Applied Soft Computing 136 (2023) 110104

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

On augmenting topological graph representations for attributed graphs

Anwar Said [a,c,*], Mudassir Shabbir [a,c], Saeed-Ul Hassan [a], Zohair Raza Hassan [a,b], Ammar Ahmed [a], Xenofon Koutsoukos [c]

[a] Department of Computer Sciences, Information Technology University, Lahore, Pakistan
[b] Rochester Institute of Technology, Rochester, NY, USA
[c] Vanderbilt University, Nashville, TN, USA

ARTICLE INFO

Article history:
Received 17 September 2021
Received in revised form 2 February 2023
Accepted 7 February 2023
Available online 10 February 2023

Keywords:
Attributed graphs
Graph augmentation
Graph classification
Graph embedding
Toxicity prediction

ABSTRACT

Graph representations based on embedding methods allow for easier analysis of the network structure and can be used for a variety of tasks, such as link prediction and node classification. These methods have been shown to be effective in a variety of settings and have become an important tool in the field of graph learning. These methods are easy to implement, and their predictions yield interpretable results. However, most graph embedding methods rely solely on graph structural information and do not consider node/edge attributes, limiting their applicability. In this paper, we propose graph-theoretic designs to incorporate node and edge attributes within the topology, enabling graph-embedding methods to seamlessly work on attributed graphs. To find ideal representation for a given attributed graph, we propose augmenting special subgraph structures within original network. We discuss the potential challenges of the proposed approach and prove some of its theoretical limitations. We test the efficacy of our approach by comparing state-of-the-art graph classification models on 15 standard bioinformatics datasets. We observe an encouraging improvement of up to 5% in classification accuracy on the augmented graphs compared to the results on the original graphs.

Code metadata

Permanent link to reproducible Capsule: https://doi.org/10.24433/CO.5927044.v1.

1. Introduction

Graph machine learning methods have proven to be effective in addressing a wide range of problems in bioinformatics, social network analysis, and network security. Their ability to model complex relationships and capture the underlying structure of biological/social systems makes them a powerful tool for solving these challenges. A prominent example is the recent breakthrough on the "protein folding problem" using graph machine learning. The protein folding problem, also known as the protein structure prediction problem, is a longstanding challenge in the field of bioinformatics. It involves predicting the three-dimensional structure of a protein from its amino acid sequence, which is crucial for understanding the protein's function and potential therapeutic applications. Graph machine learning methods have been used to tackle this problem and have achieved significant progress. These methods represent the protein as a graph, with the amino acids as nodes and chemical bonds as edges, and use graph neural networks to learn the protein's structure from this representation. This approach has shown promising results and has the potential to provide new insights into the protein folding process [1]. Significant progress has been made on other problems, including toxicity prediction, drug discovery and development, and drug similarity integration [2–4].

Along with information about the existence of pairwise interactions, a graph representation may also encode meta-information available at individual nodes and edges. For example, in a dataset of molecules, the attributes of a node may include information about the atomic number, aromaticity ionization, and metallicity of the corresponding atom. Similarly, the edges may contain information about bond types. Attributed graphs appropriately represent such datasets while preserving the associated meta-information. In most cases, these attributes can be encoded in real values. Consequently, an attributed graph can be defined

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author at: Vanderbilt University, Nashville, TN, USA.
E-mail addresses: anwar.said@itu.edu.pk (A. Said), mudassir.shabbir@vanderbilt.edu (M. Shabbir), saeed-ul-hassan@itu.edu.pk (S.-U. Hassan), zohair.raza@itu.edu.pk (Z.R. Hassan), ammar.ahmed@itu.edu.pk (A. Ahmed), xenofon.koutsoukos@vanderbilt.edu (X. Koutsoukos).

https://doi.org/10.1016/j.asoc.2023.110104
1568-4946/

as a collection, $G = (V, E, f_v, f_e)$, where $f_v : V \rightarrow \mathbb{R}^{d_v}$ (resp., $f_e : E \rightarrow \mathbb{R}^{d_e}$) assigns a d_v-dimensional (resp., d_e-dimensional) attribute vector to each node in V (resp., each edge in E) [5,6]. A graph where the dimensions of these feature vectors are zero, i.e., $d_v = d_e = 0$, is referred to as an unattributed graph.

We observe that many state-of-the-art graph embedding approaches do not use node attributes and rely only on the topological structure of the graphs for machine learning tasks. These methods are otherwise well-grounded in graph theory and structural pattern recognition. They garner special attention from the research community because of their many merits, like reproducibility, low variance, explainability, etc. [7–9]. In settings where attribute information is available, these embedding methods have been applied by ignoring the attribute information [10–12]. This is not an ideal scenario as it results in a significant loss of information potentially affecting the performance of these methods for the aforementioned applications. In contrast, Graph Neural Network-based (GNNs) approaches use graphs' structural as well as attribute information and thus have shown more promising results on various graph mining tasks where attribute information is available albeit with a high variance and less explainability [13].

In this work, our goal is to design techniques that allow graph embedding methods to utilize a graph's structural and attribute information. Any such approach should treat these representation methods as black boxes and refrain from tampering with their algorithms. Therefore, the problem reduces to augmenting a graph's topology before passing it as an input to these methods. We propose a framework to encode the node and edge attribute information within the input graph topology. The encoding is performed by appending suitable families of subgraphs at "appropriate locations" while ensuring a one-to-one correspondence between the attributed input graph and the augmented unattributed graph. To illustrate this subgraph augmentation, consider the example in Fig. 1. In Fig. 1(a), a molecule sampled from MUTAG (a nitro-aromatic compounds dataset) is shown. The molecule consists of ten aromatic and three nonmetal atoms. Among these, two are ions. In Fig. 1(b), its corresponding simple, unattributed graph is shown, which is input to graph representation methods. Note that the transformation from (a) to (b) loses all the attributes. In Fig. 1(c), a new augmented unattributed graph is shown where different sizes of cliques (a subset of vertices that has all pairwise edges) are attached at suitable locations to encode various node/edge attributes. The blue nodes are the original nodes, while the rest of colors represent various cliques used for preserving specific attributes. Unlike the traditional embedding approaches, which use a simplified version of the original graph (Fig. 1(b)), we provide the graph with attribute information that is augmented in the graph topology (Fig. 1(c)). Since graph embedding methods mostly rely on neighborhood information, we argue in this paper that this is a suitable framework to preserve attribute information and leads to improved performance of embedding methods. The contributions of this paper are as follows:

- textcolorblueWe propose an attributed graph augmentation scheme to translate graph dataset with node/edge attribute information to a graph dataset with no attributes. The proposed scheme preserves not only the topological structure of the original but also the values of the attributes of each node/edge.
- We formally define the problem of finding an ideal unattributed graph representation of an attributed graph denoted as k-HFD, and analyze the theoretical complexity of this problem.

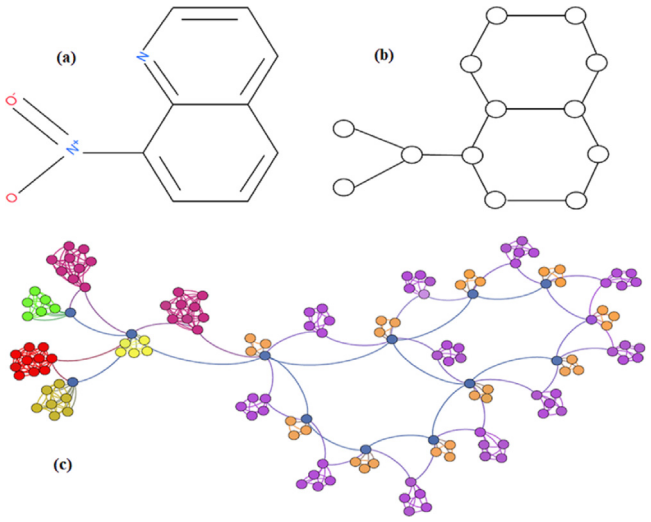


Fig. 1. Illustration of subgraph augmentation where a molecule with its corresponding simple and augmented graph is shown. The colors are provided for visualization purposes only.

- We perform extensive experiments in the graph classification setting on 15 standard bioinformatics datasets. We compare the results of 6 state-of-the-art graph embedding methods on the original graphs and the augmented graphs. Through these experiments, we demonstrate an improvement of up to 5% in the classification accuracy using the augmented graphs.

The rest of the paper is organized as follows: In Section 2, we briefly describe the existing works in this direction. In Section 3, we present the proposed procedure for attributed graph augmentation and also provide the analysis of the problem in terms of feasibility and lower/upper bounds. Section 4 provides a case study highlighting the applications of the proposed method in toxicity prediction. It also presents numerical results to evaluate the efficacy of the proposed method. Finally, Section 6 concludes the paper.

2. Related work

This section provides a brief overview of graph machine learning methods under three main categories: graph kernels, graph descriptors/embeddings, and graph neural networks. Furthermore, we present a summary of the applications of these methods in the field of bioinformatics.

2.1. Graph kernels

Graph representations were first explored in the context of graph kernels, many of which study the R-convolution framework [14]. Graph kernels usually decompose the entire graph into different subgraphs and then define kernels based on the similarity among those subgraphs [15]. For example, the family of graphlet kernels [16] considers fixed-size graphlets while shortest-path kernels are derived by comparing the paths among graphs [15]. Similarly, random walk kernels count the number of common random walks for graph comparison [17]. Alas, these kernels usually involve subgraph mining, which is a computationally expensive task, limiting their application to small-sized graphs. To overcome the computational overhead, faster methods such as the Weisfeiler–Lehman kernel [8], Neighborhood Hash Kernel [18,19] and Edge Histogram kernel [20] were introduced.

These methods generally compute a kernel matrix based on their defined procedures, which can then be used for the downstream machine learning task.

2.2. Graph descriptors

The key idea behind graph descriptors is to extract expressive graph representations that fulfill the desired properties of the embedding method. These methods usually use the spectral and statistical properties of the graph to encode both local and global-level information. For example, NetSIMILE [21] uses graph's statistical properties such as average degree, clustering coefficient, and the standard deviation of two-hop neighbors to compute fixed-length graph representation. Similarly, FGSD [10] and NetLSD [11] use spectral properties such as harmonic distance and transformed Laplacian traces for computing the graph representations. More recently, the authors in [22] consider the distribution of node features, while [23] consider the distribution of smooth graph signals for graph comparison using Wasserstein distance. Yu et al. [24] introduce a graph comparison framework based on optimal transport theory and discrete graph matching in the continuous domain. The authors in [9] use the Kirchhoff index as a measure of edge centrality for computing pair-wise distances among nodes to extract graph representations. A similar pair-wise distances based approach was presented in [12] to enhance the scalability of graph descriptors. The proposed approach extracts graph representations in both centralized and distributed environments. More recently, graph embedding methods have also been applied for industrial decisions in sewage treatment [25,26]. These methods capture global-level properties of the graphs with node pairwise similarities to extract the embedding vectors. Due to their scalability on large datasets and attractive results, descriptor methods have attracted increasing interest in the last few years.

2.3. Graph representation in bioinformatics

Graphs are a powerful formalism for representing biological systems. Many biological systems, such as DNA, RNA, molecules, metabolites, and proteins, can be naturally represented by graphs that capture many of their essential properties [27–29]. Graph representations of biological systems effectively help to understand the events and the processes within these systems that lead to pathologies and diseases [30–32]. Moreover, graph representations are widely used in machine learning to perform various tasks and have shown promising results in the last few years. For example, in [33], an end-to-end deep learning framework is proposed in which a GNN is adopted to predict properties of the molecules. Similarly, in [34–36], improved GNN architecture is proposed to improve predictions on molecular data, especially in quantum chemistry. For molecular structure generation, MolGAN [37] introduces reinforcement learning-based optimization, while [38] proposes a GNN architecture for protein interface prediction. There is a rich literature on similar approaches; the interested reader is referred to [39–41] for further reading.

Despite the massive success of graph representation methods, very few attempts have been made to improve graphical representations for attributed graphs. We consider constructing unattributed graphs from attributed graphs that incorporate the attributive and structural information of the original graph. This allows deploying graph embedding methods on attributed graphs and, thus, bridges the gap between attributed graphs and graph embedding methods. Testing on several benchmark algorithms, our empirical results showed improved results on the augmented graphs.

3. The proposed method

In this section, we present our framework for encoding attribute information using graph topology. We propose a graph augmentation approach to encode each node and/or edge attribute as a substructure within the topology of the input graph, as shown in Fig. 2. The goal is to perform lossless encoding to ensure that no information is lost in the translation, i.e., the original graph structure as well as the values of the node/edge attributes should be fully recoverable from the final graph. We would also like to make sure that the size of the final graph does not blow up to an extent that standard graph embedding methods cannot be used anymore. And lastly, the final representations should be expressive enough to produce comparable results on standard learning tasks. After such an ideal augmentation is performed, one can apply any graph embedding method to obtain feature vectors that can be utilized in classical machine learning algorithms. In this section, we formally define the combinatorial optimization problem of augmenting an attributed graph to an unattributed graph keeping in mind the above-mentioned challenges. We explore the theoretical limitations of any algorithm that solve the augmentation problem, as well as propose our solution.

3.1. Preliminaries

We begin with an outline of the necessary definitions, notations, and terminology.

3.1.1. Graph terminology

An unattributed graph $G = (V, E)$ is defined as a collection of nodes and edges, where V is a set of nodes, and $E \subseteq V \times V$ is the set of edges. For a given G, we define $\text{mag}(G) = |V| + |E|$ as the magnitude of the graph. For ease of exposition, all unattributed graphs discussed in this work are simple, undirected, and unweighted. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs, then G_1 and G_2 are isomorphic whenever there exists a mapping $\phi : V_2 \to V_1$ such that $E_1 = \{(\phi(u), \phi(v))|(u, v) \in E_2\}$, i.e., G_1 and G_2 represent the same graph modulo permutation of vertices. We also say G_2 is a subgraph of G_1 if there exists an induced subgraph of G_1 that is isomorphic to G_2. We refer to G_1 as G_2-free if there exists no induced subgraph in G_1 isomorphic to G_2.

3.1.2. Attributed and labeled graphs

An attribute of a node (resp. edge) refers to information relevant to the node (resp. edge). Attributes allow for providing further details about the entity being modeled. For simplicity, we assume that all graph attributes take integer values. We argue that this is a reasonable assumption because there are reasonably small number values that an attribute can take in a standard bioinformatics dataset, i.e., all attributes such as element type, bond type, metallicity, etc., can be efficiently mapped or approximated to the nearest integer. Each node (resp. edge) may have multiple attributes.

Thus an attributed graph L is a quadruple (V, E, δ, θ) where V is a set of nodes and E is a set of edges; δ is a mapping from V to $\mathbb{N}^{d_v}$ that assigns a list of d_v integer attributes to each node, and θ is a mapping from E to $\mathbb{N}^{d_e}$ that gives a list of d_e integer attributes to each edge in E. Either d_v or d_e can be zero, but we assume that for an attributed graph at least one of them is nonzero. For an attributed graph L, we use $\bar{L}$ to refer to the trivial unattributed graph represented by (V, E) without node or edge attributes.

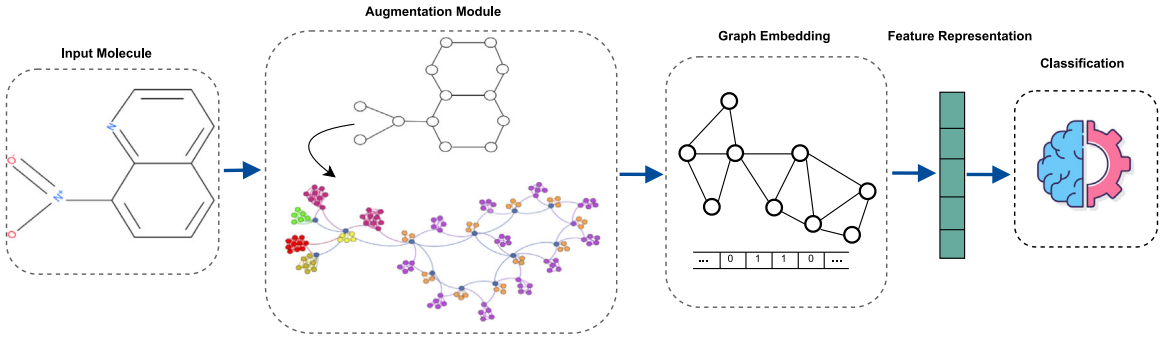


Fig. 2. Example of graph augmentation. we take a molecule as input and transform it into an augmented graph before passing it to a graph embedding method.

3.2. Encoding labeled graphs as unlabeled graphs

Let $\mathcal{L} = \{L_1, L_2, L_3, \ldots, L_k\}$ be a dataset of attributed graphs. For a given graph $L_i \in \mathcal{L}$, our goal is to construct an unattributed graph $G_i = (V_i, E_i)$ such that the attribute information in L_i is encoded in the topological structure of the graph G_i. Let $f : \mathcal{L} \to \mathcal{G}$ be a function such that $f(L_i) = G_i$. To ensure effective encoding, we propose that f must fulfill the following design properties:

(**P1**) The mapping f must be bijective; there must exist a function f^{-1} such that $f^{-1}(G_i) = L_i$ whenever $f(L_i) = G_i$.

(**P2**) On average, the increase in magnitude in the graphs, should be minimized:

$$\frac{1}{|\mathcal{L}|} \sum_{L_i \in \mathcal{L}} \text{mag}(G_i) - \text{mag}(\bar{L}_i)$$

(**P3**) The resultant graphs must be "expressive" enough for a classification model to learn relevant features.

Property **P1** ensures that L_i can be recovered from G_i, guaranteeing a lossless encoding that encapsulates the same information as the original attributed graph. Note that it also implies that for any two distinct attributed graphs $L_i, L_j \in \mathcal{L}$, the augmented unattributed graphs $f(L_i)$ and $f(L_j)$ must not be isomorphic. We note that graph magnitudes are extremely important to avoid excessive computational overhead for computing graph embeddings. Therefore property **P2** ensures that the output graphs do not explode in size.

Unfortunately, these properties are not enough to guarantee meaningful representations for classification tasks, as indicated by the lemma below.

Lemma 1. *For a given set of attributed graphs $\mathcal{L}$, a bijective mapping from $\mathcal{L}$ to $\mathcal{G}$ that minimizes the objective function,*

$$\frac{1}{|\mathcal{L}|} \sum_{L_i \in \mathcal{L}} mag(G_i) - mag(\bar{L}_i),$$

is trivial. Furthermore, such a mapping will always return the first $|\mathcal{L}|$ smallest graphs in $\mathcal{G}$, in order of magnitude defined above.

Proof. Let $\mathcal{G}' = \{G_1, G_2, \ldots, G_{|\mathcal{L}|}\}$ be the family of the $|\mathcal{L}|$ smallest graphs, with respect to magnitude. Note that this set can be found in exponential time by generating all graphs with up to $|\mathcal{L}|$ nodes, ordering the graphs with respect to their magnitudes, and keeping the first $|\mathcal{L}|$ graphs. It is clear that among all sets of graphs that can be output by a mapping from $\mathcal{L}$ to $\mathcal{G}$, the objective value cannot be less than

$$\frac{1}{|\mathcal{L}|} \sum_{L_i \in \mathcal{L}, G_i \in \mathcal{G}'} \text{mag}(G_i) - \text{mag}(\bar{L}_i).$$

Therefore, any bijective mapping from $|\mathcal{L}|$ to $\mathcal{G}'$ is a feasible solution with an optimal value, as no smaller graphs exist for the attributed graphs in $\mathcal{L}$ to be mapped to. Note that the set $\mathcal{G}'$ is not unique because multiple graphs may have the same magnitude, but an arbitrary choice suffices to complete the proof of the statement. □

The lemma above shows that while there may exist some trivial graph augmentations, they are not suited for tasks like classification because we may lose important information in the process. Therefore, we included property **P3**, which, although subjective, is necessary to obtain meaningful embeddings. For a given graph, utilizing the presence or absence of certain subgraph structures has proven to be useful in classification settings in some previous works (e.g., [42,43]). In this vein, we propose appending "appropriate" subgraphs to nodes, corresponding to the attributes within $\mathcal{L}$. Formally, let $\gamma : \mathbb{N}^d \to \mathcal{G}$ be a mapping from attribute vectors to graphs, where d may be equal to d_v or d_e, the dimensions of the node and edge attribute vectors, respectively. We present an algorithm, 1, that constructs an unattributed graph G_i for a given graph L_i. We begin by converting an attributed graph L_i to a trivial unattributed graph $\bar{L}_i$ by removing all nodes and edge attributes. In steps 2–4, for each node v, the mapping function $\gamma(\delta(v))$ maps node attributes to subgraphs and appends them to their corresponding nodes. We repeat this process for each edge attribute as well. In steps 6–9, we append a subgraph $\gamma(\theta(e))$ to both end-nodes for an edge (u, v).

Algorithm 1 Attributed Graph Augmentation

Input: L_i
Output: G_i
1: transform L_i to $G_i = \bar{L}_i$ by removing all attributes
2: **for** each node v in L_i **do**
3: append subgraph $\gamma(\delta(v))$ to G_i
4: add an edge from each node in the subgraph $\gamma(\delta(v))$ to the node v in G_i
5: **end for**
6: **for** each edge $e = (u, v) \in G_i$ **do**
7: append the subgraph $\gamma(\theta(e))$ to G_i
8: add edges from a random node in $\gamma(\theta(e))$ to both end nodes u and v in the graph G_i
9: **end for**
10: **return** G_i

We note that this mapping satisfies all three sought-out design properties, given an appropriate function γ. To obtain the original labeled graph L_i from the unattributed graph G_i, one can find all the appended subgraphs G_i, delete them from G_i, and label the node it was attached to according to γ. Note that this reverse construction is independent of the construction method and only

needs access to γ. We observe that if γ produces subgraphs that already exist as induced subgraphs in some input attributed graph L_i, then we will not be able to completely recover L_i because the reverse construction process will delete all copies including the ones that were not added via the construction process. Therefore, this requires appending graphs that do not already exist as subgraphs of any graph L_i, i.e., for all $L_i \in \mathcal{L}$, L_i must be H-free for any $H = \gamma(k)$ where k is a label used in $\mathcal{L}$. Unfortunately, attempting to meet property **P2** under this setting is a computationally challenging task. We show this by proving that the problem of finding the smallest magnitude subgraph to append even for one value of an attribute in a graph is NP-Hard. We first formally define the problem:

Problem 3.1 (*k-Magnitude H-free Discovery (k-HFD)*)**.** Given G, find a graph H with magnitude k such that G does not contain H as an induced subgraph, i.e., G is H-free. If there are multiple such graphs H, then return the one with the fewest number of nodes. If there is no such H then return NULL.

Note that some instances of k-HFD are trivial. For example, when $k = 2$, a solution will always return NULL because the only graph with magnitude 2 is two nodes with no edge between them and G will always contain such nonadjacent two nodes unless G is a complete graph. Similarly, when $k = 3$, there are only two possibilities for H, three nodes that do not have any edges with them and two nodes that have an edge between them. Such cases can be easily checked. We show that in general, the k-Magnitude H-free Discovery Problem is NP-hard in the following theorem via a simple reduction.

Theorem 3.1. *The k-Magnitude H-free Discovery Problem is NP-Hard.*

Proof. We prove the hardness result by reducing the famous k-Clique problem to k-HFD problem. The k-Clique problem can be defined as follows: given an (unattributed) graph G, does there exist a subgraph on k nodes where each pair of nodes is connected by an edge?

Let $(\hat{G}, \hat{k})$ be an instance of k-Clique problem. We create an instance of k-HFD by setting $G = \hat{G}$ and $k = \hat{k} + \binom{\hat{k}}{2}$. Note that $\hat{k} + \binom{\hat{k}}{2}$ is the magnitude of a complete graph on $\hat{k}$ nodes. If the hypothetical black box solution to k-HFD returns a graph H on $\hat{k}$ nodes, then H must be the complete graph on these nodes and $\hat{G}$ must not contain a copy of such a graph. So, we can be sure that the original graph G does not contain a clique on the required number of nodes. So, we can return NO answer to the k-Clique problem. If k-HFD returns NULL, or the H graph returned has more than $\hat{k}$ nodes, then it indicates the graph $\hat{G}$ is not $\hat{k}$-clique free, assuming there are enough nodes in the original graph. Therefore, in this case, we can return a YES answer to the original problem because the graph G contains a clique on the prescribed number of nodes. Note that if the number of nodes in H is more than $\hat{k}$, then $\hat{G}$ cannot contain a clique on $\hat{k}$ nodes because the solution to the k-HFD is required to contain the fewest number of nodes. This completes the proof of the statement. □

The above theorem implies that there is no polynomial-time solution for the problem of finding and augmenting attributed graphs unless $P = NP$. However, we assert that domain knowledge of bioinformatics can be used to design an efficient solution to this problem. For an instance, it is well-known that three-membered rings (3MR) like epoxies are highly uncommon, strained, and susceptible to ring-opening [44]. Therefore, it is plausible to use triangles, i.e., three nodes all connected to each other by three edges, to represent an attribute in an output graph. Similarly, the domain knowledge of the particular dataset can be employed to find structures that are small in size, and rarely exist in the attributed graphs representing the dataset. We observe that, in general, the graphs in the bioinformatics datasets are quite sparse in nature and do not contain medium-sized cliques. Using cliques allow us to efficiently compute γ by arranging the attributes in order of how many times they appear in the dataset $\mathcal{L}$ and assigning the smallest cliques to the most frequently appearing labels.

Note that due to a clique's symmetric nature, we can also save on extra edges by adding an edge from only one of its nodes to the corresponding nodes in the attributed graph. The corresponding structure in this case is called a "lollipop graph". In our experiments, we use both cliques and lollipop graphs.

4. Case study: Graph embedding for toxicity prediction

Measuring chemical toxicity is an important task in pharmaceutical, agricultural, and environmental sciences. In pharmacology, chemical toxicity plays a key role in drug discovery and development. Chemical toxicity is considered a major factor in disqualifying most drug candidates in the discovery process [45]. Thus, it is highly desirable to develop new methods for predicting the toxicity of chemicals.

More recently, Graph Machine Learning (GML) has shown great success in molecular generation and drug discovery [46]. Since molecules can be naturally represented by graphs, GML methods are known to be suitable candidates to leverage the topological structure along with attribute information to perform various machine learning tasks. The latest developments in the GML field, graph descriptors, and GNNs have shown promising results in the application of toxicity prediction on molecular graphs [10,43,47,48]. In the following, we provide a brief comparison of both types of methods.

GNNs: GNNs are generally based on a message passing mechanism where they exploit node feature information along with graph topology to learn the desired representation. In applications where expressive attributes are available, GNNs are found to be superior in terms of classification accuracy. Conversely, in case of limited or no feature availability, the performance of GNNs is dominated by that of graph descriptors. The following are some of the major challenges of GNN-based approaches:

- **Require large training data:** In applications where large amounts of training data is unavailable, GNNs may not be a suitable candidate to perform the desired ML task [49].
- **Robustness and guaranteed performance:** GNNs usually report large variance in results which limits its application where trusted results are necessary [50].
- **Interpretability:** The results reported by GNNs are, in general, not open to interpretation. Therefore, GNNs are not suitable for real-world applications where interpretability and explainability are important [51].
- **Lack of attribute information:** In applications where attribute information is unavailable, the performance of GNNs methods is quite limited [52].
- **Scalability on sufficiently large graphs:** Training GNNs is a computationally expensive task that scales with the order of the graph. As such, their application is limited to small or medium-sized graphs [49].

Graph descriptors: Graph descriptors are flexible graph representation methods that can be adjusted to fulfill the desired properties of the embedding vector. For example, in applications where subgraphs play an important role, graph descriptors can be adjusted to capture the information of the subgraphs [43]. Graph

Table 1
Characteristics of the chosen 15 bioinformatics datasets. Attributes $|G|$ indicates simple unattributed datasets, $\mathcal{G}$ indicates augmented datasets, $V(.)$ and $E(.)$ indicate the average number of nodes and edges in both datasets.

Dataset	$\|G\|$	avg.$\|V(G)\|$	avg.$\|E(G)\|$	avg. $\|V(\mathcal{G})\|$	avg.$\|E(\mathcal{G})\|$
MUTAG	186	17.95	19.81	213.63	692.47
PTC	343	14.71	14.72	200.58	701.17
NR-AHR	1900	17.88	18.79	259.93	959.61
NR-AR	756	21.05	22.59	345.69	1375.79
NR-AR-LBD	604	20.45	21.92	340.93	1368.34
NR-Aromates	712	19.58	20.57	298.36	1138.47
NR-ER	1866	17.83	18.70	271.19	1033.58
NR-ER-LBD	882	19.15	20.21	296.61	1142.70
NR-PPAR-GAMMA	442	18.33	19.10	272.07	1023.69
SR-ARE	2188	17.30	18.01	258.96	980.41
SR-ATAD5	674	18.06	18.88	266.26	993.70
SR-HSE	850	17.06	17.72	253.55	955.79
SR-MMP	2246	18.24	19.06	272.44	1028.29
SR-P53	1064	19.71	20.38	298.51	1132.21
NCI1	3474	29.30	31.91	428.41	1513.20

Table 2
Comparison of classification accuracies of seven graph embedding methods on 15 bioinformatics datasets. G indicates the simple undirected graphical representation of the datasets while $\mathcal{G}$ indicates the transformed graphical representation using the proposed method. Augmentation is performed on cliques of different sizes. Blue indicates improved results using the proposed approach.

Dataset	FGSD		NetLSD		HOSD		NetSIMILE		SP		WL		FWL-D	
	G	$\mathcal{G}$	G	$\mathcal{G}$	G	$\mathcal{G}$	G	$\mathcal{G}$	G	$\mathcal{G}$	G	$\mathcal{G}$	G	$\mathcal{G}$
MUTAG	87.77	89.9	81.52	85.56	86.74	89.26	82.49	86.87	87.4	87.0	82.92	83.22	86.49	89.27
PTC	56.25	58.04	58.62	59.5	60.03	59.0	55.1	54.27	57.22	57.32	51.3	51.62	53.97	60.28
NR-AHR	76.04	78.57	71.84	74.54	74.2	74.56	74.96	76.2	70.68	75.53	69.42	72.95	75.95	80.16
NR-AR	77.43	75.65	73.89	73.57	79.69	73.09	76.27	74.28	75.53	75.79	73.44	72.65	76.32	75.93
NR-AR-LBD	81.19	82.45	81.01	77.0	80.98	81.41	81.08	79.54	79.77	80.1	74.36	78.8	83.12	82.64
NR-Aromates	78.14	77.88	72.19	74.6	76.32	79.3	75.03	75.67	75.48	75.78	73.2	76.85	76.56	80.61
NR-ER	67.77	67.94	63.41	65.26	66.49	69.24	64.71	67.66	65.82	67.38	63.57	63.83	67.2	69.94
NR-ER-LBD	72.97	75.89	69.83	71.34	73.38	75.66	70.05	72.45	68.93	70.41	65.53	67.46	71.09	71.52
NR-PPAR-G	67.76	71.57	64.8	61.81	73.48	70.89	69.59	68.47	66.28	71.0	66.76	67.45	74.43	72.63
SR-ARE	68.56	70.99	63.42	67.57	66.19	69.39	66.75	67.4	65.82	68.33	64.13	65.77	67.27	74.13
SR-ATAD5	72.85	74.57	68.65	70.01	73.86	75.47	70.81	71.96	69.21	74.5	71.08	68.11	71.93	75.38
SR-HSE	63.29	66.19	58.09	60.82	61.15	65.12	59.51	63.29	61.22	63.47	61.29	60.71	60.71	64.94
SR-MMP	76.52	78.75	72.47	73.05	74.18	78.79	74.12	77.06	71.97	74.24	70.4	73.07	75.82	80.46
SR-P53	74.64	75.35	72.14	68.93	73.61	72.88	72.37	73.07	71.07	70.7	68.26	68.34	75.57	78.47
NCI1	76.51	75.9	69.64	65.18	72.76	72.77	70.12	70.03	66.26	66.34	65.69	67.24	82.21	82.81

descriptors also allow the end-user to apply machine learning classifiers of their choice. However, the main limitation of graph descriptors is that they rely only on graph topology, i.e., they do not use attribute information. Due to this, it loses valuable information, hindering the method's performance. Therefore, it is highly desirable to incorporate attribute information within the graph topology so that graph descriptor methods can be applied. Thus, the main objective of the proposed framework is to transform attributed graphs into simple unattributed graphs.

Here we consider toxicity prediction as an application to the proposed framework, as the input datasets are easily modeled as attributed graphs. In the subsequent sections, we provide the experimental setup and numerical results in different settings to evaluate the efficacy of the proposed framework.

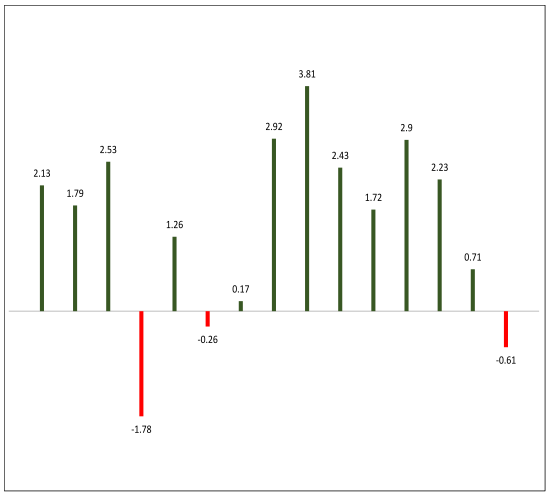
4.1. Datasets and graph descriptors

We ran multiple experiments to evaluate the performance of unattributed graph embedding methods on augmented graphs. We considered the problem of toxicity prediction as a graph classification task and chose 15 well-known bioinformatics datasets, among which the balanced versions of 12 datasets are selected from the Tox21 data challenge [53,54]. The remaining three datasets are MUTAG, PTC, and NCI1, which are well-known benchmark datasets for graph classification. The Tox21 data challenge is a grand challenge on molecule toxicity prediction consisting of ≈10K compounds with active and inactive class labels and 12 pathway assays. MUTAG is the graphical dataset representing the mutagenicity correlation among 188 nitro compounds. PTC and NCI1 are chemical compound datasets describing the toxicity of various cancer cells. Overall, the datasets we chose describe the toxicity of various molecules and classified either toxic or non-toxic. The characteristics of these datasets are presented in Table 1. Note that there is a slight difference between MUTAG, PTC, and NCI1's class distributions. Also, as we preprocess these datasets from SMILES, two instances in MUTAG and one in PTC were ignored due to getting "null objects" during the transformation. For the NCI1 dataset, we used the complete NCI dataset available on PubChem and compared instances one by one through a graph isomorphism test. A total of 3474 matched instances were found among 4110.

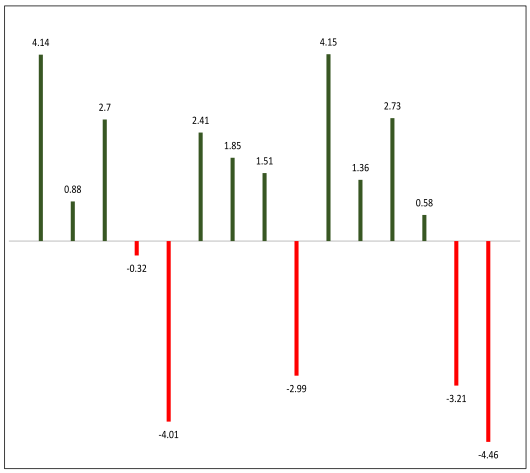
We ran seven state-of-the-art graph embedding methods: FGSD [10], NetLSD [11], NetSIMILE [21], HOSD [43], Shortest-Path Kernel (SP), [15], Weisfeiler Lehman Kernel (WL) [55] and Graph Filtration Kernel (FWL-D) [56] on the simple unattributed graphs (denoted by G) and the transformed graphs (denoted by $\mathcal{G}$) to compare the results.

Experimental setup: All experiments were conducted on a machine with an Intel (R) Xeon (R) 4110 CPU 2.10 GHz with 32 logical cores and 512 GB of RAM. We implemented the proposed method in Python with RDKit to handle molecules and NetworkX to work with graphs. We used SMILES representations of all the chosen datasets to convert molecules into the desired graphical representation with the RDKit library in Python. The code and collected datasets are made publicly available to encourage reproducibility of the results.[1] We performed experiments on

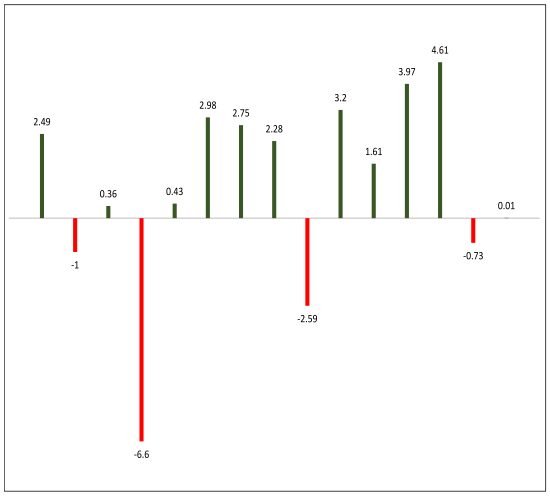
[1] https://github.com/Anwar-Said/Attributed-Graph-Augmentation



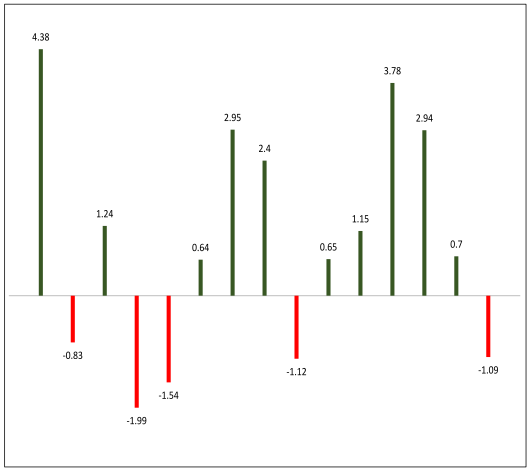
(a) FGSD



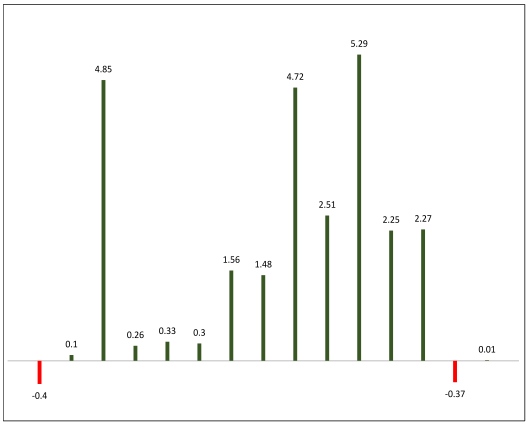
(b) NetLSD



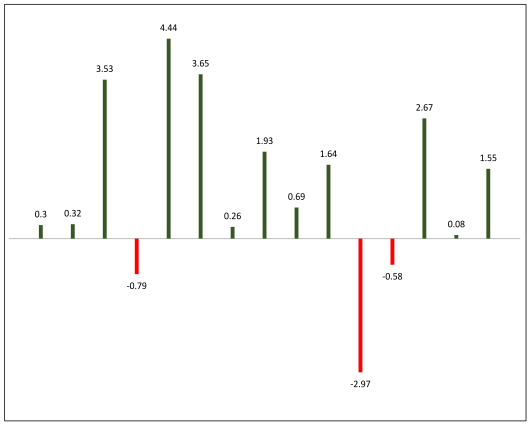
(c) HOSD



(d) NetSIMILE



(e) SP Kernel



(f) WL Kernel

Fig. 3. The improvement in classification accuracy of state-of-the-art graph embedding methods on the graph classification task. Each figure shows one algorithm's performance on 15 benchmark bioinformatics datasets and plots improvement or detriment (%) in the new results.

Table 3
Comparison of classification accuracy of six graph embedding methods on 15 bioinformatics datasets has been shown. G indicates the simple undirected graphical representation of the datasets while $\mathcal{G}$ indicates the augmented representations using the proposed method. The augmentation has been performed on lollipop graphs of different sizes. Blue indicates improved results using the proposed approach.

Dataset	FGSD		NetLSD		NetSIMILE		SP		WL		FWL-D	
	G	$\mathcal{G}$	G	$\mathcal{G}$	G	$\mathcal{G}$	G	$\mathcal{G}$	G	$\mathcal{G}$	G	$\mathcal{G}$
MUTAG	87.54	90.82	84.82	87.63	83.35	88.94	85.91	87.02	82.11	86.40	86.49	83.39
PTC	55.09	56.86	56.60	51.60	52.76	51.62	54.59	53.40	50.72	53.69	53.97	58.37
NR-AHR	75.37	77.80	72.58	73.68	74.84	75.84	70.95	74.74	70.16	71.84	75.95	79.95
NR-AR	76.87	75.78	72.78	74.10	76.19	73.68	76.08	76.09	73.18	70.93	76.32	76.05
NR-AR-LBD	81.11	83.24	80.95	79.12	81.11	80.09	80.62	80.96	73.70	74.36	83.12	84.79
NR-Aromates	75.16	77.61	77.84	72.91	74.86	74.58	75.32	74.05	73.49	73.08	76.56	78.08
NR-ER	66.57	68.52	63.41	65.44	64.71	66.04	65.39	66.73	63.25	64.85	67.20	69.13
NR-ER-LBD	74.04	76.84	69.97	72.57	70.13	72.00	69.73	70.87	65.19	64.62	71.09	72.56
NR-PPAR-G	66.75	70.75	64.71	69.22	70.13	71.64	66.07	66.74	67.22	66.51	74.43	70.83
SR-ARE	67.51	70.35	67.32	66.82	66.78	67.20	65.95	68.47	64.08	64.67	67.27	74.45
SR-ATAD5	71.40	75.95	70.05	69.89	70.90	70.28	68.27	72.25	70.04	66.47	71.93	75.81
SR-HSE	62.82	66.64	60.82	63.65	59.81	66.19	59.65	61.76	60.94	60.35	60.71	67.65
SR-MMP	74.72	78.12	70.63	73.83	74.18	76.65	71.38	74.41	69.91	71.74	75.82	80.90
SR-P53	72.96	75.40	71.08	70.69	72.31	71.03	70.98	70.33	68.82	69.09	75.57	75.47
NCI1	75.97	76.64	69.15	64.71	71.19	67.11	68.54	66.04	66.15	63.10	82.21	81.23

cliques and lollipop graphs to evaluate the effectiveness of the augmented graphs.

We considered four atom attributes and four bond types in our experimental setup. Atoms' attributes are aromaticity, positive/negative charge, and non-metallicity, for which we choose cliques of sizes 4, 6, 7, and 8 respectively. Similarly, we choose four bond types: aromatic, single, double, and triple, for which cliques of sizes 5, 9, 10, and 11 are chosen. Parameter settings for the embedding methods we used are listed below:

- For FGSD we set bin width size $= 0.01$.
- For NetLSD, we used the heat kernel with a 250-dimensional vector.
- For HOSD, we use the graphlet version with a limit of 4 and consider the graphs of magnitude ≤ 500 due to their high computational load.

For the remaining methods, we kept all parameters as default as mentioned in the actual papers. We considered the graph classification task, where we extracted graph embeddings using these methods. Then, we used a Random Forest classifier with 500 estimators to perform the classification. 10-fold cross-validation was used for each dataset. Classification accuracy is reported for all the methods on the original and transformed graphs.

All the datasets in SMILES format [57] were collected from different sources to create two types of graphical datasets: first, simple, unattributed graphs G were generated. Recall that G is constructed by converting each molecule to a simple graph where nodes represent atoms and edges represent bonds. Second, the augmented graphical representation $\mathcal{G}$ is constructed from G. Then, each algorithm was run on both datasets. In Table 2, the performance of 7 algorithms on 15 bioinformatics datasets is shown, and the classification accuracy on both datasets is reported. To highlight the augmented graphs' efficacy, we show the improvements (%) on all the datasets of each algorithm in Fig. 3. We ran each algorithm on the two versions (G and $\mathcal{G}$) of each dataset and then plotted the difference in the results shown in Fig. 3. The green bars represent an improvement in the results, while the red bars represent a detriment. We observed an improvement on each dataset by at least one embedding method. In particular, the short-path kernel performed quite well and improved results on all the datasets except MUTAG and NCI1. The highest improvement achieved was 6.86% on SR-ARE and 6.31% on PTC datasets. Similarly, the WL, FWL-D, and FGSD obtained improvements on 12 datasets, while the HOSD achieved improvements over 11 datasets. Overall, we found on average of 2% improvement on each dataset by every embedding method.

Results on Lollipop graphs: To show the capability and expressiveness of augmenting different graph structures, we also report results on lollipop graphs in Table 3. We chose lollipop graphs of magnitude 10, 11, 12, and 13 for atoms' attributes and 6, 7, 8, and 9 for bonds' attributes. We note that because of the small difference among cliques and lollipop graphs, we consider graphs of magnitudes greater than those of the cliques chosen in the first experiment. As before, each embedding method's improvement in each dataset is shown in Fig. 4. FGSD has performed quite well and improved on all the datasets except NR-AR with an average 2.49% improvement on all the datasets. Similarly, other algorithms have also shown improved results on most of the datasets. We observed an improvement on each dataset by at least one embedding method. Note that in Figs. 3 and 4 dataset labels are omitted. However, the sequence of datasets (left to right) is the same as presented in Table 1.

5. Discussion

Our empirical results support our theory that the proposed augmentation framework is powerful enough to encapsulate attribute information in unattributed graphs, and significantly improve the performance of graph embedding methods. Having numerous applications to graph embedding methods, our framework also allows encapsulating information that may not aid classification using GNNs. For example, adding atoms' labels to the feature set may not improve learning models' performance. However, using our framework, the augmentation function can produce a quite suitable transformation from the input molecule. Such transformed data can then be fed to any graph embedding or graph learning model to perform the desired task. To conclude, in light of the aforementioned results, the proposed framework is quite useful for applying to any type of attributed graph dataset to construct expressive simple graph representations. In particular, it can be used in settings where: (1) training data is limited; (2) computational resources are limited; (3) reliable results are required; and (4) the size of the graphs is huge, where other learning methods cannot be deployed.

6. Conclusions

This paper presented a framework for subgraph augmentation, which constructs unattributed graphs from attributed graphs while still preserving the attribute information. The proposed framework considers particular types of small graphs aiming at augmentation with minimum magnitude resultant graphs. We

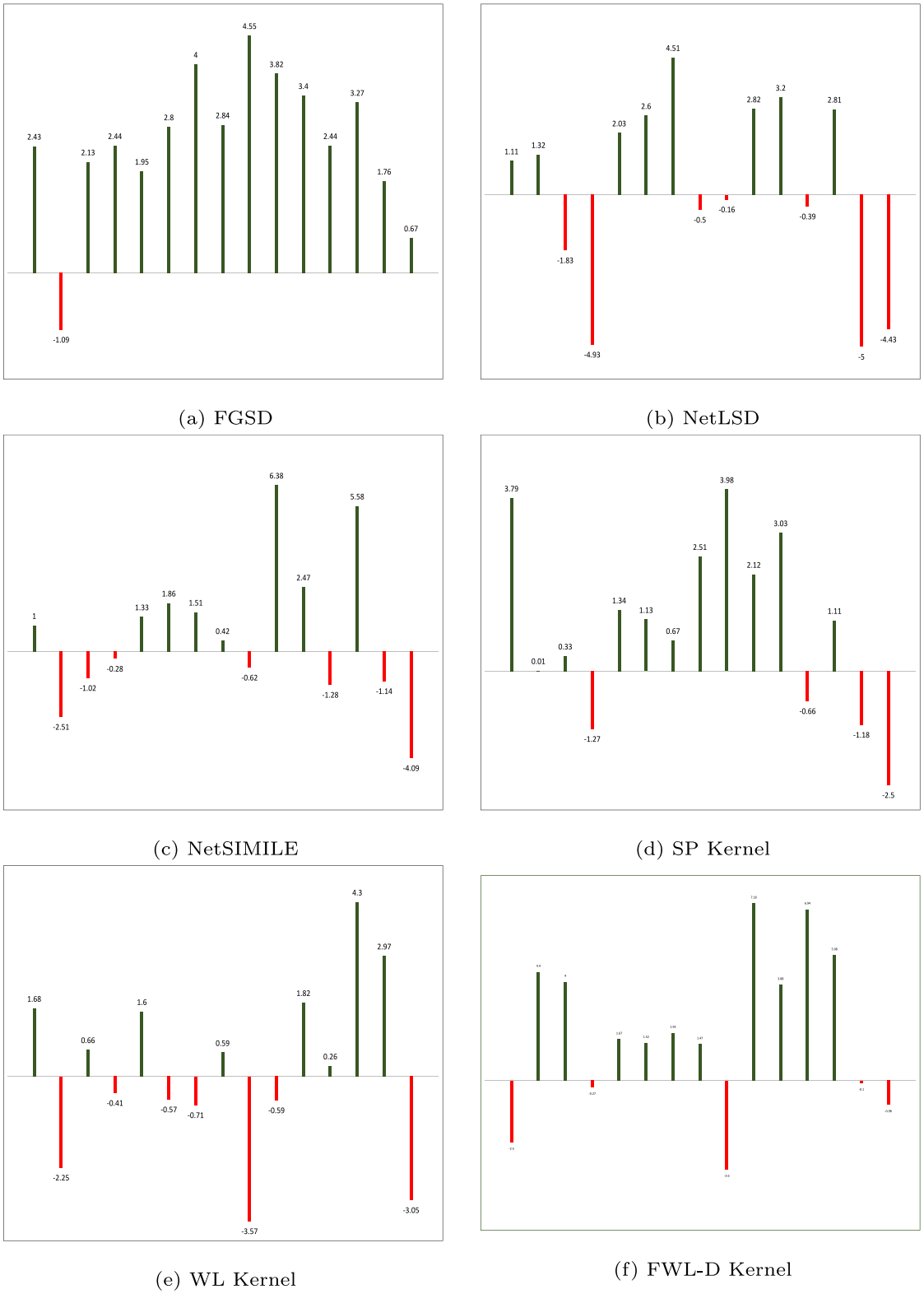


Fig. 4. Attributed graph augmentation results on lollipop graphs have been shown. Each figure shows one algorithm's performance on 15 benchmark bioinformatics datasets and plots improvement or decrements (%) in the new results.

analyzed the theoretical limitations of the problem and propose an algorithm. Using extensive experimentation on bioinformatics datasets, we showed that such augmentation improves graph embedding methods by at least 5%.

CRediT authorship contribution statement

Anwar Said: Methodology, Software, Formal analysis, Writing – original draft, Investigation, Validation, Data

curation. **Mudassir Shabbir:** Conceptualization, Formal analysis, Methodology, Writing – review & editing, Supervision, Project administration. **Saeed-Ul Hassan:** Writing – review & editing, Supervision, Validation, Project administration. **Zohair Raza Hassan:** Formal analysis, Investigation, Writing – original draft. **Ammar Ahmed:** Data curation, Investigation, Validation. **Xenofon Koutsoukos:** Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Github link is provided to access coda and data.

References

[1] A.W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A.W. Nelson, A. Bridgland, et al., Improved protein structure prediction using potentials from deep learning, Nature 577 (7792) (2020) 706–710.

[2] Y.-C. Lo, S.E. Rensi, W. Torng, R.B. Altman, Machine learning in chemoinformatics and drug discovery, Drug Discov. Today 23 (8) (2018) 1538–1546.

[3] A. Kundaje, W. Meuleman, J. Ernst, M. Bilenky, A. Yen, A. Heravi-Moussavi, P. Kheradpour, Z. Zhang, J. Wang, M.J. Ziller, et al., Integrative analysis of 111 reference human epigenomes, Nature 518 (7539) (2015) 317–330.

[4] E.P. Consortium, et al., An integrated encyclopedia of DNA elements in the human genome, Nature 489 (7414) (2012) 57–74.

[5] W.L. Hamilton, Graph representation learning, Synth. Lect. Artif. Intell. Mach. Learn. 14 (3) (2020) 1–159.

[6] G. Corso, L. Cavalleri, D. Beaini, P. Liò, P. Veličković, Principal neighbourhood aggregation for graph nets, Adv. Neural Inf. Process. Syst. 33 (2020).

[7] A.K. Wong, M. You, Entropy and distance of random graphs with application to structural pattern recognition, IEEE Trans. Pattern Anal. Mach. Intell. (5) (1985) 599–609.

[8] C. Morris, M. Ritzert, M. Fey, W.L. Hamilton, J.E. Lenssen, G. Rattan, M. Grohe, Weisfeiler and leman go neural: higher-order graph neural networks, in: Association for the Advancement of Artificial Intelligence, AAAI, AAAI Press, 2019, virtual.

[9] A. Said, S.-U. Hassan, W. Abbas, M. Shabbir, NetKI: A kirchhoff index based statistical graph embedding in nearly linear time, Neurocomputing 433 (2021) 108–118.

[10] S. Verma, Z.-L. Zhang, Hunt for the unique, stable, sparse and fast feature learning on graphs, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., Montreal, Canada, 2017, pp. 88–98.

[11] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, E. Müller, Netlsd: hearing the shape of a graph, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, London, United Kingdom, 2018, pp. 2347–2356.

[12] A. Said, S.-U. Hassan, S. Tuarob, R. Nawaz, M. Shabbir, DGSD: Distributed graph representation via graph statistical properties, Future Gener. Comput. Syst. 119 (2021) 166–175.

[13] F. Errica, M. Podda, D. Bacciu, A. Micheli, A fair comparison of graph neural networks for graph classification, in: International Conference on Learning Representations, (ICLR 2020), ACM, virtual, 2020.

[14] T. Gärtner, P. Flach, S. Wrobel, On graph kernels: Hardness results and efficient alternatives, in: Learning Theory and Kernel Machines, Springer, 2003, pp. 129–143.

[15] K.M. Borgwardt, H.-P. Kriegel, Shortest-path kernels on graphs, in: Fifth IEEE International Conference on Data Mining, (ICDM'05), IEEE, Houston, Texas, USA, 2005, p. 8.

[16] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, K. Borgwardt, Efficient graphlet kernels for large graph comparison, in: Artificial Intelligence and Statistics, PMLR, 2009, pp. 488–495.

[17] U. Kang, H. Tong, J. Sun, Fast random walk graph kernel, in: Proceedings of the 2012 SIAM International Conference on Data Mining, SIAM, 2012, pp. 828–838.

[18] S. Hido, H. Kashima, A linear-time graph kernel, in: 2009 Ninth IEEE International Conference on Data Mining, IEEE, USA, 2009, pp. 179–188.

[19] R.Y. Dougnon, P. Fournier-Viger, J.C.-W. Lin, R. Nkambou, Inferring social network user profiles using a partial social graph, J. Intell. Inf. Syst. 47 (2) (2016) 313–344.

[20] M. Sugiyama, K. Borgwardt, Halting in random walk kernels, Adv. Neural Inf. Process. Syst. 28 (2015) 1639–1647.

[21] M. Berlingerio, D. Koutra, T. Eliassi-Rad, C. Faloutsos, Network similarity via multiple social theories, in: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ACM, Niagara Ontario Canada, 2013, pp. 1439–1440.

[22] H.P. Maretic, M. El Gheche, G. Chierchia, P. Frossard, GOT: an optimal transport framework for graph comparison, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., Canada, 2019, pp. 13876–13887.

[23] R. Flamary, N. Courty, A. Rakotomamonjy, D. Tuia, Optimal transport with Laplacian regularization, in: Workshop on Optimal Transport and Machine Learning, NIPS 2014, Curran Associates, Inc., Canada, 2014.

[24] T. Yu, J. Yan, Y. Wang, W. Liu, et al., Generalizing graph matching beyond quadratic assignment model, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., Canada, 2018, pp. 853–863.

[25] Z. Guo, Y. Shen, A.K. Bashir, K. Yu, J.C.-w. Lin, Graph embedding-based intelligent industrial decision for complex sewage treatment processes, Int. J. Intell. Syst. (2021).

[26] J.M.-T. Wu, Z. Li, N. Herencsar, B. Vo, J.C.-W. Lin, A graph-based CNN-LSTM stock price prediction algorithm with leading indicators, Multimedia Syst. (2021) 1–20.

[27] M. He, S. Petoukhov, Mathematics of Bioinformatics: Theory, Methods and Applications, Vol. 19, John Wiley & Sons, New Jersey, 2011.

[28] Y. Yin, X. Zheng, B. Hu, Y. Zhang, X. Cui, EEG emotion recognition using fusion model of graph convolutional neural networks and LSTM, Appl. Soft Comput. 100 (2021) 106954.

[29] K. Skorniakov, D. Turdakov, A. Zhabotinsky, Make social networks clean again: graph embedding and stacking classifiers for bot detection, in: CIKM Workshops, ACM, Italy, 2018.

[30] M. Zitnik, F. Nguyen, B. Wang, J. Leskovec, A. Goldenberg, M.M. Hoffman, Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities, Inf. Fusion 50 (2019) 71–91.

[31] B. Yu, Y. Zhang, Y. Xie, C. Zhang, K. Pan, Influence-aware graph neural networks, Appl. Soft Comput. 104 (2021) 107169.

[32] T. Liang, X. Sheng, L. Zhou, Y. Li, H. Gao, Y. Yin, L. Chen, Mobile app recommendation via heterogeneous graph neural network in edge computing, Appl. Soft Comput. 103 (2021) 107162.

[33] D.K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., Montreal, Canada, 2015, pp. 2224–2232.

[34] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: ICML, ACM, Australia, 2017, arXiv–1704.

[35] Q. Liu, M. Nickel, D. Kiela, Hyperbolic graph neural networks, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., Canada, 2019, pp. 8230–8241.

[36] Y. Djenouri, G. Srivastava, A. Belhadi, J.C.-W. Lin, Intelligent blockchain management for distributed knowledge graphs in IoT 5G environments, Trans. Emerg. Telecommun. Technol. (2021) e4332.

[37] N. De Cao, T. Kipf, MolGAN: An implicit generative model for small molecular graphs, 2018, arXiv preprint arXiv:1805.11973, abs/1805.11973, arXiv–1805.

[38] A. Fout, J. Byrd, B. Shariat, A. Ben-Hur, Protein interface prediction using graph convolutional networks, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., Long beach CA, USA, 2017, pp. 6530–6539.

[39] Y. Ma, J. Tang, Deep Learning on Graphs, Cambridge University Presss, UK, 2020.

[40] P. Fournier-Viger, G. He, C. Cheng, J. Li, M. Zhou, J.C.-W. Lin, U. Yun, A survey of pattern mining in dynamic graphs, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 10 (6) (2020) e1372.

[41] Y. Li, C. Huang, L. Ding, Z. Li, Y. Pan, X. Gao, Deep learning in bioinformatics: Introduction, application, and perspective in the big data era, Methods 166 (2019) 4–21.

[42] R. Kondor, N. Shervashidze, K.M. Borgwardt, The graphlet spectrum, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, Montreal, Quebec, 2009, pp. 529–536.

[43] A. Ahmed, Z.R. Hassan, M. Shabbir, Interpretable multi-scale graph descriptors via structural compression, Inform. Sci. 533 (2020) 169–180.

[44] A. Schäfer, M. Weidenbruch, W. Saak, S. Pohl, Phosphasilirenes, three-membered rings containing PC double bonds, Angew. Chem. Int. Ed. Eng. 26 (8) (1987) 776–777.

[45] J. Jiang, R. Wang, G.-W. Wei, GGL-tox: geometric graph learning for toxicity prediction, J. Chem. Inf. Model. 61 (4) (2021) 1691–1700.
[46] J.M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N.M. Donghia, C.R. MacNair, S. French, L.A. Carfrae, Z. Bloom-Ackermann, et al., A deep learning approach to antibiotic discovery, Cell 180 (4) (2020) 688–702.
[47] O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, T. Langer, A compact review of molecular property prediction with graph neural networks, Drug Discov. Today: Technol. (2020).
[48] A. Said, T.D. Bowman, R.A. Abbasi, N.R. Aljohani, S.-U. Hassan, R. Nawaz, Mining network-level properties of Twitter altmetrics data, Scientometrics 120 (1) (2019) 217–235.
[49] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, T.-Y. Liu, Do transformers really perform badly for graph representation? Adv. Neural Inf. Process. Syst. 34 (2021) 28877–28888.
[50] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? in: International Conference on Learning Representations, ICLR, ACM, New Orleans, Louisiana, 2018, pp. 826–1810.
[51] H. Yuan, H. Yu, S. Gui, S. Ji, Explainability in graph neural networks: A taxonomic survey, 2020, arXiv preprint arXiv:2012.15445.
[52] C. Morris, M. Ritzert, M. Fey, W.L. Hamilton, J.E. Lenssen, G. Rattan, M. Grohe, Weisfeiler and leman go neural: Higher-order graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, No. 01, 2019, pp. 4602–4609.
[53] T.D. Challenge, Tox21 data challenge 2014, 2014.
[54] S. Pan, J. Wu, X. Zhu, CogBoost: Boosting for fast cost-sensitive graph classification, IEEE Trans. Knowl. Data Eng. 27 (11) (2015) 2933–2946.
[55] N. Shervashidze, P. Schweitzer, E.J.v. Leeuwen, K. Mehlhorn, K.M. Borgwardt, Weisfeiler-lehman graph kernels, J. Mach. Learn. Res. 12 (Sep) (2011) 2539–2561.
[56] T. Schulz, P. Welke, S. Wrobel, Graph filtration kernels, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, No. 8, 2022, pp. 8196–8203.
[57] D. Weininger, SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules, J. Chem. Inf. Comput. Sci. 28 (1) (1988) 31–36.