Annals of Mathematics and Artificial Intelligence (2023) 91:259–285
https://doi.org/10.1007/s10472-022-09811-4

Adversarial attacks on graph-level embedding methods: a case study

Maurizio Giordano[1] · **Lucia Maddalena**[1] · **Mario Manzo**[2] · **Mario Rosario Guarracino**[3]

Accepted: 27 July 2022 / Published online: 6 October 2022

Abstract
As the number of graph-level embedding techniques increases at an unprecedented speed, questions arise about their behavior and performance when training data undergo perturbations. This is the case when an external entity maliciously alters training data to invalidate the embedding. This paper explores the effects of such attacks on some graph datasets by applying different graph-level embedding techniques. The main attack strategy involves manipulating training data to produce an altered model. In this context, our goal is to go in-depth about methods, resources, experimental settings, and performance results to observe and study all the aspects that derive from the attack stage.

Keywords Adversarial attacks · Adversarial machine learning · Graph embedding · Graph neural networks · Graph classification

Mathematics subject classification (2010) 68T01 · 68T07 · 68R10

Mathematics subject classification (2020) 68T01 · 68T07 · 92B20 · 68R10

✉ Maurizio Giordano
maurizio.giordano@cnr.it

Lucia Maddalena
lucia.maddalena@cnr.it

Mario Manzo
mmanzo@unior.it

Mario Rosario Guarracino
mario.guarracino@unicas.it

[1] High Performance Computing and Networking Institute (ICAR), National Research Council (CNR), Via Pietro Castellino 111, Naples 80131, Italy

[2] Information Technology Services, University of Naples "L'Orientale", Via Nuova Marina 59, Naples 80133, Italy

[3] Department of Economics and Law, University of Cassino and Southern Lazio, Campus Folcara, Cassino 03043, Italy

1 Introduction

Graphs are playing an important role in many real-world applications, and related data analysis techniques are showing their feasibility also on large-scale problems, including drug screening [1], cancer metabolism [2], protein analysis [3], resource deployment [4], cooperative control strategies [5], and knowledge graph completion [6].

Graph embedding is an emergent research area in machine learning (ML) that includes techniques and methods to find "latent vector representations" of graphs to capture their topology and preserve relevant network properties [7–9]. The resulting graph representations can be made rich by considering several information sources, such as vertex-vertex relationships and vertex/edge attributes. Graph embedding techniques attract significant interest in the ML community since vector spaces are more amenable to data science than graphs. Indeed, while network relationships can only be processed by specific techniques from mathematics, statistics, and ML, vector spaces have a richer toolset from those disciplines. In addition, vector operations are often simpler and faster than the equivalent graph operations.

In everyday real-life applications, due to the increasing pervasiveness of ML, deep learning, and AI algorithms, the robustness and vulnerability of these algorithms have now become crucial and very important aspects of research. In the specific context of AI applied to networked data processing, application domains include cybersecurity, online financial trading, social media, big-data analytics, and bioinformatics.

Here, the question arises on how to handle situations in which input data is altered, due to acquisition noise or intentional modifications (the so-called *adversarial attacks*), leading to misleading conclusions or reduced algorithms performance. In these contexts, the real goal of an attack is to cause (intentionally or not) the malfunctioning and/or fraudulent behavior of algorithms operating on data structured as graphs. This occurs as a result of perturbations of the graphs performed by the attacker, where the changes can be more or less significant and targeted based on knowledge of the intrinsic functioning of these algorithms (parameters, implementation logic, etc.). This is the assumption of Adversarial Machine Learning (AML) [10]. This new research area studies and proposes solutions to tackle the vulnerability of ML models when their performance in different tasks is compromised through adversarial perturbations on their input. Indeed, neural networks and many other ML techniques suffer this problem when input modifications occur during training, testing, or deploying phases. Notable application domains of AML are Computer Vision [11, 12], Natural Language Processing (NLP) [13], and Cybersecurity [14].

Among the reviews on graph adversarial attacks, [15, 16] mainly focus on GNN-based methods, while [17] also covers attack and defense models for non-GNN methods. The *Graph Adversarial Learning Literature repository*, produced by Sun et al. [17], has a curated selection of more than 110 adversarial attack and defense studies on graph-structured data, as well as links to downloadable programs. The *Awesome Graph Adversarial Learning repository*, built and maintained by Chen et al. [18], includes links to 271 relevant publications published in the last five years. Literature on adversarial attacks is mainly focused on poisoning strategies, such as *backdoor attacks* [19], where the training stage of the target model is perturbed and its behavior is normal unless a specific *trigger* is present in the test samples, training to mislead graph prediction [20]. The general aim is to affect the performance of graph-level tasks, and none of these works shares our goal, which is to compare the robustness to adversarial attacks of different graph-level embedding methods.

In this work, we address a specific application domain for AML: the study of the vulnerability of ML models applied to the classification/prediction of biological networks. In our

assumption, an "adversarial attack" on a biological network concerns any type of perturbation to the structure of the graph due both to the noise introduced by the experimental environment from which the biological data are extracted and to the lack of information due to corrupted sources, or incomplete pre-processing of raw data. Therefore, in our mind, we consider less likely, but still possible, a scenario in which a "real" attacker can intentionally and fraudulently modify the biological networks processed by the ML models.

In this context, we aim to study and measure the robustness of some graph embedding methods, two based on neural networks and another based on statistics, when we alter the training stage of these models through adversarial perturbations to the input data. The altered models of graph-to-vector transformers (*embedders*) are evaluated with respect to their robustness by measuring classification performance on unperturbed test graphs. In this work, we keep on developing the research activities carried out in [21, 22], which, as far as we know, are the only literature contributions to the robustness analysis of graph embedding techniques.

The paper is structured as follows. Section 2 introduces definitions, types, and properties of graph embedding and adversarial attack taxonomies. Section 3 discusses our approach to adversarial attacks on graph-embedding methods and the related experimental study by delving into methods, resources, experimental settings, and performance results. Section 4 reports concluding considerations and future directions for the work.

2 Background

2.1 Graph embedding definitions

The general term of *graph embedding methods* denotes a plethora of techniques and methodologies to translate large and complex graphs into a reduced vector space, which is often called *latent space*. In other words, any procedure that constructs a vector representation of a graph in order to simplify and/or make a certain machine learning task more efficient is called graph embedding.

Definition 1 (**Graph Embedding**) A graph embedding is a mapping ϕ from a collection of graph substructures (most commonly either all nodes, or all edges, or certain subgraphs, or even the whole graph) to $\mathbb{R}^d$.

Graph embedding techniques differ in which aspects of the graph we try to represent:

- *Node-level embeddings* describe the connectivity of the graph. Each node in the graph is associated with a vector representation. Node-level embeddings target node prediction, reconstruction, and graph clustering.
- *Edge-level embeddings* describe traversals across the graph. Each edge in the graph is associated with a vector representation. Edge-level embeddings target edge prediction, reconstruction, and graph clustering.
- *Graph-level embeddings* encode the entire graph into a single vector. Each element in a set of graphs is associated with a vector representation. Graph-level embeddings target graph classification and graph matching.

In this work, we focus on graph-level embedding (see Fig. 1), more precisely in the realm of graph classification tasks in the biological networks domain. Graph-level embedding, also known as whole-graph embedding, can be formally defined as:

Definition 2 (**Graph-level Embedding**) Given a set of graphs $\mathcal{G} = \{G_1, \ldots, G_m\}$, a graph-level embedding is a mapping function $\phi : \mathcal{G} \rightarrow \mathbb{R}^d$ where $d \in \mathbb{N}$, such that ϕ preserves some proximity measure defined on $\mathcal{G}$.

Choosing an appropriate embedding dimension d is challenging but necessary, and above all crucial, to generate embeddings applicable to a multitude of tasks. A general rule of thumb is "small enough to be efficient and large enough to be effective". The criticality concerning the final latent space dimension is that it should express all valuable information needed to accomplish the machine learning task on graphs.

When talking about graph embedding techniques, it is important to be aware of another distinction:

- In *transductive embedding*, the vector representation for a new graph is produced by the embedding function (or model), requiring to process the new graph jointly with previous graphs. At each new graph embedding, the vectors for previous graphs change.
- In *inductive embedding*, the vector representation (embedding) for a new graph is produced by the embedding function (or model), requiring only the processing of the new graph. The embedding for older graphs does not change when we perform the embedding of a new graph.

The embedding process can be *unsupervised* or *supervised*. Thus, transductive supervised embedding methods cannot be used as models for prediction and classification tasks on unknown graphs.

2.2 Adversarial attack taxonomies

Several surveys in the literature [15–17] propose different taxonomies for graph/network attacks based on the goals, knowledge, and resources of the attackers. Adversarial samples of graph data can be produced either through *node-level* perturbations or by *edge-level*

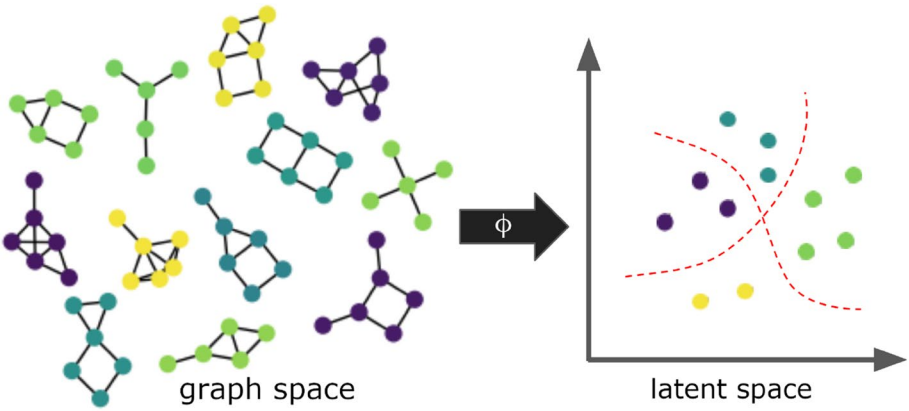

Fig. 1 Graph-level embedding

perturbations. *Node-level* attacks may consist in adding/removing nodes and/or modifying target node features. *Edge-level* attacks may consist in adding/removing edges between nodes and/or modifying target edge features. In both cases, the number of modified nodes/edges is often referred to as the *perturbation budget*, which is used to evaluate the magnitude of the perturbation.

Evasion attacks refer to modifications to only the testing data on which a model is applied to accomplish the requested task (e.g., classification, regression, clustering, matching, etc.). In evasion attacks, there is no need to know the model insights (architecture, parameters, and so on). *Poisoning attacks* aim to affect the model's performance by adding adversarial samples into the training dataset. The majority of adversarial attacks in graph-based machine learning is of this type. In addition, in the case the task of the trained model is performed in the transductive learning setting, any evasion attack results in a poisoned attack since the model is re-trained after testing.

Concerning the amount of knowledge the attacker has about the target models, the types of attacks are classified as: 1) *white-box* attack, in which the attacker can retrieve all the useful information about the target system to successfully complete the attack, such as the underlying model, its architecture and parameters, etc.; 2) *gray-box* attack, in which the attacker can only obtain limited information about the target system to perform the attack. This type of attack is more dangerous to the system than white-box attacks, as it only needs partial information to work; 3) *black-box* attack, in which the attacker has no information about the system. Generally, it is only allowed to do black-box queries on limited samples at most, and thus it cannot make poisoning attacks on the trained model. However, if a black-box attack works, it is more dangerous than the other two since the attacker succeeds with no information at all.

Regarding the goal, the attack can be *targeted* if the attacker pursues a specific goal, such as the prediction of wrong labels in a graph classification task accomplished by a trained model. On the other hand, the attack is *untargeted* when the attacker aims at a general malfunctioning or degradation in performance of the model under attack.

3 Graph embedding adversarial attack

The present experimental analysis compares the behavior of three graph-level embedding methods under attack conditions for the graph classification task. This section is organized as follows. In Section 3.1, we describe the datasets considered in the experiments for the robustness evaluation of embedding models. In Section 3.2, we introduce the adopted attack strategies and their rationale. Then, in Section 3.3, we briefly describe the embedding methods considered in the experiments. In Section 3.4, we present the proposed experimental pipeline that trains the embedding models on attacked graphs and evaluates their performance on test graphs. In the last Section 3.5, we discuss the experimental results.

3.1 Datasets

We consider three graph datasets having varying properties, as detailed in Table 1.

MUTAG is a popular benchmark dataset composed of networks of 188 mutagenic aromatic and heteroaromatic nitro compounds [23]. The two classes indicate whether or not the compound has mutagenic effects on a bacterium. The nodes represent the atoms of the compound, while the edges represent the chemical bonds between them. The graphs contain both vertex and edge labels.

Table 1 Main properties of the adopted datasets

Property	MUTAG	PROTEINS	Kidney
# graphs	188	1113	299
# classes	2	2	3
# samples per class	125/63	663/450	159/90/50
Average # nodes	17.93	39.06	1034
Average # edges	19.79	72.82	3226.00
Average edge density	0.138	0.212	0.006
# distinct node labels	7	3	1034
Edge weights	✗	✗	✓
Minimum diameter	5	1	7
Maximum diameter	15	54	7
Average degree	2.19	3.73	6.24

The **PROTEINS** dataset consists of 1113 graphs corresponding to protein molecules, subdivided into two classes according to whether or not they are enzymes [24]. The nodes represent Secondary Structure Elements of three different types (helix, sheet, or turn). Edges connect two nodes if they are neighbors in the amino-acid sequence or in the 3D space.

The **Kidney** dataset includes tissue-specific metabolic networks created for validating related research [25–27]. It contains networks representing 299 patients divided into three disease classes: 159 clear cell Renal Cell Carcinoma (KIRC), 90 Papillary Renal Cell Carcinoma (KIRP), and 50 Solid Tissue Normal samples.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 [28] (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 [25]. This results in graphs having the same topology (dictated by the metabolic model) but different edge weights (dictated by the gene expression values for each patient). The simplification procedure described in [26] is applied to reduce the complexity of the network, leading to reduce the number of nodes from 4022 to 1034. For our experimental study, we augmented the Kidney graphs with additional numerical labels storing the weighted degree of each node. Indeed, Kidney graphs have all the same structure (same set of vertices and edges), while they differ in the weights associated with edges among different graph samples. Since two of the embedding methods under study cannot process edge weights, we decided to provide this information as node labels. With this choice, we could apply these embedding methods efficiently in this particular graph domain.

While MUTAG is a small dataset of small graphs (few nodes and edges each), PROTEINS is a much larger set of small graphs, and Kidney is a medium-sized dataset of large graphs. The first two datasets can be considered historical benchmarks since they have been widely adopted as benchmarks in several works on graph classification in the last decade. In contrast, the larger-scale Kidney dataset is here adopted as a more challenging benchmark.

3.2 Attack strategies

The attack strategies we consider are edge-level perturbations of the input graphs consisting in the removal of a chosen budget of their edges. Which edges are to be removed is decided according to four different criteria:

Random - It is the baseline removal strategy, where edges to be removed are randomly chosen.
Betweenness Centrality - It measures the centrality of an edge e defined [29] as the sum of the fraction of all-pairs shortest paths that pass through e

$$c_B(e) = \sum_{i,j \in V} \frac{\sigma(i,j \mid e)}{\sigma(i,j)}, \tag{1}$$

where V is the set of nodes, $\sigma(i,j)$ is the number of shortest-paths connecting vertices i and j, and $\sigma(i,j \mid e)$ is the number of those paths passing through the edge e.
Eigenvector Centrality - Known also as *eigen-centrality* [30], it describes the importance of a node in a graph based on that of its adjacent nodes. Let $A = (a_{i,j})$ be the adjacency matrix of a graph G. We can compute a weight x_i for node i in terms of the weights x_j for the other nodes j as

$$x_i = \lambda^{-1} \sum_j a_{ij} x_j, \tag{2}$$

where λ is a constant. Equation 2 can be rewritten as $Ax = \lambda x$ so that x is an eigenvector of the adjacency matrix A. The components of the eigenvector associated with the maximum eigenvalue measure the relative importance among the nodes, providing their ranking. To obtain the centrality of edges, rather than nodes, we use the eigenvector centrality computed on the *line graph* of G (i.e., the graph where nodes represent the edges of G and two vertices are adjacent if their corresponding edges in G are incident).
PageRank - The PageRank algorithm is a variant of the Eigenvector Centrality originally designed for ranking web content, using hyperlinks between pages as a measure of importance [31]. Let $A = (a_{i,j})$ be the adjacency matrix of a directed graph. The PageRank centrality x_i of node i is given by:

$$x_i = \alpha \sum_k \frac{a_{k,i}}{d_k} x_k + \beta_i, \tag{3}$$

where α and β_i are user-defined constants and $d_k = \max(o_k, 1)$, with o_k denoting the out-degree of node k. Thus, PageRank is determined by an endogenous component, namely the so-called *damping factor* α, that considers the network topology, and an exogenous component $\beta = (\beta_i)$, the so-called *personalization vector*, that is independent of the network structure. As for the eigenvector centrality, we compute the PageRank centrality of edges using the *line graphs* of the original graphs. In the experiments, we used $\alpha = 0.85$ and a null β vector.

The motivation for removing edges based on their centrality measure is based on the assumption that the most significant information in biological networks relies on some relevant links, where the relevance is measured in terms of the link centrality measure. For example, in the Kidney dataset (see Section 3.1), metabolites are linked if they appear in the same metabolic

reaction. Therefore, high centrality links identify metabolites that are involved in many metabolic reactions and thus represent significant information for the networks. Similarly, in the MUTAG and PROTEINS datasets, edges with high centrality identify significant backbone structures of chemical compounds and proteins, respectively (Fig. 2).

Our focus is to study the effects of *targeted* attacks on the graph embedding process. Indeed, the choice of an edge removal criterion based on edges relevance shows that we envision a data degradation or an attack causing a significant information loss in biological networks.

Figures 3 and 4 show pictures of a Kidney graph before and after edge removal, in particular, with the removal of edges with highest betweenness centrality score (see Fig. 3), or randomly chosen (see Fig. 4). It can be noted how the former edge selection strategy, when compared to the baseline edge removal, focuses more on the deletion of entire clusters of nodes characterized by a great number of high centrality connections (see the rightmost node cluster of Fig. 3a, which is disrupted in isolated nodes after edge removal in Fig. 3b).

The described adversarial attacks to the embedding models under study are *poisoning attacks*. Indeed, we limited graph perturbations only to the sets of graphs adopted to build the embedding models, which are then used to infer embeddings of testing graphs in an inductive learning environment, as detailed in Section 3.4.

3.3 Embedding methods

Two embedding methods considered in the experiments rely on neural networks models: Netpro2vec [32] and Graph2Vec [33]. These models learn a function that maps graphs into a numerical lower-dimensional space. This mapping is optimized in a learning process that uses one by one a set of training graph samples. The third method, FEATHER [34], is a probabilistic embedding model. Probabilistic models exploit the extraction of random walks in the graph to learn its global structure together with the local neighborhood connectivity. FEATHER behaves as

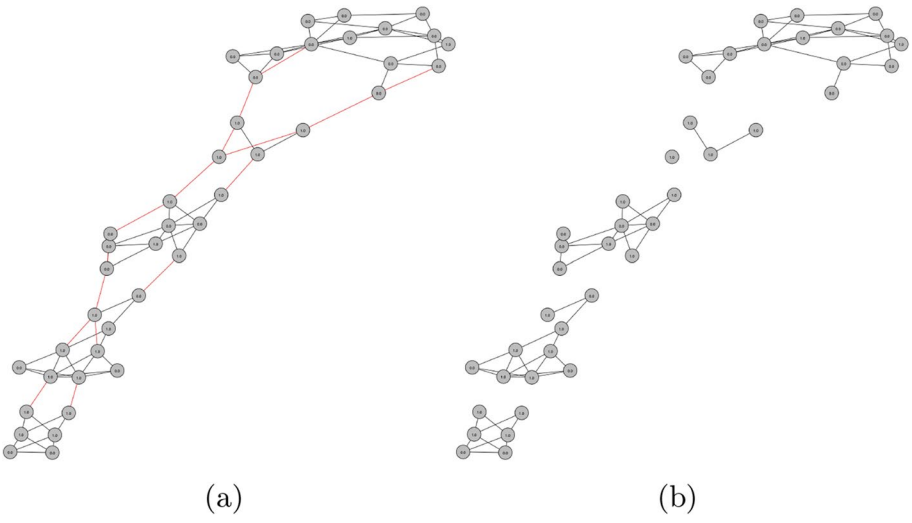

Fig. 2 A graph from PROTEINS dataset before (**a**) and after (**b**) a 20% budget of edge removal based on betweenness centrality (red edges in the left picture indicate those removed in the right picture)

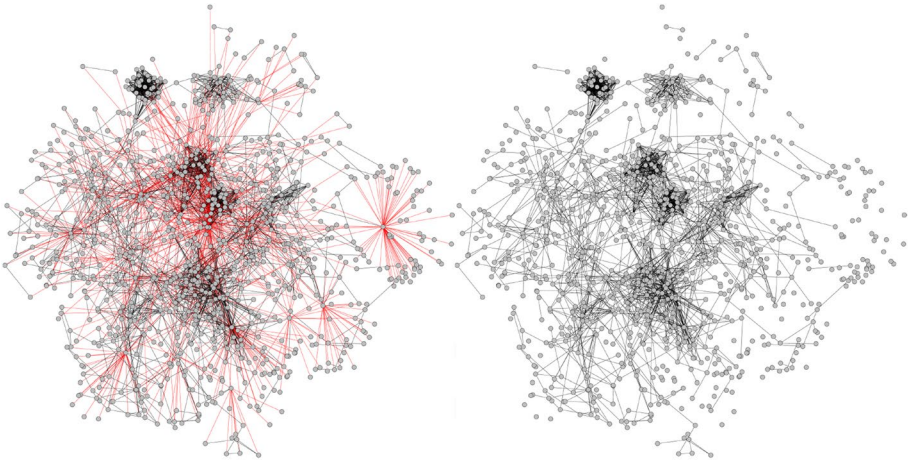

Fig. 3 A graph from the Kidney dataset before (**a**) and after (**b**) a 30% budget of edge removal based on betweenness centrality (red edges in the left picture indicate those removed in the right picture)

an embedding function performing graph-level embedding on each graph separately. Details are described in the following subsections.

3.3.1 Inductive Netpro2vec

Netpro2vec [32] is an unsupervised graph-level embedding method that exploits node proximity information (under different metrics) to transform graphs into textual documents while preserving their significant structural properties. Netpro2vec relies on an NLP learning model, called SkipGram [35], to extract, from each document-based graph,

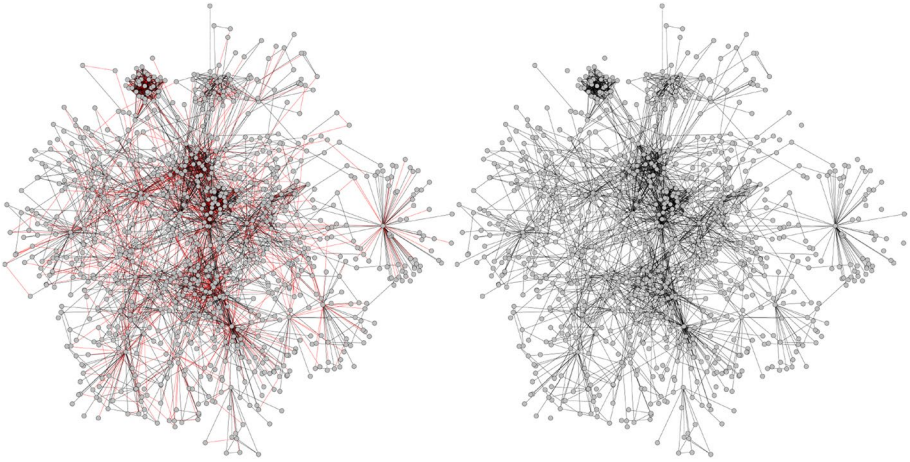

Fig. 4 A graph from the Kidney dataset (same as Fig. 3) before (**a**) and after (**b**) a 30% budget of random edge removal (red edges in the left picture indicate those removed in the right picture)

the meaningful features in terms of vectors, i.e., the *embeddings*. Such a new graph representation can be used for several machine learning tasks, such as unsupervised clustering and supervised classification of graphs. The main advantage of Netpro2vec is that it provides efficient embeddings completely independent of the task and nature of the data.

The current Netpro2vec implementation cannot be directly used in our experimental study since it provides a programming interface with the only support of transductive embedding. Nonetheless, by exploiting the Doc2Vec [36] facility to infer vector representation of new documents (in our case, graphs) based on a pre-trained embedding model, we developed a new API for the method, that we call *iNetpro2vec*, to support also inductive embedding.

3.3.2 Graph2Vec

Graph2Vec [33] is a neural method for learning graph-level embeddings in an unsupervised manner. First, the method relabels nodes through a recursive node relabeling algorithm assigning to each node a label uniquely representing the node's rooted subgraph (neighborhood). After recursion, the final node labels form a vocabulary of words, and graphs are represented as a set of words (a document) in this vocabulary. Like Netpro2vec, Graph2Vec relies on the Doc2Vec learning model to learn the graph embeddings. The initial labels of nodes are, by default, the node degrees, although the user can specify them as an additional input. Graph2Vec is a popular method among graph-level embedding techniques, and it has proved to have good performance throughout many graph domains.

Graph2Vec graph-level embeddings are learned in a transductive manner. Since this method shares with Netpro2vec the same NLP processing technique, also in this case, it is possible to use the Doc2Vec facility to infer embeddings of new samples based on a pre-trained neural model. Thus, in the current work, we developed a new API that we call *iGraph2Vec*, to enable the method to operate in inductive learning mode.

3.3.3 FEATHER

FEATHER [34] is a method that uses an r-scale random walk weighted characteristic function to describe the distribution of graph node features at multiple scales. Assuming the neighborhood of a node u at scale r consists of nodes that can be reached by a random walk in r steps from source node u, this characteristic function has probability weights defined by the transition probabilities of random walks in r steps from source node u. FEATHER is a probabilistic embedding method: by exploiting random walks, it learns multi-scale node features of the graph that are aggregated by mean pooling to obtain a numerical vector (embedding) representing the entire graph structure and its local neighborhood connectivity. Therefore, the resulting embedder can be applied to each graph sample separately.

3.4 Experimental pipeline

The experimental pipeline is summarized in the pseudo-code of Algorithm 1. The graph dataset is loaded (line 1) and split ten times into a training set with ninety percent of the samples and a test set used for evaluation (line 2). The dataset partitions

are non-overlapping, thus ensuring that all graphs in the dataset are used for testing exactly once. After dataset splitting, the training samples are attacked according to the chosen adversarial attack strategy and parameter (*budget* of the attack) (line 3), while the test samples are unaltered. The embedding method is initialized, and its parameters are set (line 4). The embedding model is built (trained) on the altered samples (line 5). The so-trained model is applied to produce embedding on both training (line 6) and testing (line 7) samples. Once the embedding vectors are obtained, an SVM classifier with a linear kernel is applied to fit the training vectors and predict the test vectors (line 8). Scores for all cross-validation folds are collected, and performances are computed.

Input: graph $dataset$, graph labels y
Output: embedding validation scores
1: $graphs, y \leftarrow Load(dataset)$
2: **for** $train\_graphs, y\_train, test\_graphs, y\_test \leftarrow Split(graphs, y)$ **do**
3: $train\_graphs \leftarrow Attack(train\_graphs)$ ▷ attack training graphs
4: $model \leftarrow GEmbedder()$ ▷ init and set parameters for embedder
5: $GEmbedder.fit(train\_graphs)$ ▷ build the model
6: $X \leftarrow model.get\_embedding(train\_graphs)$ ▷ embed the train set
7: $Xt \leftarrow model.infer(test\_graphs)$ ▷ infer embedding of test set
8: $scores \leftarrow SVM(X, y).predict(Xt)$ ▷ embedding validation
9: **end for**

Algorithm 1 The experimental pipeline

3.5 Performance evaluation

The experimental results are all reported in Tables 3, 4, 5, 6, 7, 8, 9, 10 and 11 in the Appendix in terms of accuracy, precision, F-measure, recall, and Matthews Correlation Coefficient (MCC) [37]. In Figs. 5, 6 and 7, we plotted the MCC scores obtained by the stratified 10-fold cross-validation on the embeddings produced by all methods when applied to each dataset and by varying the type of attack (random, betweenness, eigenvector, and PageRank centrality-based attack) and the budget of poisoning (percentage of edge removal).

In the MUTAG benchmark, all the methods show a performance degradation towards the null classification for attack budgets higher than 20%. This is due to the small size of the original graphs and the consequent scarcity of graph information survived to the edge removal attacks. The worst behavior is observed for FEATHER, while the best for iNetpro2vec, which partially succeeds in extracting distinctive information from the built vocabulary under moderate attacks.

In the case of PROTEINS, the performance of all the methods is low and similar, even with no attacks. iNetpro2vec is the only method showing less degradation of MCC scores when increasing the poisoning budget to the maximum. In our interpretation, this effect is partially due to the inherent robustness of the method. Indeed, if we look at the structure of a graph from the PROTEINS dataset (e.g., Fig. 2), we observe that edge removal attacks lead to the division of the protein graph into disjoint groups of atoms, leaving intra-group connectivity unaltered. In our understanding, iNetpro2vec relies more on intra-group than

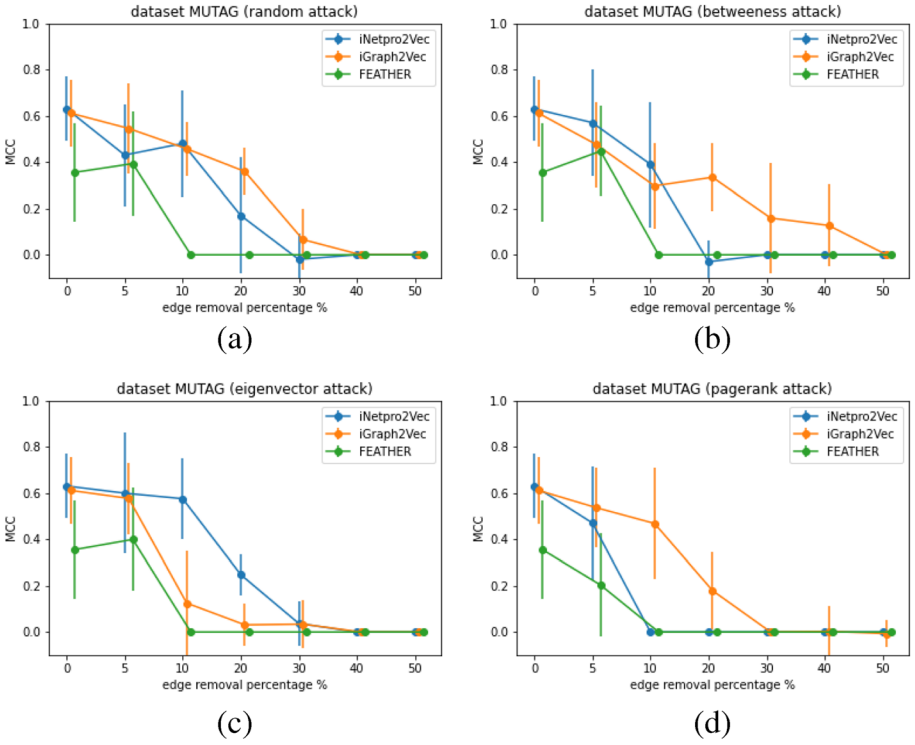

Fig. 5 Plots of performance measures (MCC) on the MUTAG dataset for baseline (random) attacks (**a**), betweenness centrality-based attacks (**b**), eigenvector centrality-based attacks (**c**), and PageRank-based centrality attacks (**d**)

inter-group connections to characterize the graph embedding. This is a plausible explanation for the almost flat trend of MCCs in the plots.

In the case of the more challenging Kidney dataset, iNetpro2vec always performs better than FEATHER. This method clearly suffers in robustness under increasing edge removal attacks, and its performance under all the attack strategies soon degrades towards the null classification. iNetpro2vec also outperforms iGraph2Vec in the unattacked case and with attack budgets less than %20. In the other poisoning percentages, iGraph2Vec and iNetpro2vec show similar robustness when the attack increases and across the different strategies. In particular, the two methods show good robustness within a range of 20% for random and betweenness strategies and within a larger range of budgets in the case of eigenvector and PageRank centrality-based edge removal.

Overall, iNetpro2vec appears more robust to edge removal attacks than the other methods. This is particularly evident in the PROTEIN benchmark, where the gap is larger as the attack budget increases. The same holds in the Kidney benchmark, although, for this domain of high-scale and weighted graphs, iGraph2Vec performs lightly worse with low-budget attacks but similarly with larger budgets.

It should be observed that some of the compared methods seem to improve, rather than decrease, their performance under small budgets (generally 5%) of edge removal attacks. This is the case of FEATHER on the MUTAG and PROTEINS datasets and of iNetpro2vec

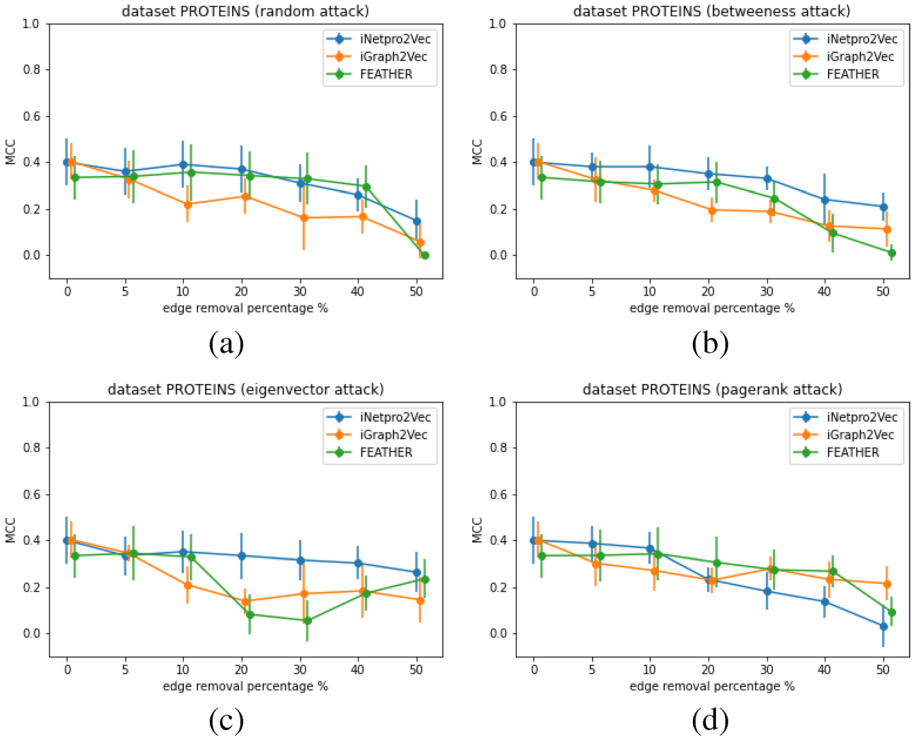

Fig. 6 Plots of performance measures (MCC) on the PROTEINS dataset for baseline (random) attacks (**a**), betweenness centrality-based attacks (**b**), eigenvector centrality-based attacks (**c**), and PageRank-based centrality attacks (**d**)

and Graph2Vec on the Kidney dataset. However, by applying the two sample T-test to quantify the difference between the population of MCC means in the case of unattacked graphs and of 5% budget of poisoning, and by examining the relative p-value, it comes out that the reported unexpected increase is not statistically significant and therefore cannot be considered a real improvement in performance.

As a general comment on the attack strategies, the eigenvector and PageRank centrality-based edge removal strategies have similar effects on the methods' performance. This was expected since, as already discussed in Section 3, the PageRank centrality is a variant of the eigenvector one.

To conclude the performance evaluation of the compared methods, for all the embedding methods we report the parameter settings in Table 12 and the execution times recorded during the experiments in Table 2. We measured the average execution times of the experimental pipeline when applied to each pair dataset/method on an iMac Retina 5K with a 4GHz Intel Core i7 quad-core and 32GB of RAM 1600 MHz DDR3. We observe that the FEATHER algorithm is much faster than the other two methods in the case of small graphs (MUTAG and PROTEINS datasets). However, it is the slowest when dealing with the much larger Kidney graphs, for which the two neural network-based methods require approximately 2/3 of the time.

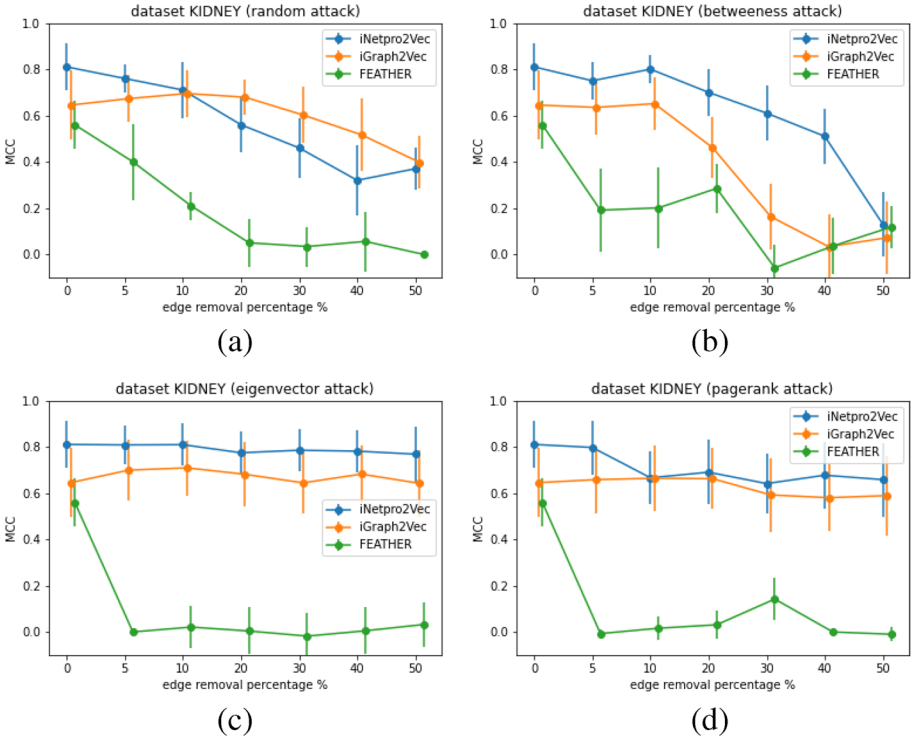

Fig. 7 Plots of performance measures (MCC) on the Kidney dataset for baseline (random) attacks (**a**), betweenness centrality-based attacks (**b**), eigenvector centrality-based attacks (**c**), and PageRank centrality-based attacks (**d**)

4 Conclusions and future work

As a general conclusion, from our experimental study, iNetpro2vec shows a very good robustness of its embedding models across all the considered benchmarks and when the model training set is poisoned even with targeted attacks involving more central connections of nodes. It behaves similarly to iGraph2Vec in the Kidney benchmark, consisting of large-scale weighted and highly connected graphs. In this domain, the FEATHER method has no success. This is further proof that iNetpro2vec provides efficient embeddings independently from the nature of data and for different tasks (graph classification, graph similarity matching, and so on).

Table 2 Average time estimates for each experiment execution

Method	iNetpro2vec	iGraph2Vec	FEATHER
MUTAG	40s	12s	1s
PROTEINS	3m	30s	17s
Kidney	25m	24m	38m

Future works are in the following directions: definitely, look at datasets with different characteristics in terms of density, structure, and position of nodes and edges as the attack strategies act above all on these aspects; furthermore, another important issue concerns the application of additional attack strategies in order to evaluate the behavior of graph embedding methods.

Appendix A: Performance measures of graph-embedding methods

In this appendix, we include tables reporting measures of 10-fold classification accuracy (acc), precision (prec), F-measure (f1), recall, and Matthews Correlation Coefficients (MCC) obtained in all the experiments. One table is reported for each experiment bunch, referring to the classification performance of one graph-embedding method (iNetpro2vec, iGraph2Vec, or FEATHER) when applied to one dataset (MUTAG, PROTEINS, or Kidney). In each table, we report the performance results when the dataset is unattacked (first row) and in the case of different percentages of edge removal (budget). The rows are grouped according to the criterion adopted for edge removal (random, betweenness, eigenvector, or pagerank) (Tables 3, 4, 5, 6, 7, 8, 9, 10 and 11).

Table 3 Performances of iNetpro2Vec on MUTAG dataset under the different attacks

Attack	Budget	acc	prec	f1	Recall	MCC
unattacked	0	0.82±0.07	0.83±0.08	0.80±0.08	0.80±0.09	0.62±0.15
Random	5	0.71±0.10	0.71±0.10	0.69±0.10	0.72±0.11	0.43±0.22
	10	0.74±0.12	0.74±0.12	0.72±0.12	0.74±0.12	0.48±0.23
	20	0.65±0.09	0.58±0.15	0.57±0.12	0.58±0.11	0.17±0.25
	30	0.62±0.09	0.37±0.08	0.42±0.06	0.49±0.06	−0.02±0.11
	40	0.66±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.66±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
Betweeness	5	0.81±0.09	0.80±0.12	0.78±0.11	0.78±0.12	0.57±0.23
	10	0.75±0.10	0.70±0.18	0.67±0.15	0.68±0.13	0.39±0.27
	20	0.33±0.02	0.17±0.01	0.25±0.01	0.49±0.02	−0.03±0.09
	30	0.66±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	40	0.66±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.66±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
Eigenvector	5	0.82±0.10	0.81±0.13	0.79±0.13	0.79±0.14	0.60±0.26
	10	0.80±0.07	0.78±0.08	0.78±0.08	0.79±0.09	0.57±0.17
	20	0.46±0.07	0.68±0.04	0.43±0.08	0.59±0.05	0.25±0.09
	30	0.39±0.10	0.36±0.22	0.30±0.05	0.51±0.03	0.04±0.10
	40	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
Pagerank	5	0.73±0.11	0.72±0.12	0.71±0.12	0.75±0.13	0.47±0.24
	10	0.46±0.16	0.23±0.08	0.31±0.07	0.50±0.00	0.00±0.00
	20	0.34±0.02	0.17±0.01	0.25±0.01	0.50±0.00	0.00±0.00
	30	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	40	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00

Table 4 Performances of iGraph2Vec on MUTAG dataset under the different attacks

Attack	Budget	acc	prec	f1	Recall	MCC
Unattacked	0	0.82±0.06	0.82±0.06	0.79±0.09	0.80±0.09	0.61±0.14
Random	5	0.80±0.08	0.80±0.10	0.76±0.11	0.76±0.11	0.55±0.20
	10	0.77±0.05	0.82±0.08	0.68±0.07	0.67±0.05	0.46±0.12
	20	0.73±0.03	0.83±0.08	0.58±0.06	0.60±0.04	0.36±0.10
	30	0.68±0.03	0.44±0.21	0.43±0.06	0.52±0.03	0.07±0.13
	40	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
Betweeness	5	0.76±0.08	0.74±0.09	0.73±0.09	0.74±0.10	0.47±0.18
	10	0.72±0.06	0.71±0.21	0.56±0.11	0.59±0.07	0.30±0.19
	20	0.72±0.06	0.71±0.11	0.64±0.07	0.64±0.06	0.33±0.15
	30	0.68±0.06	0.64±0.20	0.52±0.10	0.55±0.07	0.16±0.24
	40	0.50±0.14	0.51±0.19	0.46±0.15	0.56±0.09	0.13±0.18
	50	0.37±0.11	0.19±0.05	0.27±0.05	0.50±0.00	0.00±0.00
Eigenvector	5	0.81±0.07	0.81±0.10	0.77±0.08	0.77±0.08	0.58±0.16
	10	0.69±0.06	0.52±0.24	0.48±0.12	0.54±0.08	0.12±0.23
	20	0.67±0.02	0.38±0.15	0.41±0.04	0.51±0.02	0.03±0.09
	30	0.67±0.03	0.38±0.16	0.41±0.05	0.51±0.03	0.03±0.10
	40	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
Pagerank	5	0.79±0.07	0.77±0.08	0.76±0.10	0.77±0.09	0.54±0.17
	10	0.78±0.09	0.76±0.12	0.71±0.14	0.71±0.13	0.47±0.24
	20	0.68±0.04	0.65±0.20	0.52±0.07	0.55±0.04	0.18±0.17
	30	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	40	0.66±0.03	0.37±0.10	0.42±0.06	0.50±0.04	0.00±0.11
	50	0.66±0.02	0.36±0.07	0.41±0.03	0.50±0.01	−0.01±0.06

Table 5 Performances of FEATHER on MUTAG dataset under the different attacks

Attack	Budget	acc	prec	f1	Recall	MCC
Unattacked	0	0.73±0.08	0.69±0.12	0.67±0.11	0.66±0.10	0.35±0.21
Random	5	0.76±0.07	0.73±0.21	0.64±0.14	0.65±0.10	0.39±0.23
	10	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	20	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	30	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	40	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
Betweeness	5	0.77±0.06	0.78±0.16	0.68±0.12	0.68±0.09	0.45±0.20
	10	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	20	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	30	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	40	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
Eigenvector	5	0.76±0.07	0.73±0.17	0.66±0.12	0.66±0.10	0.40±0.22
	10	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	20	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	30	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	40	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
Pagerank	5	0.71±0.06	0.57±0.24	0.52±0.14	0.57±0.09	0.20±0.22
	10	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	20	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	30	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	40	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00
	50	0.67±0.02	0.33±0.01	0.40±0.01	0.50±0.00	0.00±0.00

Table 6 Performances of iNetpro2Vec on PROTEINS dataset under the different attacks

Attack	Budget	acc	prec	f1	recall	MCC
Unattacked	0	0.71±0.05	0.70±0.06	0.69±0.06	0.69±0.06	0.39±0.11
Random	5	0.70±0.05	0.69±0.05	0.68±0.05	0.68±0.05	0.36±0.10
	10	0.71±0.05	0.71±0.05	0.69±0.05	0.69±0.05	0.39±0.10
	20	0.70±0.05	0.70±0.06	0.67±0.05	0.67±0.05	0.37±0.10
	30	0.68±0.03	0.69±0.05	0.62±0.04	0.63±0.03	0.31±0.08
	40	0.66±0.02	0.71±0.05	0.55±0.04	0.58±0.03	0.26±0.07
	50	0.62±0.02	0.69±0.16	0.44±0.04	0.53±0.02	0.15±0.09
Betweeness	5	0.71±0.03	0.70±0.03	0.68±0.03	0.68±0.03	0.38±0.06
	10	0.71±0.04	0.70±0.04	0.68±0.04	0.68±0.04	0.38±0.09
	20	0.70±0.03	0.70±0.04	0.66±0.04	0.66±0.04	0.35±0.07
	30	0.69±0.02	0.70±0.03	0.63±0.03	0.63±0.02	0.33±0.05
	40	0.64±0.04	0.70±0.09	0.53±0.04	0.57±0.03	0.24±0.11
	50	0.63±0.02	0.73±0.07	0.48±0.03	0.55±0.02	0.21±0.06
Eigenvector	5	0.68±0.04	0.67±0.04	0.67±0.04	0.67±0.04	0.33±0.08
	10	0.69±0.04	0.69±0.05	0.67±0.04	0.67±0.04	0.35±0.09
	20	0.69±0.04	0.69±0.06	0.64±0.05	0.64±0.04	0.33±0.10
	30	0.68±0.04	0.69±0.05	0.62±0.04	0.63±0.04	0.32±0.09
	40	0.68±0.03	0.70±0.05	0.60±0.03	0.62±0.03	0.30±0.07
	50	0.66±0.03	0.69±0.06	0.57±0.04	0.59±0.03	0.26±0.09
Pagerank	5	0.70±0.04	0.69±0.04	0.69±0.04	0.69±0.04	0.39±0.08
	10	0.70±0.03	0.69±0.04	0.68±0.03	0.68±0.03	0.37±0.07
	20	0.65±0.02	0.65±0.04	0.57±0.02	0.59±0.02	0.23±0.05
	30	0.63±0.03	0.65±0.07	0.51±0.04	0.56±0.03	0.18±0.08
	40	0.62±0.02	0.66±0.08	0.46±0.04	0.53±0.02	0.14±0.07
	50	0.60±0.01	0.54±0.21	0.39±0.02	0.51±0.01	0.03±0.09

Table 7 Performances of iGraph2Vec on PROTEINS dataset under the different attacks

Attack	Budget	acc	prec	f1	Recall	MCC
Unattacked	0	0.72±0.04	0.71±0.04	0.69±0.04	0.69±0.04	0.40±0.08
Random	5	0.69±0.03	0.70±0.05	0.62±0.04	0.63±0.04	0.33±0.08
	10	0.64±0.02	0.71±0.08	0.50±0.03	0.56±0.02	0.22±0.08
	20	0.64±0.02	0.76±0.07	0.50±0.04	0.56±0.02	0.25±0.07
	30	0.62±0.03	0.66±0.23	0.44±0.06	0.54±0.03	0.16±0.14
	40	0.62±0.01	0.72±0.15	0.44±0.03	0.53±0.02	0.17±0.07
	50	0.60±0.01	0.57±0.24	0.39±0.02	0.51±0.01	0.06±0.07
Betweeness	5	0.68±0.04	0.71±0.06	0.62±0.05	0.63±0.04	0.33±0.10
	10	0.66±0.02	0.72±0.04	0.55±0.04	0.59±0.03	0.28±0.05
	20	0.63±0.01	0.74±0.08	0.47±0.02	0.54±0.01	0.20±0.05
	30	0.62±0.01	0.78±0.06	0.44±0.03	0.53±0.01	0.19±0.05
	40	0.61±0.01	0.69±0.20	0.41±0.03	0.52±0.01	0.13±0.07
	50	0.61±0.01	0.66±0.20	0.41±0.03	0.52±0.01	0.11±0.07
Eigenvector	5	0.69±0.01	0.71±0.02	0.63±0.02	0.64±0.02	0.34±0.04
	10	0.64±0.02	0.69±0.08	0.51±0.04	0.56±0.03	0.21±0.08
	20	0.61±0.01	0.67±0.08	0.46±0.03	0.53±0.01	0.14±0.06
	30	0.63±0.03	0.66±0.14	0.48±0.06	0.55±0.04	0.17±0.13
	40	0.63±0.02	0.69±0.14	0.47±0.05	0.55±0.03	0.18±0.12
	50	0.62±0.02	0.65±0.15	0.45±0.05	0.53±0.03	0.14±0.10
Pagerank	5	0.65±0.04	0.65±0.05	0.65±0.05	0.65±0.05	0.30±0.10
	10	0.66±0.04	0.64±0.04	0.63±0.05	0.63±0.04	0.27±0.09
	20	0.64±0.02	0.71±0.05	0.51±0.03	0.56±0.02	0.23±0.06
	30	0.66±0.01	0.73±0.05	0.55±0.03	0.59±0.02	0.28±0.05
	40	0.64±0.02	0.69±0.07	0.54±0.04	0.57±0.03	0.23±0.08
	50	0.63±0.02	0.73±0.07	0.48±0.03	0.55±0.02	0.21±0.07

Table 8 Performances of FEATHER on PROTEINS dataset under the different attacks

Attack	Budget	acc	prec	f1	Recall	MCC
Unattacked	0	0.69±0.04	0.71±0.06	0.62±0.05	0.63±0.04	0.33±0.09
Random	5	0.69±0.04	0.72±0.07	0.62±0.06	0.63±0.05	0.34±0.12
	10	0.70±0.05	0.71±0.08	0.65±0.05	0.65±0.05	0.36±0.12
	20	0.69±0.04	0.70±0.07	0.65±0.04	0.65±0.04	0.34±0.10
	30	0.69±0.04	0.70±0.07	0.63±0.05	0.64±0.04	0.33±0.11
	40	0.67±0.03	0.72±0.07	0.58±0.04	0.60±0.03	0.30±0.09
	50	0.60±0.00	0.30±0.00	0.37±0.00	0.50±0.00	0.00±0.00
Betweeness	5	0.68±0.03	0.72±0.06	0.59±0.04	0.61±0.04	0.32±0.09
	10	0.67±0.03	0.71±0.06	0.59±0.04	0.61±0.03	0.31±0.09
	20	0.67±0.03	0.73±0.06	0.58±0.05	0.61±0.03	0.31±0.09
	30	0.64±0.02	0.74±0.07	0.51±0.04	0.56±0.02	0.24±0.07
	40	0.61±0.01	0.64±0.23	0.40±0.02	0.51±0.01	0.10±0.08
	50	0.60±0.00	0.35±0.15	0.38±0.01	0.50±0.00	0.01±0.04
Eigenvector	5	0.69±0.04	0.71±0.07	0.63±0.06	0.64±0.05	0.34±0.12
	10	0.68±0.04	0.73±0.07	0.60±0.04	0.62±0.04	0.33±0.10
	20	0.56±0.04	0.54±0.04	0.54±0.04	0.54±0.04	0.08±0.09
	30	0.56±0.04	0.53±0.05	0.52±0.04	0.53±0.04	0.06±0.09
	40	0.60±0.03	0.59±0.04	0.58±0.04	0.59±0.04	0.17±0.07
	50	0.64±0.03	0.63±0.04	0.60±0.05	0.61±0.04	0.23±0.08
Pagerank	5	0.69±0.04	0.70±0.06	0.63±0.06	0.64±0.05	0.34±0.11
	10	0.69±0.05	0.69±0.07	0.65±0.05	0.65±0.04	0.34±0.12
	20	0.67±0.04	0.69±0.08	0.62±0.05	0.62±0.04	0.31±0.11
	30	0.66±0.03	0.69±0.07	0.58±0.03	0.60±0.03	0.27±0.09
	40	0.66±0.02	0.72±0.06	0.55±0.03	0.58±0.02	0.27±0.07
	50	0.61±0.01	0.62±0.22	0.40±0.03	0.51±0.01	0.09±0.06

Table 9 Performances of iNetpro2Vec on Kidney dataset under the different attacks

Attack	Budget	acc	prec	f1	Recall	MCC
Unattacked	0	0.87±0.05	0.90±0.04	0.88±0.05	0.87±0.06	0.79±0.09
Random	5	0.85±0.04	0.88±0.03	0.86±0.04	0.87±0.05	0.76±0.06
	10	0.82±0.08	0.85±0.06	0.84±0.07	0.85±0.07	0.71±0.12
	20	0.67±0.12	0.81±0.05	0.71±0.11	0.74±0.09	0.56±0.12
	30	0.60±0.12	0.68±0.09	0.61±0.10	0.66±0.08	0.46±0.13
	40	0.49±0.12	0.59±0.18	0.47±0.14	0.53±0.09	0.32±0.15
	50	0.61±0.08	0.59±0.15	0.50±0.09	0.53±0.07	0.37±0.09
Betweeness	5	0.85±0.04	0.87±0.03	0.86±0.05	0.86±0.06	0.75±0.08
	10	0.87±0.04	0.90±0.03	0.88±0.04	0.89±0.05	0.80±0.06
	20	0.81±0.06	0.87±0.04	0.81±0.07	0.79±0.08	0.70±0.10
	30	0.72±0.09	0.81±0.05	0.75±0.08	0.78±0.08	0.61±0.12
	40	0.67±0.07	0.74±0.08	0.70±0.07	0.72±0.07	0.51±0.12
	50	0.54±0.05	0.37±0.11	0.33±0.06	0.38±0.04	0.13±0.14
Eigenvector	5	0.88±0.05	0.90±0.04	0.89±0.05	0.89±0.06	0.81±0.08
	10	0.88±0.06	0.91±0.05	0.89±0.05	0.89±0.06	0.81±0.09
	20	0.86±0.06	0.89±0.05	0.87±0.06	0.87±0.06	0.77±0.09
	30	0.86±0.06	0.89±0.05	0.87±0.05	0.88±0.06	0.79±0.09
	40	0.86±0.06	0.90±0.04	0.86±0.07	0.86±0.07	0.78±0.09
	50	0.86±0.07	0.89±0.06	0.86±0.07	0.86±0.08	0.77±0.12
Pagerank	5	0.87±0.08	0.90±0.06	0.88±0.07	0.89±0.07	0.80±0.12
	10	0.79±0.08	0.81±0.08	0.80±0.07	0.82±0.06	0.67±0.12
	20	0.80±0.09	0.83±0.07	0.82±0.09	0.83±0.09	0.69±0.14
	30	0.78±0.08	0.82±0.05	0.80±0.07	0.80±0.08	0.64±0.13
	40	0.80±0.09	0.84±0.07	0.81±0.08	0.81±0.09	0.68±0.14
	50	0.79±0.10	0.84±0.07	0.80±0.10	0.79±0.10	0.66±0.16

Table 10 Performances of iGraph2Vec on Kidney dataset under the different attacks

Attack	Budget	acc	prec	f1	recall	MCC
Unattacked	0	0.77±0.10	0.79±0.10	0.79±0.10	0.81±0.08	0.65±0.15
Random	5	0.79±0.07	0.81±0.06	0.81±0.06	0.83±0.06	0.67±0.10
	10	0.81±0.06	0.83±0.06	0.82±0.07	0.84±0.07	0.69±0.10
	20	0.80±0.05	0.82±0.05	0.81±0.06	0.81±0.06	0.68±0.08
	30	0.74±0.10	0.80±0.07	0.74±0.10	0.75±0.08	0.60±0.12
	40	0.67±0.12	0.78±0.08	0.66±0.13	0.67±0.11	0.52±0.16
	50	0.62±0.09	0.71±0.10	0.57±0.07	0.57±0.06	0.40±0.12
Betweeness	5	0.77±0.08	0.80±0.08	0.79±0.07	0.80±0.06	0.64±0.12
	10	0.78±0.07	0.81±0.08	0.77±0.08	0.77±0.08	0.65±0.11
	20	0.69±0.07	0.78±0.07	0.65±0.08	0.62±0.07	0.46±0.13
	30	0.53±0.07	0.37±0.10	0.37±0.06	0.41±0.07	0.16±0.14
	40	0.45±0.09	0.29±0.05	0.31±0.06	0.34±0.06	0.03±0.14
	50	0.49±0.08	0.30±0.06	0.33±0.07	0.37±0.07	0.07±0.16
Eigenvector	5	0.81±0.08	0.81±0.09	0.82±0.08	0.84±0.07	0.70±0.13
	10	0.82±0.07	0.83±0.08	0.83±0.07	0.84±0.07	0.71±0.12
	20	0.80±0.09	0.83±0.09	0.82±0.09	0.82±0.08	0.68±0.14
	30	0.78±0.08	0.82±0.07	0.80±0.08	0.79±0.08	0.64±0.13
	40	0.81±0.07	0.85±0.06	0.81±0.08	0.79±0.09	0.68±0.12
	50	0.79±0.08	0.84±0.09	0.77±0.09	0.74±0.10	0.64±0.14
Pagerank	5	0.79±0.09	0.80±0.10	0.80±0.09	0.81±0.08	0.66±0.14
	10	0.79±0.09	0.79±0.10	0.80±0.09	0.82±0.08	0.66±0.14
	20	0.80±0.08	0.82±0.06	0.81±0.08	0.81±0.09	0.66±0.13
	30	0.76±0.09	0.79±0.08	0.76±0.09	0.75±0.09	0.59±0.16
	40	0.75±0.07	0.80±0.06	0.74±0.09	0.73±0.10	0.58±0.14
	50	0.75±0.09	0.81±0.07	0.74±0.11	0.73±0.12	0.59±0.17

Table 11 Performances of FEATHER on Kidney dataset under the different attacks

Attack	Budget	acc	prec	f1	Recall	MCC
Unattacked	0	0.74±0.06	0.76±0.10	0.72±0.08	0.72±0.08	0.56±0.10
Random	5	0.61±0.12	0.67±0.14	0.59±0.12	0.62±0.11	0.40±0.16
	10	0.47±0.09	0.46±0.13	0.38±0.06	0.48±0.05	0.21±0.06
	20	0.19±0.03	0.23±0.23	0.14±0.06	0.36±0.04	0.05±0.10
	30	0.18±0.02	0.16±0.15	0.12±0.04	0.35±0.02	0.03±0.09
	40	0.27±0.11	0.24±0.17	0.21±0.11	0.40±0.07	0.06±0.13
	50	0.17±0.00	0.06±0.00	0.10±0.00	0.33±0.00	0.00±0.00
Betweeness	5	0.56±0.09	0.43±0.16	0.41±0.09	0.43±0.08	0.19±0.18
	10	0.46±0.15	0.41±0.17	0.32±0.14	0.42±0.09	0.20±0.17
	20	0.55±0.11	0.51±0.12	0.45±0.10	0.49±0.07	0.28±0.11
	30	0.31±0.07	0.15±0.07	0.17±0.03	0.31±0.03	−0.06±0.10
	40	0.32±0.04	0.22±0.15	0.19±0.04	0.34±0.03	0.04±0.12
	50	0.36±0.06	0.29±0.13	0.27±0.07	0.39±0.04	0.12±0.09
Eigenvector	5	0.53±0.01	0.18±0.00	0.23±0.00	0.33±0.00	0.00±0.00
	10	0.26±0.16	0.13±0.14	0.16±0.12	0.35±0.04	0.02±0.09
	20	0.43±0.14	0.18±0.08	0.22±0.08	0.35±0.07	0.00±0.10
	30	0.27±0.14	0.17±0.10	0.16±0.06	0.31±0.05	−0.02±0.10
	40	0.22±0.05	0.19±0.12	0.16±0.05	0.32±0.07	0.01±0.10
	50	0.37±0.10	0.19±0.12	0.20±0.06	0.34±0.04	0.03±0.10
Pagerank	5	0.46±0.14	0.16±0.04	0.21±0.05	0.34±0.01	−0.01±0.02
	10	0.29±0.02	0.12±0.06	0.16±0.02	0.34±0.02	0.02±0.05
	20	0.30±0.01	0.12±0.04	0.18±0.05	0.35±0.06	0.03±0.06
	30	0.29±0.07	0.26±0.08	0.25±0.05	0.42±0.06	0.14±0.09
	40	0.19±0.05	0.07±0.02	0.11±0.02	0.33±0.00	0.00±0.00
	50	0.23±0.13	0.09±0.07	0.12±0.06	0.33±0.01	−0.01±0.03

Appedix B: Parameter settings of graph-embedding methods

Table 12 reports the parameter settings for the software implementations of Netpro2vec,[1] Graph2Vec,[2] and FEATHER[2] adopted in the experiments. These parameters have been experimentally chosen to optimize MCC performance.

Table 12 Parameter settings for the embedding methods used in the experiments for each dataset. In the case of FEATHER, the embedding size is not an input parameter, and it is set to 500

	iNetpro2vec	iGraph2Vec	FEATHER
MUTAG	dimensions=512, epochs=400, min-count=2, prob_type=[ndd], cut_off=[0.1], extractor=[1], agg_by=[1], learning-rate=0.025, down-sampling=0.0001	dimensions=256, epochs=200, min-count=3, wl-iterations=3, learning-rate=0.025, down-sampling=0.0001	order=5, eval_points=25, theta_max=2.5, pooling=mean
PROTEINS	dimensions=256, epochs=200, min-count=2, prob_type=[ndd,tm1], cut_off=[0,0], extractor=[2,2], agg_by=[1,0], learning-rate=0.025, down-sampling=0.0001	dimensions=256, epochs=25, min-count=3, wl-iterations=3, learning-rate=0.025, down-sampling=0.0001	order=5, eval_points=25, theta_max=2.5, pooling=mean
Kidney	dimensions=256, epochs=200, min-count=2, prob_type=[ndd,tm1], cut_off=[0,0], extractor=[1,1], agg_by=[1,0], learning-rate=0.025, down-sampling=0.0001	dimensions=256, epochs=200, min-count=3, wl-iterations=3, learning-rate=0.025, down-sampling=0.0001	order=5, eval_points=25, theta_max=2.5, pooling=mean

[1] available at https://github.com/cds-group/Netpro2vec

[2] available at https://karateclub.readthedocs.io

Acknowledgements Mario Manzo thanks Prof. Alfredo Petrosino for the guidance and supervision during the years of working together.

Funding This work has been partially funded by the 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". 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).

Data availability Data and algorithms used in the current work are all available as open source.

Code availability The software used in the current experimental study is publicly available for reproducibility of results.

Declarations

Ethics approval and consent to participate Datasets used in the current work are all from secondary sources, where primary ethics approval had been obtained for data acquisition.

Consent for publication Not applicable.

Conflicts of interest The authors declare that they have no competing interests.

References

1. Vlietstra, W.J., Vos, R., Sijbers, A.M., van Mulligen, E.M., Kors, J.A.: Using predicate and provenance information from a knowledge graph for drug efficacy screening. J. Biomed. Semantics **9**(1), 1–10 (2018)
2. Manipur, I., Granata, I., Maddalena, L., Guarracino, M.R.: Clustering analysis of tumor metabolic networks. BMC Bioinformatics **21**(10), 349 (2020). https://doi.org/10.1186/s12859-020-03564-9
3. Thorne, T., Stumpf, M.P.: Graph spectral analysis of protein interaction network evolution. J. R. Soc. Interface. **9**(75), 2653–2666 (2012)
4. Ding, S., Chen, C., Zhang, Q., Xin, B., Pardalos, P.M.: Metaheuristics for Resource Deployment Under Uncertainty in Complex Systems. CRC Press, (2021)
5. Chen, C., Wu, X., Chen, J., et al.: Dynamic grouping of heterogeneous agents for exploration and strike missions. Front. Inform. Technol. Electron. Eng. **23**, 86–100 (2022). https://doi.org/10.1631/FITEE.2000352
6. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29 (2015)
7. Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(09), 1616–1637 (2018). https://doi.org/10.1109/TKDE.2018.2807452
8. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowl.-Based Syst. **151**, 78–94 (2018). https://doi.org/10.1016/j.knosys.2018.03.022
9. Maddalena, L., Manipur, I., Manzo, M., Guarracino, M.R.: On whole-graph embedding techniques. In: Mondaini, R.P. (ed.) Trends in Biomathematics: Chaos and Control in Epidemics, Ecosystems, and Cells: Selected Works from the 20th BIOMAT Consortium Lectures, Rio de Janeiro, Brazil, 2020, pp. 115–131. Springer, (2021). https://doi.org/10.1007/978-3-030-73241-7_8

10. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I.P., Tygar, J.D.: Adversarial machine learning. In: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence. AISec '11, pp. 43–58. Association for Computing Machinery, (2011). https://doi.org/10.1145/2046684.2046692
11. 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
12. Qiu, S., Liu, Q., Zhou, S., Wu, C.: Review of artificial intelligence adversarial attack and defense technologies. Appl. Sci.**9**(5) (2019). https://doi.org/10.3390/app9050909
13. Gao, J., Lanchantin, J., Soffa, M.L., Qi, Y.: 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
14. 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.) Research in Attacks, Intrusions, and Defenses, pp. 490–510. Springer, (2018)
15. Jin, W., Li, Y., Xu, H., Wang, Y., Ji, S., Aggarwal, C., Tang, J.: Adversarial attacks and defenses on graphs. SIGKDD Explor. Newsl. **22**(2), 19–34 (2021). https://doi.org/10.1145/3447556.3447566
16. Chen, L., Li, J., Peng, J., Xie, T., Cao, Z., Xu, K., He, X., Zheng, Z.: A survey of adversarial learning on graphs. (2020). arXiv:2003.05730. Accessed 29 Sept 2022
17. Sun, L., Wang, J., Yu, P.S., Li, B.: Adversarial attack and defense on graph data: A survey. (2020). arXiv:1812.10528. Accessed 29 Sept 2022
18. Chen, L., Wang, S., Yan, X.: Centroid-based clustering for graph datasets. In: Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), pp. 2144–2147 (2012)
19. Xi, Z., Pang, R., Ji, S., Wang, T.: Graph backdoor. In: 30th USENIX Security Symposium (USENIX Security 21) (2021)
20. Zhang, Z., Jia, J., Wang, B., Gong, N.Z.: Backdoor attacks to graph neural networks. In: Proceedings of the 26th ACM Symposium on Access Control Models and Technologies, pp. 15–26 (2021)
21. Manzo, M., Giordano, M., Maddalena, L., Guarracino, M.R.: Performance evaluation of adversarial attacks on whole-graph embedding models. In: Simos, D.E., Pardalos, P.M., Kotsireas, I.S.K. (eds.) Learning and Intelligent Optimization. LNCS. Springer, (2021)
22. Maddalena, L., Giordano, M., Manzo, M., Guarracino, M.R.: Whole-graph embedding and adversarial attacks for life sciences. In: Mondaini, R.P. (ed.) Trends in Biomathematics: Chaos and Control in Epidemics, Ecosystems, and Cells: Selected Works from the 21st BIOMAT Consortium Lectures, 2021. Springer, (2022)
23. Debnath, A., Lopez de Compadre, R., Debnath, G., Shusterman, A., Hansch, C.: 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
24. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Fifth IEEE International Conference on Data Mining (ICDM'05), p. 8 (2005). https://doi.org/10.1109/ICDM.2005.132
25. Granata, I., Guarracino, M.R., Kalyagin, V.A., Maddalena, L., Manipur, I., Pardalos, P.M.: Supervised classification of metabolic networks. In: 2018 IEEE Int. Conf. on Bioinformatics and Biomedicine (BIBM), pp. 2688–2693. IEEE (2018)
26. Granata, I., Guarracino, M.R., Kalyagin, V.A., Maddalena, L., Manipur, I., Pardalos, P.M.: Model simplification for supervised classification of metabolic networks. Ann. Math. Artif. Intell. **88**(1), 91–104 (2020)
27. Manipur, I., Granata, I., Maddalena, L., Guarracino, M.R.: Clustering analysis of tumor metabolic networks. BMC Bioinformatics (2020). https://doi.org/10.1186/s12859-020-03564-9
28. Uhlén, M., Fagerberg, L., Hallström, B.M., Lindskog, C., Oksvold, P., Mardinoglu, A., Sivertsson, Å, Kampf, C., Sjöstedt, E., Asplund, A., et al.: Tissue-based map of the human proteome. Science **347**(6220) (2015)
29. Brandes, U.: On variants of shortest-path betweenness centrality and their generic computation. Social Networks **30**(2), 136–145 (2008). https://doi.org/10.1016/j.socnet.2007.11.001
30. Bonacich, P.: Power and centrality: A family of measures. Am. J. Sociol. **92**(5), 1170–1182 (1987). Accessed 2022 June 01
31. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120. (1999). http://ilpubs.stanford.edu:8090/422/
32. Manipur, I., Manzo, M., Granata, I., Giordano, M., Maddalena, L., Guarracino, M.: Netpro2vec: a graph embedding framework for biomedical applications. IEEE/ACM Trans. Comput. Biol. Bioinform. 1–1 (2021). https://doi.org/10.1109/TCBB.2021.3078089

33. Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S.: graph2vec: Learning distributed representations of graphs. (2017). arXiv:1707.05005
34. 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. CIKM '20, pp. 1325–1334. Association for Computing Machinery, (2020). https://doi.org/10.1145/3340531.3411866
35. Mikolov, T., Le, Q.V., Sutskever, I.: Exploiting similarities among languages for machine translation. (2013). arXiv:1309.4168. Accessed 29 Sept 2022
36. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 1188–1196. PMLR, (2014). https://proceedings.mlr.press/v32/le14.html. Accessed 29 Sept 2022
37. Matthews, B.W.: Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA) - Protein Structure **405**(2), 442–451 (1975). https://doi.org/10.1016/0005-2795(75)90109-9