 0.11	$\mathbf{0.79 \pm 0.10}$	0.76 ± 0.10
	Netpro2vec$^{ndd+tm1}$	$\mathbf{91.27 \pm 4.45}$	$\mathbf{87.33 \pm 5.86}$	$\mathbf{90.91 \pm 5.60}$	$\mathbf{0.86 \pm 0.07}$	0.79 ± 0.09	$\mathbf{0.85 \pm 0.09}$
MUTAG	GL2Vec	76.11 ± 8.48	59.09 ± 10.57	67.82 ± 9.44	0.31 ± 0.24	0.05 ± 0.24	0.27 ± 0.21
	Graph2vec	66.32 ± 9.72	64.07 ± 9.79	63.63 ± 10.02	0.39 ± 0.24	0.15 ± 0.24	0.15 ± 0.24
	IGE	83.72 ± 7.92	83.22 ± 8.13	$\mathbf{84.93 \pm 7.45}$	0.61 ± 0.16	0.63 ± 0.17	$\mathbf{0.67 \pm 0.16}$
	NetLSD	$\mathbf{86.23 \pm 7.68}$	$\mathbf{83.71 \pm 8.03}$	83.38 ± 7.54	0.69 ± 0.16	$\mathbf{0.63 \pm 0.18}$	0.62 ± 0.17
	FGSD	86.01 ± 7.77	78.68 ± 9.57	81.21 ± 8.79	$\mathbf{0.70 \pm 0.16}$	0.55 ± 0.19	0.60 ± 0.18
	FeatherGraph	82.40 ± 8.24	69.72 ± 7.22	68.23 ± 7.20	0.60 ± 0.17	0.23 ± 0.22	0.17 ± 0.23
	Netpro2vec^{ndd}	71.42 ± 9.19	61.75 ± 8.78	62.34 ± 0.06	0.35 ± 0.21	0.00 ± 0.22	0.12 ± 0.23
	Netpro2vec$^{ndd+tm1}$	72.06 ± 9.64	73.95 ± 8.60	76.51 ± 9.27	0.34 ± 0.19	0.41 ± 0.20	0.47 ± 0.21

For the LFR dataset, the performance on unattacked graphs is high for all methods. Indeed, as already shown in [31], most of the considered methods succeed in producing whole-graph embeddings that lead to an almost perfect linear separation between the two LFR classes. In the case of moderate attacks, NetLSD and FGSD (but also FeatherGraph and Netpro2vec with NDD) respond better to both the types of adversarial attack, showing a lower reduction in Accuracy and MCC values as compared to the other methods. For stronger attacks, FeatherGraph reveals the most robust method, experiencing only a slight performance decrease.

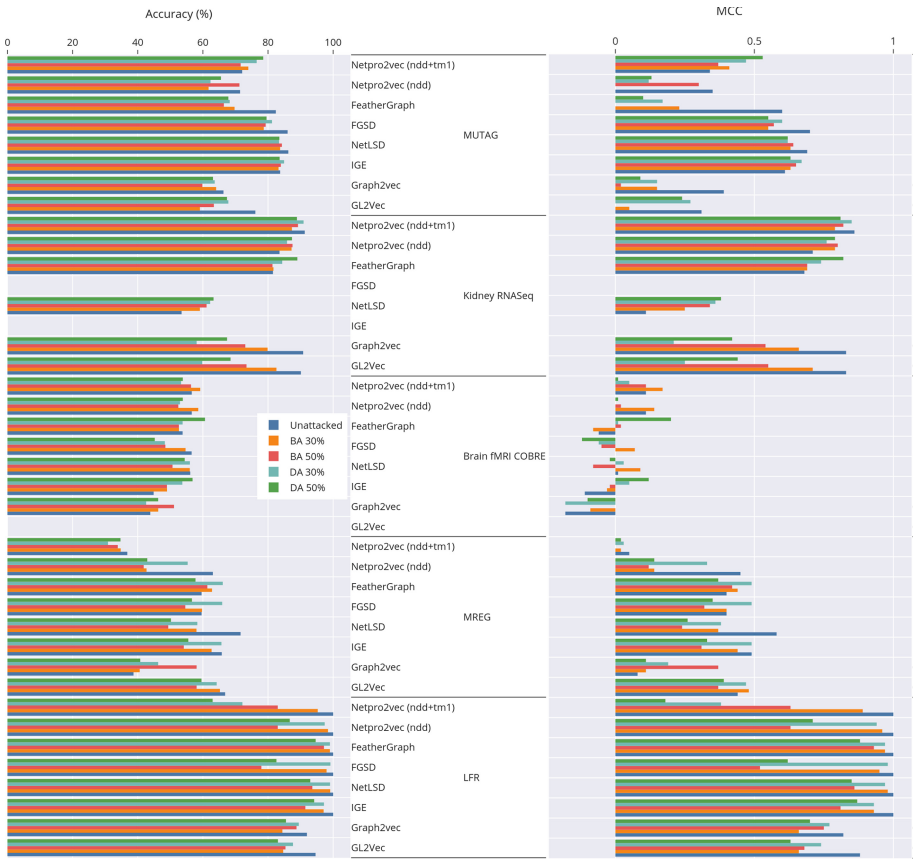

Fig. 1. Bar plots of the accuracy (%) and MCC of adversarial attacks on whole-graph embedding models

In the case of the MREG dataset, discordant results can be observed. Indeed, the best-performing method using unattacked data (NetLSD) experiences a dramatic performance decrease when handling both types of attacks. In contrast, the second-best method (GL2vec) is subject to a much lower performance decrease under attack, and this behavior holds whichever the strength of the attack. On the other side, the attacks can even be beneficial for classification performance, as for FeatherGraph that improves its performance under the moderate BA and the stronger DA.

For both the fMRI and Kidney datasets, Netpro2vec, mainly when based on NDD+TM1, appears to be the method that best exploits the network edges' weights. At the same time, it proves to be quite robust to adversarial attacks, experiencing slightly decreased performance for both moderate and strong attacks. Instead, NetLSD improves its performance when handling moderate DAs, showing the best performance among all the compared methods, and

Table 3. Accuracy (%) and MCC (mean and std over ten iterations) of adversarial attacks on whole-graph embedding models (50% of attacked nodes). In boldface the best results for each dataset and each attack.

Dataset	Method	Accuracy			MCC		
		Unattacked	BA	DA	Unattacked	BA	DA
LFR	GL2Vec	94.59 ± 1.75	85.36 ± 2.65	83.04 ± 2.56	0.88 ± 0.03	0.68 ± 0.05	0.63 ± 0.05
	Graph2vec	91.94 ± 2.04	88.74 ± 2.36	85.51 ± 2.69	0.82 ± 0.04	0.75 ± 0.05	0.70 ± 0.05
	IGE	$\mathbf{100.00 \pm 0.00}$	91.46 ± 2.10	94.17 ± 1.85	$\mathbf{1.00 \pm 0.00}$	0.81 ± 0.04	0.87 ± 0.03
	NetLSD	$\mathbf{100.00 \pm 0.00}$	93.60 ± 1.93	92.97 ± 1.99	$\mathbf{1.00 \pm 0.00}$	0.86 ± 0.04	0.85 ± 0.04
	FGSD	$\mathbf{100.00 \pm 0.00}$	77.96 ± 2.54	82.58 ± 2.97	$\mathbf{1.00 \pm 0.00}$	0.52 ± 0.05	0.62 ± 0.06
	FeatherGraph	$\mathbf{100.00 \pm 0.00}$	$\mathbf{97.17 \pm 1.19}$	$\mathbf{94.62 \pm 1.79}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.93 \pm 0.02}$	$\mathbf{0.88 \pm 0.03}$
	Netpro2vec^{ndd}	$\mathbf{100.00 \pm 0.00}$	82.99 ± 2.55	86.67 ± 2.40	$\mathbf{1.00 \pm 0.00}$	0.63 ± 0.05	0.71 ± 0.05
	Netpro2vec$^{ndd+tm1}$	$\mathbf{100.00 \pm 0.00}$	82.99 ± 2.55	62.99 ± 3.73	$\mathbf{1.00 \pm 0.00}$	0.63 ± 0.05	0.18 ± 0.08
MREG	GL2Vec	66.83 ± 7.14	58.07 ± 8.46	$\mathbf{59.57 \pm 8.28}$	0.44 ± 0.10	0.37 ± 0.12	$\mathbf{0.39 \pm 0.12}$
	Graph2vec	38.70 ± 8.46	58.07 ± 8.46	40.77 ± 8.18	0.08 ± 0.13	0.37 ± 0.12	0.11 ± 0.12
	IGE	65.80 ± 8.60	54.13 ± 6.73	55.53 ± 7.47	0.49 ± 0.12	0.31 ± 0.10	0.33 ± 0.11
	NetLSD	$\mathbf{71.57 \pm 7.05}$	49.37 ± 7.78	50.20 ± 7.48	$\mathbf{0.58 \pm 0.10}$	0.24 ± 0.11	0.26 ± 0.11
	FGSD	59.60 ± 6.08	54.60 ± 7.98	56.63 ± 8.13	0.40 ± 0.09	0.32 ± 0.12	0.35 ± 0.12
	FeatherGraph	59.60 ± 6.08	$\mathbf{61.37 \pm 8.02}$	57.73 ± 8.18	0.40 ± 0.09	$\mathbf{0.42 \pm 0.12}$	0.37 ± 0.12
	Netpro2vec^{ndd}	63.07 ± 7.60	41.83 ± 9.18	42.93 ± 7.92	0.45 ± 0.11	0.12 ± 0.14	0.14 ± 0.12
	Netpro2vec$^{ndd+tm1}$	36.80 ± 8.48	33.87 ± 8.71	34.70 ± 7.93	0.05 ± 0.12	0.00 ± 0.13	0.02 ± 0.12
Brain fMRI COBRE	GL2Vec	No conv	No conv	No conv	No conv	No conv	No conv
	Graph2vec	43.85 ± 11.27	51.13 ± 11.93	46.27 ± 13.14	-0.18 ± 0.24	-0.00 ± 0.25	-0.10 ± 0.27
	IGE	44.88 ± 14.70	48.96 ± 13.21	56.83 ± 13.16	-0.11 ± 0.30	-0.02 ± 0.27	0.12 ± 0.27
	NetLSD	56.12 ± 6.59	50.68 ± 10.30	54.41 ± 6.25	0.01 ± 0.18	-0.08 ± 0.20	-0.02 ± 0.13
	FGSD	56.54 ± 2.20	48.49 ± 13.51	45.22 ± 11.42	0.00 ± 0.00	-0.05 ± 0.28	-0.12 ± 0.23
	FeatherGraph	53.77 ± 5.89	52.63 ± 12.04	$\mathbf{60.65 \pm 13.51}$	-0.06 ± 0.16	0.02 ± 0.26	$\mathbf{0.20 \pm 0.28}$
	Netpro2vec^{ndd}	$\mathbf{56.58 \pm 12.74}$	52.46 ± 13.24	53.83 ± 11.31	$\mathbf{0.11 \pm 0.27}$	0.02 ± 0.28	0.01 ± 0.26
	Netpro2vec$^{ndd+tm1}$	$\mathbf{56.58 \pm 12.74}$	$\mathbf{56.35 \pm 12.46}$	53.83 ± 11.31	$\mathbf{0.11 \pm 0.27}$	$\mathbf{0.11 \pm 0.25}$	0.01 ± 0.26
Kidney RNASeq	GL2Vec	90.09 ± 4.74	73.39 ± 7.56	68.49 ± 7.34	0.83 ± 0.08	0.55 ± 0.12	0.44 ± 0.14
	Graph2vec	90.79 ± 5.11	73.02 ± 7.30	67.42 ± 7.44	0.83 ± 0.08	0.54 ± 0.12	0.42 ± 0.14
	IGE	No conv	No conv	No conv	No conv	No conv	No conv
	NetLSD	53.46 ± 7.02	61.13 ± 8.00	63.27 ± 7.76	0.11 ± 0.16	0.34 ± 0.14	0.38 ± 0.13
	FGSD	No conv	No conv	No conv	No conv	No conv	No conv
	FeatherGraph	81.51 ± 7.96	81.37 ± 6.83	$\mathbf{89.00 \pm 4.79}$	0.68 ± 0.13	0.69 ± 0.11	$\mathbf{0.82 \pm 0.07}$
	Netpro2vec^{ndd}	83.53 ± 6.42	87.52 ± 5.66	87.35 ± 5.30	0.71 ± 0.11	0.80 ± 0.10	0.79 ± 0.08
	Netpro2vec$^{ndd+tm1}$	$\mathbf{91.27 \pm 4.45}$	$\mathbf{89.20 \pm 5.36}$	88.87 ± 5.68	$\mathbf{0.86 \pm 0.07}$	$\mathbf{0.82 \pm 0.09}$	0.81 ± 0.09
MUTAG	GL2Vec	76.11 ± 8.48	63.37 ± 9.25	67.39 ± 10.10	0.31 ± 0.24	0.00 ± 0.22	0.24 ± 0.24
	Graph2vec	66.32 ± 9.72	59.86 ± 9.32	63.10 ± 8.28	0.39 ± 0.24	0.02 ± 0.24	0.09 ± 0.21
	IGE	83.72 ± 7.92	83.93 ± 7.85	83.56 ± 7.47	0.61 ± 0.16	$\mathbf{0.65 \pm 0.17}$	$\mathbf{0.63 \pm 0.17}$
	NetLSD	$\mathbf{86.23 \pm 7.68}$	$\mathbf{84.25 \pm 7.61}$	83.48 ± 7.77	0.69 ± 0.16	0.64 ± 0.16	0.62 ± 0.17
	FGSD	86.01 ± 7.77	79.27 ± 8.80	$\mathbf{79.54 \pm 9.17}$	$\mathbf{0.70 \pm 0.16}$	0.57 ± 0.17	0.55 ± 0.20
	FeatherGraph	82.40 ± 8.24	66.44 ± 2.30	67.82 ± 4.63	0.60 ± 0.17	0.00 ± 0.00	0.10 ± 0.18
	Netpro2vec^{ndd}	71.42 ± 9.19	71.22 ± 8.09	65.55 ± 8.54	0.35 ± 0.21	0.30 ± 0.22	0.13 ± 0.23
	Netpro2vec$^{ndd+tm1}$	72.06 ± 9.64	71.60 ± 11.32	78.52 ± 8.30	0.34 ± 0.19	0.37 ± 0.20	0.53 ± 0.18

the same can be said for FeatherGraph under strong attack. Other methods, such as GL2vec on fMRI or IGE and FGSD on Kidney, fail to reach convergence in all the unattacked and attacked cases, yielding no classification model.

On the MUTAG dataset, the best-performing method (NetLSD) experiences a tiny performance decrease in handling moderate and strong adversarial attacks. The same can also be said for IGE and Netpro2vec, which maintain similar performance, if not better, regardless of the attacks.

Overall, we can conclude that FeatherGraph, NetLSD, Netpro2vec, and IGE appear to be more robust than the other methods under both moderate and strong adversarial attacks. We also observed unexpected behaviors in some cases,

where an improvement in performance rather than a degradation occurs. Indeed, removing central (important) nodes does not always weaken the significance of the graph description. Therefore, these nodes can be considered important but not fundamental for the transformation from a graph to a vector space. This point deserves further attention in future experiments.

We wish to emphasize that the above analysis is primarily intended to investigate the robustness to adversarial attacks of the considered methods rather than their performance for the classification task. Indeed, further optimization steps, such as feature selection or class balancing, have been purposely omitted for all the methods, which would certainly help in achieving better classification performance.

5 Conclusions and Future Work

We have analyzed and compared different whole-graph embedding methods to understand their behavior under adversarial attacks better. We have performed attacks on graphs supposing the subsequent data analysis task is supervised classification. During the attacks, we have analyzed the unique features of each embedding method to highlight strengths and weaknesses, varying the type of attack and dataset. In this regard, the robustness of the graph analysis task model is an important issue. Future works concern many directions. First, extending the analysis on different types of datasets and attacks to propose defense mechanisms that can partially or entirely erase the highlighted limits of existing solutions. Besides, it would be interesting to analyze the embedding features that the methods create for the classification task. Methods like SHapley Additive exPlanations (SHAP) [28] could be applied to learn feature importance and explain the model output.

Acknowledgments. This work has been partially funded by BiBiNet project (H35F21000430002) within POR-Lazio FESR 2014–2020. It was carried out also within the activities of the authors as members of the ICAR-CNR INdAM Research Unit and partially supported by the INdAM research project "Computational Intelligence methods for Digital Health". Mario Manzo thanks Prof. Alfredo Petrosino for the guidance and supervision during the years of working together. The work of Mario R. Guarracino was conducted within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE).

References

1. Aine, C.J., Jeremy Bockholt, H., Bustillo, J.R., et al.: Multimodal neuroimaging in schizophrenia: description and dissemination. Neuroinformatics **15**(4), 343–364 (2017)
2. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: a survey. IEEE Access **6**, 14410–14430 (2018). https://doi.org/10.1109/ACCESS.2018.2807385

3. Arroyo-Relión, J.D., et al.: Network classification with applications to brain connectomics [Internet]. Ann. Appl. Stat. **13**(3), 1648 (2019)
4. Beygelzimer, A., et al.: Improving network robustness by edge modification. Physica A Stat. Mech. Appl. **357**(3–4), 593–612 (2005)
5. Cai, H., Zheng, V.W., Chang, K.C.-C.: A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018)
6. Chen, H., Koga, H.: GL2vec: graph embedding enriched by line graphs with edge features. In: Gedeon, T., Wong, K.W., Lee, M. (eds.) ICONIP 2019. LNCS, vol. 11955, pp. 3–14. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36718-3_1
7. Chen, J., et al.: GraphAttacker: a general multi-task graphattack framework. In: arXiv preprint arXiv:2101.06855 (2021)
8. Chen, L., et al.: A survey of adversarial learning on graphs. In: CoRR abs/2003.05730 (2020). arXiv: 2003.05730
9. Chung, H.W., Datta, A., Waites, C.: GABO: graph augmentations with bi-level optimization. In: arXiv preprint arXiv:2104.00722 (2021)
10. Chvátal, V.: Tough graphs and Hamiltonian circuits. Discret. Math. **5**(3), 215–228 (1973)
11. Dai, H., et al.: Adversarial attack on graph structured data. In: CoRR abs/1806.02371 (2018). arXiv: 1806.02371
12. Dalvi, N., et al.: Adversarial classification. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2004, pp. 99–108. Association for Computing Machinery, Seattle (2004). https://doi.org/10.1145/1014052.1014066. ISBN 1581138881
13. Debnath, A.K., et al.: Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. J. Med. Chem. **34** (1991). https://doi.org/10.1021/jm00106a046
14. Faber, L., Moghaddam, A.K., Wattenhofer, R.: Contrastive graph neural network explanation. In: arXiv preprint arXiv:2010.13663 (2020)
15. Galland, A., Lelarge, M.: Invariant embedding for graph classification. In: ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Representations (2019)
16. Gao, J., et al.: Black-box generation of adversarial text sequences to evade deep learning classifiers. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 50–56 (2018). https://doi.org/10.1109/SPW.2018.00016
17. Granata, I., et al.: A short journey through whole graph embedding techniques. In: International Conference on Network Analysis (NET 2020) (2020)
18. Granata, I., et al.: Model simplification for supervised classification of metabolic networks. Ann. Math. Artif. Intell. **88**(1), 91–104 (2020)
19. Granata, I., Guarracino, M.R., Maddalena, L., Manipur, I.: Network distances for weighted digraphs. In: Kochetov, Y., Bykadorov, I., Gruzdeva, T. (eds.) MOTOR 2020. CCIS, vol. 1275, pp. 389–408. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58657-7_31 ISBN 978-3-030-58657-7
20. Granata, I., et al.: Supervised classification of metabolic networks. In: 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 2688–2693. IEEE (2018)
21. Gutiérrez-Gómez, L., Delvenne, J.-C.: Unsupervised network embeddings with node identity awareness. Appl. Netw. Sci. **4**(1), 82 (2019)
22. Holme, P., et al.: Attack vulnerability of complex networks. Phys. Rev. E **65**(5), 056109 (2002)

23. Huang, L., et al.: Adversarial machine learning. In: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, AISec 2011, pp. 43–58. Association for Computing Machinery, Chicago (2011). https://doi.org/10.1145/2046684.2046692. ISBN 9781450310031
24. Jin, H., et al.: Certified robustness of graph convolution networks for graph classification under topological attacks. In: Advances in Neural Information Processing Systems, vol. 33 (2020)
25. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E **78**(4), 046110 (2008)
26. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)
27. Lin, Y., et al.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 29, January 2015
28. Lundberg, S.M., Lee, S.-I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, vol. 30, pp. 4765–4774 (2017)
29. Maddalena, L., et al.: On whole graph embedding techniques. In: International Symposium on Mathematical and Computational Biology (BIOMAT 2020), November 2020
30. Manipur, I., et al.: Clustering analysis of tumor metabolic networks. BMC Bioinform. (2020). https://doi.org/10.1186/s12859-020-03564-9. ISSN 1471–2105
31. Manipur, I., et al.: Netpro2vec: a graph embedding framework for biomedical applications. IEEE/ACM Trans. Comput. Biol. Bioinform. (2021)
32. Matthews, B.W.: Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA) - Protein Struct. **405**(2), 442–451 (1975). https://doi.org/10.1016/0005-2795(75)90109-9. ISSN 0005–2795
33. Narayanan, A., et al.: graph2vec: learning distributed representations of graphs. In: ArXiv abs/1707.05005 (2017)
34. Qiu, S., et al.: Review of artificial intelligence adversarial attack and defense technologies. Appl. Sci. **9**(5) (2019). https://doi.org/10.3390/app9050909. ISSN 2076–3417
35. Řehůřrek, R., Sojka, P.: Software framework for topic modelling with large corpora. English. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50. ELRA, Valletta, May 2010
36. Rosenberg, I., Shabtai, A., Rokach, L., Elovici, Y.: Generic black-box end-to-end attack against state of the art API call based malware classifiers. In: Bailey, M., Holz, T., Stamatogiannakis, M., Ioannidis, S. (eds.) RAID 2018. LNCS, vol. 11050, pp. 490–510. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00470-5_23 ISBN 978-3-030-00470-5
37. Rozemberczki, B., Kiss, O., Sarkar, R.: Karate Club: an API oriented open-source Python framework for unsupervised learning on graphs. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM 2020). ACM (2020)
38. Rozemberczki, B., Sarkar, R.: Characteristic functions on graphs: birds of a feather, from statistical descriptors to parametric models. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 1325–1334 (2020)
39. Sun, L., et al.: Adversarial attack and defense on graph data: a survey. In: CoRR abs/1812.10528 (2018). arXiv: 1812.10528

40. Tang, H., et al.: Adversarial attack on hierarchical graph pooling neural networks. In: arXiv preprint arXiv:2005.11560 (2020)
41. Thorne, T., Stumpf, M.P.H.: Graph spectral analysis of protein interaction network evolution. J. Royal Soc. Interface **9**(75), 2653–2666 (2012)
42. Tong, H., et al.: On the vulnerability of large graphs. In: 2010 IEEE International Conference on Data Mining, pp. 1091–1096. IEEE (2010)
43. Tsitsulin, A., et al.: NetLSD: hearing the shape of a graph. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2347–2356 (2018)
44. Uhlén, M., et al.: Tissue-based map of the human proteome. Science **347**(6220) (2015)
45. Verma, S., Zhang, Z.-L.: Hunt for the unique, stable, sparse and fast feature learning on graphs. In: Advances in Neural Information Processing Systems, pp. 88–98 (2017)
46. Vlietstra, W.J., et al.: Using predicate and provenance information from a knowledge graph for drug efficacy screening. J. Biomed. Semant. **9**(1), 1–10 (2018)
47. Wang, S., et al.: Joint embedding of graphs. IEEE Trans. Pattern Anal. Mach. Intell. **43**(4), 1324–1336 (2021). https://doi.org/10.1109/TPAMI.2019.2948619
48. Xi, Z., et al.: Graph backdoor. In: 30th USENIX Security Symposium (USENIX Security 2021) (2021)
49. You, Y., et al.: Graph contrastive learning with augmentations. In: Advances in Neural Information Processing Systems, vol. 33 (2020)