Machine Learning at the Forefront of Molecular Analysis

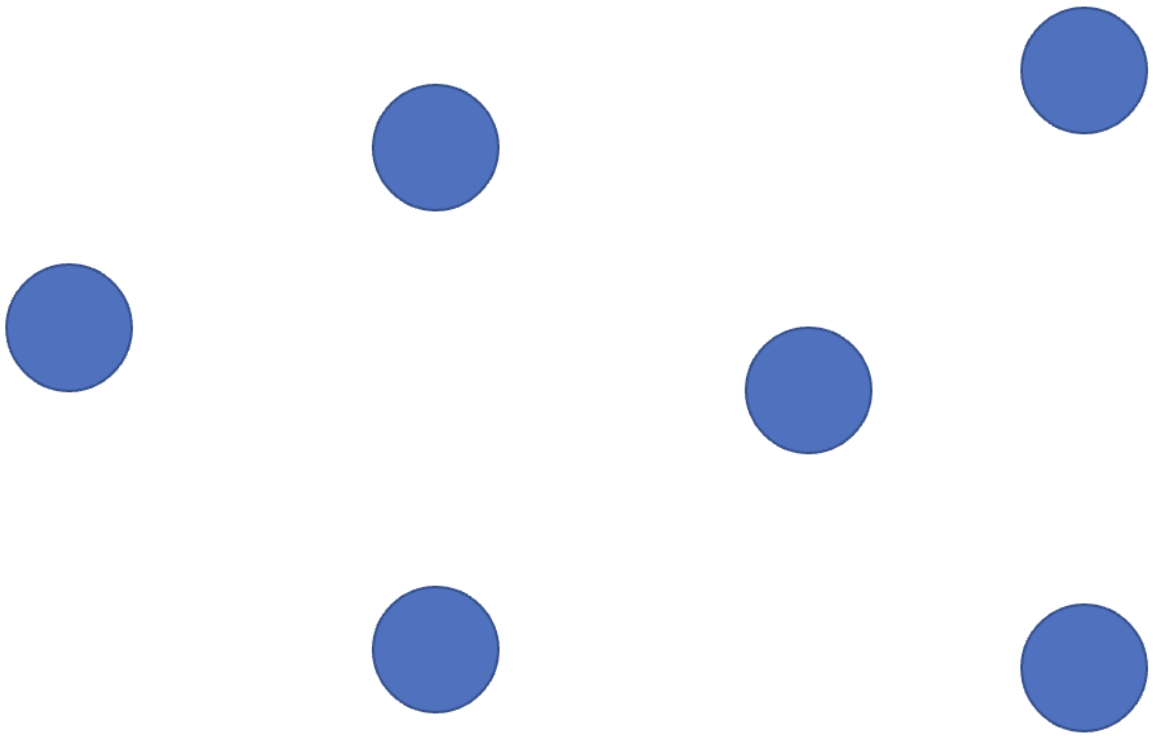
Agricultural Sciences, Tennessee State University

Anwar Said
Postdoctoral Research Fellow
Institute for Software and Integrated Systems,
Vanderbilt University, TN
March 14, 2023

Agenda

- Introduction to graphs
- Graph representation problem
- Applications
- Challenges
- Graph Neural Networks
- Graph descriptors
- Toxicity prediction with graph augmentation

Introduction to Graphs



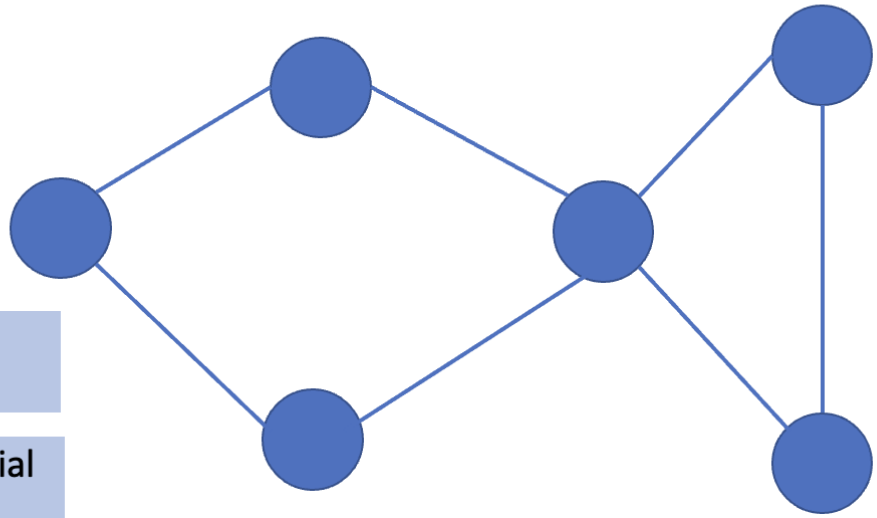
Atom in a molecule

User in a social network

Item in ecommerce network

Protein in PPI

Graphs



Bond in a molecule

Friendship in a social network

Transaction in ECN

Path in traffic network

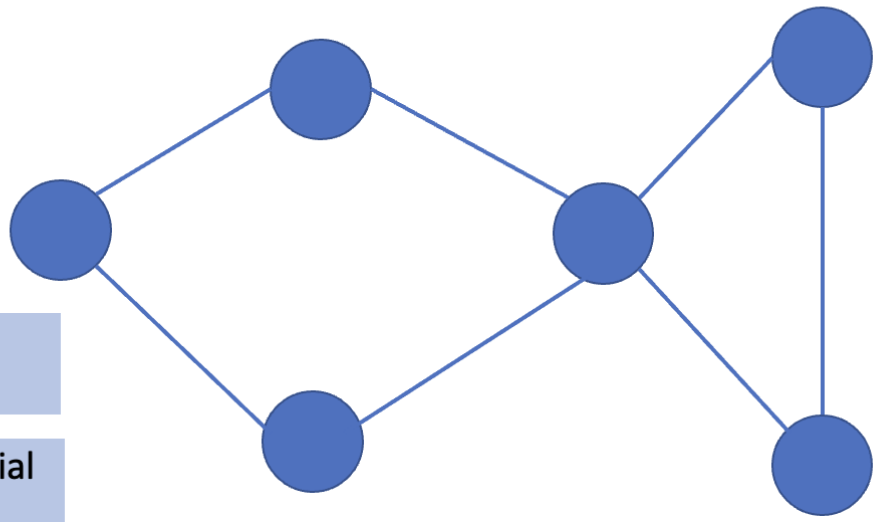
Atom in a molecule

User in a social network

Item in ecommerce network

Protein in PPI

Introduction



Atom in a molecule

User in a social network

Item in ecommerce network

Protein in PPI

Bond in a molecule

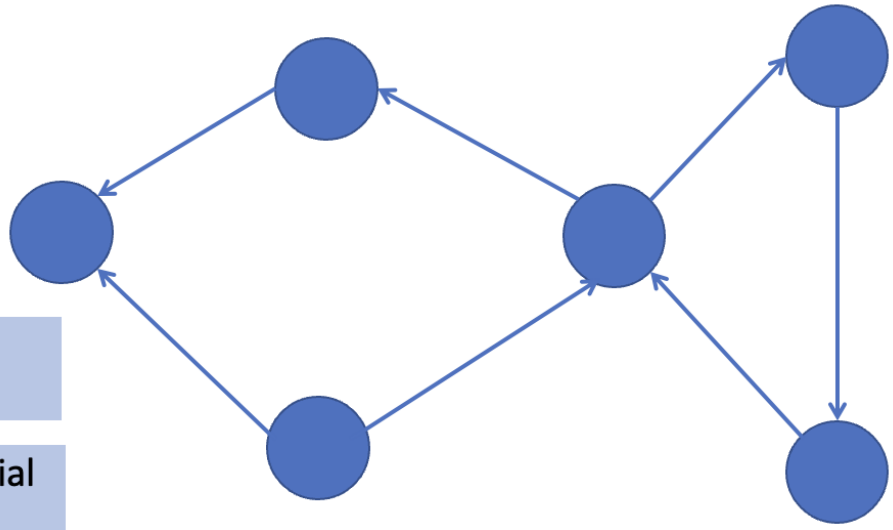
Friendship in a social network

Transaction in ECN

Path in traffic network

$$G = (V, E) \quad where \quad V = \{v_1, v_2, \ldots, v_n\} \quad \text{and} \quad E \subseteq V \times V$$

Directed Graphs



Atom in a molecule

User in a social network

Item in ecommerce network

Protein in PPI

Bond in a molecule

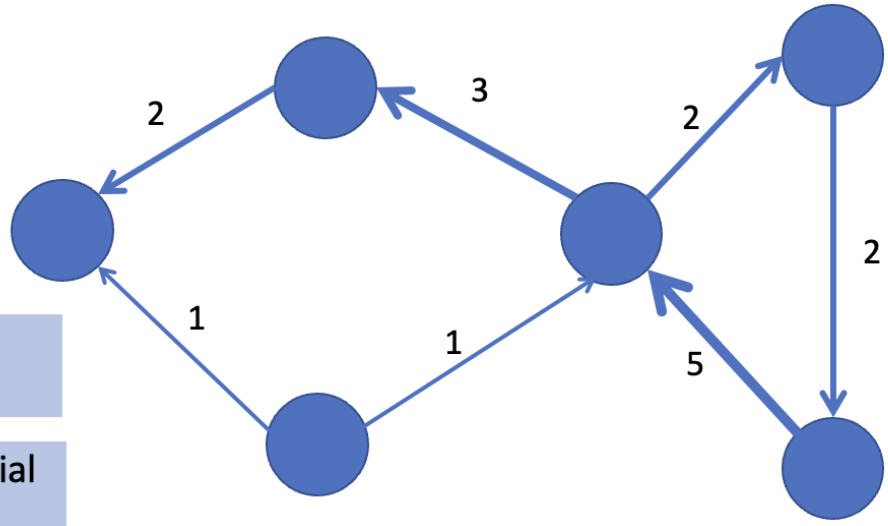
Friendship in a social network

Transaction in ECN

Path in traffic network

$$G = (V, E) \quad where \quad V = \{v_1, v_2, \ldots, v_n\} \quad \text{and} \quad E \subseteq V \times V$$

Directed Weighted Graph



Bond in a molecule

Friendship in a social network

Transaction in ECN

Path in traffic network

Atom in a molecule

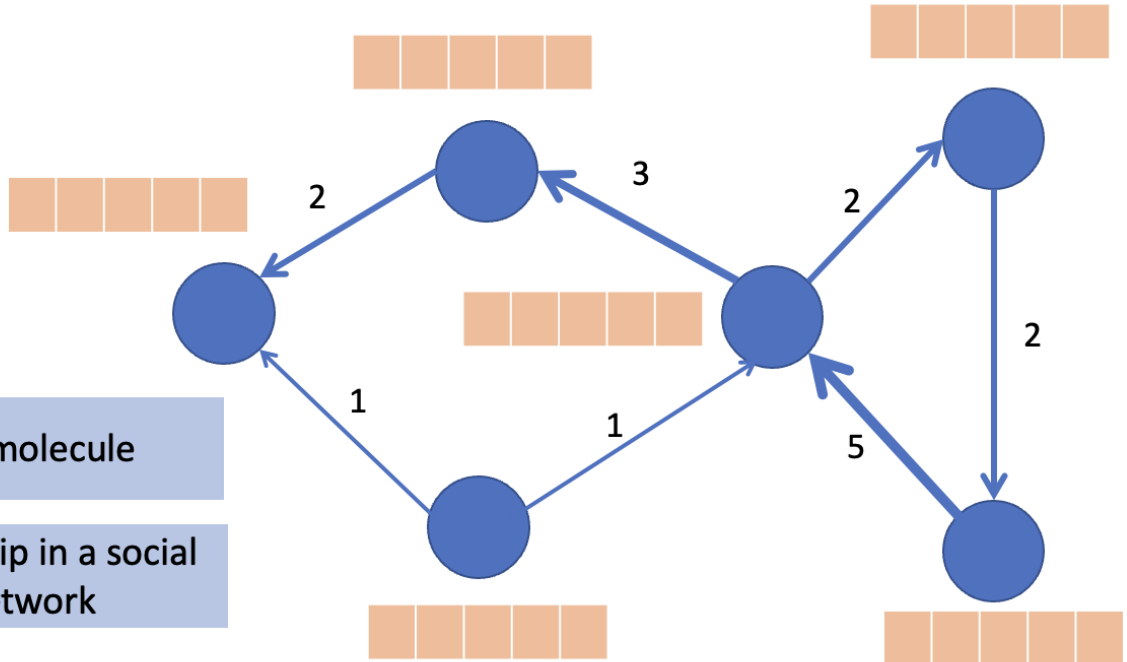
User in a social network

Item in ecommerce network

Protein in PPI

$$G = (V, E, W) \quad where \quad V = \{v_1, v_2, \ldots, v_n\} \quad \text{and} \quad E \subseteq V \times V$$

Attributed Graph



Atom in a molecule

User in a social network

Item in ecommerce network

Protein in PPI

Bond in a molecule

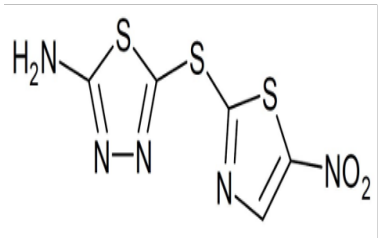
Friendship in a social network

Transaction in ECN

Path in traffic network

$$G = (V, E, X, W) \quad where \quad V = \{v_1, v_2, \ldots, v_n\} \quad \text{and} \quad E \subseteq V \times V, X^{N \times d}$$

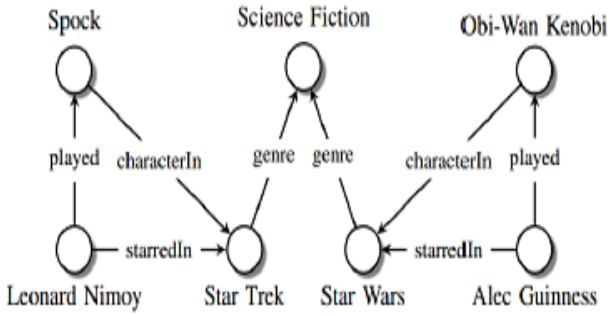
Networks are all around us?



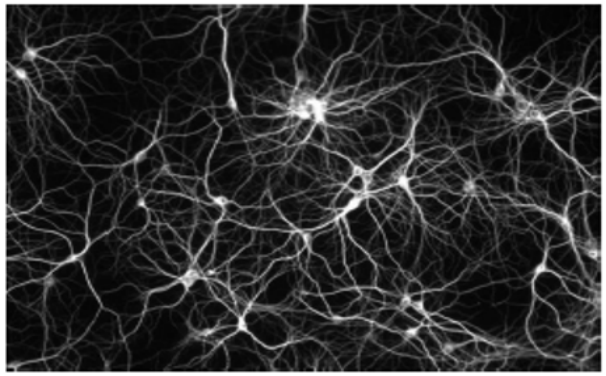
Molecular Network

Social Network

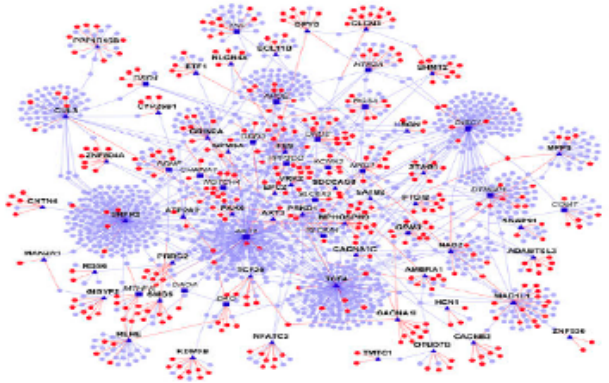
Communication Network



Knowledge Graph

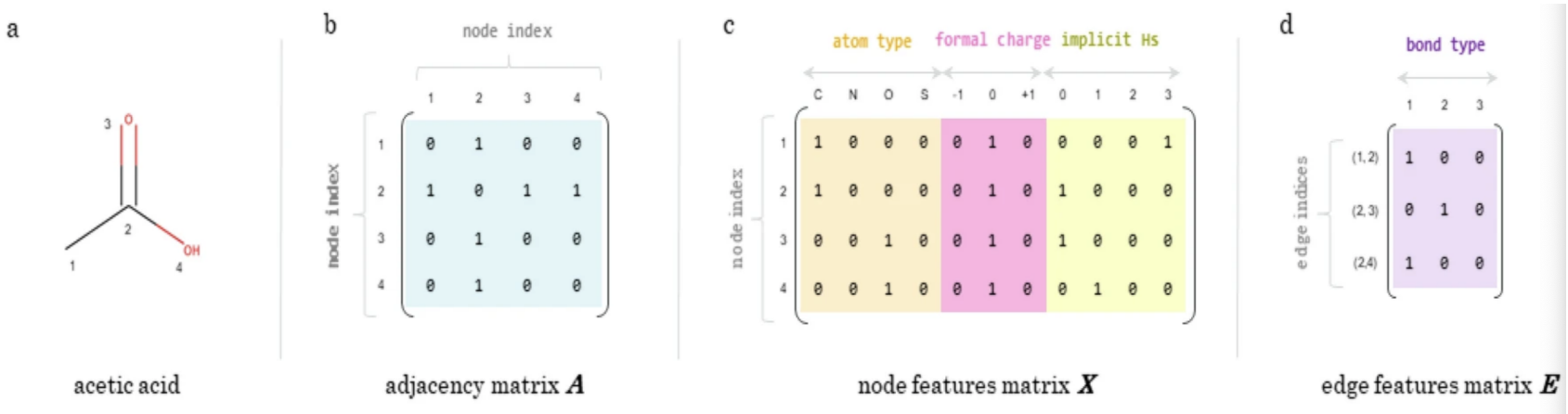


Network of Neurons



PPI

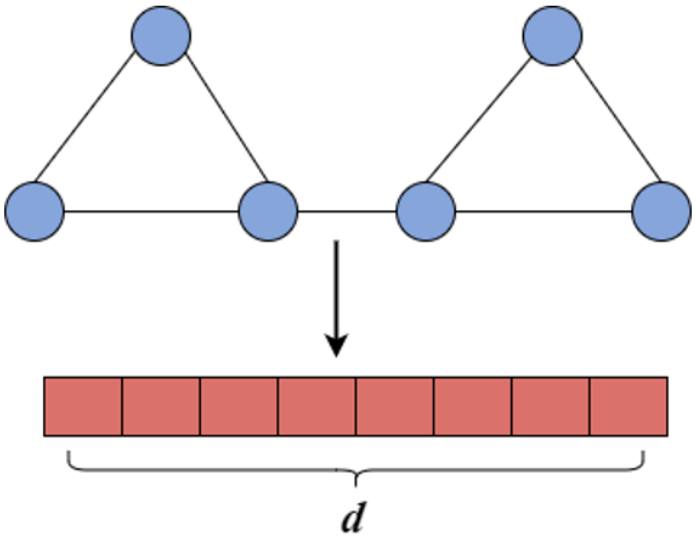
Example



acetic acid | adjacency matrix $\boldsymbol{A}$ | node features matrix $\boldsymbol{X}$ | edge features matrix $\boldsymbol{E}$

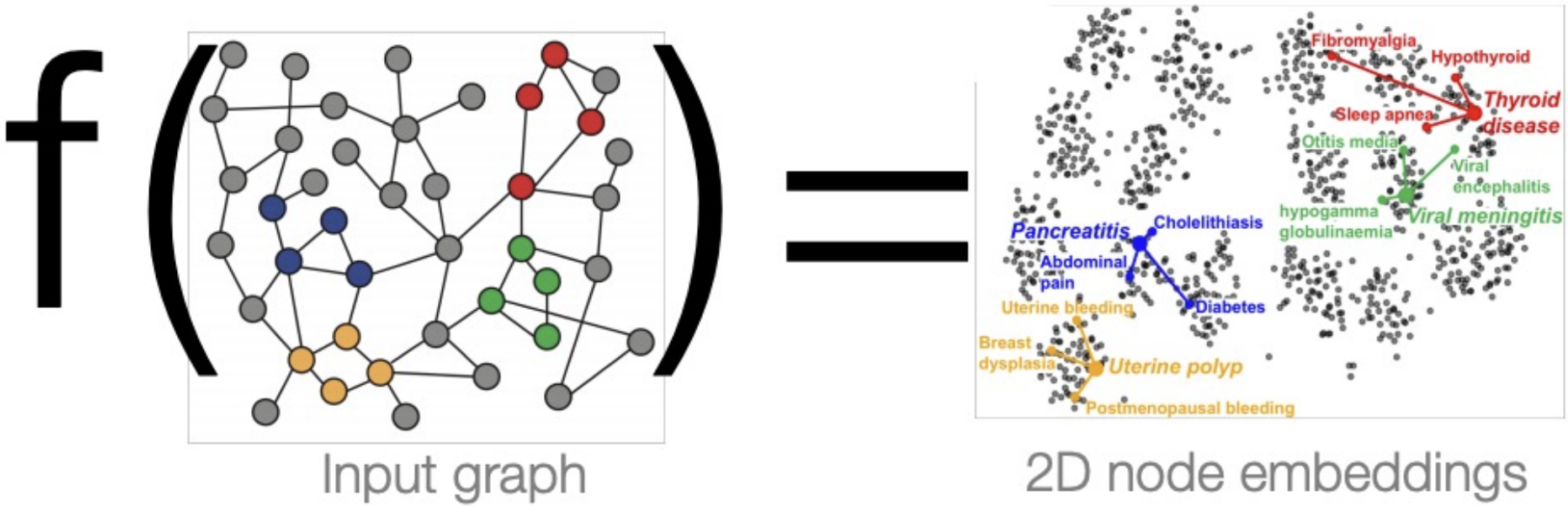
David et al., (2020)

Graph Representation Problem

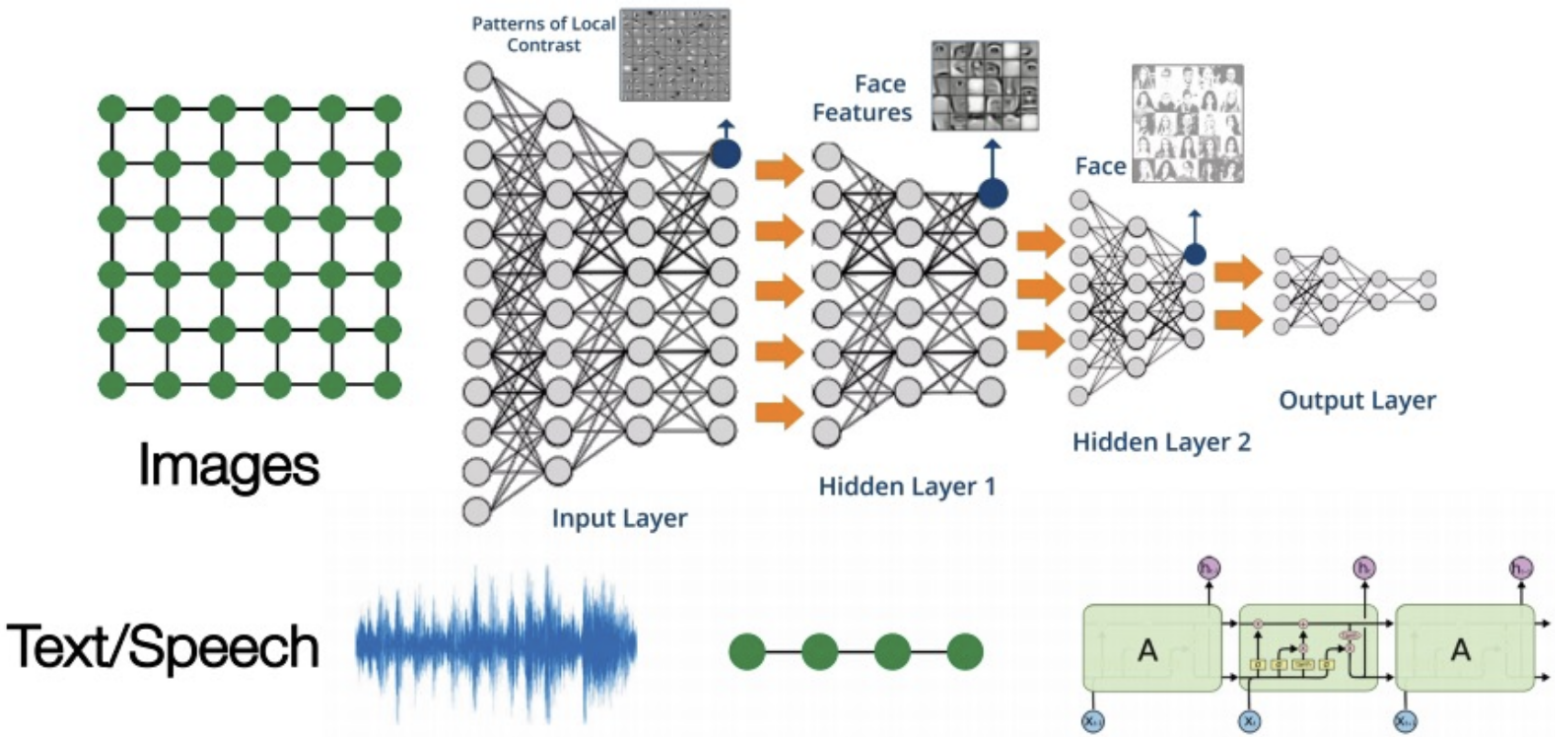


- Given a graph representing real-world entities and their relationships
- The goal is to design a function $f : G \rightarrow \mathbb{R}^d$
- To perform any downstream Machine Learning task

Learning a Mapping Function

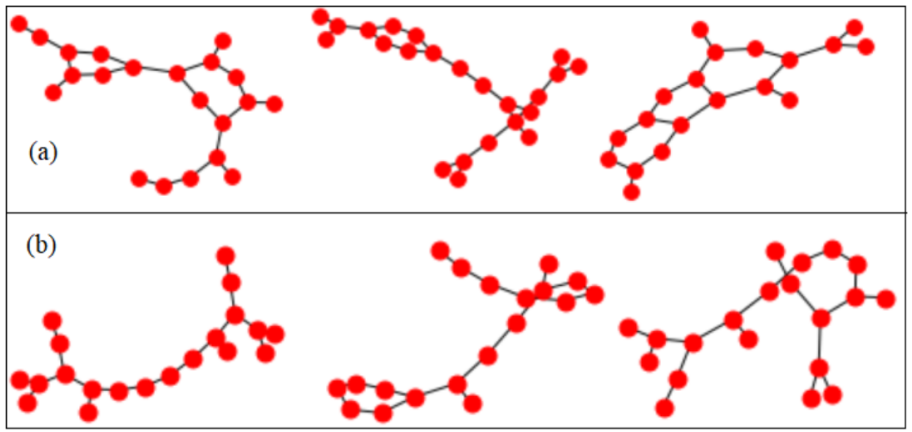


Deep Learning Methods: Fixed Input Size

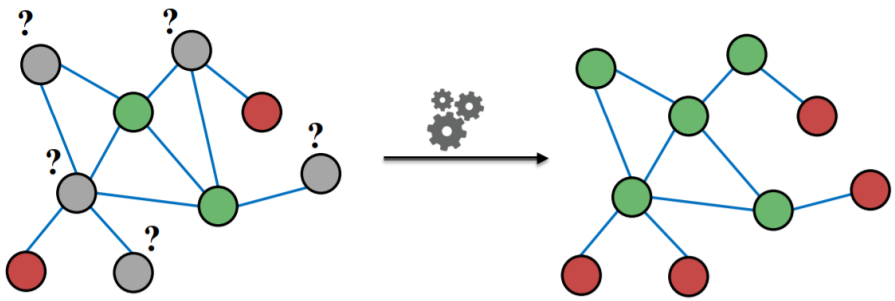


Machine Learning on Graphs

Graph Classification

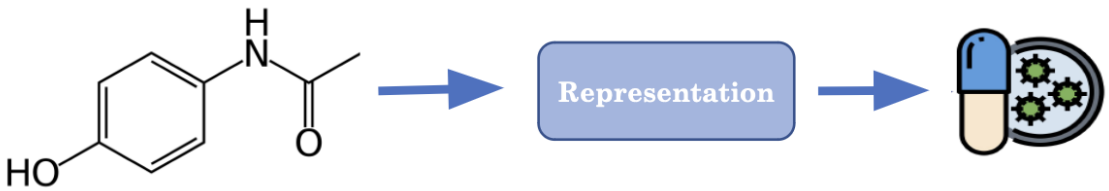


Node Classification



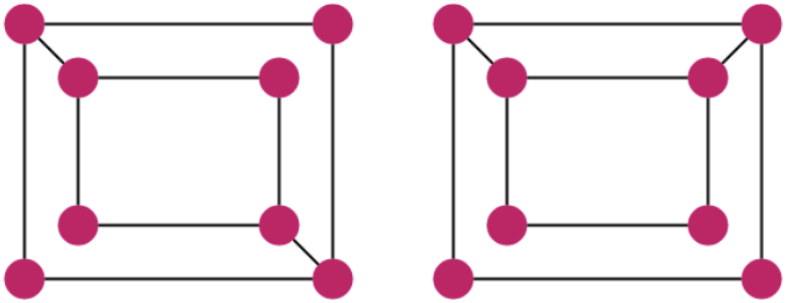
Applications

- Predict whether molecule is a potent drug?
- Toxicity Prediction (whether the molecule is toxic?)
- Property identification (HIV virus replication, HOMO LUMO gap etc)
- Antibiotic discovery
- Drug discovery and development



Applications

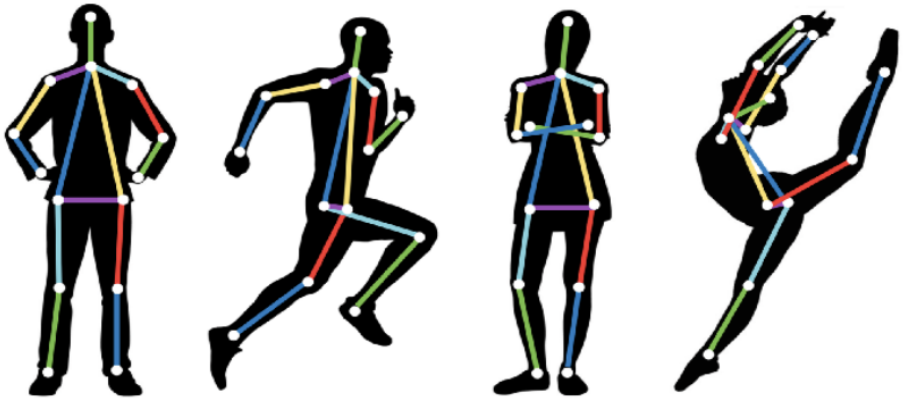
Inexact graph matching



Document classification

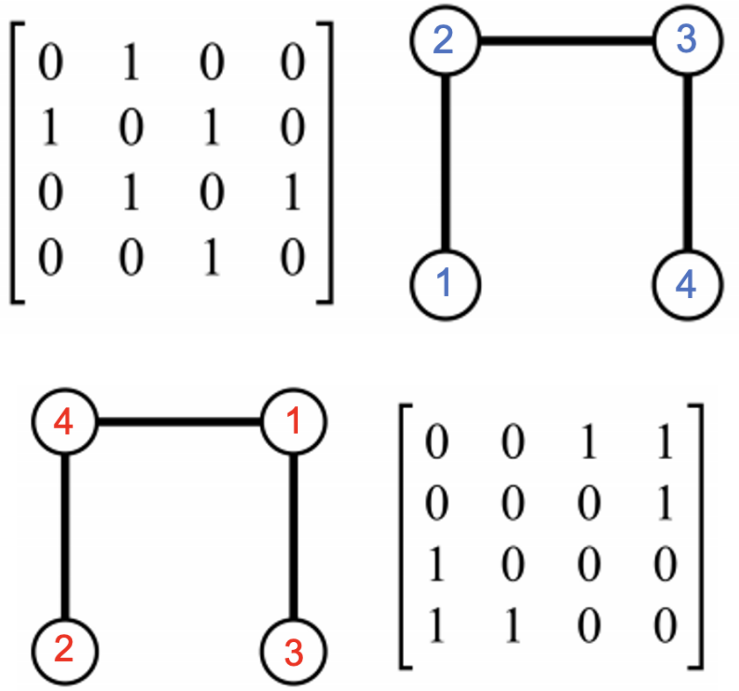
Traffic forecasting

Pose estimation



Challenges

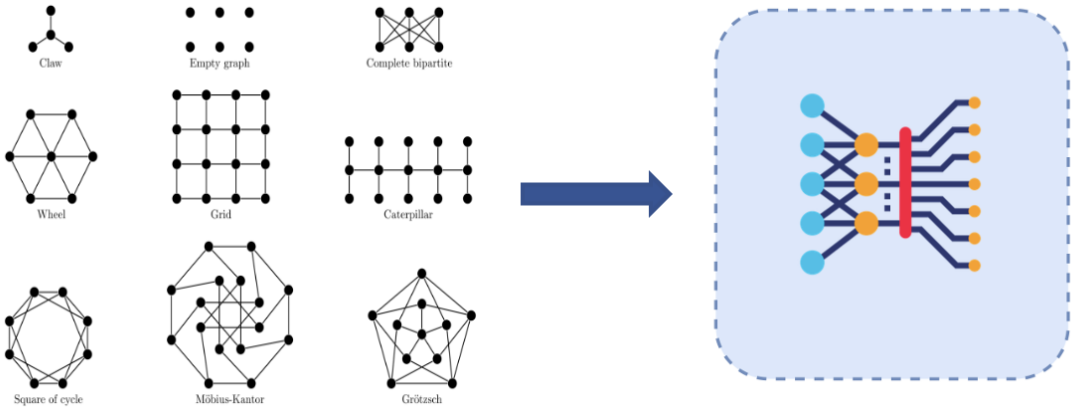
- Permutation invariance



- $O(n)$ number of parameters

Challenges

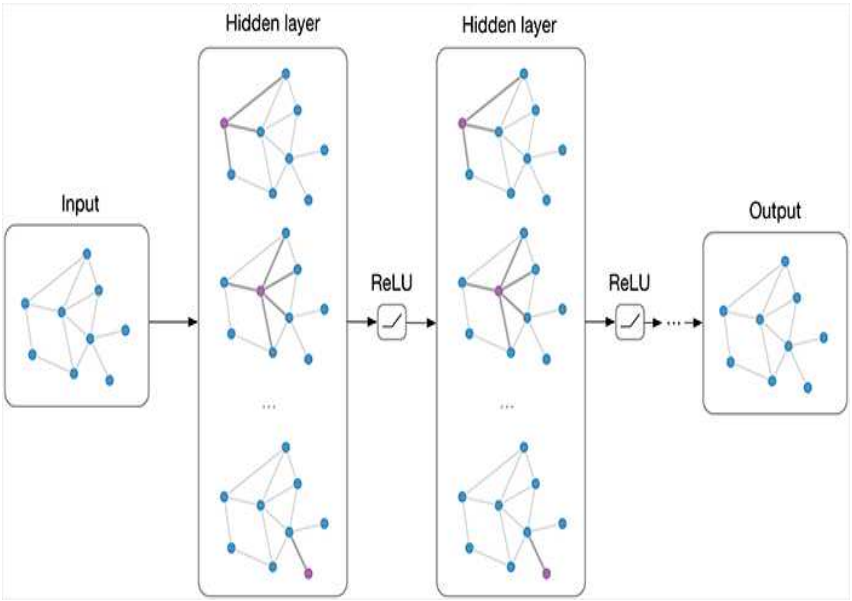
- Size invariance



- Computational complexity
- Multimodal features
- Complex topological structures (large number of combinations $(2^{\binom{n}{2}})$)

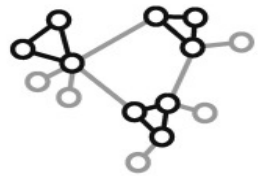
Approaches

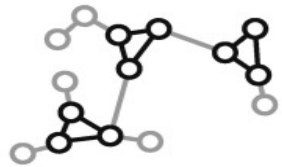
Graph Neural Networks



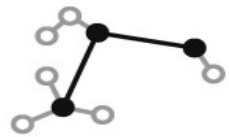
(kipf et al., 2017)

Graph Kernels/descriptors

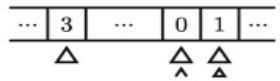


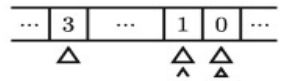


(a) Two graphs of equal order and size. Note that both graphs have the same number of triangles. Relevant sub-graphs have been highlighted.



(b) The higher-order graphs formed after compressing all triangles. Compressed nodes have been represented as black, and relevant portions of the graph have been highlighted.

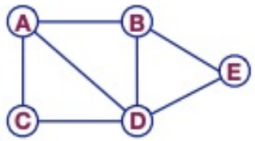




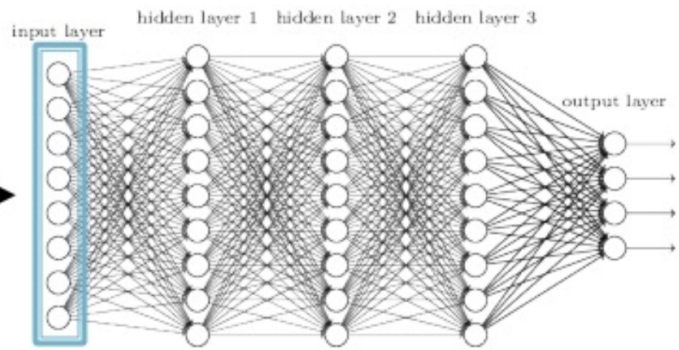
(c) Relevant values from the graphs' HOSD descriptors. From left to right, the values represent the number of triangles in the original graph, the number of three-paths with three compressed nodes in the triangle-compressed graph, and the number of triangles with three compressed nodes in the triangle-compressed graph.

(Ahmed et al., 2021)

Naive Approach



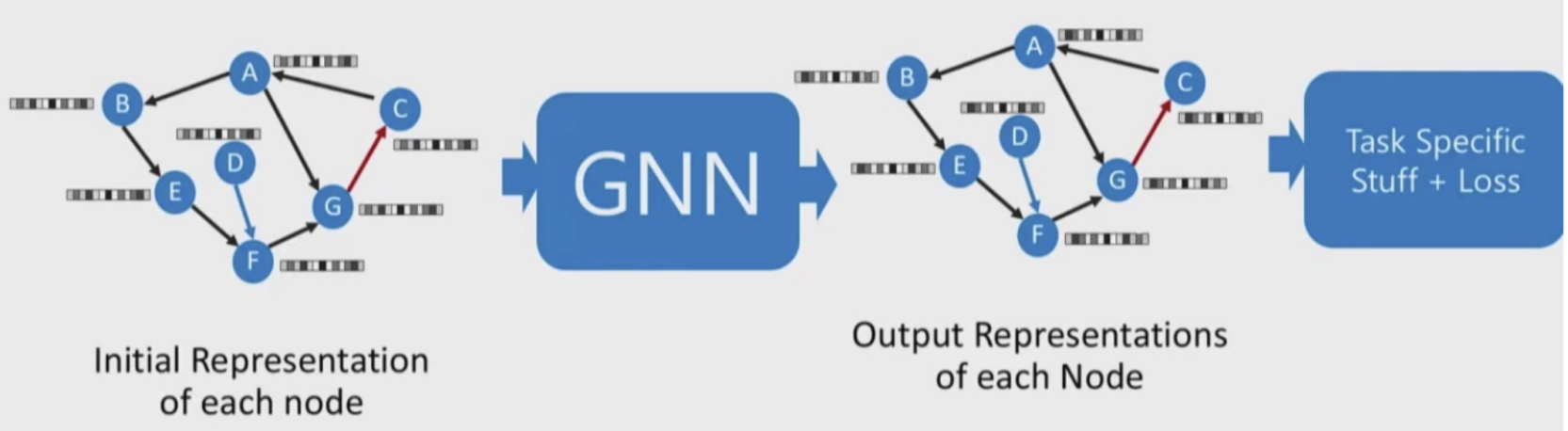
	A	B	C	D	E	Feat	
A	0	1	1	1	0	1	0
B	1	0	0	1	1	0	0
C	1	0	0	1	0	0	1
D	1	1	1	0	1	1	1
E	0	1	0	1	0	1	0



Feed adjacency matrix with node features to a neural network

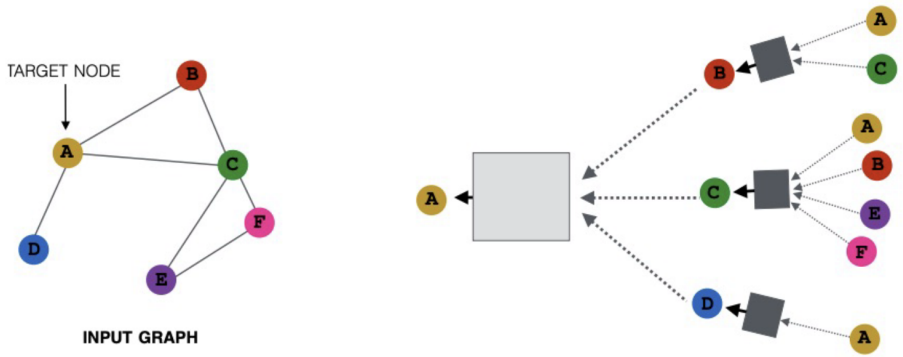
Problems?

General Framework



Graph Neural Networks (GNNs)

- End-to-end deep learning framework
- nodes/graph representations are learned using message passing mechanism



(hamilton et al.,2017)

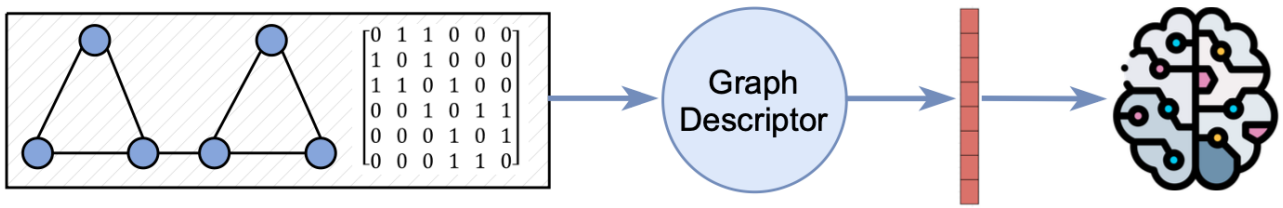
$$a_v^k = f^k\left(h_u^{(k-1)} : u \in \mathcal{N}(v)\right)$$
$$h_v^k = g^k\left(h_v^{(k-1)}, a_v^k\right)$$

Graph-level predictions:

$$h_G = \text{READOUT}\left(h_v^k \quad | v \in G\right)$$

Graph Descriptors

- Use graph-theoretic measures to encode graph structures in one shot
- Unlike GNNs, they do not involve any end-to-end learning
- Seek to encode graph's structure based on both node and graph-level properties
- Node-level properties include node degrees, average clustering coefficient, and number of edges in egonetwork [1]
- The graph-level properties include graph spectrum, and aggregated node-level information [2]



1. Berlinger et al., Network similarity via multiple social theories, ICASNAM2013
2. Verma, et al., Hunt for the unique, stable, sparse and fast feature learning on graphs, NeurIPS 2018

Pros and Cons

Pros:

- Easy to interpretation
- Flexible to apply any classical ML method
- Easy to train
- Provide stable results

Cons:

- Intractable on sufficiently large graphs
- Do not use node/edge features
- Inferior performance

Recent Work

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.

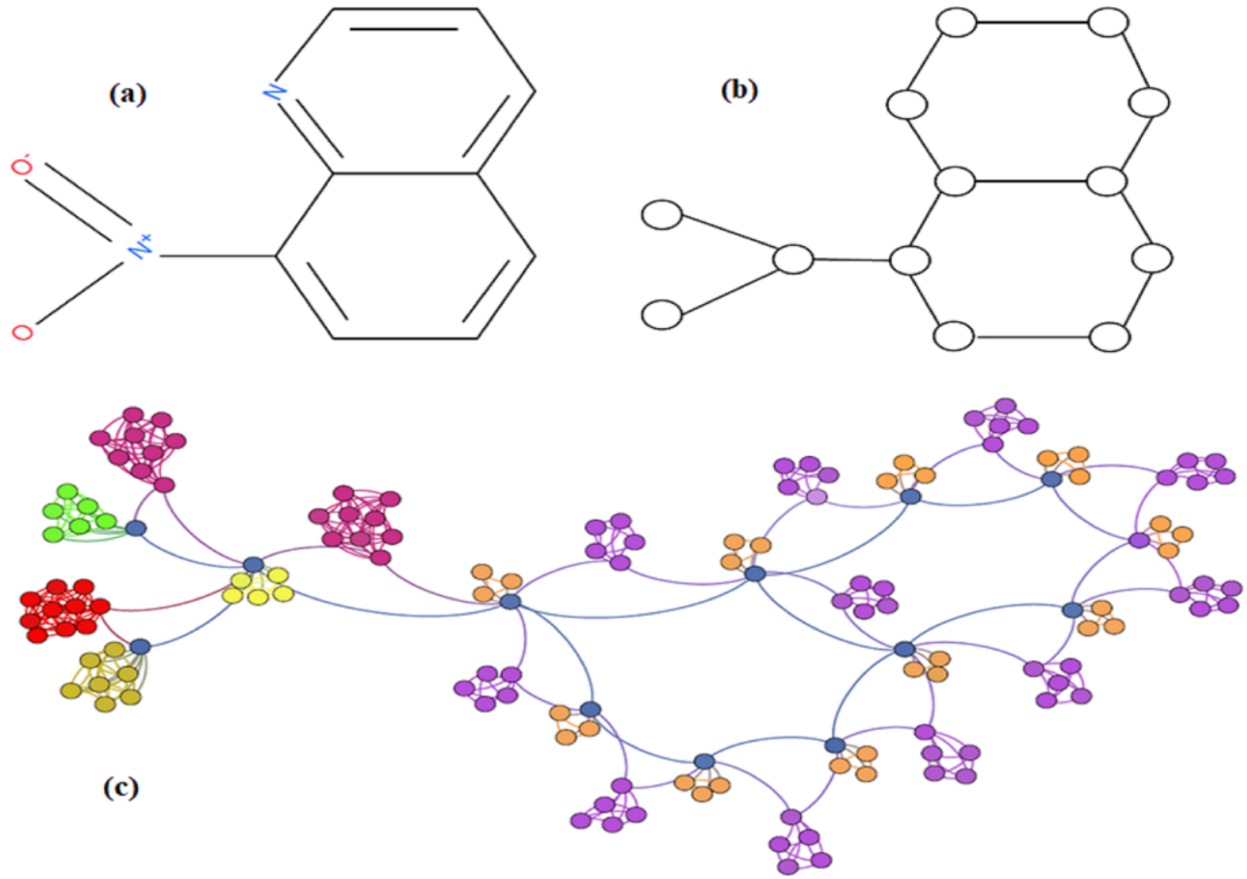
Motivation

Challenge: Most of the descriptors can't handle node/edge features
Research question: How can we enable graph descriptors on attributed graphs?

Key Idea: Augmenting special subgraphs to preserve attribute information in a simple graph

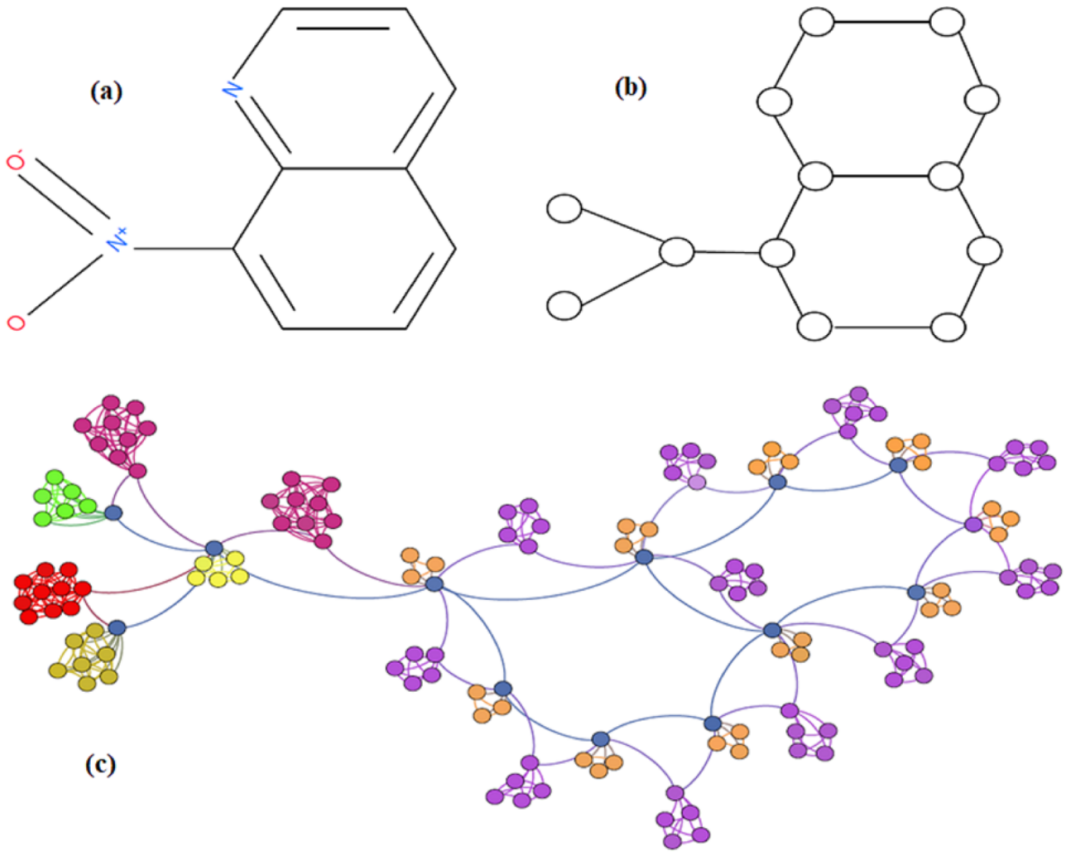
Graph Augmentation

Key Idea: Augmenting special subgraphs to preserve attribute information in a simple graph



Graph Augmentation

- recover the original graph?
- expressive enough?
- small magnitude?



Design Properties

- Let $L = (V, E, \delta, \theta)$ be an attributed graph
- $\bar{L} = (V, E)$, simple graph
- $G = (V', E')$, transformed graph

We propose the following properties that f must fulfill

- f must be bijective; there must exist a function $f^{-1}(G_i) = L_i$, whenever $f(L_i) = G_i$
- On average, the increase in magnitude should be minimized;

$$\frac{1}{|\mathcal{L}|} \sum_{L_i \in \mathcal{L}} mag(G_i) - mag(\bar{L_i}) \quad \text{where} \quad mag = |V + E|$$

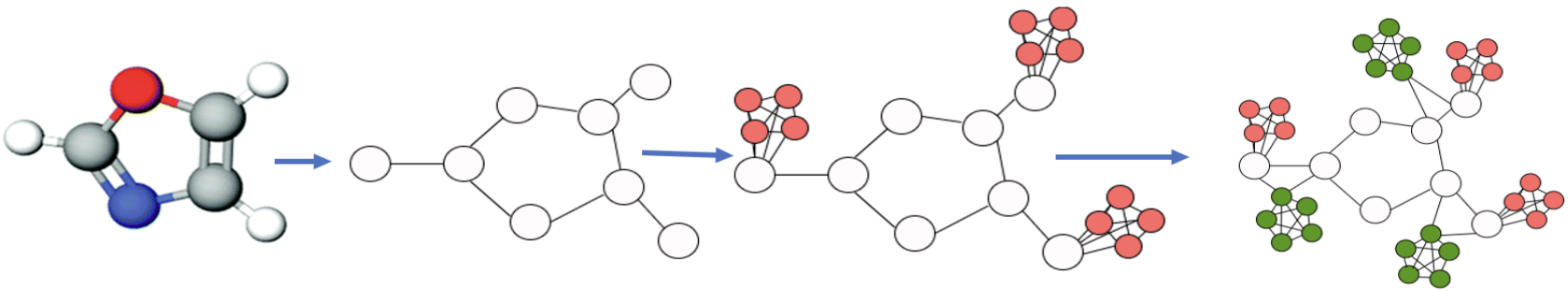
- The resultant graph must be expressive enough for a classification model to learn relevant features

Attributed Graph Augmentation Procedure

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



Analysis

Lemma

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. We order $\mathcal{G}'$ with respect to graphs' magnitudes and keep the first $|\mathcal{L}|$. Then the lower bound of the objective function is:

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

$k-$magnitude $H-$free Discovery(k-HFD)

- Ensuring property 1 requires appending subgraphs that are not exist in the attributed graph
- Computationally challenging!

Problem (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 is no such H then return NULL, else return H.

Cases:

- when $k = 2$, always return NULL unless G is a complete graph.
- when $k = 3$, we have two possibilities: three nodes with no edge or two nodes with a single edge. Such cases can be easily checked.
- in general, we show that k-HFD is NP-hard.

$k-$HFD is NP-hard

Theorem

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

We reduce the famous k-Clique problem to k-HFD problem. **k-Clique problem:** given a graph G, does there exist a subgraph on k nodes where each pair of nodes is connected by an edge?

Proof.

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 $k = \hat{k} + \binom{\hat{k}}{2}$.

- If $k-$HFD returns a graph H on $\hat{k}$ nodes, then we return NO answer to the clique problem - because H must be a clique, and G does not contain clique of the given number of nodes

- If $k-$HFD returns NULL, or $|H| > \hat{k}$ then $\hat{G}$ is not clique free and we return YES.

□

Theorem Results

- The Theorem implies that there is no polynomial time solution for the problem of finding and augmenting attributed graphs
- However, we assert that domain knowledge of bioinformatics can be used to design efficient solutions
- For instance, it is plausible to use rings and star graphs because they most likely exist in molecules
- We observed that bioinformatics graphs are typically sparse and lack cliques of medium size.

Case Study: Toxicity Prediction

- Chemical toxicology prediction is an important task in drug discovery
- We consider Toxicity prediction challenge [1] as a case study
- We consider graph classification setting for the experiments

Graph Classification:

Problem

Given a set of graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$, with their corresponding binary labels, $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$, the goal is to learn a representation vector $\boldsymbol{h_G}$ that helps predict the label for an unseen graph, $y_G = f(\boldsymbol{h_G})$.

[1]-http://bioinf.jku.at/research/DeepTox/tox21.html

Datasets

Dataset	$\lvert G \rvert$	avg.$\lvert V(G) \rvert$	avg.$\lvert E(G) \rvert$	avg. $\lvert V(\mathcal{G}) \rvert$	avg.$\lvert E(\mathcal{G}) \rvert$
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

Dataset Preprocessing

- Obtain each dataset in a SMILE format and preprocess it using RDKit in Python
- Construct two versions of each dataset: G and $\mathcal{G}$
  - Simple graph (ignore attribute information and keep the simple graphs)
  - Augmented graphs using the proposed graph augmentation approach
- Consider four atom attributes: aromacity, positive and negative charge, and non-metallicity. We choose cliques of size 4, 6, 7 and 8
- Consider four bond types: aromatic, single, double and triple for which we choose cliques of size 5, 9, 10 and 11.

Baselines

- We consider seven different graph kernels and graph descriptors as baselines
- **FGSD:** a spectral graph descriptor that uses spectral distances among pair of nodes
- **NetLSD:** uses the idea of heat kernels
- **HOSD:** uses subgraphs' counting with compression
- **NetSIMILE**: uses statistical node and graph-level properties
- **Shortest Path Kernel:** based on pair-wise distances
- **WL kernel:** based on the idea of color refinement based on nodes' neighborhood

Results

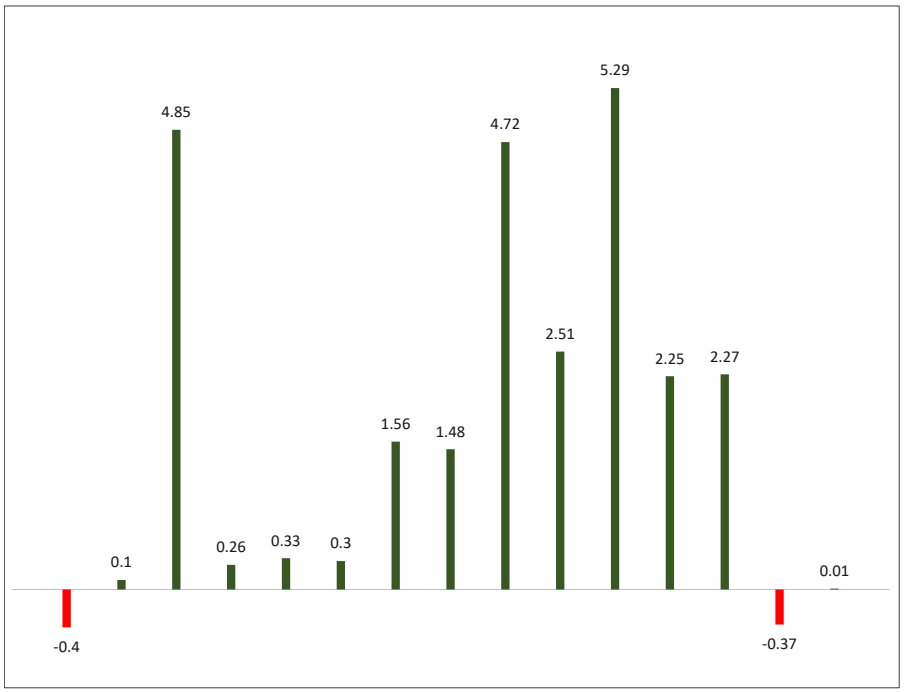


Figure: Shortest Path Kernels

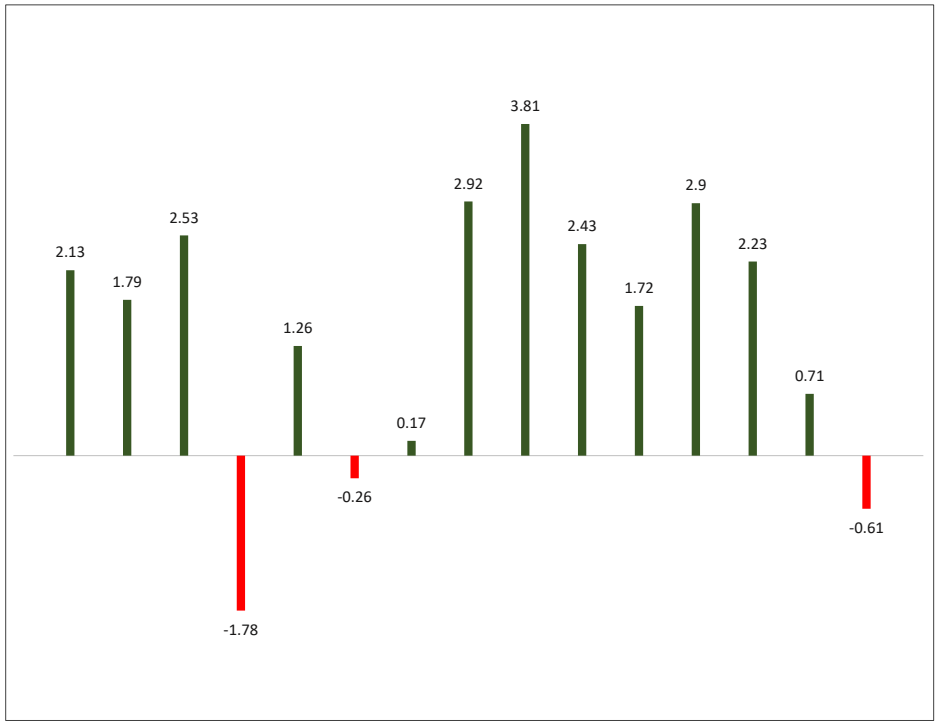


Figure: FGSD

Results continued...

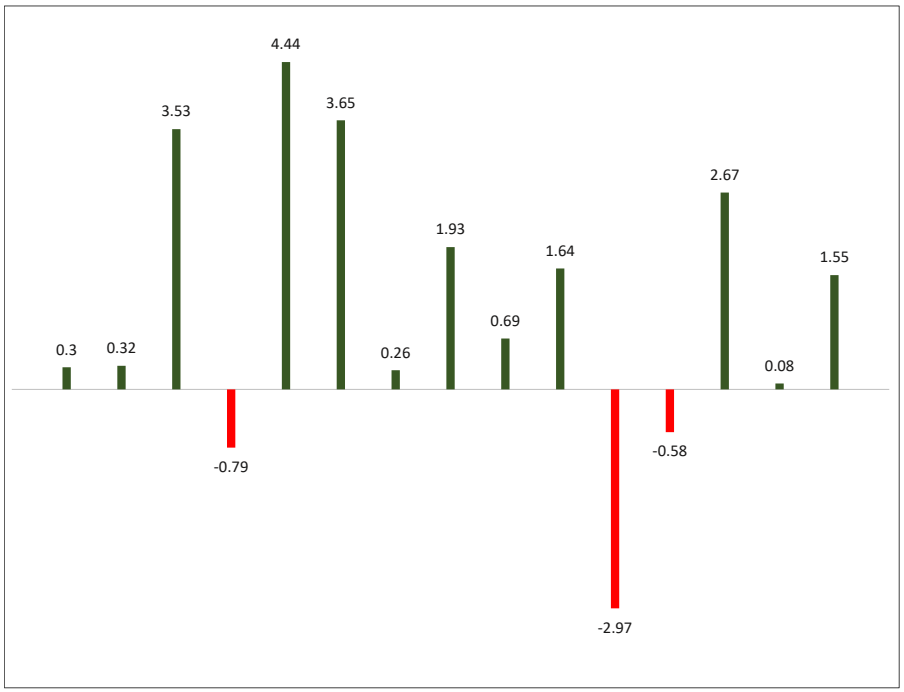


Figure: WL Kernel

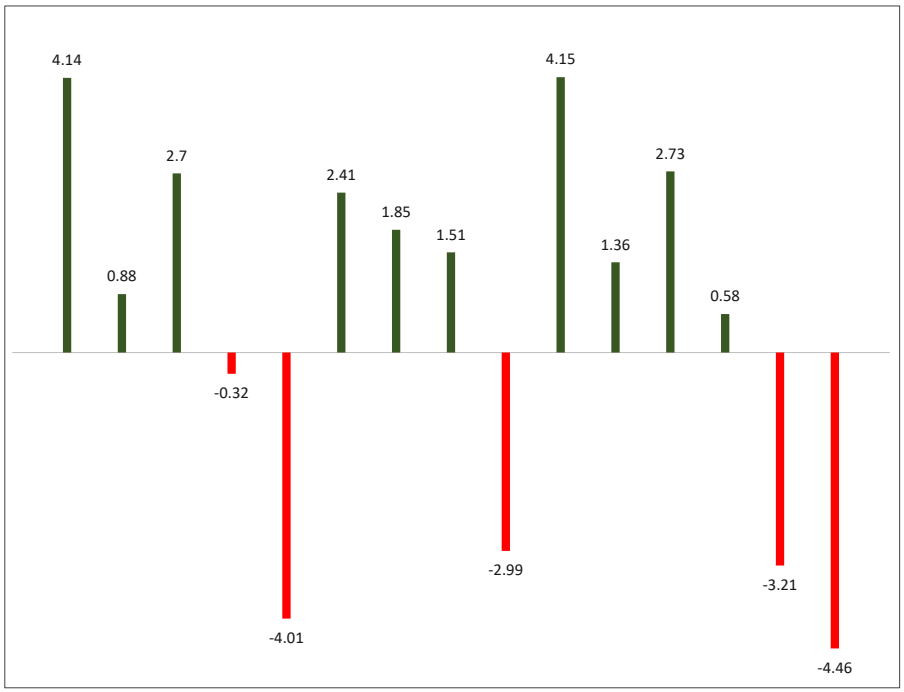


Figure: NetLSD

Results continued...

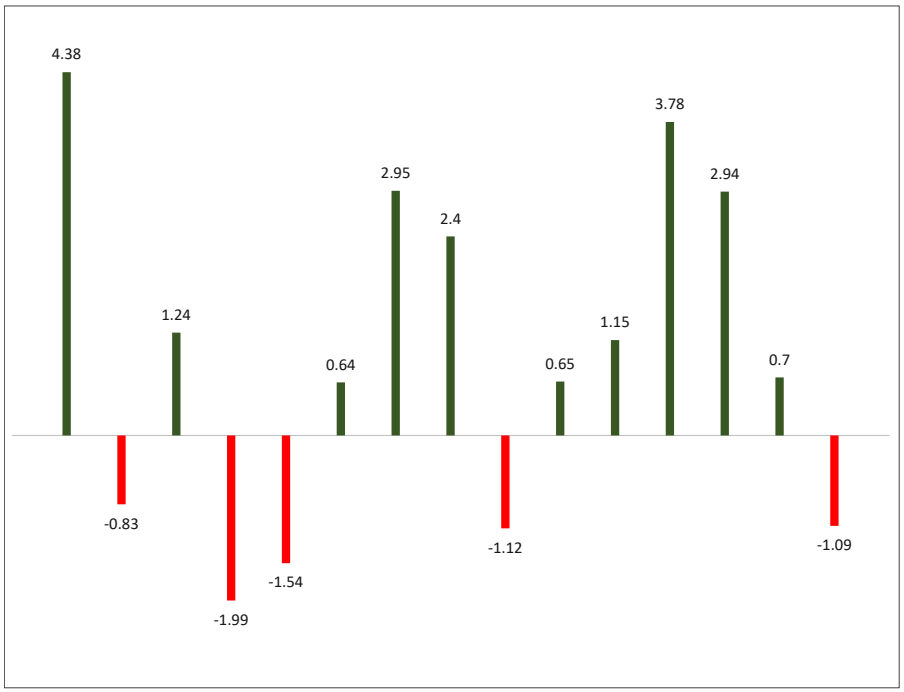


Figure: NetSimile

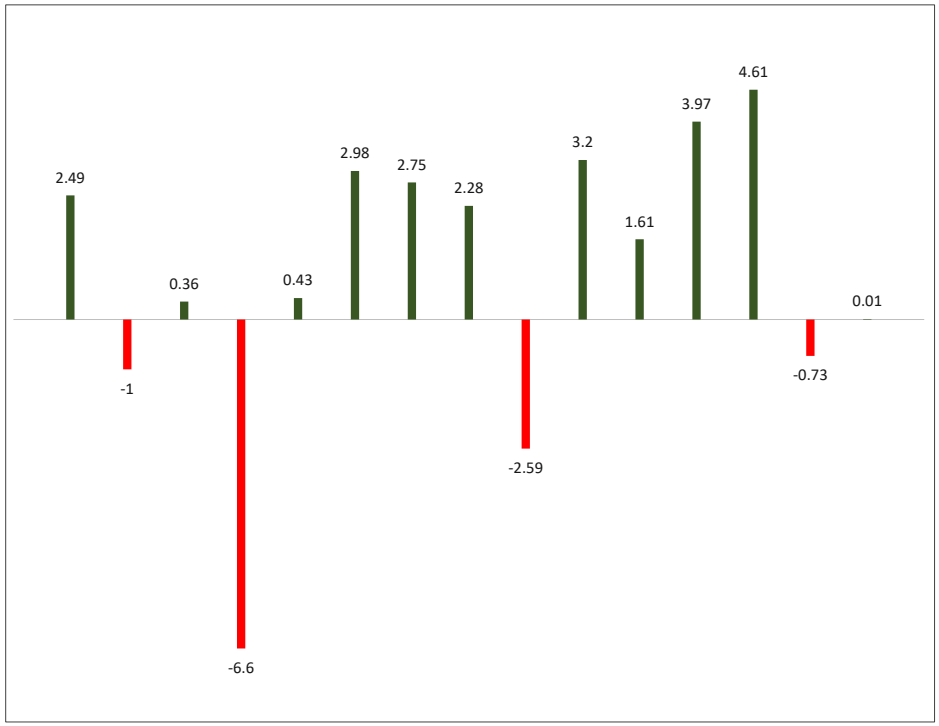


Figure: HOSD

Results

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

Table: Comparison of classification accuracies of seven graph embedding methods on 15 bioinformatics datasets.

Conclusion

- Introduced a new augmentation framework
- Discovered a new problem and provide a proof for NP-hardness
- The results support our theory
- We also perform the same experiments for augmenting lollipop graphs and retrieved almost the same results

Implementation:
The source code of the proposed method is made publicly available on both CodeOcean and GitHub.

Thank you for your attention!
Any questions?