# Study of deep neural networks on graph data in a generative learning regime

by

Chao Jiang

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 6, 2022

Keywords: Graph Neural Network, Adversarial Generative Network, Poisoning attack

## Abstract

Graph-formatted data is ubiquitous among different domains from social networks and academic citation networks to drug-target interactions and others. Graph neural networks (GNNs) have achieved outstanding performance in applying node classification, link prediction, and node clustering, etc. However, there are two common questions asked by researchers.

First, how to get a considerable amount of labeled high quality data? Data quality is crucial in training deep neural network models. However, most of the current works in this area have focused on improving a model's performance with the assumption that the preprocessed data are clean. Our first result is about improving data quality by removing noise information. Here we build a real knowledge graph from data sets LitCovid and Pubtator. The multiple types of biomedical associations of the real knowledge graphs, including the COVID-19-related ones, are based upon the co-occurring biomedical entities retrieved from recent literature. However, the applications derived from these raw graphs (e.g., association predictions amongst genes, drugs, and diseases) have a high probability of false-positive predictions as the co-occurrences in literature do not always mean a true biomedical association between two entities. We proposed a framework that utilized generative-based deep neural networks to generate a graph that can distinguish the unknown associations in the raw training graph. Two Generative Adversarial Network models, NetGAN and CELL, were adopted for the edge classification (i.e., link prediction), leveraging unlabeled link information based on the real knowledge graph. The performance of link prediction, especially in the extreme case of training data versus test data at a ratio of 1:9, demonstrated that the promised method still achieved favorable results (AUCROC $> 0.8$ for synthetic and 0.7 for real dataset) despite the limited amount of testing data available.

Second, what is the decision-making process of GNNs as it often remains a black box? In addition, many of the models are vulnerable to adversarial attack. Our second result focuses on the study of the robustness of GNNs. Recent studies revealed that the GNNs are vulnerable to adversarial attacks, where feeding GNNs with poisoned data at training time can lead them to yield devastative test accuracy. However, the prior studies mainly posit that the adversaries can access freely and manipulate the original graph, while obtaining such access could be too costly in practice. To fill this gap, we propose a novel attacking paradigm, named Generative Adversarial Fake Node Camouflaging(GAFNC), with its crux laying in crafting a set of fake nodes in a generative-adversarial regime. These nodes carry camouflaged malicious features and can poison the victim GNN by passing their harmful messages to the original graph via learned topological structures. These messages can maximize the devastation of classification accuracy (*i.e.*, global attack) or enforce the victim GNN to misclassify a targeted node set into prescribed classes (*i.e.*, target attack).

## Acknowledgments

I would like first to thank my advisor, Dr. Richard Chapman for his patient guidance and support in my Ph.D. study. Not only did he gave me plenty of freedom, he has also supported in choosing my own areas and topics. He also offered me plenty of cooperation opportunities to develop as an independent researcher. Without his continued support and encouragement, I could not finish this dissertation work.

I would also thank all my committee members, Dr. Cheryl Seals, Dr. Shubhra Karmaker, Dr. Anh Nguyen, and my university reader, Dr. Shiwen Mao for offering valuable advice and assistance towards my dissertation work.

I would also like to thank to Dr. Nansu Zong, at Mayo Clinic, who provided the mentorship on the work of "Deep Denoising of Raw Biomedical Knowledge Graph from COVID-19 Literature, LitCovid and Pubtator". Without his insightful suggestions and guidance on the project, I would not able to complete my first work (Chapter 3) of this dissertation.

I would finally like to thank Dr. Yi He from Old Dominion University for his generous and patient guidance. I was able to have an invaluable and productive remote internship experience.

I extend my special thanks to my friends Yanan Wu, Xiaopu Peng, Chen Jiang, Sicheng Li, Enkai Bi, Bo Hui, Jing Zhao, Wenwen Ye, Xuewei Ding, and Jiachen Liu, for their support and company during my time at Auburn. I would also like to extend my special thanks to my parents, my girlfriend Kathryn Aw, and my family for their endless love and encouragement throughout my life.

Table of Contents

List of Figures

List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

With the explosion of the amount of available data and computation power, Artificial Intelligence has achieved tremendous progress. Examples include image-classification [1], generation [2] and objective detection [3] in computer vision; Chat bot [4] for answering customer questions, speech recognition [5] in virtual assistants and Grammarly [6] which simplifies writing by conducting grammar checks. As such, researchers came up with two frequent questions unanswered questions.

First, how can we obtain high quality labeled data? Most of the excellent performances of machine learning tasks are formed from a large amount of available high quality labeled data. Sophisticated AI algorithms or models that we see today would not perform well without clean and quality data. Data has been increasingly important in our development of AI-models. As Dr. Andrew Ng states ”Data is food for AI”, and true to his words, attention has shifted from academic and industrial researchers using model-centric AI to data-centric AI. Regarding the shortage of training data , especially for image data [7], we would first need to create the generative models for generating more data. One exmaple is the Generative adversarial networks [8] (GAN) as it is widely used in data augmentation [9] and can generate realistic or specific style of images [10]. GAN also has been used for generating text data [11] and even graph data [12]. However, the quality of labeling the training data still plays a key in supervised learning tasks. Startup company Scale AI which focuses on providing high quality training data for leading machine learning teams has raised a total of 602.6 million dollars in 2021. Within this dissertation, we explore the capability of applying GAN for generating similar graphs and used those graphs for filtering or removing noise or invalid links in our raw-built biomedical knowledge graph. This is the first study that uses GAN to denoise a biomedical knowledge graph to the best of my knowledge.

Second, what is the deep neural network decision process? Many works have explained the capability of convolution networks [13] in computer vision. However, it's more complex and not easy to interpret the result of a neural network as the graph are represented as vectors in the model. Here, we will focus our studies on graph data instead of regular structured data such as images and sequences. Graph data provides a universal representation of data. In other words, various data are presented as graphs for example, social networks, protein-protein interaction networks, knowledge graphs, etc. It becomes more challenging when dealing with graph data, as there are different types of edges among graphs and each node has different dimensional features. In addition, many real world problems can be modeled as a set of computational tasks on graphs. Examples include anomalous nodes detection, link prediction for knowledge graph completion, community detection through graph clustering , and so on. As graph data becomes increasingly popular, the number of deep neural networks designed for representation learning on graphs has increased, such as network embedding [14, 15, 16] and graph convolution networks [17] and graph attention [18], etc. Here, we want to explore the explainable and interpretable of the decision making of GNN by analyzing its robustness. We will try to propose an attack algorithm on a graph neural network (GNN) to explore the robustness of GNN. Compared with previous attacks on GNN, our plan of attack is to insert poisoned camouflaged nodes instead of manipulating the origin graph directly, and prove an attack success ratio in node classification tasks.

## 1.2 Thesis overview

The outline of this dissertation is as shown in Figure 1.1. As stated in our motivation, we first want to explore the capability of automatically labeling graph data, and second, the robustness of GNN on graph. The first result applies GAN for graph link prediction through a case study on the denoising of a biomedical knowledge graph. The second work is about a data poisoning attack on graph neural networks.

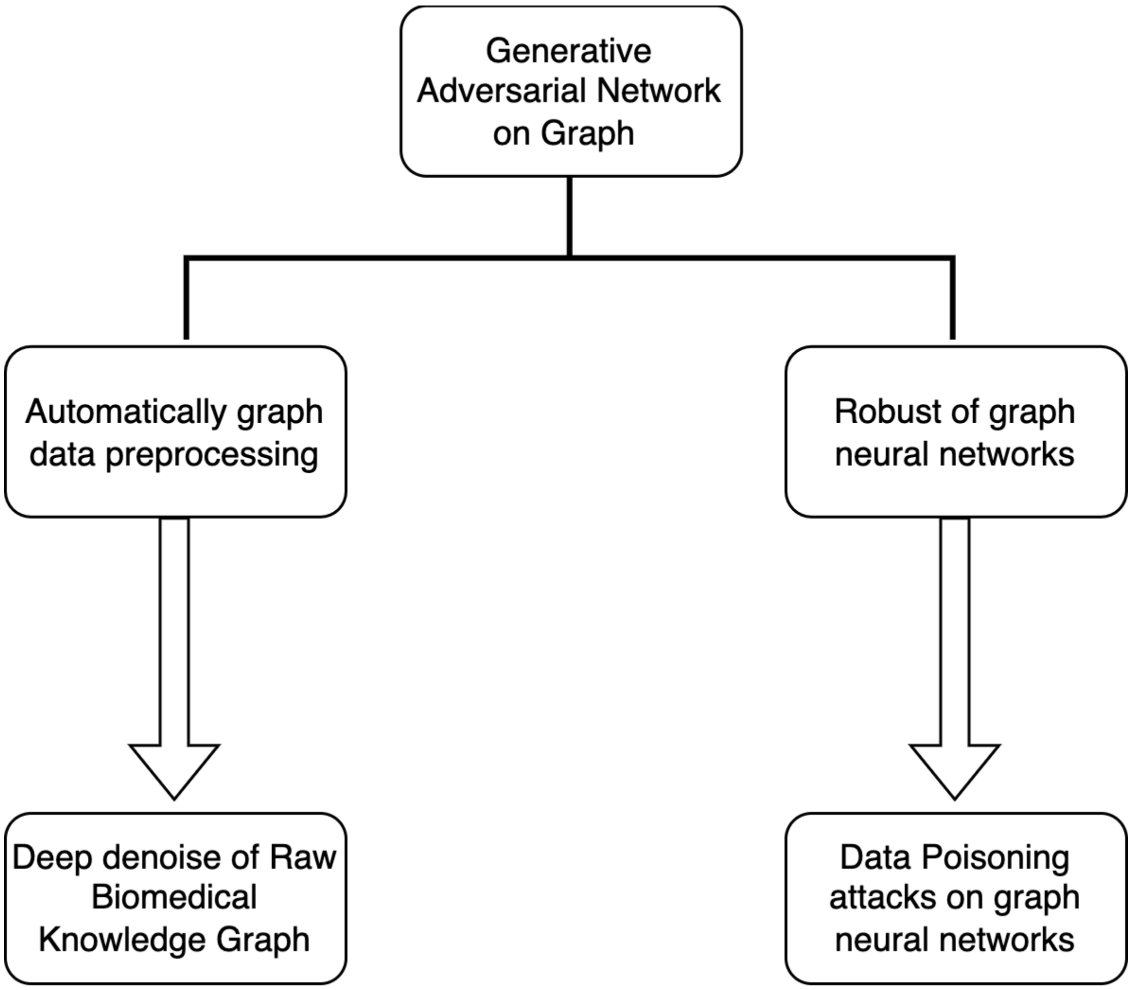

Generative Adversarial Network on Graph
Automatically graph data preprocessing
Robust of graph neural networks
Deep denoise of Raw Biomedical Knowledge Graph
Data Poisoning attacks on graph neural networks

Figure 1.1: Thesis Overview.

# Chapter 2

# Background

## 2.1 Graph representation

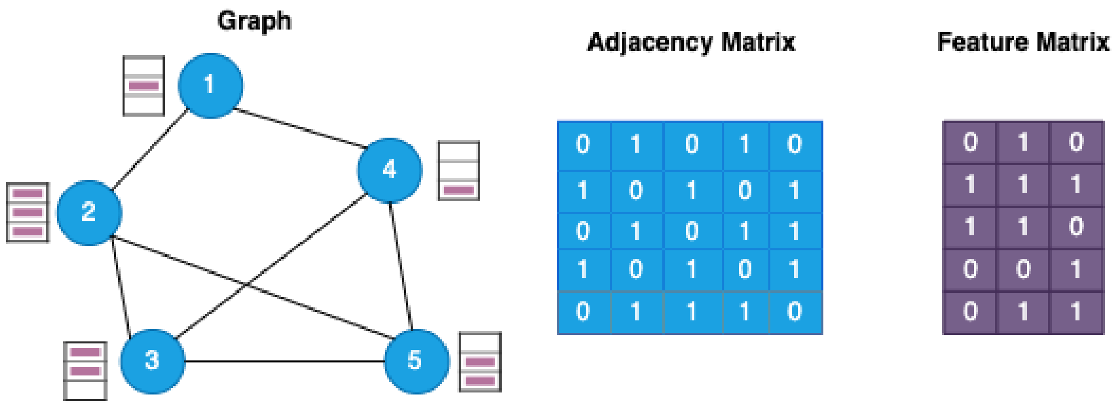


Figure 2.1: Graph representation

Most graphs can be represented by an adjacency matrix as per Figure 2.1, where there is a total of five nodes and seven edges. Each node has a 3-dimensional binary feature. Based on the direction of the edges, the graph can be categorized as a directed graph or an undirected graph. Based on the value of the edge, the graph can divide into a weighted graph or an unweighted graph. In addition, based on the type and number of edges or nodes, there are heterogeneous graphs, bipartite graphs, multi-dimensional graphs, signed graphs, hypergraphs, and dynamic graphs. Due to its universal and natural representation of the complex world, many deep learning works are proposed based on graph data.

In many real life cases, graphs consist of a large number of nodes with very sparse edges, and the degree of nodes obeys power-law distribution as per Figure 2.2 [19]. Instead of representing a graph as an adjacency matrix, we explore the work NetGAN [20] which uses random walks sampled from the graph as a representation. The random walks can

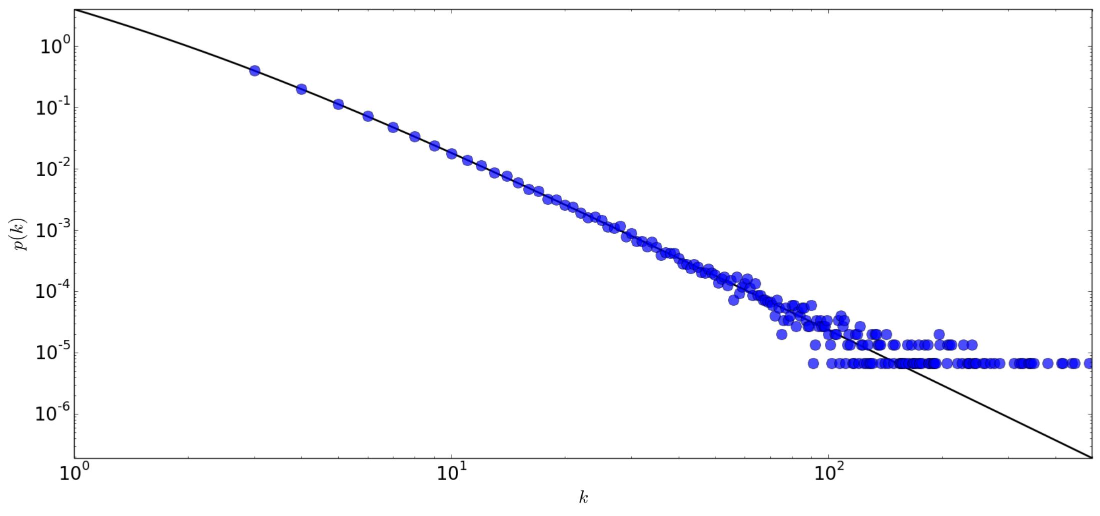
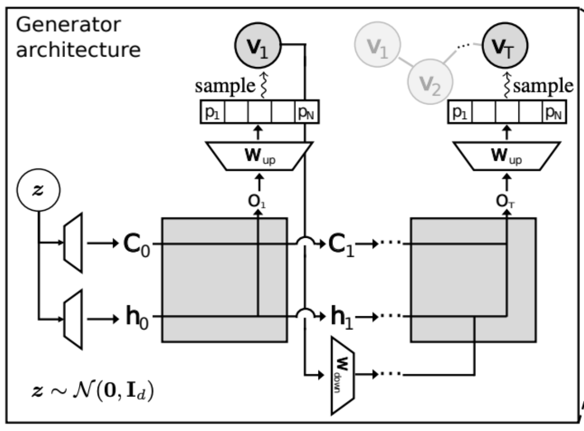


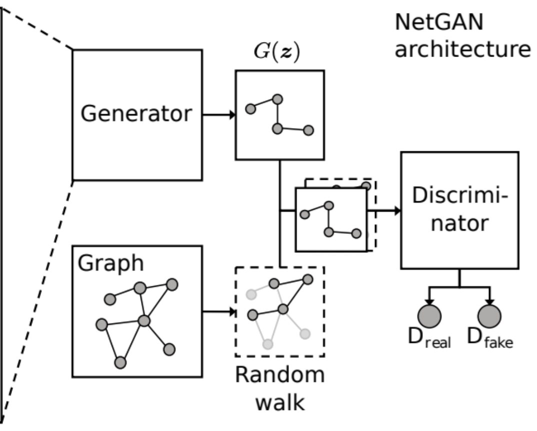
Figure 2.2: Power law degree distribution

be stored more efficiently, considering that the adjacent matrix has a lot of zero values. Also, in a deep learning framework, we need fewer learnable parameters to fit our model thus avoiding overfitting when training deep neural networks. Here, given an unweighted, undirected, and connected graph $G = (V, E)$ with nodes $V = [N] = 1, , N$, edges $E \subseteq V \times V$, adjacency matrix $A \in \{0, 1\}^{N \times N}$, and degree matrix $D = diag(d) \in R^{N \times N}$, a single random walk of length $T$ is an ordered tuple $R = (v_0, ..., v_T) \in V^{T+1}$. In NetGAN, as shown in Figure 2.3 in work [20]. The generator G was modeled as a sequence process based on Long short-term memory architecture (LSTM)[21] $f_\theta$ parameterized by $\theta$. At each step $t$, given the current memory state $m_t$ that consists of the cell state $C_t$, the hidden state $h_t$, node $v_t$, and the $f_\theta$ output distribution $p_{t+1}$ over the next node $v_{t+1}$ in the form of logits. Then the next node represented as a one-hot vector is sampled from the categorical distribution $v_{t+1} \sim Cat(\sigma(p_{t+1}))$, where $\sigma(\cdot)$ denotes the softmax function. This is proven by the equation 2.1 below.

$$\begin{gathered}(m_{t+1}, p_{t+1}) = f_\theta(m_t, v_t) \\ v_{t+1} \sim Cat(\sigma(p_{t+1}))\end{gathered} \tag{2.1}$$



(a) Generator architecture



(b) NetGAN architecture

Figure 2.3: Overview of Framework in NetGAN

NetGAN is a generative model for graphs mainly used for generating new graph with similar structures and graph statistics (for example, max degree, triangle count, Character path length, etc.) as a given input graph. The generator G tries to learn an implicit probabilistic model for generating random walks of fixed length T. The discriminator D tries to distinguish the generated random walk from a node-sequence sampled from the input graph. Both generator and discriminator use the Long short-term memory architecture (LSTM)[21] and trained with Wasserstein loss [22]. As the original NetGAN did not consider the weight of the link when sampling the random walk, we decided to adopt the sampling method proposed in [23]. Also, when the graph is very sparse and we only want the edge related to some specific nodes, we add a filter in the construction of the training graph to prefer local homomorphism.

## 2.2 Generative Adversarial Networks for Discrete Data

Generative Adversarial Networks (GANs) are used widely in the computer vision area. Typically, GANs consist of two models: a generator (G) and a discriminator (D) shown in Figure 2.4. D and G are commonly implemented by any type of neural network as long as the mapping from the prior noise space to data space is differentiable. The generator tries to capture the distribution of the real data for new data example generation. The discriminator

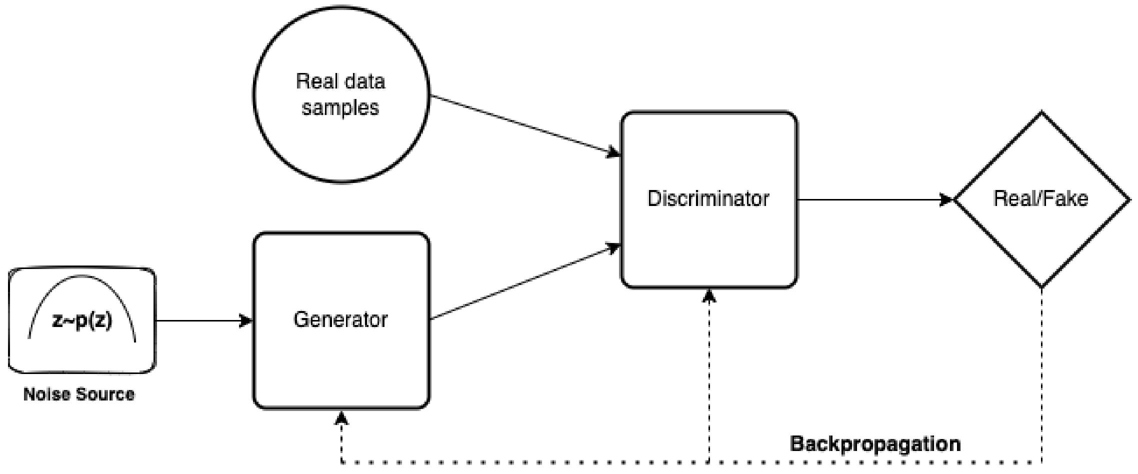


Figure 2.4: The original GAN model

discriminates generated examples from the real examples as accurately as possible. The optimization of GANs is a minimax optimization problem. The optimization terminates at a saddle point whereby the generator will be at the minimum, and the discriminator will be at the maximum. In other words, the optimization goal is to reach Nash equilibrium [24]. Finally, the generator in theory, has the capability to capture the true distribution of real examples.

However, few recent works proposed applies GANs to sequence discrete data generation. e.g. natural language generation [25]. The reason behind this phenomenon is that the generator network in a GAN is designed to be able to adjust the output continuously instead of doing discrete data generation, and the discrete outputs from the generative model make it difficult to pass the gradient update from the discriminator to the generator. The development of recurrent neural networks widely applied to produce sequences of tokens, is used in machine translation [26, 27]. There has been an increase in publishings after [28] proposed that sequence data generation can turn into a sequential decision-making process. These publications open the door for solving this problem by applying reinforcement learning techniques [26, 29, 30].

Although GAN could be used to generate graphs via random walks, it can also remove noise based on the generated graph. The first work in which applied graph generation is

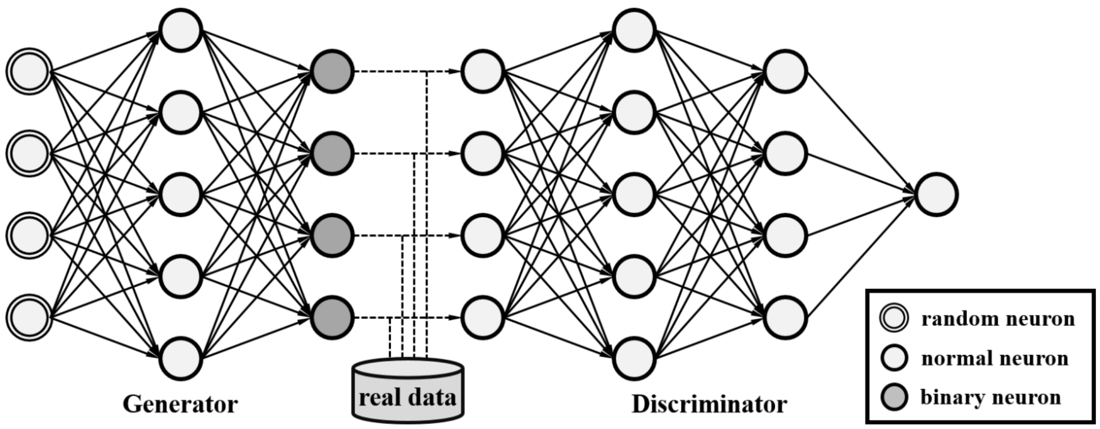


Figure 2.5: Generative Adversarial Networks with Binary Neurons

proposed by [31], takes the graph as an image. Inspired by the idea from the [31], we investigated NetGAN and CELL[32] in link prediction when the labeled edges are limited. And in our second work, we are trying to generate poisoned features for newly added nodes. We applied the GAN with binary neurons in our attack framework, as show in Figure 2.5 which details the framework of binary neurons [33]. Here, we provided the implementation of both *deterministic binary neuron* and *stochastic binary neuron* for flexibility.

## 2.3 Adversarial Attack on Graph Neural Networks

### 2.3.1 Graph Neural Networks (GNNs)

GNNs originated from the idea of extending neural networks from Euclidean spaces to discrete graphs [34]. The key idea lies in learning enriched node embeddings that recursively aggregate information from their neighborhood through *message-passing*. As shown in Figure 2.6 from work [35], the model first defines the computation graph then all the nodes update their status through the defined aggregation function. Depending on the various realization of aggregation functions suggested by [36, 37], existing GNN works form two categories: spectral-based methods and spatial-based methods. Motivated by the successes of convolutional neural networks, the spectral-based methods generalize the concept and design of convolution filters from continuous image pixels to the domain of graph spectral [38, 39, 40, 41, 38, 42]. However, these spectral-based GNNs usually require the whole

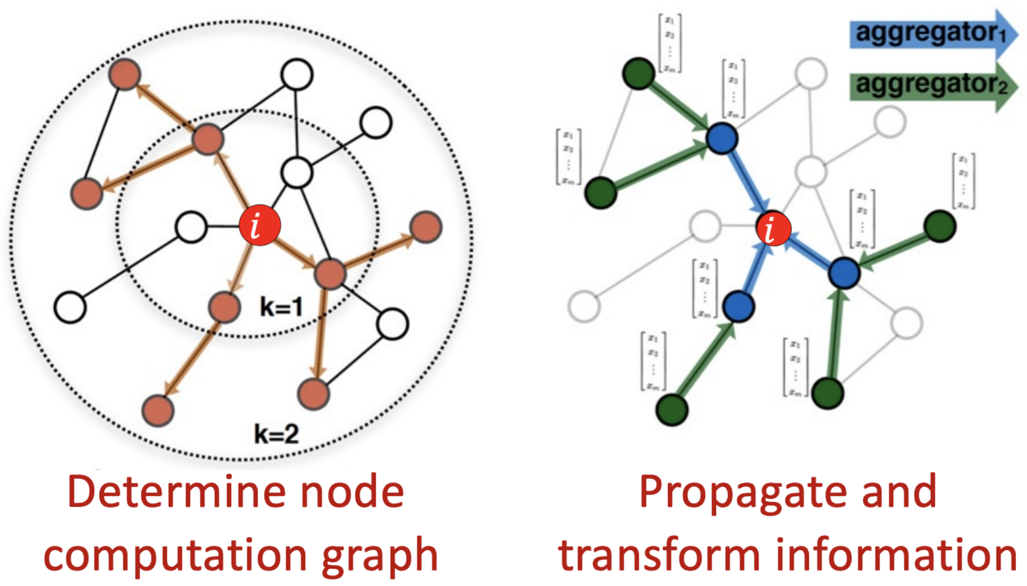

k=1
k=2
aggregator$_1$
aggregator$_2$
Determine node computation graph
Propagate and transform information

Figure 2.6: Learn how to propagate information across the graph to compute node features

graph as the input, thereby being bottlenecked for graphs at a real-world scale due to the high memory and computational cost.

To improve efficiency and scalability, subsequent spatial-based methods [43, 44] were proposed, defining message aggregation operations directly on the graph topological structures with subgraph windows or node sequences. Representative works include the Graph Attention Networks [45, 46], which assigns different weights for neighboring regions in the message-passing process. In such a way, the GNN models are able to operate on graphs with stochastic inputs, instead of the entire topology, thereby leading to much improved memory efficiency. Despite their differences, the main idea behind the two threads of GNNs remains constant, where an embedding space *harmonizing* both graph topological structure and nodal features is desired. We have modeled a generic GNN as a victim model which is discussed in Section 4.2 to respect this key concept without losing generality.

We note that our GAFNC paradigm is general and can be exploited to attack GNNs with various architectures, with the only leverage being the gradient information of the victim GNN. During implementation, attacking the graph convolutional network (GCN) is exemplified to substantiate the viability of the GAFNC paradigm.

### 2.3.2 Adversarial Attack and Defense on Graphs

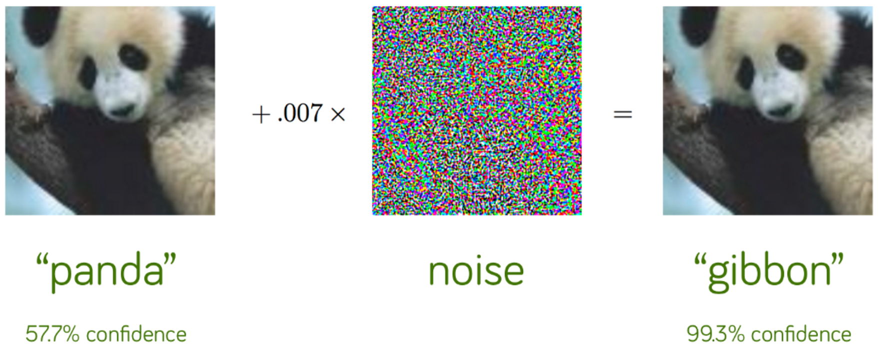


Figure 2.7: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet.

The robustness and vulnerability of deep neural networks have been extensively studied in continuous Euclidean domains, including images [47, 48], acoustics [49], and natural languages [50]. As per Figure 2.7 in [51], when there is a small intentional perturbation to the clean image, the deep model would make a false prediction. This domain was highly motivated by [52]'s seminar work, which reports that human-imperceptibly small perturbations on the input data can be escalated through the layer-by-layer representation of a deep network. Hence, it leads to the shift of correct classification boundaries and thus disastrous accuracy results. Starting from [53], the security issues of GNNs has been brought into wide attention. Mainstream attack techniques on GNNs are categorized into *evasion* attacks and *poisoning* attacks. Defense techniques to counter against both attack techniques are proposed in [54, 55, 56, 57]. To compare, in the evasion attack, the adversaries are tasked to mislead the models to make incorrect predictions on the manipulated data in the *test time*. However, the models can still keep their accuracies when given normal, undistorted data samples [58, 59].

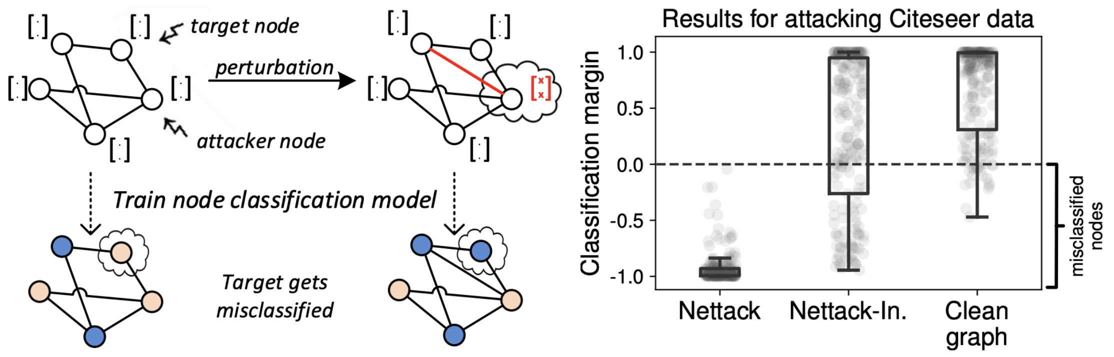


Figure 2.8: Small perturbations of the graph structure and node features lead to misclassification of the target.

Poisoning attacks, in contrast, strive to reduce the performance of GNN models during *training*, so that the poisoned models can not give a reliable prediction on any test samples [60, 61, 62]. So by this definition, our GAFNC approach falls into the category of poisoning attack. Prior arts mainly posit that the adversaries are given the privilege to manipulate the training data arbitrarily, such as adding or deleting graph edges [60, 62] or changing nodal features [61]. As per Figure 2.8 from [63]. Also, this assumption could be too costly to be realized in real-world applications, *e.g.*, hacking into a social network infrastructure such as Twitter or Facebook, limiting the practicality of the existing proposals. To fill the gap, our GAFNC paradigm aims to craft a set of adversarial nodes and propagate their camouflaged malicious features via strategically learned connections to the original graph, which needs *no access* to the training data and thus is more practical.

We note that the recent studies of *node injection* attack focusing on creating new nodes can use either evasion [64, 65, 66] or poison [67, 68] settings. However, these studies only consider destroying the model performance in test or training. They fail to consider *stealthiness* of the injected nodes. That said, their crafted nodes usually carry vicious contents that prominently differs from the original nodes. Our GAFNC approach performs better against these node injection attackers by leveraging a generative adversarial network (GAN) [69, 70] to generate *camouflaged* adversarial nodes, Our GAFNC process two properties: 1) The generated nodes enforced look like and follow a similar feature distribution as the original nodes;

2) The connectivity between the new and the old nodes shares the same graph properties *e.g.*, sparsity, average degree, as the original graph. As such, the fake nodes generated by our GAFNC approach can fool outlier detectors [71] and hence it performs poisoning attack in a more stealthy fashion.

# Chapter 3

# Deep Denoising of Raw Biomedical Knowledge Graph from COVID-19 Literature, LitCovid and Pubtator[1]

## 3.1 Introduction

The effects of the pandemic caused by the coronavirus disease 2019 (COVID-19) linger in 2022—globally affecting over 11.6 million people in the past year, and accounting for over 2.5 million deaths in more than 220 countries [72]. With the continuous accumulation of peer-reviewed publications on the topic, a literature hub serves as a means to track the most up-to-date scientific information about the virus [73]—encompassing research on treatment, diagnosis, and prevention of COVID-19. A knowledge base built upon the integration of the biomedical entities, from such a literature hub, provides tremendous value to the exploration of the explicit or implicit associations amongst diverse biomedical entities as investigators attempt to answer clinical questions related to COVID-19. A number of recently published journal articles have included graph-based analysis of COVID-19 datasets [74]. For example, Groza [75] has analyzed how a semantically annotated dataset would be helpful in detecting and preventing potentially harmful misinformation regarding the spread of COVID-19 based on CORD-19-on-FHIR [76].

Most knowledge graphs constructed for COVID-19 are currently based upon the co-occurring biomedical entities of recent literature. A knowledge graph of co-occurring concepts, such as the one created by Oniani et al. [77] can help researchers find associations among genes, drugs, and diseases related to COVID-19. Utilizing knowledge graphs with heterogeneous biomedical associations (e.g., gene-drug, disease-drug, drug-side effect) in these

[1]This work was done at Mayo Clinic, under the supervision of Dr. Nansu Zong

types of applications, however, results in a high probability of false-positive predictions because the co-occurrence in literature does not always mean a true biomedical association between the two entities. These co-occurrence edges, hence, are considered, "noise," due to its untrue associations. For example, the term, "glucose," may co-occur with the term, "yellow fever," but there is no real medical association between the two terms. Noise removal can be beneficial to downstream applications, such as link prediction [78], representation learning [79, 14], and node classification [80].

The manual processes of cleaning data and removing noise are resource-intensive. Therefore, an automated denoising method is ideal in facilitating the curation of knowledge graphs. Existing methods for denoising knowledge graphs can be divided into two groups: internal and external [81]. For the internal method, the predefined semantics or rules [82] are used for non-numerical data. The outlier detection [83] removes noise by modeling the real facts data as a distribution for numerical data. As for external methods, a pre-trained Graph Neural Network integrates heterogeneous data sources, [84] not only improving the performance of link prediction but also reducing the training time of the existing GNN model. In this work, our investigated methods can be categorized as an internal method where data augmentation with Generative Adversarial Network (GAN) removes noise. GAN has been widely applied in the medical imaging process [85] to denoise CT images based on generative adversarial networks with Wasserstein distance and perceptual similarity. Zhou et al. [86] has previously shown improvement of ultrasonic image quality and noise reduction caused device limitations through the construction of a two-stage GAN. Other than the application of generating images, GAN has mainly been used for generating discrete medical data toward contributing to the scenario of diagnosis of disease with few labels [87] or unbalanced classification [88].

**Specific contributions of this work are as follows:**

1. To the best knowledge of the authors, our study is the first study that uses GAN to denoise a biomedical knowledge graph. we propose a framework that generates a

similar graph from a raw knowledge graph to distinguish the true and false edges of association based on generative-based deep neural networks. Two recent generative-based models, Cross-Entropy Low-rank Logits (CELL) [89] and a generative-based graph method, NetGAN [90], have been adopted as a component to remove noise and retain true associations.

2. Our framework has generated two datasets. The first one is synthetic data generation which simulates the process of labeling and the quality of unlabeled data with annotation ration and noise ratio. Second, we build a real dataset from raw RDF turtle and human annotation 500 edges.

3. Extensive experiments are carried out over two datasets. The performance of link prediction, especially in the extreme case of training data versus test data at a ratio of 1:9, demonstrated that the proposed method still achieved favorable results (area under the receiver operating characteristic curve $> 0.8$ for the synthetic data set and 0.7 for the real data set), despite the limited amount of testing data available.

## 3.2 Problem definition

Given a network G(V, E) where V stands for a set of vertices (i.e. biomedical concepts in the literature) and E represents the edges among two vertices (i.e. the co-occurrence of two concepts), two kinds of edges exist, which are denoted as L (known true associations) and U (unknown true associations). Please note, if no edge exists between two vertices, we call it a false association. The goal of this study is to find the true associations among U (i.e., denoise U). Specifically, a proposed method should have the capacity to determine whether an unknown true association from U is a true association or false.

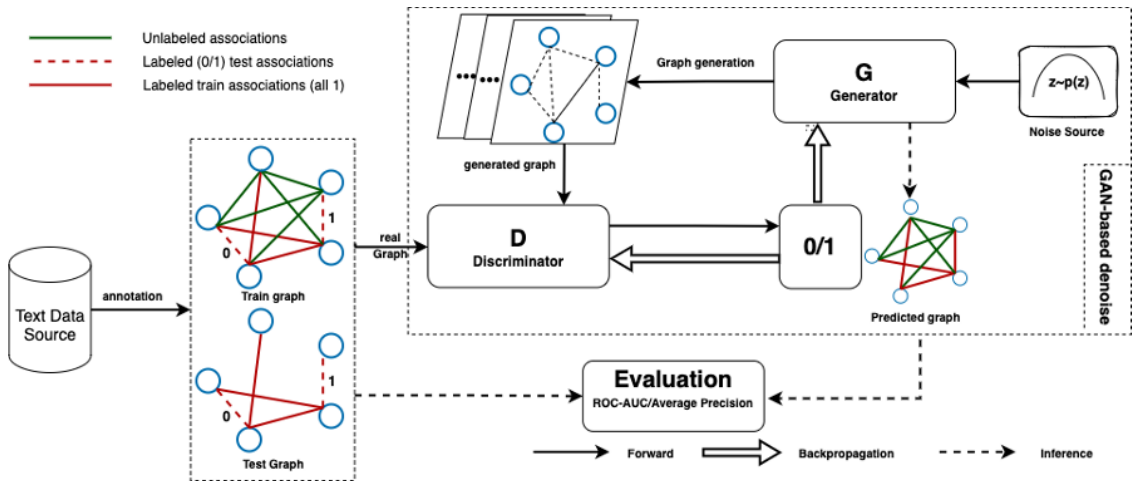


Figure 3.1: Overview of our investigation process.

## 3.3 Methodology

### 3.3.1 Framework

As this problem could be considered a classification of an unknown edge with a small number of known true associations and a large number of unknown true associations, we define this classification problem as few-shot learning (FSL) [91]. We propose a framework that utilized generative-based deep neural networks (e.g., NetGAN and CELL) to denoise the unknown true associations in U, based on similar networks generated. This framework is divided into three parts. We first briefly describe the GAN-based denoising graph adopted behind the development of the framework followed by an introduction of data preparation, which is comprised of two strategies: 1) synthetic data generation and 2) real dataset collection and annotation. Then, a comprehensive design of our experiments is conducted to verify our assumptions.

### 3.3.2 Denoise based on generative-based deep neural networks

We adopted NetGAN to generate a new network that would be used to distinguish the unknown associations in the raw training data (i.e, graph). To achieve this, we randomly sampled walks from the raw graph consisting of unlabeled edges and trained a generator to learn the walks sampled and a discriminator on how to separate a real walk and a fake one. After achieving equilibrium among the discriminator and generator, the random walk sample from the generator was used for filtering the unreal edges in the raw graph. As in work [89], sampling enough random walks was sufficient to reconstruct the graph. Both the generator and discriminator used the Long short-term memory architecture (LSTM) [92] and were trained with the Wasserstein loss [93]. The generator G generated large numbers of random walks (nodes sequence) of fixed length. The discriminator D distinguished the sequence of the nodes sampled from G and x which were sampled from the real graph (including unlabeled associations) with randomly started nodes. D and G played the following minimax game with the value function V(D,G):

$$\min_{G}\max_{D} V(D,G) = E_{x\sim p_{label}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.1}$$

Finally, the D generated a similar authentic graph network that could not be distinguished by the discriminator G.

To generate the probability of the edges, CELL approximated it with a score matrix $S$, which was computed by $\frac{S}{nT} \xrightarrow[n,T->\infty]{a.s.} diag(\pi)P$, where $n$ is the number of random walks, $T$ is each length of a random walk and $diag(\pi)$ is the stationary distribution matrix. $P$ a transition matrix that proximates the unbiased random walk used in NetGAN. $P$ can be low-rank approximated by $W$, which is the logit transition matrix and is solved by the objective function as:

$$\min_{W\in R^{N\times N}} -\sum_{k,l=1}^{N} A_{k,l}\log_{\sigma_{rows}}(W)_{k,l} - \sum_{k,l=1}^{N} A_{k,l}[D_{k,l}\leq k]\log_{\sigma_{rows}}(W)_{k,l} \tag{3.2}$$

where the A is the adjacency matrix and s.t. $rank(W) \leq H$.

Instead of the random walk as origin implementation, we further adapted node2vec[94] for the random sampling process in NetGAN to more flexibly control the preference of DFS or BFS transition among walks. Also, through constraining the edge generation length with k in the above loss function in CELL, we adapted the method to allocate more preference to the local homophily of the graph.

## 3.4 Experiments

### 3.4.1 Data preparation

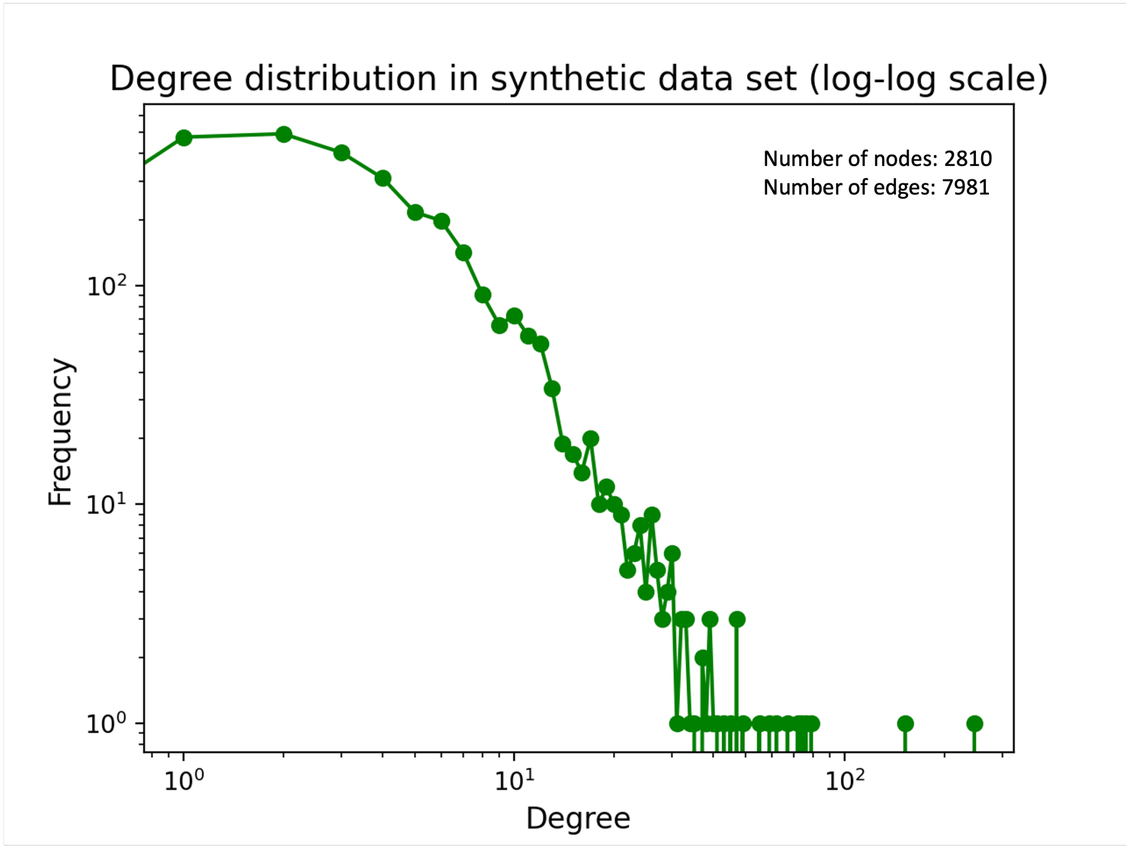
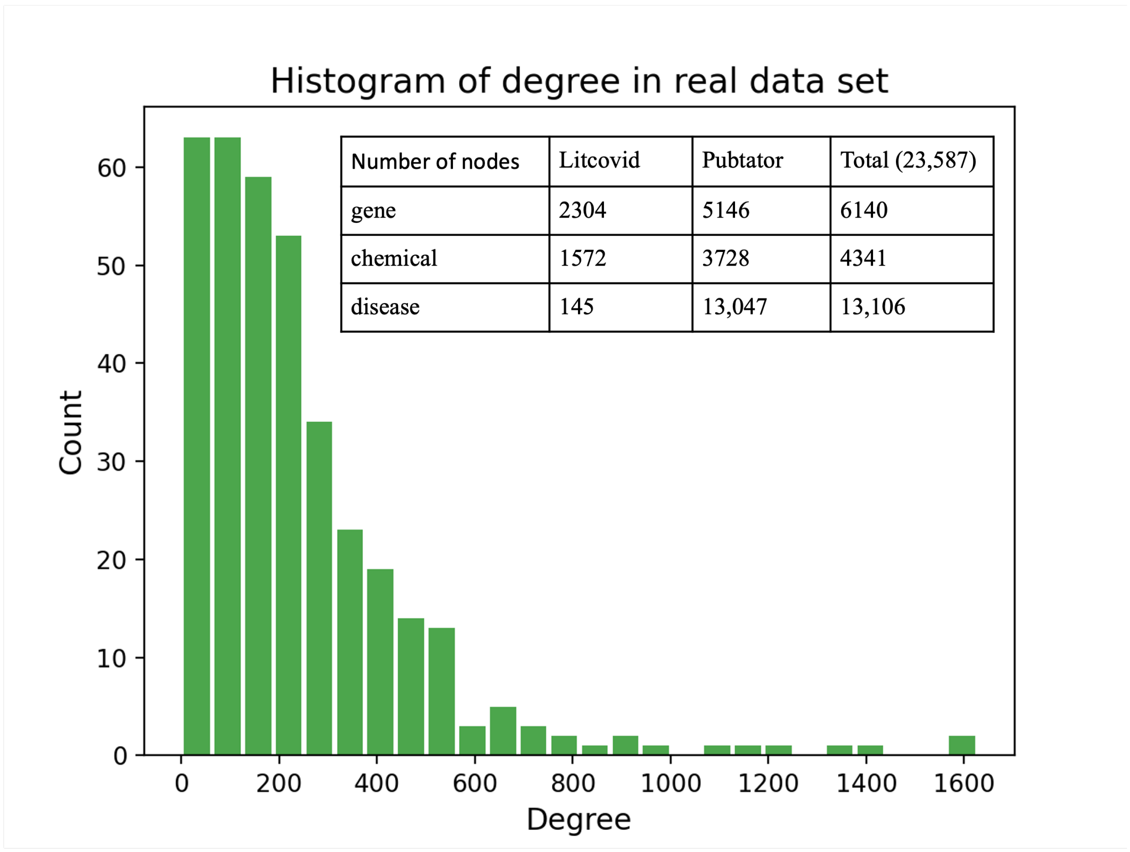


Figure 3.2: Synthetic dataset

We generated two datasets for this study: 1) a synthetic dataset based on CORA-ML, and 2) a real dataset extracted from CORD-19-on-FHIR datasets [76]. First, we defined two types of associations: labeled associations denoted as (L) [red colored] and unlabeled associations represented as (U) [green colored] (Figure 3.1), based on two types of association; this was used to construct our training and test graphs. The training graph consisted of both the labeled (L) and unlabeled (U) associations, while there were only labeled (L) associations



| Number of nodes | Litcovid | Pubtator | Total (23,587) |
|---|---|---|---|
| gene | 2304 | 5146 | 6140 |
| chemical | 1572 | 3728 | 4341 |
| disease | 145 | 13,047 | 13,106 |

Figure 3.3: Real dataset

in the test graph, as we need the ground truth for evaluating the performance of our proposed methods. The histogram of each dataset is given below, where (Figure 3.2) is the synthetic dataset. This does not include the false associations added in our subsequent experiments. (Figure 3.3) shows the histogram of degree in the real dataset.

**Synthetic Dataset** The synthetic dataset was formed based on CORA-ML with the same preprocessing work as NetGAN [20]; we chose the largest connected component (LCC) in the graph. The final total number of nodes and edges is shown in the top right corner of (Figure 3.2). We took as ground truth the existing edges as true associations, and the nonexistent edges as false associations, to test our proposed methods.

**Synthetic dataset generation** First, a positive association graph was constructed from the origin adjacency matrix labeled as 1. Second, a negative association graph was sampled based on the 0-labeled elements in the origin adjacency matrix. Finally, we merged the two types of associations in one graph as our synthetic dataset. (Also, as a restriction of NetGAN, we also colored the minimum spanning tree as red which will be kept in the training set.) An example is shown in (Figure 3.4) for the whole pipline of constructing the synthetic dataset.

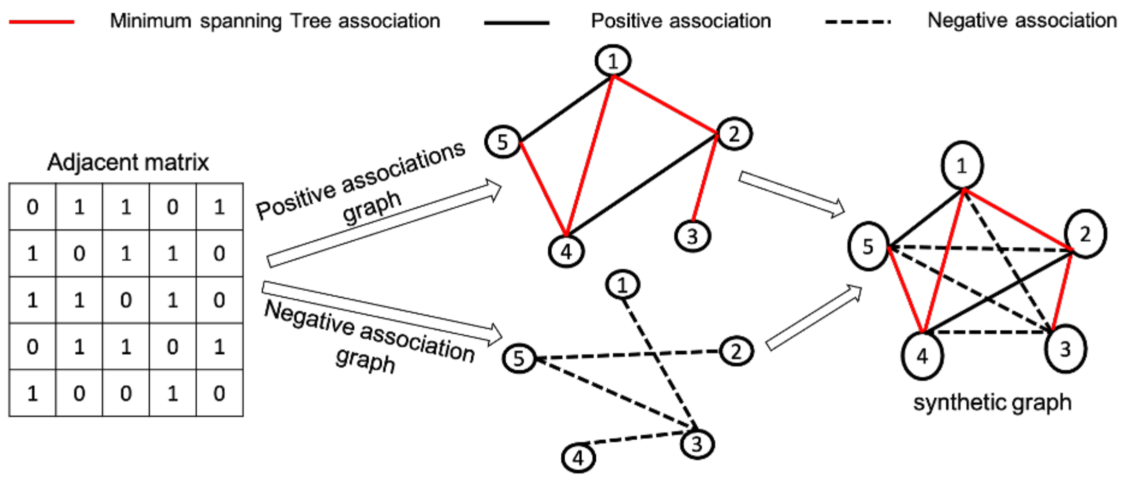


Figure 3.4: Synthetic dataset construction pipeline

| | # Litcovid | # Pubtator |
|---|---|---|
| Chem-Dise | 1572-104 | 3728-7748 |
| Gene-Dise | 2304-124 | 5146-8540 |
| Gene-Chem(filtered) | 1499-1318 | 4725-3352 |
| Gene-Chem(queried) | 2350-2014 | 20,881-13,518 |

Table 3.1: The raw dataset with node numbers counted from Litcovid and Pubtator

**Real dataset** From CORD-19-on-FHIR [76] (a linked data version of the COVID-19 Open Research Dataset (CORD-19) data represented in FHIR RDF by mining the CORD-19 dataset and adding semantic annotations), we utilized two annotated networks (Litcovid [95], Pubtator [96]), extracting the COVID-19 related terms in our SPARQL query with three types of biomedical concepts (i.e., Gene, Chemical, Mutation/Disease). After merging identical IDs from both Litcovid and Pubtator, we were able to obtain a new dataset with a total of 23578 nodes (Figure 3.3). Finally, we randomly chose a proportional number of edges with a total of 500 associations (i.e., Chemical-Disease, Gene-Disease, Gene-Chemical associations) from a total of 288270 edges in the whole graph and manually labeled them as our labeled dataset.

| Type | # Number | # Labeled(ground truth) |
|---|---|---|
| Chem-Dise | 228,148 | 350 |
| Gene-Dise | 32,611 | 100 |
| Chem-Dise(filtered) | 27,511 | 50 |
| total | 288,270 | 500 |

Table 3.2: Constitution of our real dataset with different types of edges

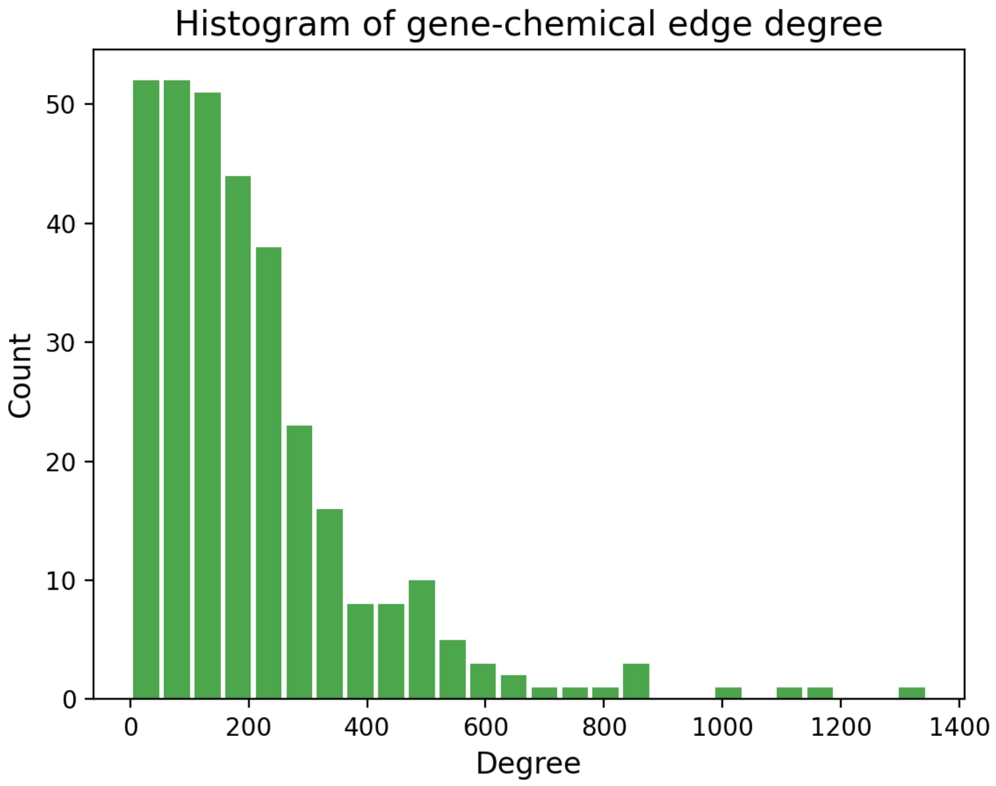


Figure 3.5: Degree of gene-chem types of edges in real dataset

**Real dataset preprocessing and statistic** (Table 3.1) shows the detailed number of associations we preprocessed and collected from the raw data, as well as data filtered from the Gene-Chem associations (by requiring that the vertices must appear in our previous two types of associations) (Chem-Dise, Gene-Dise). After merging the identical node id within

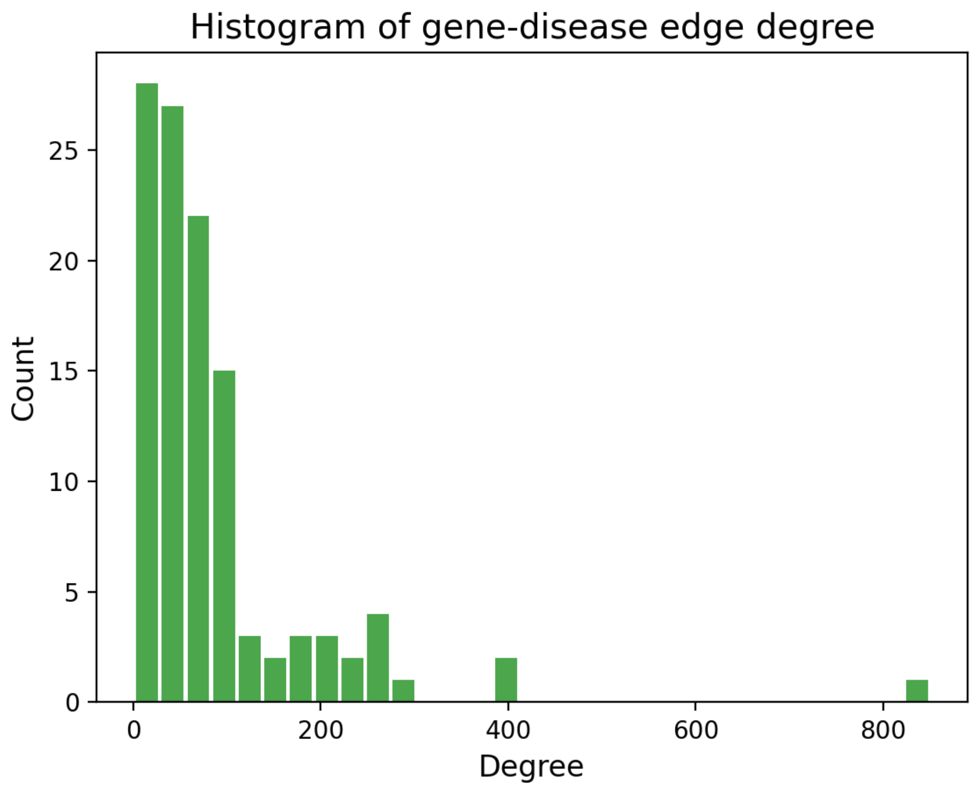


Figure 3.6: Degree of gene-dise types of edges in real dataset

the same category, we got a total of 23587 nodes. (Table 3.2) shows the number of the three types of edges and the corresponding labeled ground truth. Except for the histogram of degree in all associations reported in the manuscripts. Here we report the histogram degree of each type of association in our (Figure 3.5, 3.6, 3.7).

### 3.4.2 Experiment design

We conducted experiments on both a synthetic dataset (i.e., CORA-ML) and a real dataset extracted from CORD-19-on-FHIR, in order to investigate the capability of our proposed methods of incorporating unlabeled information for improving the link prediction performance despite limited annotation. We analyzed the performance of our models with multiple tasks based on two types of ratios to mimic the percentages of noise and annotation during the data curation: 1) Noise Ratio (NR), the percentage of true and false association

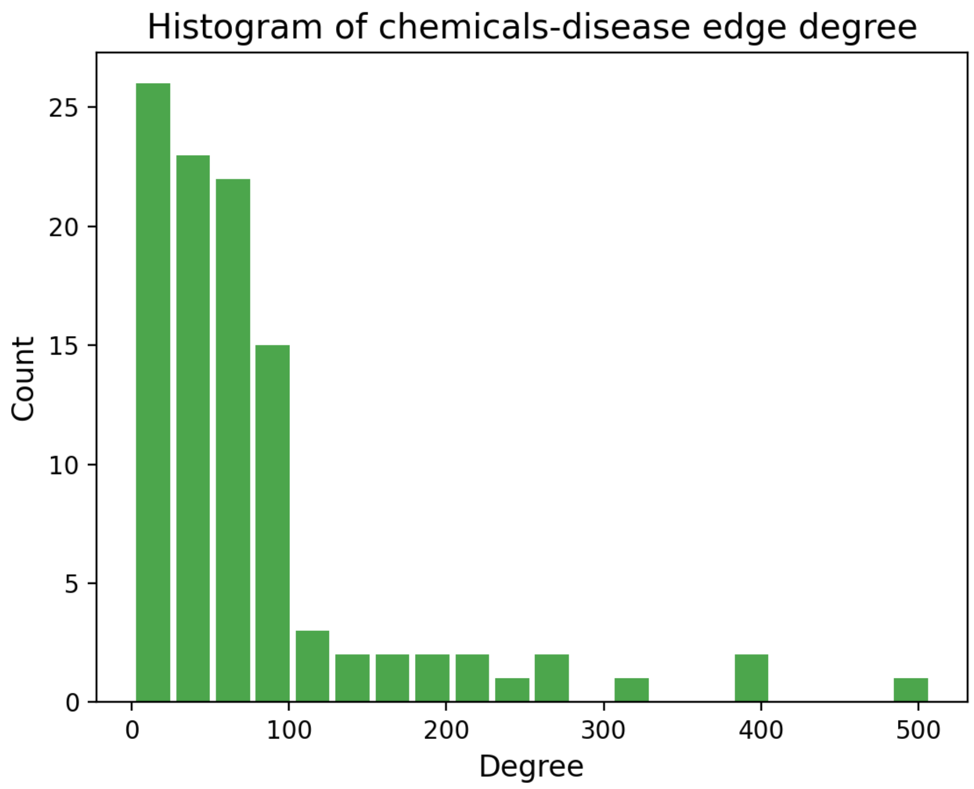


Figure 3.7: Degree of chem-dise types of edges in real dataset

in the unlabeled edges; and 2) Annotation Ratio (AR), the percentage of training and testing association in the labeled edges.

**Task (1): Test of Annotation Ratio (AR) over the synthetic dataset.**

We wanted to understand how many annotations were needed during the data curation for our proposed method to predict the true and false associations. We set a fixed NR and evaluated the performance of the tested method in two cases: one included the unlabeled data (i.e., training set=labeled true and false associations + unlabeled associations), and another one did not include the unlabeled data (i.e., training set=labeled true and false associations). The unlabeled associations were taken as true associations for training. In the experiment, we tested the performances based on different AR to mimic the percentage of the annotations already completed during the data curation. In practice, the AR varied from 1:9 to 9:1. For each ratio, we repeated 10 times with a random sampling of the training and testing sets to get the average results.

**Task (2): Test of Noise Ratio (NR) over the synthetic dataset.**

In this task, we wanted to understand how many false associations were deemed as true associations in unlabeled data for training because it affected the performance of the proposed method in prediction. We wanted to see whether the proposed method was robust enough to learn useful information for prediction, especially from unlabeled edges with more noise. With a fixed AR of 1:1, we tested the proposed method when the false edges were more than the real edges in the unlabeled data. In practice, the NR varied from 1:1 to 1:9.

**Task (3): Test over the real dataset.**

After the same training of annotation, two of the co-workers (C.J. and Y.Y.) manually labeled the 500 edges from 288270 to simulate an extreme use case for data curation, and the coworker (V.N.) verified the annotation by random sampling the edges. Among the 500 edges, three types consisted of Chem-Dise, Gene-Chem, and Gene-Dise. Each edge was marked as true, false, and unknown. In practice, the annotations for the gene-chem were excluded and marked as unknown in the final evaluation after discussion and reaching consensus among the authors that they were conducted without enough confidence. Thus, in our final result report of the ROC curve, we only considered two types of associations Chem-Dise and Gene-Dise.

### 3.4.3 Setting and evaluation metrics

For each proposed method (i.e., NetGAN and CELL), a grid search strategy was adapted for obtaining the best hyperparameters. In our experiment, we defined the search range by referencing the original settings in the articles. For NetGAN, the parameter ranges for the grid search are specified as walk $p = 0.01, 0.1, 1, 10, 100$ and $q = 0.01, 0.1, 1, 10, 100$. For CELL, the parameter ranges are specified as rank $H = 9, 20$, learning rate $lr = 0.01, 0.05, 0.1$, and weight decay $weight_{decay} = 1e-5, 1e-6, 1e-7$. In practice, the origin NetGAN was obtained from [97] , and the origin CELL was obtained from [98].

In the evaluation step, we chose the area under the receiver operating characteristic curve (AUC ROC) and Average Precision (AP) as the metrics of link prediction for our proposed methods in both synthetic and real datasets. In the implementation, both AUC ROC and AP scores were calculated by scikit-learn [99]. The visualization of predicted results in our real dataset was a plot with Cytoscape [100], an open-source software platform for visualizing complex networks and integrating these with any type of attribute data.

## 3.5 Results and Conclusion

### 3.5.1 Results

**Task (1): Comparison of the link prediction results in a graph with/without the unlabeled associations among different annotation ratios.**

We conducted our experiments in two scenarios, one was the base case (dash line was used in Figure 3.8) where we tested our models without using the unlabeled information, without explicitly stating as the base case, all of our statements in the following would be the default case (solid line is used in our Figure 3.8) that indicated that we included the unlabeled associations in the link prediction tasks. We reported the AUC ROC score in figure 3.8. Here, the Figure 3.8 displayed the AUC ROC curve with a fixed annotation ratio of 0.5. The dashed line named "Base NetGAN" meant the method of NetGAN without incorporating the unlabeled information. Detailed information about the adaption of NetGAN can be found in our appendix. The same goes for "Base CELL" which stands for the method CELL runs in the base case. There was no big difference between the two methods when considering the base case with AUC ROC Score as 0.597 for NetGAN and 0.591 for CELL. However, when unlabeled information was taken into consideration, both methods achieved better performance compared with the base case, the AUC ROC Score of NetGAN has 0.724 and CELL achieved 0.828. The Figure 3.9 showed the performance of the proposed methods in

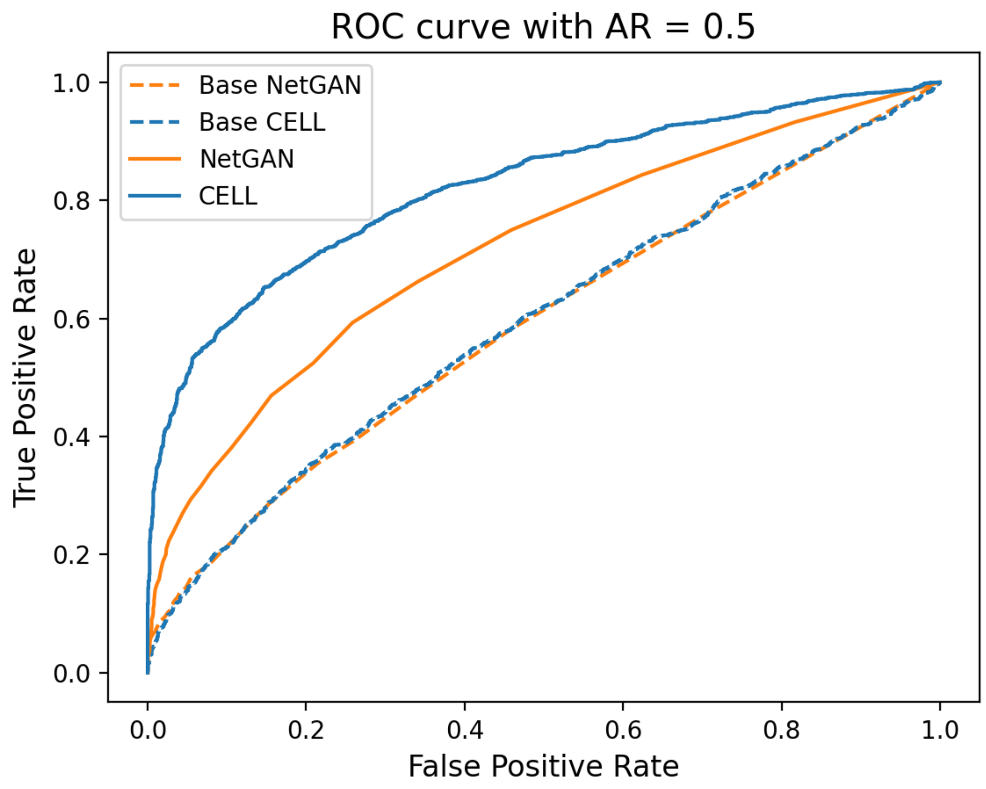


Figure 3.8: AUC ROC performance of NetGAN and CELL with/without unlabeled information at AR=0.5

different annotation ratios ranging from 0.1 to 0.9. We saw the CELL had overall better performance.

**Task (2): How are the models performing with different noise ratios in the unlabeled edges.** We tested the performance of methods when the unlabeled information contained a different ratio of by a factor of 10 (Figure 3.10). CELL demonstrated exceptional performance when the noise ratio equaled 1:1. And even in the extreme case where the true vs false ratio reached 1:9, CELL still had better performance compared to NetGAN with an AUC score of around 0.7. The performance of CELL had less variance compared with NetGAN in all noise ratios. In other words, CELL had a relatively better capability and stability to utilize unlabeled data compared with NetGAN when dealing with the complexity of the noise ratio in the unknown information.

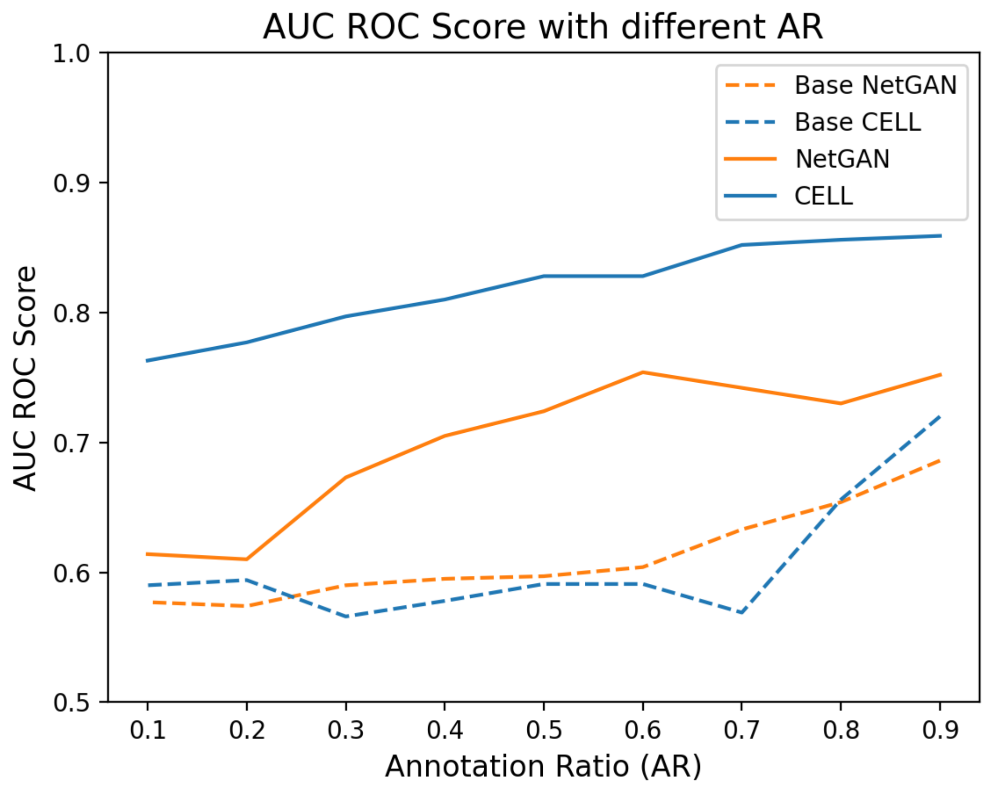


Figure 3.9: AUC ROC performance of NetGAN and CELL with/without unlabeled information with different AR

**Task (3): The performance of proposed models in our collected real dataset.** After our exploration of our methods in task 2, we conducted our methods on a real dataset. Although the noise ratio was unknown in our real dataset, the proposed methods still performed better than random classification with the incorporation of unknown associations. Also, compared with NetGAN, CELL still had an impressive result with ROC-AUC achieved up to around 0.706 when the test and train ratio was 1:1 and the unknown association occupied about 99.95% as shown in Figure 3.11.

The good performance of CELL showed that it had an excellent capability to predict the true association with the use of unlabeled data. We reported the ROC-AUC value of each type of association separately in (Figure 3.12).

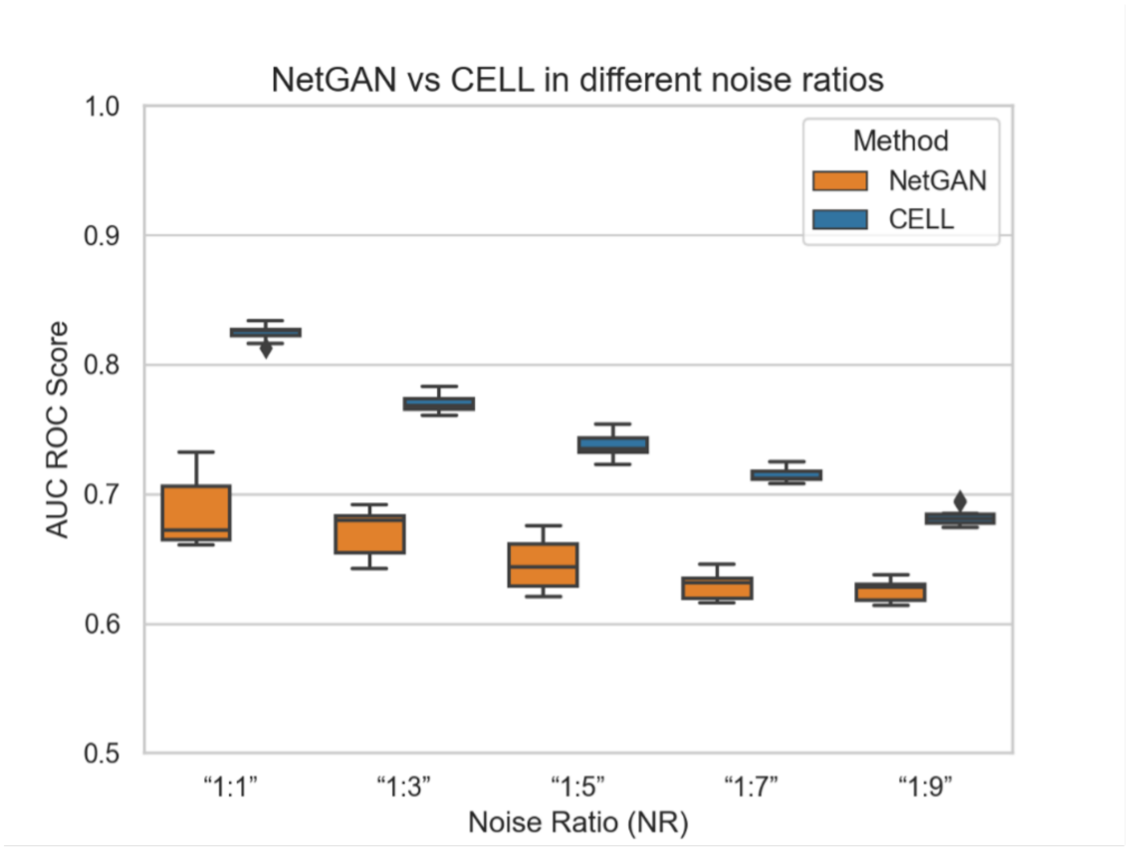


Figure 3.10: Performance of AUC Score in different noise ratios (NR)

Combining the figure of edge degree of each type of association in (Figure 3.5, 3.6, 3.7), we concluded that, as the degree is larger, there would be more noise contained in each edge. Thus, the results would be affected correspondingly. The performance of average precision for our proposed models in our synthetic and real datasets can be found in Figure (3.13 and 3.14).

**Denoised knowledge graph generated from the real dataset.** We trained the adapted NetGAN with the whole real dataset, and plot the predicted denoised knowledge graph in (Figure 3.15), where the edges are generated based on the score matrix calculated following the generation method used in NetGAN. There are a total of 21,016 edges in our visualization consisting of Gene-Chem (7562), Gene-Dise (7613), and Chem-Dise (5841). Three different colors (red, green, and blue) stand for three different types of associations/edges (Gene-Chem, Gene-Dise, Chem-Dise). The prediction can be found at [101].

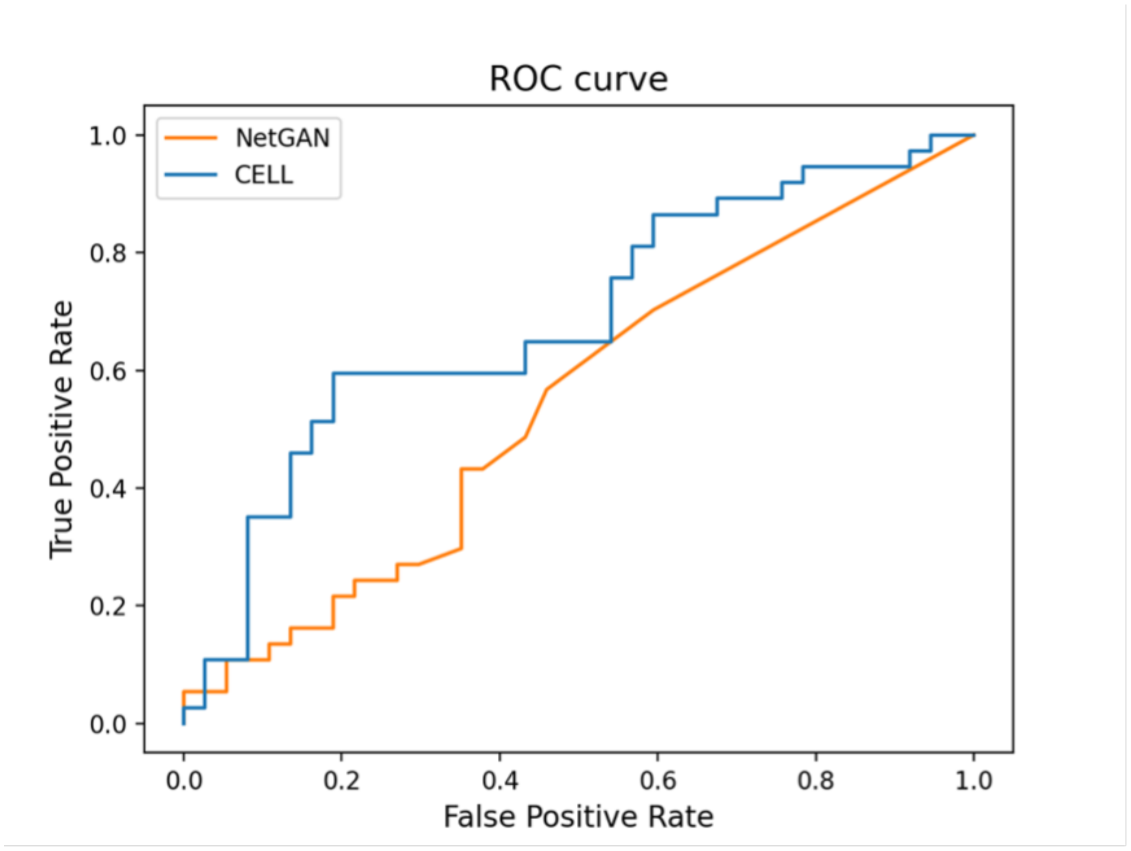


Figure 3.11: Performance on real dataset

### 3.5.2 Discussion and Conclusion

Despite the capability and stability of our proposed methods shown in this study, there are a few limitations that need to be discussed.

Firstly, the associations labeled in the real dataset are limited due to the limited resources. In addition, with the reality of vagueness or missing concepts found in different biomedical literature, there will be some bias. A large sample of annotated associations may provide a solution to reduce this bias and thus is needed in our future work. One way to potentially accomplish this goal would include utilizing NLP methods to standardize the concepts prior to annotation, which may improve the construction of knowledge graph input to our methods. Another way includes collaborating with professional annotators to both increase the number of annotations as well as improve the quality.

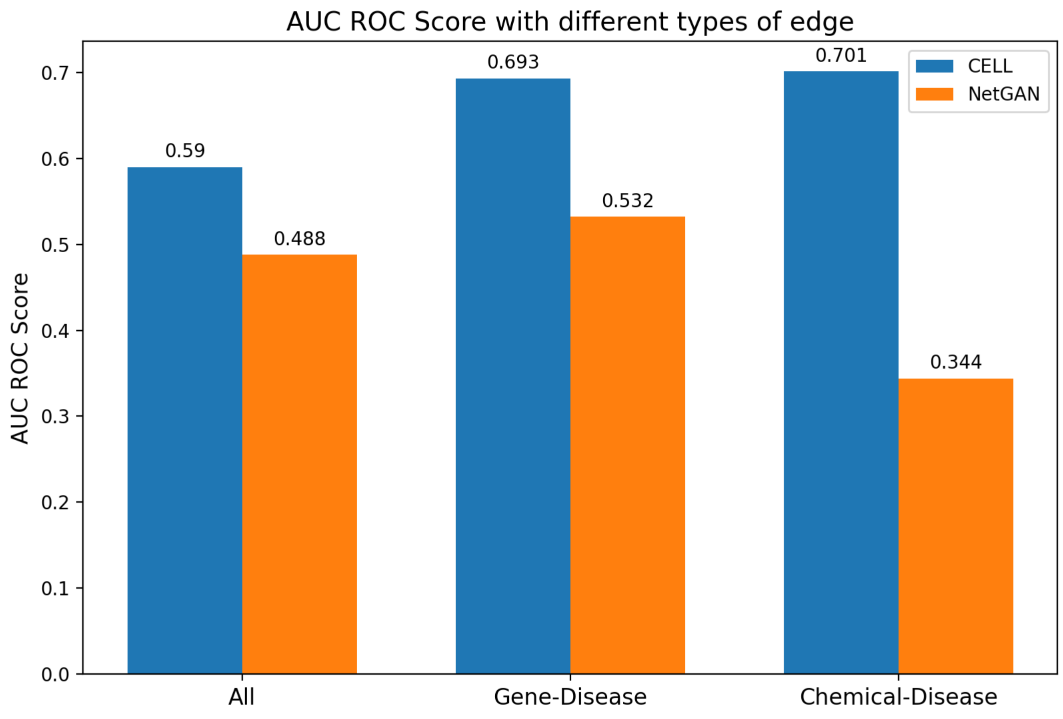


Figure 3.12: AUC ROC Score for different types of associations in the real dataset.

Secondly, while we have achieved notable improvement with AUC around 0.7 in our real dataset compared with random classification, there is still a gap between the experimental results in a controlled environment compared to the adaptation of the proposed method for data curation in real-world scenarios. Performance improvement is still needed. The complexity of our investigated algorithm comes from the module of LSTM which generates random walks for reconstructing the graph. An adaption of binary neurons [102] that directly produces the discrete adjacency matrix for the graph may have the potential to significantly improve the efficacy of our investigated methods as reconstruction of the adjacency matrix from random walks will not be needed. Another approach to improving the performance of removing noise in our investigated methods could be looking into the possibilities of transfer learning or external methods as we discussed earlier, such as [103]. By importing prior knowledge into the process of graph generation, we could employ the knowledge from an

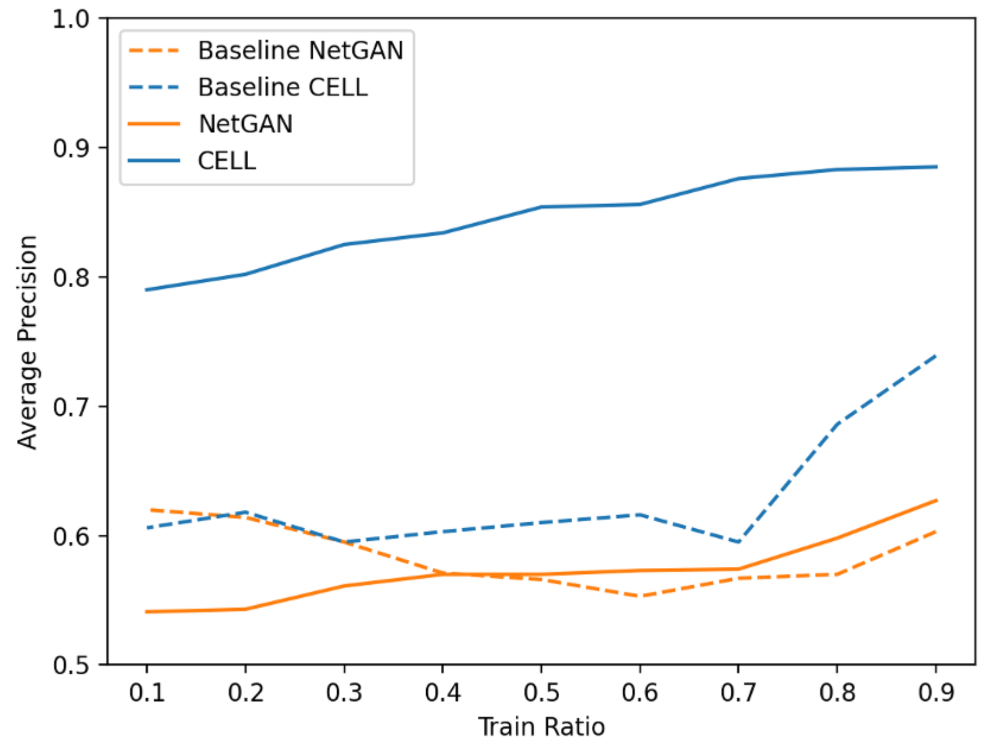


Figure 3.13: AP performance of NetGAN and CELL with/without unlabeled information in synthetic dataset.

already built dataset [104] to help us remove the false associations when constructing our biomedical graph.

Thirdly, our evaluation was based on the logic of classifying the true or false associations directly, and intentionally not focused on the impact evaluation of the denoised datasets generated in our work on the downstream applications (e.g., the prediction for drug-target association and protein-protein interaction). While we assume the performance will be improved in those applications [36], we acknowledge that there has not yet been any scientific proof to support that. The whole data stream, including the methods of data processing, data curation (i.e., denoising method proposed in this study), and the application needs to be investigated further to fill this gap—ideal in providing convincing evidence of the impact of our proposed method in denoising knowledge base construction.

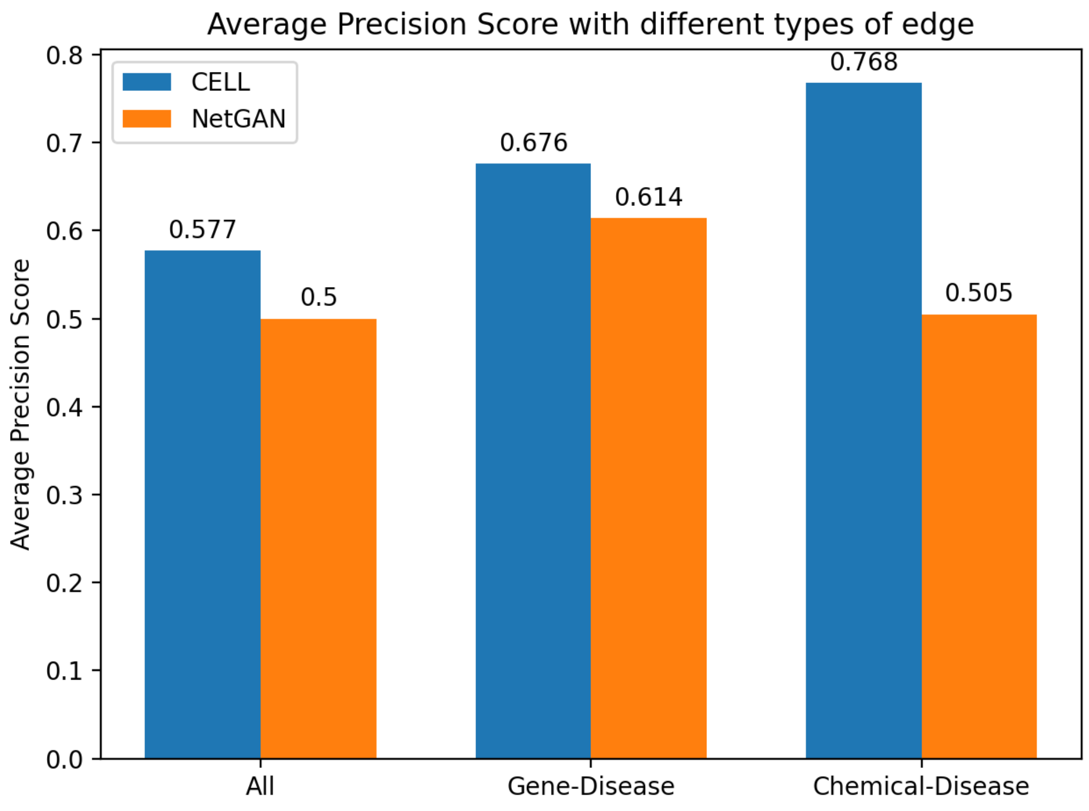


Figure 3.14: Average Precision Score for different types of associations in real dataset.

In this study, we proposed a method to automate the denoising of a knowledge graph generated via the counting of co-occurrence from biomedical literature. Our work can be considered as the pre-processing part for the curation of the knowledge graph. We adopted the state-of-the-art generate-based graph methods, NetGAN and CELL, to leverage the unlabeled co-occurred biomedical entities in the training process by the perturbation of the original graph in the determination of an unknown edge. Two datasets (i.e., synthetic and real datasets) were used to evaluate proposed methods in three link prediction tasks, and our experiments showed promising results achieved with both synthetic and real datasets.

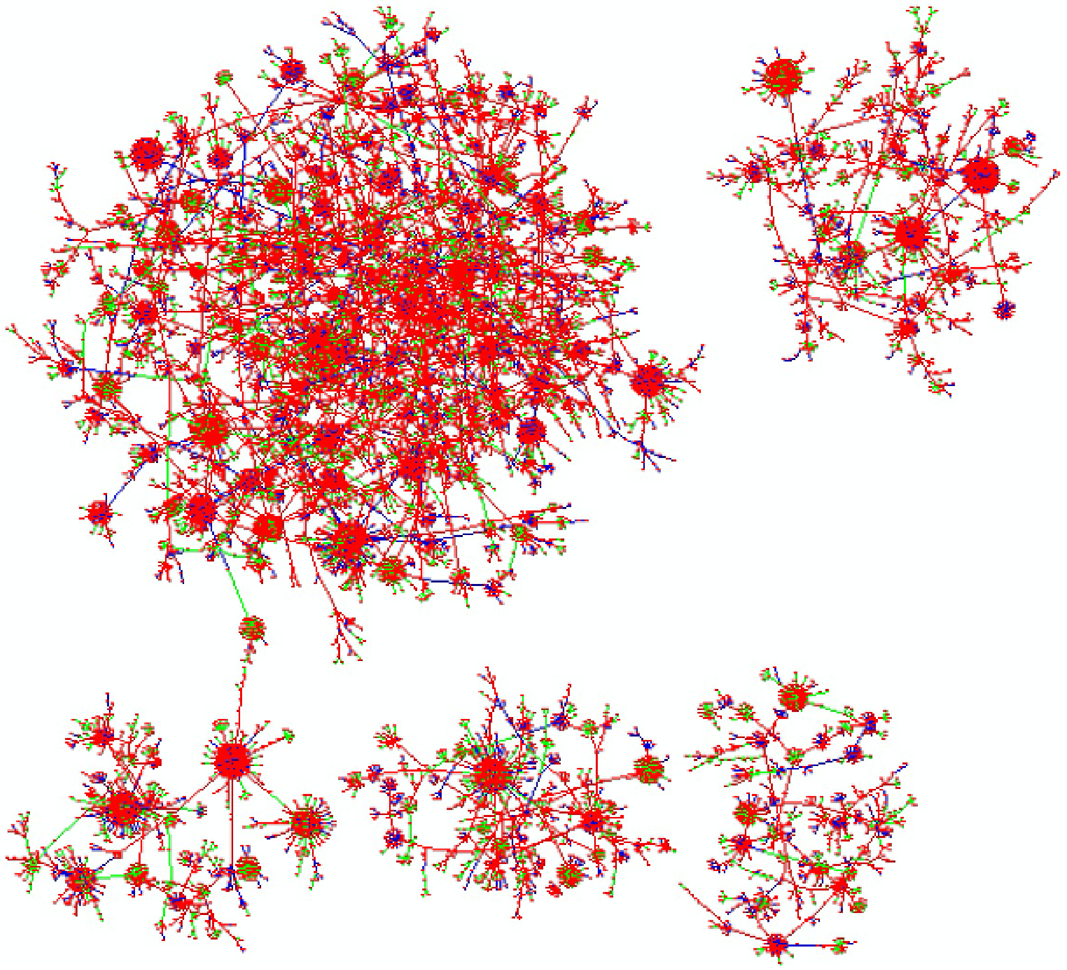


Figure 3.15: Visualization of the predicted knowledge graph based on the real covid-19 related dataset.

# Chapter 4

# Camouflaged Poisoning Attack on Graph Neural Networks

## 4.1 Introduction

Graph neural networks (GNNs) have enabled many web applications and multimedia systems to envision a higher level of automation in data analytics, such as malicious user detection in social media [105, 106], interest group discovery in recommender systems [107, 108, 109, 110], event prediction in service networks [111], to just name a few. Data in such applications are represented by graphs, in which the information is encoded in two channels – 1) the contents of the nodes and 2) the topological structure that encodes their correlations. The key to the success of GNN lies in its harmonized representation learning of the two information channels through *message-passing*. At each GNN layer, the node representations are enriched by collecting information from their immediate neighbors; By stacking multiple such layers, the neighborhood information traverses through the graph, allowing the nodes acquire knowledge about their wider surroundings.

Despite its effectiveness, this message-passing mechanism requires robustness and trustworthiness of GNNs. Indeed, recent results show that the training data of GNNs can be easily *poisoned* by making slight perturbations to the input graph, *e.g.*, manipulating the edge connections [53, 60] or changing contents of existing nodes [58]. Training the GNNs with the poisoned graph data shall lead to diminished or even disastrous test performance. Spurred by this awareness, several researchers addressed the problem of defending GNNs via detecting and correcting the distorted graph information, such as adversarial edge correction [112, 113, 114] and nodal perturbation detection [115, 116].

Although the field of GNN attack and defense is popular, most existing methods are built upon a strong assumption, namely, the adversaries have full access to the training graph data and can make any changes at will. This assumption is restrictive in many real applications. Take online business networks for example, where the attackers aiming to poison and fail an anti-scam system must hack into the customer/merchant accounts to change their contents (*i.e.*, nodal features) or connection with other users (*i.e.*, edges), which is indeed onerous, costly, and time-consuming. To lift this assumption, one may think to perform a node injection attack [64, 65, 67, 66, 68], where a set of fake nodes (*e.g.*, user accounts) are created and added to poison the graph. Also, this idea does not work well in practice, as the fake nodes usually carry vicious contents which are so *prominent* that their pattern differs from the benign nodes drastically, and simple classifiers, *e.g.*, an outlier detector [71], can easily spotlight and hence screen out the added fake nodes.

Motivated by this, we in this work mainly investigate two questions:

1. Can we poison GNNs *without* manipulating the original training graph?

2. Can the poisoning attack be performed in a *stealthy* fashion, such that current outlier detectors are fooled?

Our affirmative answers are provided through a novel attacking paradigm against GNNs, termed *Generative Adversarial Fake Node Camouflaging* (GAFNC). The key idea of GAFNC is to leverage the power of Generative Adversarial Networks (GAN) [69, 70] to synthesize an *inconspicuously small* set of adversarial nodes, each of which carries *seemingly authentic* yet distorted features. By connecting these new nodes to the original graph with different strategies, the malicious information hidden behind them are aggregated through message-passing, poisoning the victim GNN in two ways. First, our GAFNC can perform a *global attack*, where the new nodes scatter across the graph and strive to lower the classification accuracy of the victim GNN over all original nodes uniformly. Second, we can control the poisoning attack at a finer level of granularity by performing a *target attack*, where the new nodes neighbor a set of target nodes, forming a deliberate topology that encourages the victim GNN to misclassify the target nodes into prescribed classes.

**Specific contributions of this work are as follows:**

1. This is the first work to investigate the poisoning attacks against GNNs in a generative-adversarial regime. Our explored problem is novel in the sense that 1) we do *not* require access nor manipulation of the original nodes or edges and 2) our attack is at the training time, while the prior studies mainly focus on test time, evasion attacks; The challenges and goals thus differ and have be discussed in Section 2.2.

2. A novel GAFNC paradigm is proposed to realize both global and target attacks, where seemly-authentic nodes that form legitimate connectivities with the original graph are generated. Technical details are scrutinized in Sections 4.2, and 4.3.

3. Extensive experiments are carried out over four widely used real-world graph datasets, with the results demonstrating the viability, effectiveness, and stealthiness of our proposed attacking approach. Section 4.4 extrapolates the findings.

## 4.2 Problem definition

**Node Classification with GNNs.** This work uses the following conventions. Let $\mathcal{G} = (\mathbf{V}, \mathbf{A}, \mathbf{X})$ denote an attributed graph, where $\mathbf{V} = \{v_1, \ldots, v_N\}$ is the set of $N$ nodes, $\mathbf{A} \in \{0,1\}^{N \times N}$ is the adjacency matrix encoding the graph topology, where $\mathbf{A}_{i,j} = 1$ if an edge exists between two nodes $v_i$ and $v_j$ and $\mathbf{A}_{i,j} = 0$ otherwise. Denoted by $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{N \times D}$ is the nodal feature matrix, where each node is associated with a $D$-dimensional feature vector.

The problem of node classification on graphs is usually framed in a *transductive learning* setting. Our work follows this convention. Given a small subset of nodes being labeled, denoted by $\mathbf{V}_L = \{(v_1, y_1), \ldots, (v_L, y_L)\}$, where $y_i \in \{C_1, C_2, \cdots, C_k\}$ is the true label of $v_i$ and there are $k$ possible classes in total. The goal is to learn a function that can accurately predict the labels of the remaining unlabeled nodes $\mathbf{V}_U := \mathbf{V} \setminus \mathbf{V}_L$, where $|\mathbf{V}_U| > |\mathbf{V}_L|$.

GNNs have manifested their remarkable classification performance in the literature [36, 37]. Despite their many variants, the core function of GNNs is *message-passing*, where each node is represented in an enriched latent space by aggregating information from its neighbors. A generic formulation of this representation learning mechanism takes a recursive form as follows.

$$\mathbf{h}_i^l = \phi^l \left( \mathbf{h}_i^{l-1}, \mathrm{Agg} \left( \{ \psi^l(\mathbf{h}_j^{l-1}) \mid j \in \mathcal{N}_i \} \right) \right), \qquad (4.1)$$
$$\forall l = 1, 2, \ldots, M, \quad \mathbf{h}_i^0 = \mathbf{x}_i,$$

where, for node $v_i$ at the $l$-th layer, $\mathbf{h}_i^l$ denotes its latent representation, $\mathcal{N}_i$ encloses its first-order neighbors, $\mathrm{Agg}(\cdot)$ aggregates information/messages from its immediate context, and $\phi^l$ and $\psi^l$ implement arbitrary learnable transformations. In this work, we implement $\phi^l$ and $\psi^l$ with a non-linear activation and a linear mapping, for the sake of simplicity and without loss of generality, reducing Eq. (4.1) to the following format.

$$\mathbf{H}^l = \sigma(\hat{\mathbf{A}}\mathbf{H}^{l-1}\mathbf{W}^l), \quad \forall l = 1, 2, \ldots, M, \quad \mathbf{H}^0 = \mathbf{X} \qquad (4.2)$$

where $\mathbf{H}^l \in \mathbb{R}^{N \times Z^l}$ stacks the node representations at the $l$-th layer, and $\mathbf{W}^l \in \mathbb{R}^{Z^{l-1} \times Z^l}$ is the corresponding learnable weight matrix that linearly maps the input to a different latent space. $\hat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$ is the symmetric normalized adjacency matrix of the (undirected) input graph $\mathcal{G}$ after adding the self-loop $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ and $\widetilde{\mathbf{D}}$ is the degree matrix with $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{ij}$. Denoted by $\sigma(\cdot)$ is the non-linear activation function (*e.g.*, ReLU). As such, a GNN having in total $M$ layers allows the messages to travel across the graph by means of an $M$-hop neighboring context, yielding node representations that harmonize nodal features and graph topology, thereby uplifting the classification performance.

To learn the weights in Eq. (4.2), a straightforward and popular idea is to minimize the cross-entropy loss function that leverages the limited labeled nodes, defined as follows.

$$\min_{\mathbf{W}^1,\ldots,\mathbf{W}^M} \sum_{v_i \in \mathbf{V}_L} \mathrm{P}(y_i \neq \hat{y}_i) := -\sum_{v_i \in \mathbf{V}_L} y_i \log \hat{y}_i, \tag{4.3}$$

where $\hat{y}_i = \mathrm{softmax}(\mathbf{h}_i^M)$ is the predicted label of the $i$-th node, and $\mathbf{h}_i^M$ is the $i$-th column of $\mathbf{H}^M$ being the output of the last GNN layer. The optimization process of Eq. (4.3) starts from the $M$-th layer with parameters $\mathbf{W}_M$, and the gradient information back-propagates with chain rule layer-by-layer until $\mathbf{W}_1$ is updated. This process repeats multiple times until convergence is achieved.

**Threat Model** In this work, we restrict our interest to attacking GNN models with linear aggregation function, modeled by Eqs. (4.1) and (4.2), for two reasons. First, our GAFNC attacking approach operates in an end-to-end fashion with no assumption made on the model architecture – the only leverage is the gradient information of the victim model. Therefore, a linear realization delivers simplicity, helping to facilitate the understanding of how our poisoning attack is succeeded. Second, a GNN with linear aggregation is also known as a graph convolutional network (GCN), which has manifested its effectiveness in a variety of real-world applications [38, 44, 117], necessitating an analysis on its robustness and reliability towards adversaries who may or may not have access to the training data.

**Opportunities.** For adversaries who have access to the training data and can freely manipulate the nodes and edges of the original graph, the attacking opportunities appear naturally. In particular, two key components of Eq. (4.2), *i.e.*, the graph connectivity represented by $\hat{\mathbf{A}}$ and the nodal feature denoted by $\mathbf{X}$, can be deliberately distorted to perform *poisoning attack on graphs*, where the GNN models trained on the distorted graph cannot reach the same level of accuracy as on the clean data. More specifically, two types of poisoning attacks

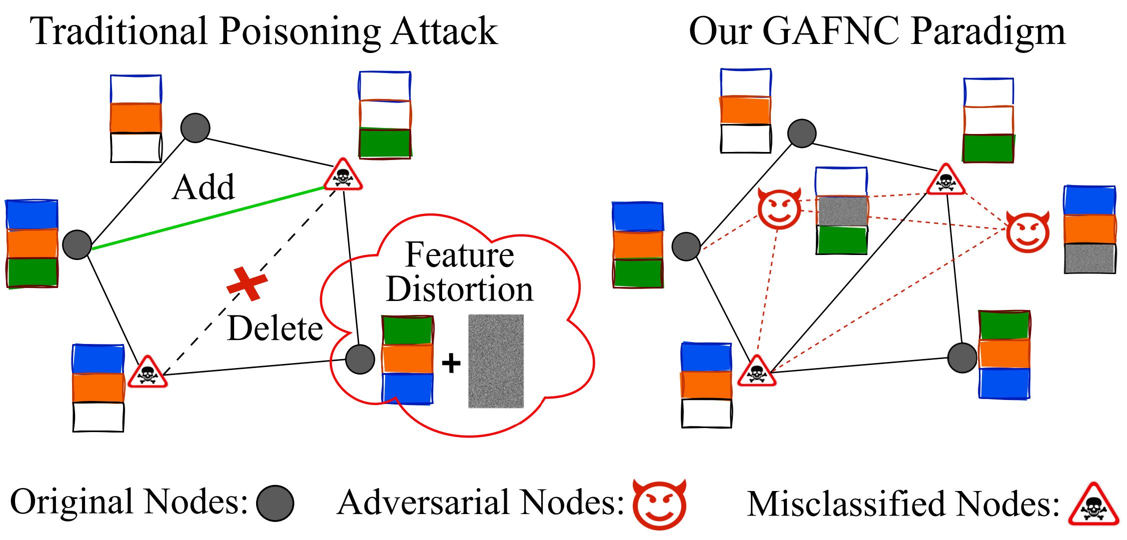


Figure 4.1: An illustrative comparison of traditional poisoning attacks on graph (left) and our GAFNC paradigm (right).

are considered, which take the forms as follows.

$$\text{Global Attack:} \quad \underset{\mathbf{A}',\mathbf{X}'}{\arg\max} \sum_{v_i \in \mathbf{V}} \mathrm{P}\left(y_i \neq \hat{y}_i \mid \mathbf{W}^1, \ldots, \mathbf{W}^M\right), \tag{4.4}$$

$$\text{Target Attack:} \quad \underset{\mathbf{A}',\mathbf{X}'}{\arg\max} \sum_{v_i \in \mathbf{V}_\Delta} \mathrm{P}\left(y_\Delta = \hat{y}_i \mid \mathbf{W}^1, \ldots, \mathbf{W}^M\right), \tag{4.5}$$

$$s.t. \quad \|\mathbf{A}' - \mathbf{A}\|_{\mathrm{F}} \leq \epsilon_A, \|\mathbf{X}' - \mathbf{X}\|_{\mathrm{F}} \leq \epsilon_X, \tag{4.6}$$

where Eq. (4.4) defines the *global attack* in which the task is to maximize the probability that the victim model gives incorrect predictions over all nodes uniformly, Eq. (4.5) defines the *target attack* in which the victim model is encouraged to predict a set of target nodes $\mathbf{V}_\Delta$ into prescribed classes $y_\Delta$. The constraints in Eq. (4.6) ensure that the distorted graph topology $\mathbf{A}'$ and nodal feature matrix $\mathbf{X}'$ do not differ from those of the original graph significantly and the distortions are within the Frobenius distances of $\epsilon_A$ and $\epsilon_X$, respectively. Intuitively, changing the graph connectivity to $\mathbf{A}'$, *e.g.*, adding new edges or deleting existing edges, is more effective than perturbing nodal features to $\mathbf{X}'$, as $\hat{\mathbf{A}}$ is involved in the computation of every layer while $\mathbf{X}$ is at the first layer only. A recent study further substantiate this

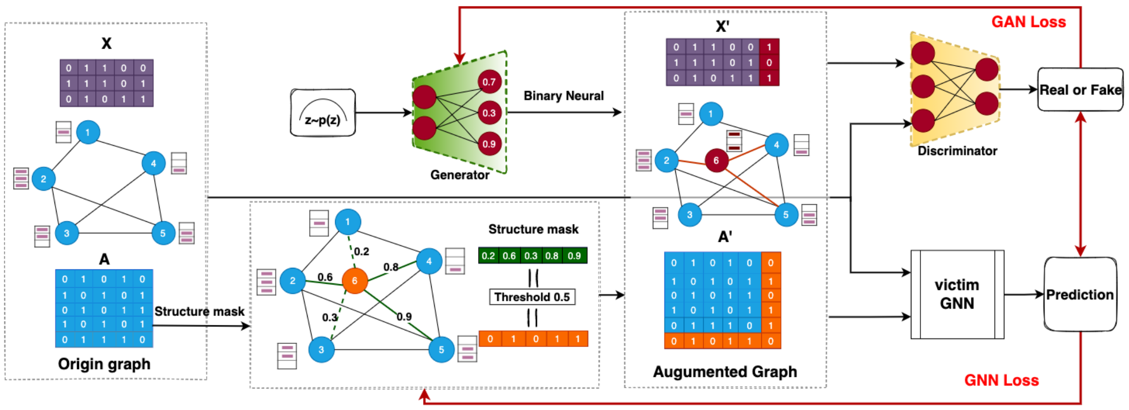


Figure 4.2: Pipeline of our proposed GAFNC. Blue colored are the original nodes and the corresponding adjacency matrix. Purple colored are the feature matrix of origin graph. Red colored are the new artificial node and its feature contents. Orange colored are the edges between the new node and the original graph and how the adjacency is augmented.

intuition [112]. Most existing poisoning attack methods can be framed in the modeling of Eqs. (4.4), (4.5), and (4.6).

**The Challenge and Limitation** The aforementioned threat model suffers from a prominent drawback – it requires access to the original graph data. Obtaining such access in real-world applications, however, can be very difficult. For example, to harm a scam detection system in an online social network such as Facebook or Twitter or a business network such as Amazon or ebay, the attackers have to hack into the central server or the user accounts, so as to perform data poisoning by manipulating the nodal features (*e.g.*, the contents that the users posted) or graph connectivity (*e.g.*, the follower-followee relationship), which is time-consuming and labour-costly and thus close to impossible in practice.

**Our Idea** To lift the limitation of requiring data access, we propose to add artificial nodes in the graph, letting their malicious messages propagate to their immediate contexts, and gradually to their $M$-hop neighbors by taking advantage of the message-passing of GNNs, such that the entire graph can be eventually poisoned. We term such an attacking paradigm *Generative Adversarial Fake Node Camouflaging* (GAFNC). Figure 4.1 illustrates a comparison between the traditional poisoning attack methods and our GAFNC proposal, where the main difference is that GAFNC does not request the permission of making any change on the original graph and hence is more practical. For example, an adversary can register many fake accounts in social/business networks, which is much cheaper and feasible than hacking into the server or real accounts.

The problem then is how would these artificial nodes look. To make the attack practical, the added nodes should not carry strong patterns, *e.g.*, a new account that keeps posting spam contents and/or sending out friend requests with high frequency – such nodes can be so easily detected by naïve network defending system such as an anomaly detector [118]. To this end, we in this study impose three requirements on the added nodes, so that our attack is *stealthy.*

1. The features of the artificial nodes follow a similar yet non-identical distribution as those of the original nodes, such that their malicious information is camouflaged.

2. The edges between the added nodes and the original graph are capable of effectively passing the poisonous messages of the new nodes yet should not be too dense; The sparsity of such new edges should be similar to that of the original graph.

3. The total number of added nodes should be small – otherwise, neatly crafting a large group of seemingly-authentic nodes would again be overly expensive.

In this work, we strictly restrict the number of new nodes to be less than 1% of the graph scale. To meet the three requirements, our key idea is to frame the node generation process in a generative adversarial network (GAN) regime, where the contents of the artificial nodes and their connectivity with the original graph are learned jointly by solving a bi-objective two-player minimax game.

## 4.3 Methodology

In this section, we elaborate the building blocks of our GAFNC paradigm, with the pipeline delineated in Figure 4.2. In a nutshell, two questions are solved in order to perform poisoning attacks on graphs without manipulating the original data. First, *what features should the added nodes carry*, such that they look authentic, can camouflage their poisonous contents, and would not be detected and rejected by simple classifiers running on graphs. To answer this question, the generative adversarial network (GAN) is employed to ensure that the features of the generated nodes follow a similar distribution of those of the original nodes, as shown in the top panel in Figure 4.2. We shall scrutinize this block in Section 4.3.1.

Second, *how should the added nodes connect to the original graph*, such that two types of poisoning attacks can be effectively implemented in which, for *global* attack, the goal is to maximize the accuracy degradation of the victim GNN classifier and, for *target* attack, the goal is to encourage the targeted node to be misclassified into a prescribed class. To this end, we draw insights from soft mask learning, which was devised for capturing important image regions in computer vision [119, 120], to devise the *edge mask learning* block, as shown in the bottom panel in Figure 4.2. The objective of this block is to identify a *sparse* topological structure, through which the malicious contents hidden in the newly added nodes can be propagated so as to poison the entire graph with high efficacy. The details of the edge mask learning are presented in Section 4.3.2.

### 4.3.1 Adversarial Node Generation

The learning objective of our GAFNC paradigm can be framed in a generic minimax game as follows.

$$\min_{D,\Phi} \max_{G} \mathcal{L}_{\text{GNN}}(G, \Phi, \mathcal{G}, \mathcal{G}') + \mathcal{L}_{\text{GAN}}(G, D, \mathcal{G}'), \tag{4.7}$$

where $G$, $D$, and $\Phi$ represent the generator, the discriminator, and the victim GNN model, respectively. Denoted by $\mathcal{G}' = (\mathbf{V}', \mathbf{A}', \mathbf{X}')$ is the augmented graph, where $\mathbf{V}' = \{v_1, \ldots, v_N, v_{N+1}, \ldots, v_{N+m}\}$ includes both the $N$ original nodes and the added $m$ artificial nodes, $\mathbf{A}' = \begin{bmatrix} \mathbf{A} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \in \mathbb{R}^{(N+m)\times(N+m)}$ in which $\mathbf{B}$ and $\mathbf{C}$ encode the edges between the old and new nodes and that among the new nodes, respectively, and $\mathbf{X}' = [\mathbf{X}, \mathbf{X}_{new}]^\top \in \mathbb{R}^{(N+m)\times D}$ in which $\mathbf{X}_{new}$ stacks the feature vectors of the artificial nodes. The goal of $G$ is to generate $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{X}_{new}$ that together deliver a stationary point of solving Eq. (4.7). The first term is the GNN loss term, which varies in accordance with the different attack settings and thus shall be detailed later on. The second term in Eq. (4.7) is the GAN loss term, which is defined as follows.

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D, \mathcal{G}') = \mathbb{E}_{v_i \in \mathbf{V}} \left[1 - \log D(\mathbf{A}, \mathbf{x}_i)\right] \\ +\mathbb{E}_{v_j \in \{v_{N+1}, \cdots, v_{N+m}\}} \left[\log \left(D(\mathbf{A}', G(\mathbf{z}_j))\right)\right],\end{aligned} \tag{4.8}$$

where $\mathbf{A}'$ and $\mathbf{z}_j$ are the generated connections and nodal features from $G$. The intuition behind Eq. (4.8) is that, with $D$ fixed, the second term of Eq. (4.8) with respect to $G$ is maximized such that the features of the nodes $v_{N+1}, \ldots, v_{N+m}$ that are generated by drawing from a latent code $z_j$ follows a similar distribution of those of the original nodes $\mathbf{V}$, striving to fool the classifier at the current round. In an alternative fashion, with $G$ fixed, minimizing Eq. (4.8) equals to maximizing the first term in which $D$ is guided to identify the original nodes and meanwhile minimizing the second term in which $D$ aims to screen out the generated nodes. A Graph Convolutional Network (GCN) is selected to implement the discriminator $D$ due to its capability of tracing changes in both graph topology and nodal features.

To implement the generator $G$, a fully-connected decoder would suffice in most tasks. However, we note that the nodes in several tasks may carry binary features, such as scholarly graphs, where each node represents a research article and 1 or 0 in a specific entry of the feature vector indicates the existence or not of the corresponding keyword [121, 122] and bioinformatics graphs, where each node is a chemical medicine with the binary features indicating whether or not a particular molecule is a part of this medicine. For such tasks, the decoder architecture would suffer from the discreteness and hence yield inferior performance in generating seemingly authentic nodes. To solve the issue, we adapt the idea of *binary neurons* suggested in the BinaryGAN [33], where the output layer of a generator was binarized through sigmoid-adjusted, straight-through estimators for generating data in continues domain. Two types of binary neurons, named *deterministic binary neuron* (DBN) and stochastic binary neuron (SBN), are both implemented in this work for the sake of efficacy. The formulation are as follows.

$$DBN(x) = \mu(\sigma(x) - 0.5), \tag{4.9}$$

$$SBN(x) = \mu(\sigma(x) - v), \quad v \sim U[0,1], \tag{4.10}$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ denote the unit step function and the sigmoid, respectively. $\sigma(x)$ is the preactivated outputs, and $U[0,1]$ denotes a uniform distribution.

### 4.3.2 Graph Poisoning with Edge Mask Learning

Thus far we have elaborated what features are in the newly added nodes, the question now is how to connect them to the original graph. Conceptually, for different attacking purposes, the resultant connectivities would also differ. As such, we in the next present details of how the two types of poisoning attacks are realized by *learning* the topological structure of the augmented graph via extrapolating the GNN loss term in Eq. (4.7).

**Global Attack** We first introduce the global attack, where the only objective is to lower the classification accuracy in general of a GNN model over all nodes uniformly. In practice, such an attack has a broad effect. For example, by making up a small set of seemly legitimate merchants, an online business infrastructure such as Amazon or ebay would weaken its capability of detecting scam sellers, incurring huge economic loss [123]. For another example, by spreading fake users across an online social network such as Facebook or Twitter, the platforms would lose their agility in screening bots and malicious users, so that rumors and fake news cannot be stopped in their early spreading stage, leaving space for attackers to corrupt democracy in the worst case [124, 125].

To realize the global attack, we define the GNN loss term in Eq. (4.7) that is to be maximized as follows,

$$\mathcal{L}_{\text{GNN}}(G, \Phi, \mathcal{G}, \mathcal{G}') = \sum_{v_j \in \mathbf{V}'} -y_i \log \hat{y}_i + \lambda_1 \Omega(\mathbf{S}) + \lambda_2 H(\mathbf{S}), \tag{4.11}$$

where $\hat{y}_i = P_\Phi(\mathbf{S} \odot \mathbf{K}(\mathbf{1}), \mathbf{X}')$ denotes the prediction of the node $i$ in the augmented graph. $\mathbf{S} \in \mathbb{R}^{(N+m)\times(N+m)}$ is the edge mask and $\mathbf{K}(\mathbf{1}) \in \mathbb{R}^{(N+m)\times(N+m)}$ stands for a full 1' matrix. With their scales controlled by tuned parameters $\lambda_1$ and $\lambda_2$ are the regularization terms, with $\Omega = \sum |(\sigma(\mathbf{S})|$ representing the sum of $\ell_1$ norm of the edge mask, and $H(\mathbf{S}) = -\Omega * \log(\Omega + \epsilon) - (1 - \Omega) * \log(1 - \Omega + \epsilon)$ with $\epsilon = 1e - 15$ is the entropy of $\Omega$. Note, to ensure that the topology of the original graph can be kept unchanged during optimization, we split our edge mask into two parts in implementation, namely, the part corresponding to the edges of the origin graph that is fixed as-is, and the part that is trainable representing the edges encoded by $\mathbf{B} \in \{0, 1\}^{N\times m}$ and $\mathbf{C} \in \{0, 1\}^{m\times m}$ in $\mathbf{A}'$.

The intuition behind Eq. (4.11) is to maximize the cross entropy of $y$ and $\hat{y}$. One more thing to note is that we exclude the output for the new node in the second term of equation Eq. (4.11) for two reasons. First, we cannot decide the groundtruth labels of newly added nodes at the initial iterations, as which label assignment would lead to the highest attacking efficacy remains unclear. Enforcing a priori assignment would introduce noises and thus slows down the training efficiency. Second, in this way we could align the numbers of the nodes that are unlabeled and need to be predicted in the original and augmented graph, launching a fair performance comparison of the GNN models, being either healthy or poisoned, in a transductive learning regime.

**Target Attack** The global attack leaves the question, *which nodes are misclassified into which classes* in a black-box, thereby preventing a further analysis on the GNN robustness. We now give details of the target attack, where the GNN is enforced to misclassify specified nodes into pre-designated classes. Given a target node $v_i$ with label $y_i = C_t$, the goal is to encourage a misclassification of this node such that $\hat{y}_i = C_k \neq C_t$, with $C_k$ being a desired class. The attack is straightforward and can be realized by defining the the GNN loss term in Eq. (4.7) as follows.

$$\mathcal{L}_{\text{GNN}}(G, \Phi, \mathcal{G}, \mathcal{G}') = \sum_{\substack{v_j \in \mathbf{V}', \\ v_i \in C_t}} -C_k \log \hat{y}_i + \lambda_1 \Omega(\mathbf{S}) + \lambda_2 H(\mathbf{S}), \tag{4.12}$$

where the predictive function of $\hat{y}_i$ and the two regularizers $\Omega(\mathbf{S})$ and $H(\mathbf{S})$ share the same definitions as in Eq. (4.11). The intuition behind Eq. (4.12) is to minimize the probability that the predicted label $\hat{y}_i$ is not the pre-designated class $C_k$, so that all nodes in the victim class $C_t$ are likely to be misclassified by the poisoned GNN.

## 4.4 Experiments

### 4.4.1 Data preparation

We benchmark our experiments on four real-world datasets, which are all widely-used in the literature. As a convention, we adapt the largest connected components search from [53] to filter out the very small sets of nodes in which the nodes are mutually connected and do not connect to the entire graph. The statistics of the datasets are summarized in the table 4.1 below.

| Datasets | # Nodes | # Edges | # Features | # Classes |
|---|---|---|---|---|
| Cora | 2485 | 5069 | 1433 | 7 |
| CiteSeer | 3327 | 4732 | 3703 | 6 |
| PubMed | 19,717 | 44,338 | 500 | 3 |
| Amazon | 7487 | 11,9043 | 745 | 8 |

Table 4.1: Statistics of the studied datasets

A detailed introduction of the studied datasets is given below.

- **Cora** [121] and **CiteSeer** [122] represent two citation network that contains a collection of scientific publications. Each node is a research article and is associated with a 0/1-valued word vector, where each entry represents the absence/presence of the corresponding unique word. As each article cites or is cited by other papers, the links between pairs of nodes are naturally formed. The dimension of the nodal vectors indicates the volume of the used dictionary. For example, in Cora each nodal vector is 1433-dimensioned, meaning that the dictionary encodes 1433 unique words in total. The learning task in both datasets is to classify the research articles into a number of research domains. For example, the class labels in Cora represent 7 domains in machine learning, *e.g.*, neural networks, probabilistic methods, genetic algorithms, and others. Our task in Cora is thus to synthesize a small number of articles that makes the GNN incapable of classifying existing articles into the 7 classes correspondingly.

- **PubMed** [126] is formed by extracting diabetes-related publications from PubMed database. It differs from Cora and CiteSeer in three aspects. First is its edge sparsity, which is 6$\times$ higher than Cora and CiteSeer on average. Second is its finer learning granularity, as PubMed focuses on diabetes-related articles only with a larger sample size, while Cora and CiteSeer both cover a boarder research scope. Third, PubMed represents the nodal features in a denser modality, where each publication is described by a TF-IDF [127] weighted word vector in a 500-dimensional latent space. Per these unique properties of PubMed, the experiment carried out on it could substantiate the generalizability of our attacking approach.

- **Amazon** [128] is is a dataset profiling the co-purchase behaviors of Amazon users, with the nodes representing the products, the edges pinpointing the pairs of products that are frequently bought together, the nodal features being the product reviews encoded with the bag-of-words, and the class labels indicating the product category. Amazon has a significantly smaller sparsity compared to the other three datasets due to the Matthew effect in E-commerce [129], where the more frequently a pair of products has been co-purchased before, the more likely either one product in this pair is recommended if the other is purchased. A successful attacking on this dataset could hint the viability of applying our GAFNC paradigm to undermine the deep-learning-based recommender systems [130], helping the online business users to build and uplift the awareness of analyzing and reinforcing the robustness of their systems.

### 4.4.2 Experiment design

To fully investigate the roles that different building blocks play in our GAFNC pipeline, ablation study is a must. To this end, we devise five model variants (denoted by V1 – V5) by controlling different blocks, based on the combination of imposing attacks on graph topology and nodal features along with randomness, as summarized in the below table.

| | V1 | V2 | V3 | V4 | V5 | GAFNC |
|---|---|---|---|---|---|---|
| Features Learned? | ✗ | ✗ | ✓ | ✂ | ✂ | ✓ |
| Edges Learned? | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ |

Table 4.2: Model Variants

**V1**: *Purely Random Attack*, where a number of new nodes with randomly generated features are connected to the original graph with randomly selected nodes.

**V2**: *Random feature attack with learned topology*, which differs from V1 in the sense that the new nodes having randomly generated features are learned to strategically connect to the original in a way that maximizes the attacking efficacy.

**V3**: *Random topology attack with learned features*, which is parallel to V2 with randomly connected edges yet learnable nodal features.

**V4**: *Random topology attack with sampled features*, which differs from V3 by having the newly added nodes sampled from the original graph.

**V5**: *Learnable topology attack with sampled features*, which evolves from V4 with the connectivities between the new nodes and original graph learned with edge masks.

**GAFNC**, which further upgrades V5 with a higher degree of freedom, both nodal features and the topology of augmented graph are purely learned from data through the method described in Section 4.3.

### 4.4.3 Setting and evaluation metrics

The computational environment is Ubuntu 18.04.5 LTS with CPUs of Intel Xeon Gold 6248R and GPUs of NVIDIA Tesla V100S (on a 4096-bit memory bus), 376GB of RAM and 50TB of HDD. The victim model is implemented as a 2-layer GCN with *ReLU* as activation in a hidden dimension of 16 and *IdenticalPool* as the readout in the last layer. To ensure comparison fairness, we fixed the number of maximum training epochs at 300 with a .005 learning rate. All experiments were conducted by splitting the datasets with a training/validation/test ratio of 7:1.5:1.5. To satisfy a semi-supervised setting in node classification, 20% nodes in the training set are associated with labels and 80% training nodes remain unlabeled. To offset randomness, 10-fold cross-validation is executed by shuffling the split and averaged results were reported. For all experiments, a maximum ratio of 0.44% adversarial nodes were added to perform the attack.

To carry out a comprehensive evaluation, we employ Accuracy, Precision, Recall, and Macro-F1 to evaluate the model performance and Attack Success Rate (ASR) to gauge the attacking efficacy. In a nutshell, to respect the multi-class nature of the studied datasets, Precision, Recall, and Macro-F1 that give an extrapolation of correct classification ratio in finer-level of granularity can eliminate the inherent bias of the commonly used Accuracy. For the global attack, ASR indicates the overall misclassification rate; for the target attack, ASR summarizes the rate to misclassify nodes into the target class.

To carry out a comprehensive evaluation, we employ accuracy, precision, recall, and macro-F1 as the four metrics to gauge the model performance. Due to the multi-class nature of the studied datasets, we decompose the classification problem into multiple binary classification subproblems, in each of which one class is selected as the target class (positive) and the others are merged into a non-target class (negative). Note, each subproblem suffers from imbalance, where the number of samples in the negative class significantly outweighs that of the target class. As such, we define a few terms at first, and then introduce the intuition behind these metrics. True Positive (TP) and True Negative (TN) denote correct predictions, where the model prediction aligns with the groundtruth. False Positive (FP) and False Negative (FN) indicate misclassifications, where FP means the model predicts the input as positive but the true label is negative, and FN is vice versa.

**Accuracy** $= (TP + TN)/(TP + TN + FP + FN)$, which is the most commonly-used and straightforward metric in node classification. It defines how likely the model would give correct prediction in general, yet may introduce bias in imbalanced or multi-class settings.

**Precision** $= TP/(TP+FP)$ and **Recall** $= TP/(TP+FN)$ decompose the Accuracy metric into a finer-level of granularity by favoring the positive class with small sample size, where Precision measures how likely a predicted positive sample is indeed positive, and Recall gives the ratio of correctly predicted positive samples in that class.

**Macro-F1** $= 2 * Precision * Recall/(Precision + Recall)$ harmonizes Precision and Recall and is a more comprehensive and robust metric. We would like to note that, although Accuracy is mostly used in the literature, lowering the model performance with poisoning attacks is relatively easy under the Accuracy metric, as in many real tasks there exists pairs of classes being naturally difficult to distinguish (*e.g.*, ML, AI, and IR in CiteSeer). Macro-F1 that gives a better measure of incorrectly classified cases is thus a better metric to evaluate our attacking efficacy on.

## 4.5 Results and Conclusion

### 4.5.1 Results

Two Tables and 16 Figures in this section present the experimental results, from which we answer the four research questions as follows.

| Datasets | Baseline | V1 | V2 | V3 | V4 | V5 | Mettack | Our-Global | Our-Target |
|---|---|---|---|---|---|---|---|---|---|
| Cora | 0.150 | 0.215 | 0.430 | 0.230 | 0.229 | 0.333 | 0.259 | **0.488** | **0.714** |
| CiteSeer | 0.310 | 0.320 | 0.430 | 0.445 | 0.353 | 0.401 | 0.315 | **0.697** | **0.717** |
| PubMed | 0.122 | 0.418 | 0.484 | 0.485 | 0.438 | 0.476 | 0.192 | **0.516** | **0.537** |
| Amazon | 0.061 | 0.136 | 0.278 | 0.216 | 0.118 | 0.186 | - | **0.369** | **0.405** |

Table 4.3: Comparative results of the victim GNN and our GAFNC approach and its five variants and Mettack on four real-world graph datasets in terms of Attack Success Rate (ASR). "Baseline" is the error rate of the GNN trained on clear, unpoisoned data.

**Q1.** *How vulnerable is GNN to our GAFNC attack?*

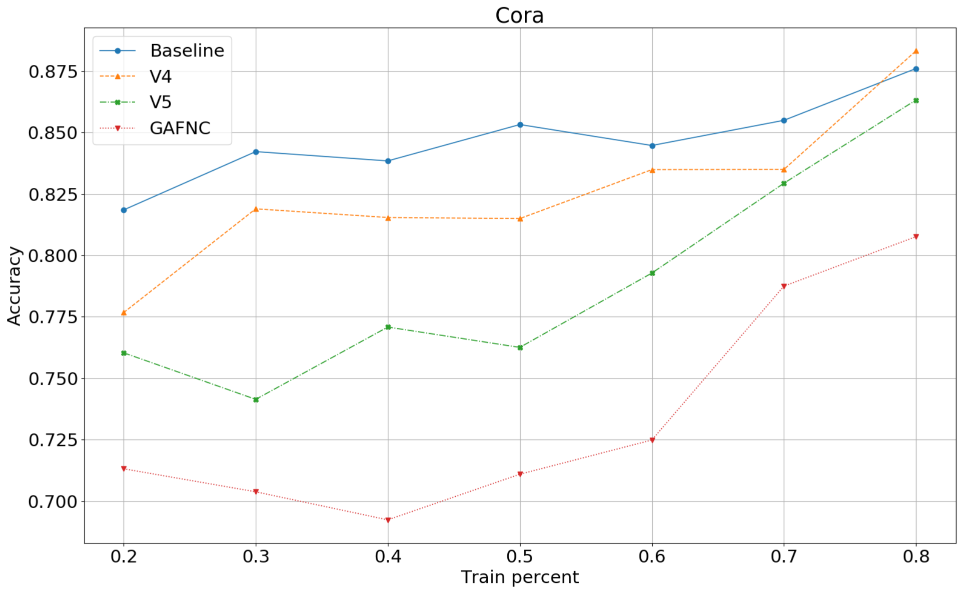


Figure 4.3: Performance of the classification accuracy of the baseline GNN on the clear datasets and the datasets poisoned by our GAFNC methods and its V4 and V5 variants, where the ratios of training data to the original datasets vary from 20% to 80%. A fixed number of 20 adversarial nodes were added to perform the poisoning attack on Cora.

The answer is immediate from the comparison between our GAFNC approach and the baseline in Table 4.3, where our approach leads to an increased ASR of 72.77%, with a 71.06% global ASR and a 74.49% target ASR on average. We note that such an attack efficacy is achieved by adding only 20 adversarial nodes, revealing the vulnerability of GNN

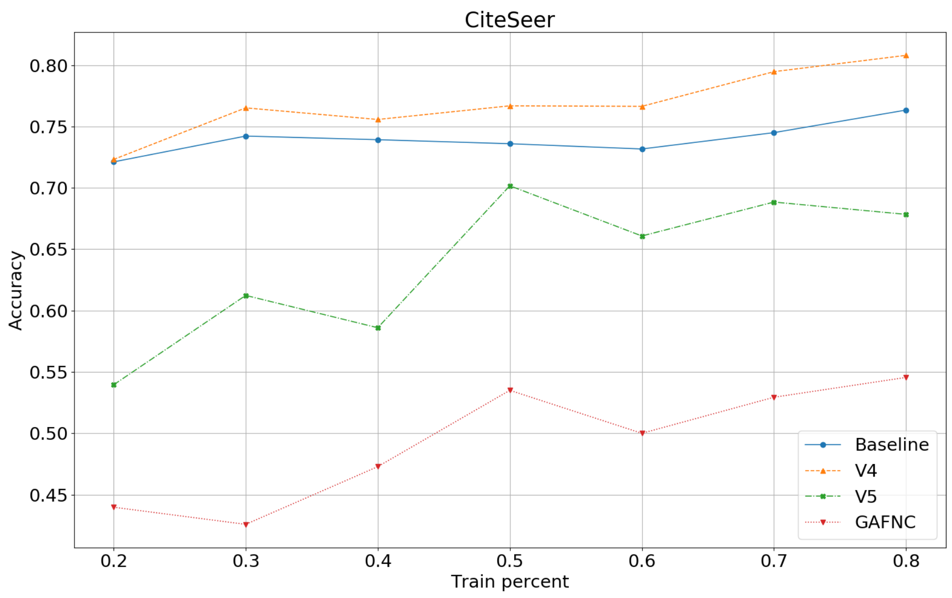


Figure 4.4: Performance of the classification accuracy of the baseline GNN on the clear datasets and the datasets poisoned by our GAFNC methods and its V4 and V5 variants, where the ratios of training data to the original datasets vary from 20% to 80%. A fixed number of 20 adversarial nodes were added to perform the poisoning attack on CiteSeer.

models because they can be attacked at very low cost in our generative-adversarial poisoning regime.

To further explore this question, we make the following observations from Table 4.3, Figures 4.3, 4.4, 4.5, and 4.6. *First*, as the node classification task is semi-supervised, given a larger number of training nodes can intuitively leads to a better model performance. The increasing trends of classification accuracy with a larger proportion of training nodes in Figures 4.3, 4.4, 4.5, and 4.6 validate this intuition. However, our GAFNC lead to the most flat increasing trends, meaning that our GAFNC attack keeps its efficacy even if more groundtruth labels are given in the training phase. *Second*, the attacking efficacy of V4 and V5 is affected by the training data ratios largely. In Cora and CiteSeer, V4 and V5 even improved the classification accuracy of the victim GNN. This phenomena is also termed as the positive effect of adversarial training [57], where a jointly learning of benign and malicious nodes can increase the robustness of GNNs. Our GAFNC does not suffer this issue and can

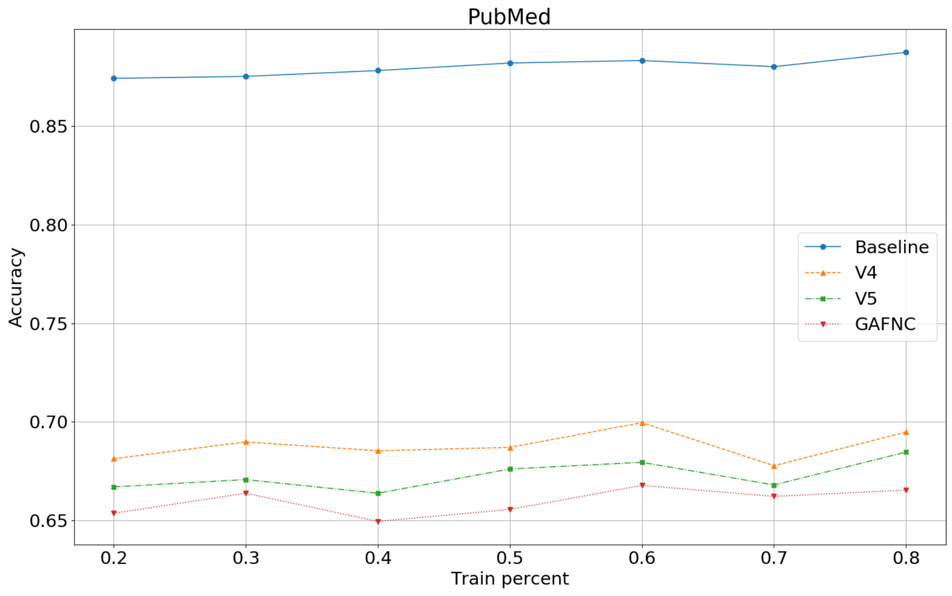


Figure 4.5: Performance of the classification accuracy of the baseline GNN on the clear datasets and the datasets poisoned by our GAFNC methods and its V4 and V5 variants, where the ratios of training data to the original datasets vary from 20% to 80%. A fixed number of 20 adversarial nodes were added to perform the poisoning attack on PubMed.

always perform effective poisoning however the imbalanced ratio between the benign nodes (*e.g.*, 1988 in Cora and 15,773 in PubMed) for training and the added adversarial nodes (*i.e.*, 20 in our study). This suggests that adding more healthy nodes cannot uplift the classification performance of the GNN models once being poisoned by our GAFNC attack.

We further added a more strong baseline Mettack [62] for comparison. The experiment source code is downloaded from the work Deep Robust [131]. And the setting of the method is default with a perturbation ratio of 0.05. We did not report the result of Mettack on Amazon as it did not finish within a reasonable time due to limited computer resources. Comparing Mettack with five model variants can show that adding poisoning node attack through edge mask learning and generative of feature matrix is more efficient compared with the meta-learning for disturbance of origin graph.

**Q2.** *How poisonous are the adversarial nodes?*

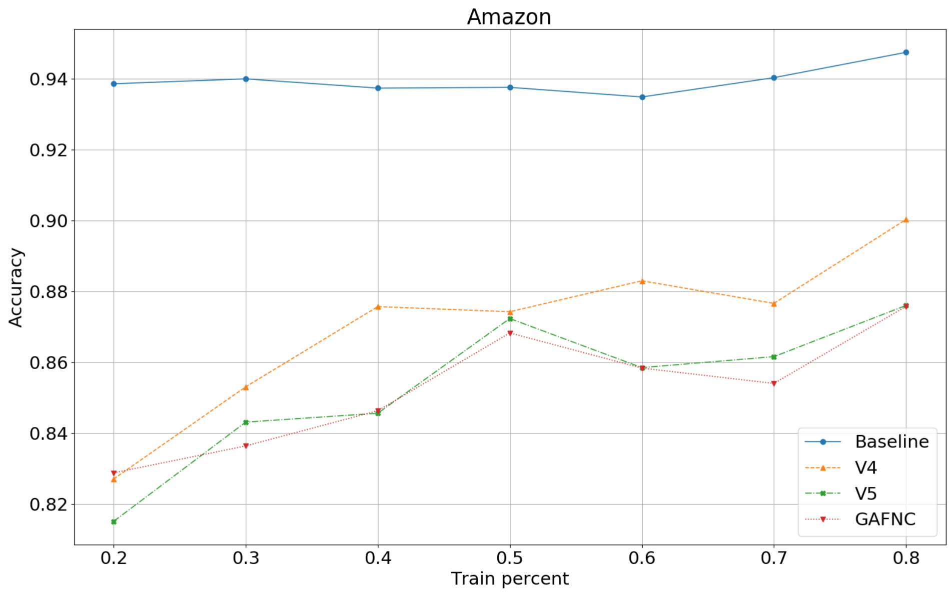


Figure 4.6: Performance of the classification accuracy of the baseline GNN on the clear datasets and the datasets poisoned by our GAFNC methods and its V4 and V5 variants, where the ratios of training data to the original datasets vary from 20% to 80%. A fixed number of 20 adversarial nodes were added to perform the poisoning attack on Amazon.

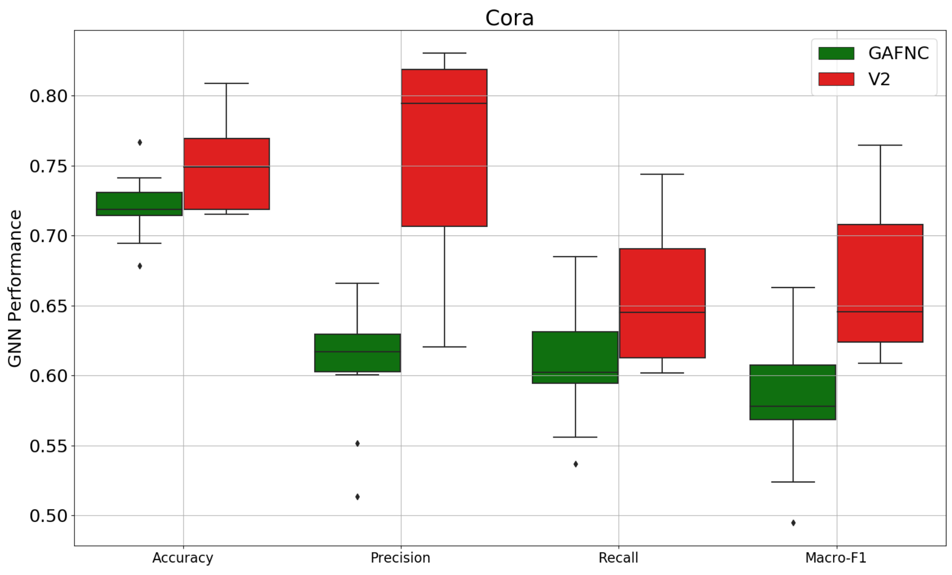


Figure 4.7: A boxplot comparison between GAFNC and V2 with a 25th – 75th interquartile range on Cora.

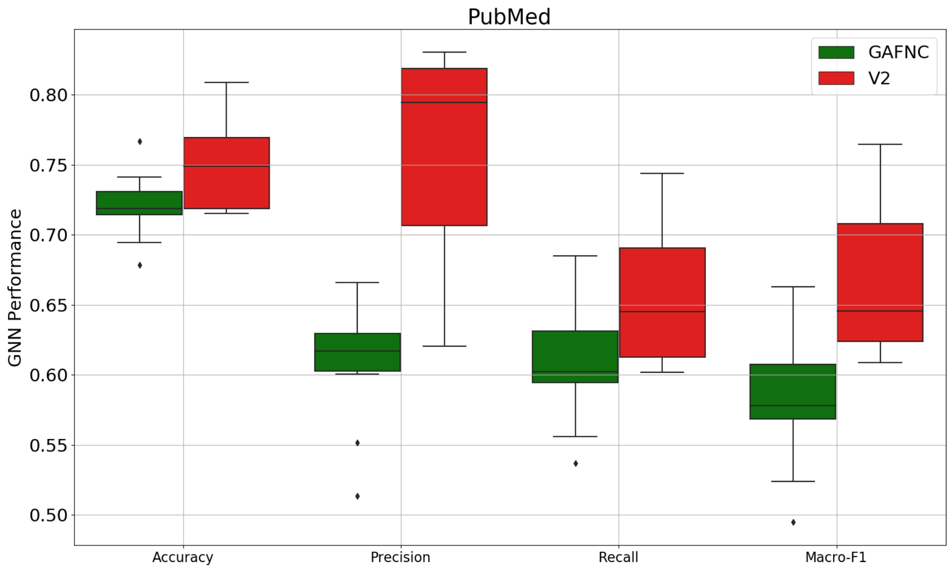


Figure 4.8: A boxplot comparison between GAFNC and V2 with a 25th – 75th interquartile range on PubMed.

The answer are provided in two aspects, namely, the attacking efficacy and the attacking robustness. The efficacy is extracted by comparing our GAFNC to V5 and comparing V3 to V1, where the impact of graph topology is controlled. First, with both learned topology, we observe from Table 4.3 that GAFNC enjoys a 16.31% higher ASR than V5 on average, which suggests that, with strategical connections solely, our GAN-generated nodes carry more poisonous contents. Second, with both random connection, V3 outperforms V1 in terms of ASR by 13.1%, which indicates that, even being connected to the original graph randomly, the GAN-generated nodes may also achieve malicious purposes because of their poisonous features. In addition, comparing GAFNC to V2 as shown in Figure 4.7, 4.8, we observe that with a fixed topological structure, the features learned by GAFNC makes the attacking efficacy more *robust*, where the performance variances of the resultant model are significantly smaller across all metrics.

**Q3.** *How effective can edge mask generate poisoning connections?*

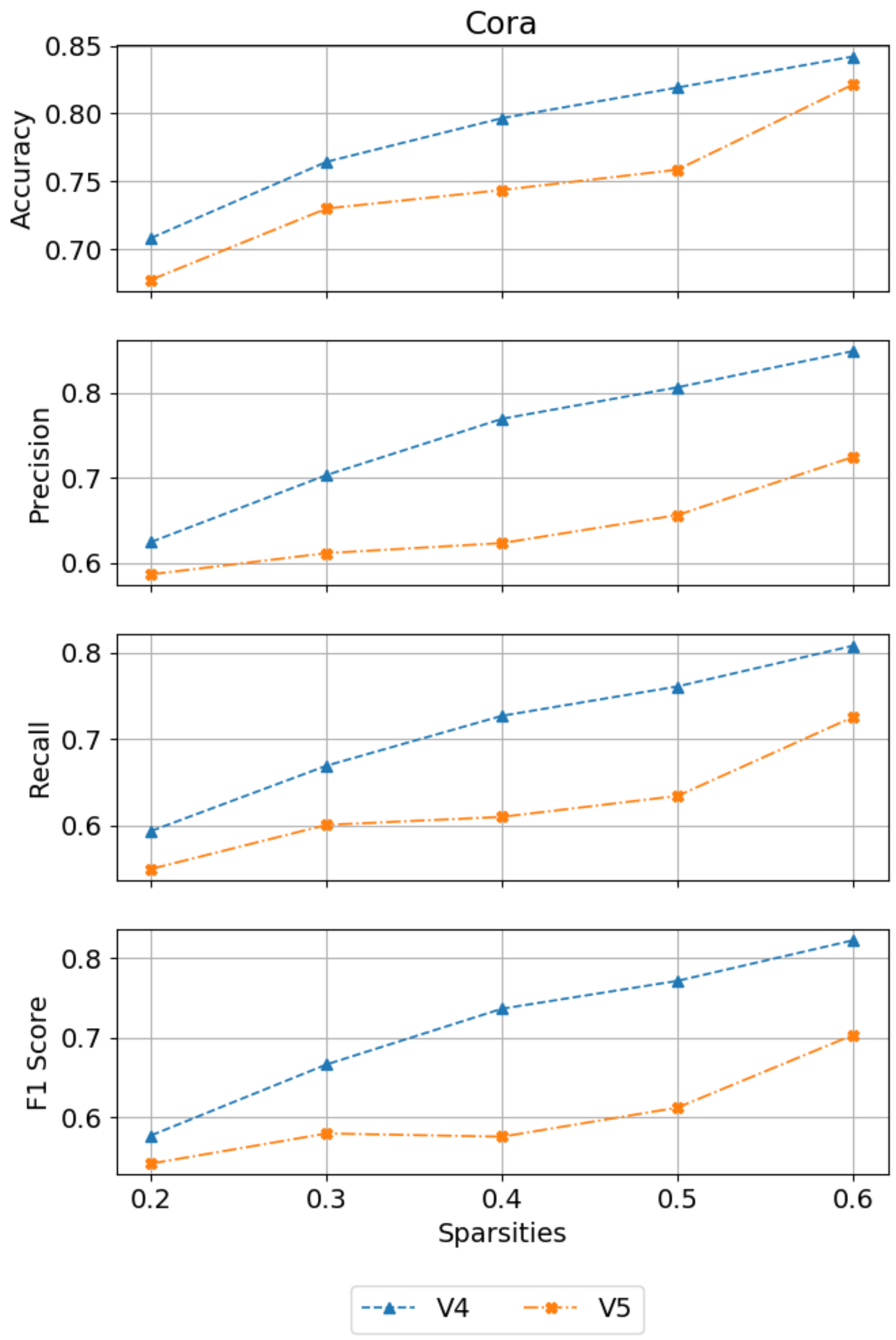


Figure 4.9: Attacking efficacy in four datasets *w.r.t.* different Sparsities on Cora

First, we compared the effect of V4 to V5 with nodal features both randomly generated. From Figures 4.9, 4.10, 4.11, and 4.12, we can seen that, our learned edges are more effective

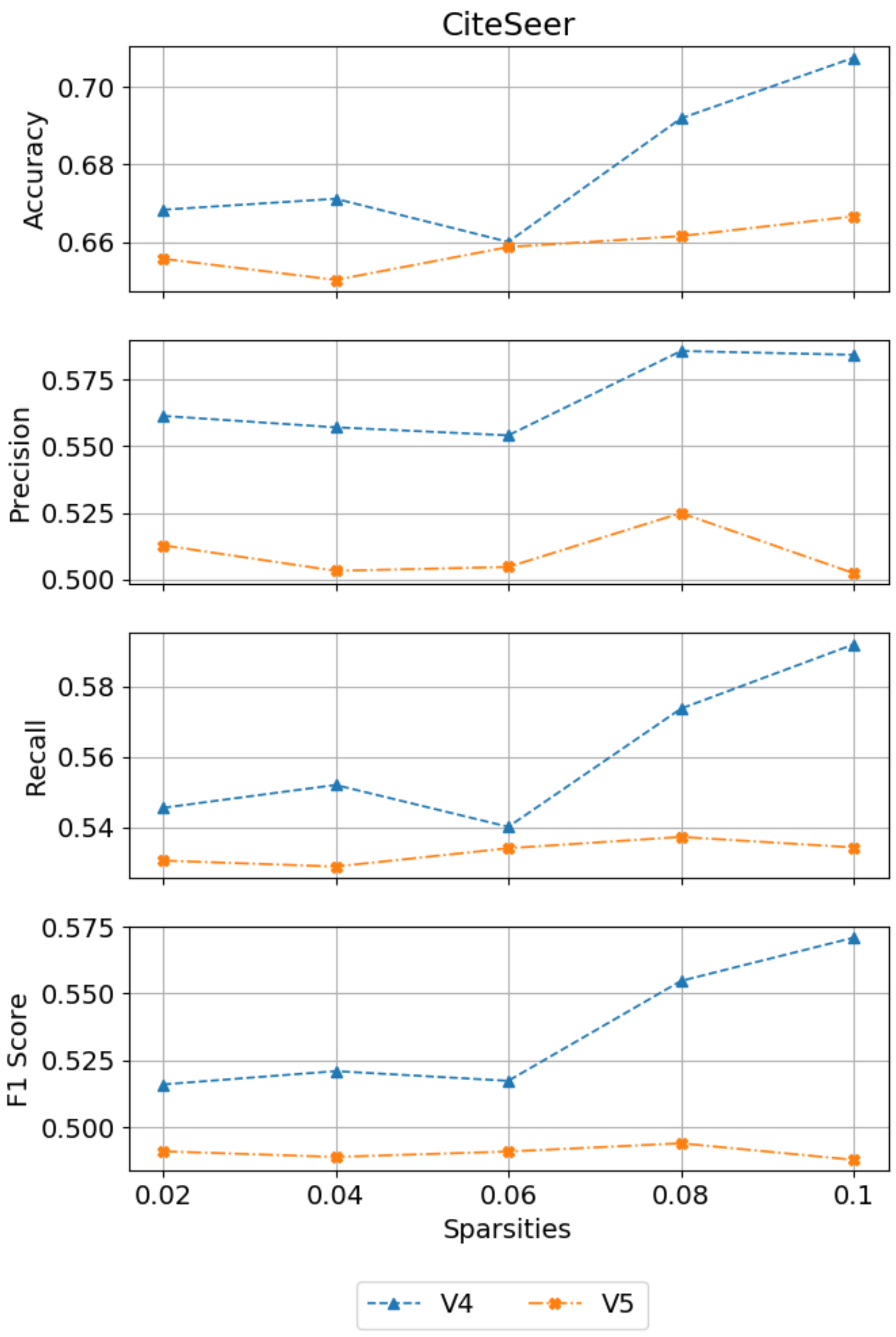


Figure 4.10: Attacking efficacy in four datasets *w.r.t.* different Sparsities on CiteSeer

compared with randomly connections among four datasets with respect to four evaluation metrics Accuracy, Precision, Recall and F1-Score. Also, we control the effect of poisonous

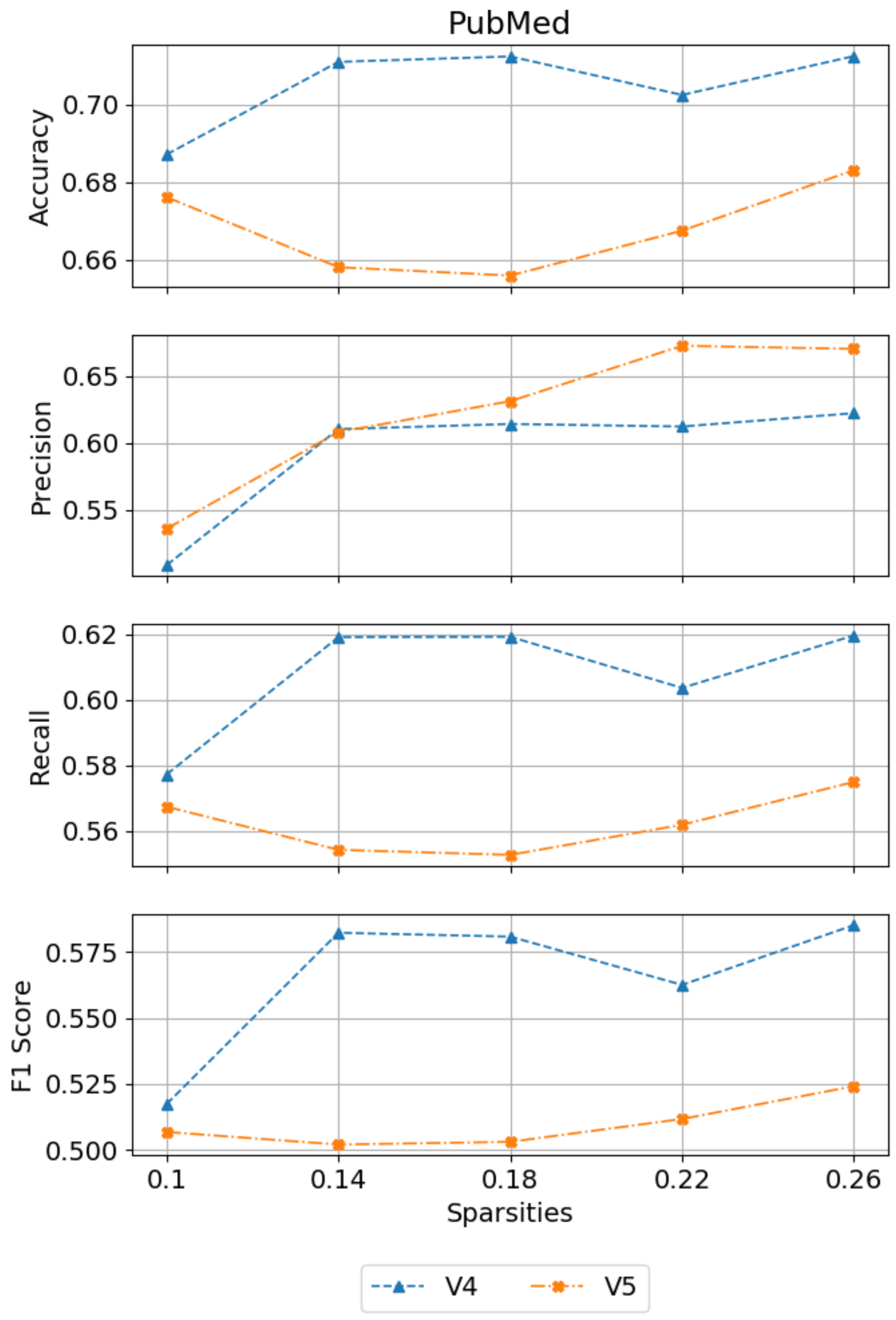


Figure 4.11: Attacking efficacy in four datasets *w.r.t.* different Sparsities on PubMed

nodal features by comparing GAFNC to V3 and V2 to V1. We observe from Table 4.3 that GAFNC and V2 excel in ratios of 18.22% and 21.64% to their competitors, respectively.

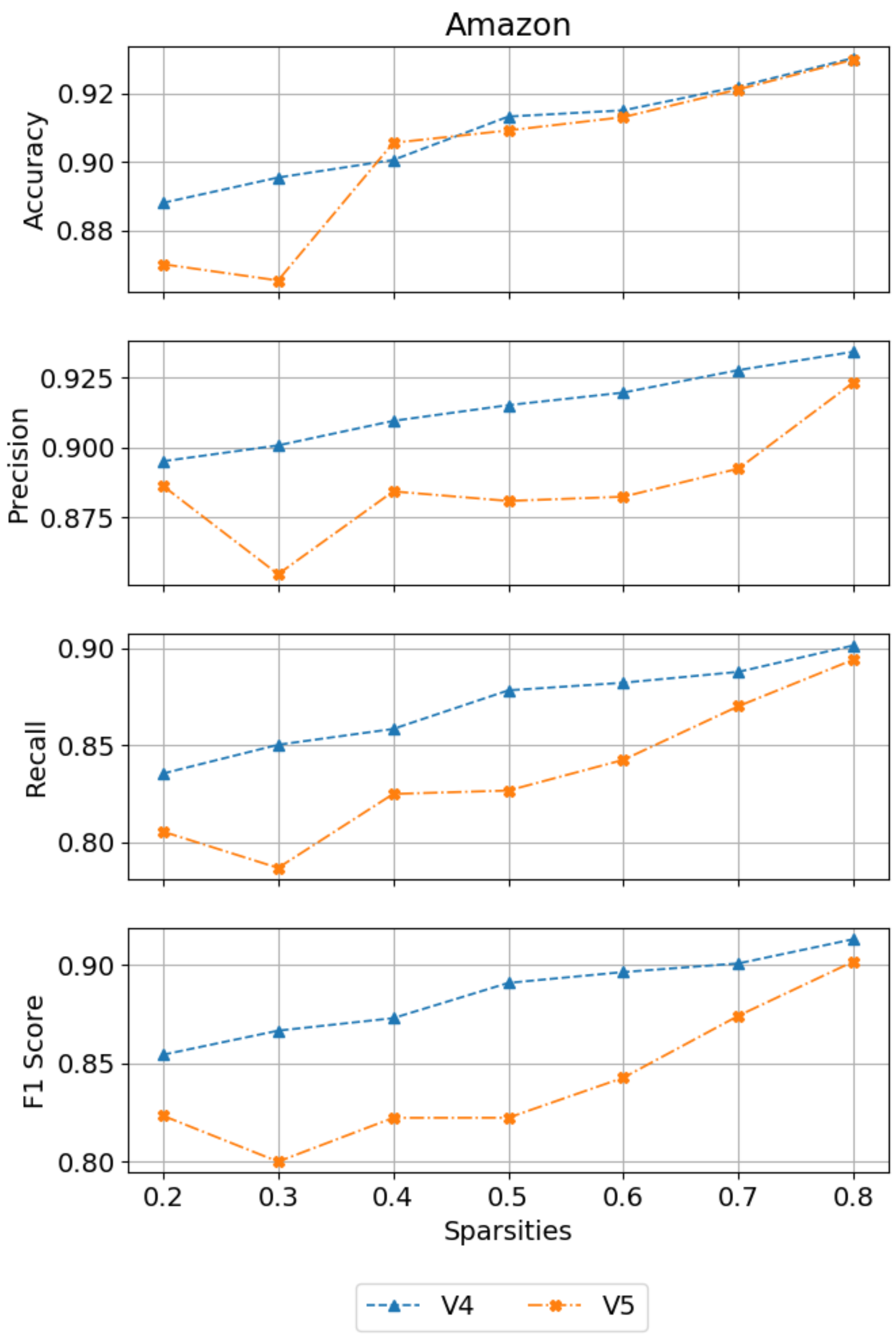


Figure 4.12: Attacking efficacy in four datasets *w.r.t.* different Sparsities on Amazon

This finding reveals that the edge mask can learn how to connect the added nodes to the original graph in a way that their malicious contents (being smartly generated in GAFNC or randomly generated in V2) can be propagated so as to poison the entire graph most effectively. In addition, our GAFNC ends up with the highest ASR in CiteSeer with a 20.22% higher ratio than in the other three datasets, where the key attribute that differentiates CiteSeer from the other three is its substantially lower sparsity. This shows that the graph density matters, where a denser graph is more flexible because of the choices of routes are many, but in a sparse graph the poisonous messages cannot be effectively propagated without strategical edges. The edge mask in GAFNC provides such an apparatus to place the connections between the new nodes and the original graph strategically, thereby arriving at the remarkable attacking success rates.

**Q4.** *How stealthy is our GAFNC poisoning attack?*

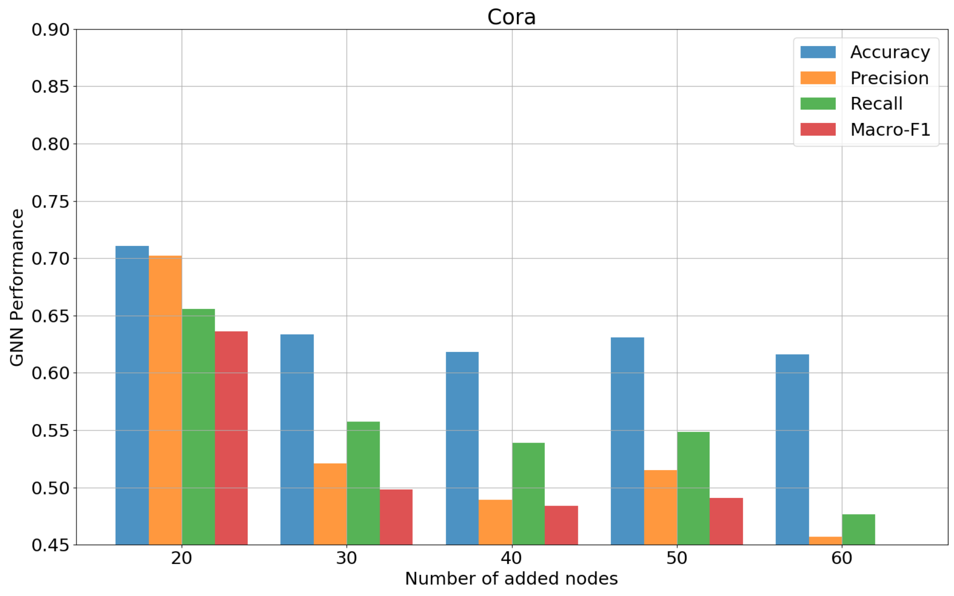


Figure 4.13: Attacking efficacy in four datasets *w.r.t.* different numbers of added nodes on Cora.

The stealthiness of the adversarial nodes added by our GAFNC approach comes from three aspects.

(1) The total number of added nodes is very small, where we note that all the empirical studies were conducted with only 20 adversarial nodes added and connected. As shown in

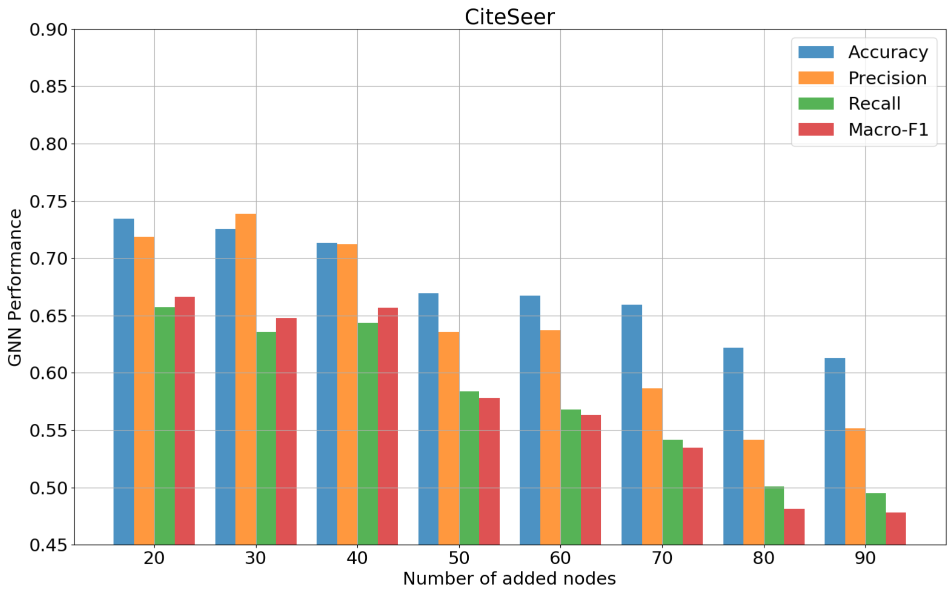


Figure 4.14: Attacking efficacy in four datasets *w.r.t.* different numbers of added nodes on CiteSeer.

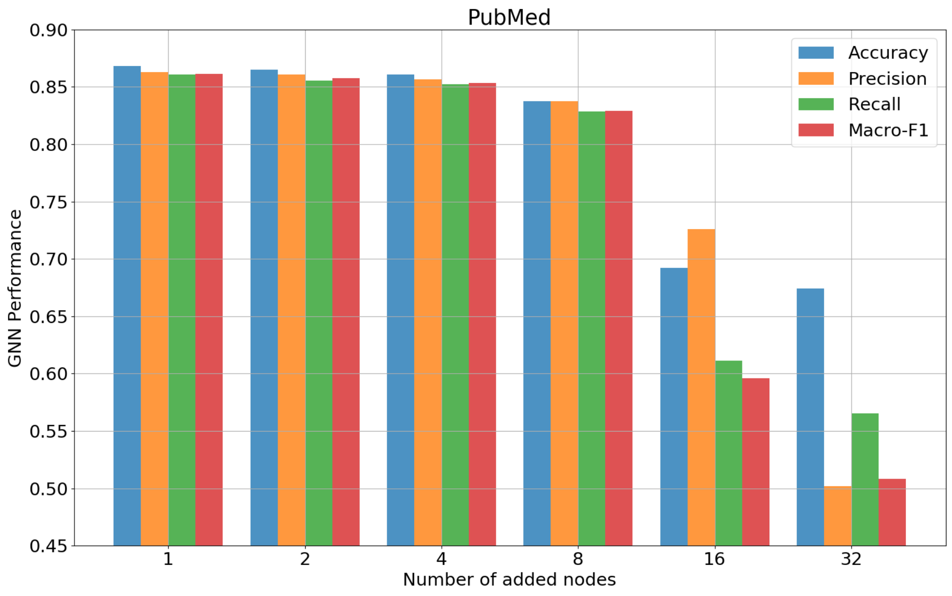


Figure 4.15: Attacking efficacy in four datasets *w.r.t.* different numbers of added nodes on PubMed.

Figures 4.13, 4.14, 4.15, and 4.16, with more nodes added, the performance of the victim GNN can be further degraded, with the macro-F1 ends up with 23.93%, 15.13%, 40.98%,

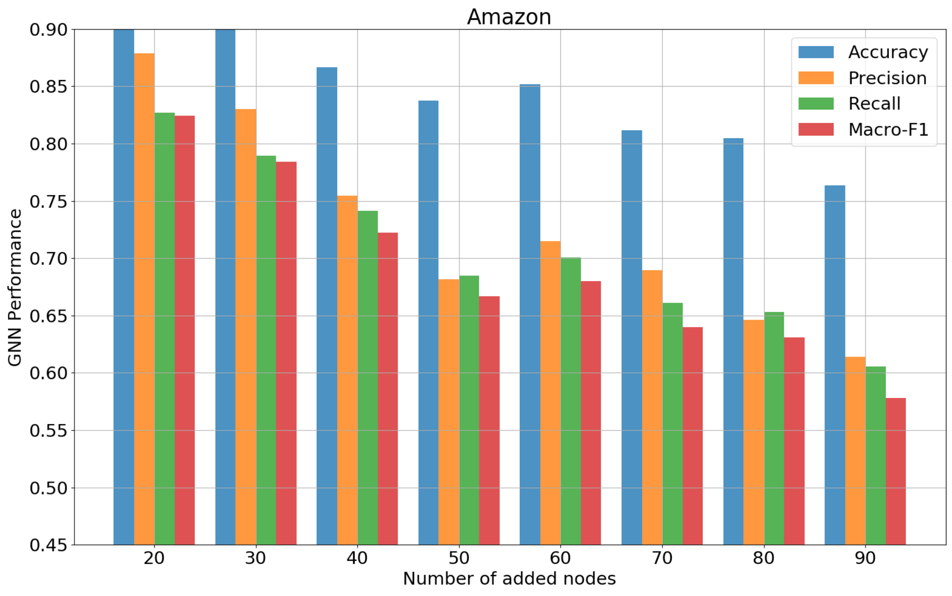


Figure 4.16: Attacking efficacy in four datasets *w.r.t.* different numbers of added nodes on Amazon.

and 29.87% in Cora, CiteSeer, PubMed, and Amazon, respectively, if we increase the number of added nodes to 1% of the original graph. Yet, this ratio of added nodes is still rather negligible.

(2) The nodes generated by GAFNC are seemingly-authentic, *i.e.*, the new nodes follow a similar distribution as the original nodes. To see this, a state-of-the-art outlier detector COPOD [132] is applied to distinguish our generated nodes from the nodes in the original datasets. We first try COPOD on the nodes with randomly generated features, where COPOD is capable of detecting those nodes 100% from the original ones. However, with our GAFNC nodes, COPOD ends up with a 12% detection ratio, leaving most of the adversarial nodes undetected. This validates that the malicious features of our GAFNC generated nodes are camouflaged and can hence perform poisoning attacks stealthily.

(3) Our GAFNC approach allows a target attack beyond 1-hop. A qualitative result from the Cora dataset is exemplified in Figure 4.17, where the objective of the target attack is to encourage a misclassification of Node #1336 into a prescribed class. After 1973 epochs of

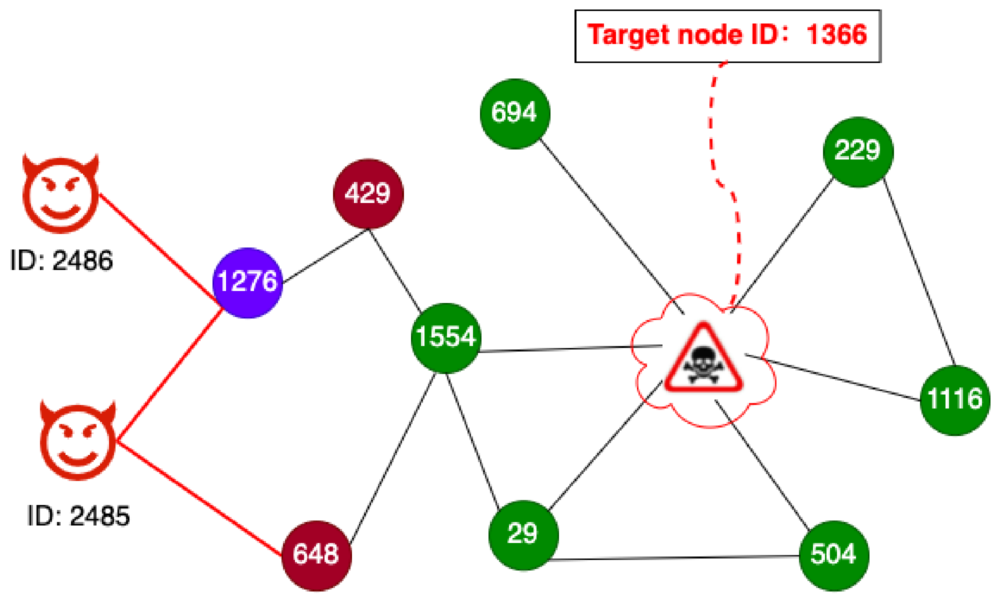


Figure 4.17: A successful 3-hop target attack in the Cora dataset.

poisoning, we observe that by adding two adversarial nodes as the second- and third-order neighbors of the victim node, the malicious messages can be propagated in such a stealthy path that the immediate neighbors of the Node #1336 can still be classified correctly by the poisoned GNN, but still leads to the misclassification of the victim #1336. Considering the large size of neighborhood of each node in more than 1-hop, such a *multi-hop* target attack make our poisoning approach even more difficult to be detected in practice.

### 4.5.2 Discussion and Conclusion

This work propose a novel attacking paradigm, named Generative Adversarial Fake Node Camouflaging (GAFNC), its crux lying in crafting a set of fake nodes in a generative-adversarial regime. These nodes carry camouflaged malicious features and can poison the victim GNN by passing their malicious messages to the original graph via learned topological structures. They 1) maximize the devastation of classification accuracy (i.e., global attack) or 2) enforce the victim GNN to misclassify a targeted node set into prescribed classes (i.e., target attack). We benchmark our experiments on four real-world graph datasets, and the results substantiate the viability, effectiveness, and stealthiness of our proposed poisoning attack approach by evaluating four metrics.

# Chapter 5

# Conclusion and Future Work

## 5.1 Conclusion

In this dissertation, we focus on using a GAN technique for dealing with graph data. Through two case studies, we prove that GAN can benefit from graph data preprocessing and contribute to adversarial data poisoning attacks on GNNs.

In the first work, we described a novel framework that utilized a generative-based deep neural network to distinguish the unknown associations in the raw training graph. Two Generative Adversarial Network models, NetGAN and CELL, were adopted for the edge classification (ie, link prediction), leveraging unlabeled link information based on a real knowledge graph built from LitCovid and Pubtator. The performance of link prediction, especially in the extreme case of training data versus test data at a ratio of 1: 9, demonstrated that the promised method still achieved favorable results (AUCROC $> 0.8$ for synthetic and 0.7 for real dataset) despite the limited amount of testing data available. Our preliminary findings showed the proposed framework achieved promising results for removing noise in data preprocessing of the biomedical knowledge graph improves the performance of downstream applications by providing cleaner data.

In the second work, we explored the robustness of graph neural network (GNN) in node classification from an adversary perspective. A novel data poisoning attack paradigm on graphs, named Generative Adversarial Fake Node Camouflaging (GAFNC), was proposed, with its essence being the creation of adversarial nodes that are seemingly authentic yet carry malicious contents. The key idea of GAFNC is to leverage the message-passing mechanism of GNN, such that the connectivity between this set of crafted nodes and the original graph is established in a way that expedites the propagation of the hidden malicious contents over the entire graph. Two attacking goals were realized, namely, 1) global attack, where the GNN trained on such poisoned graph would suffer poor prediction performance in general, with the margin to that of a healthy GNN is maximized, and 2) target attack, where the GNN is encouraged to misclassify a target node into a prescribed class. The practicality of our proposal lies in that, compared to the prior arts requiring to either distort the nodal features or add/delete the existing linkages or both, our GAFNC paradigm does not entail any manipulation on the original graph (note that such manipulation could be very expensive in practice such as hacking into an internet infrastructure *e.g.*, Amazon or Facebook). Empirical study on four widely used, real-world graph datasets and extensive ablation studies together substantiated the viability, effectiveness, and stealthiness of our proposed GAFNC poisoning attack approach.

## 5.2 Future Work

- I aim to continue working on the denoising of the knowledge graph and to explore the potential solution in two directions with respect to the effectiveness and efficiency of the current framework. First, to utilize existing knowledge graphs through transfer learning so that we can make better decisions by considering prior experience. Second, to design more advanced generative models for generating graphs, as the previous generation of graphs through random walks are not easy to parallel and are time-consuming. For example, using binaryGAN[102] to create an adjacent matrix as a graph would be an alternative solution.

- I aim to continue my work on the robustness of GNNs. First, to focus on injecting domain knowledge into the design of GNNs for explainability or interpretability. For example, for drug-drug protein interaction networks, we can first design the rules based on the biological properties of small molecules that interact with one another. Then, we can produce the hyper-graph for future small molecules predictions if GNNs can capture the prior knowledge. Second, to work on the defense model for detecting the newly added poison nodes.

# Bibliography

[1] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2016.

[2] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE international conference on computer vision*, pages 2745–2754, 2017.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[4] Anbang Xu, Zhe Liu, Yufan Guo, Vibha Sinha, and Rama Akkiraju. A new chatbot for customer service on social media. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 3506–3510, 2017.

[5] Matthew B Hoy. Alexa, siri, cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88, 2018.

[6] Muhamad Nova. Utilizing grammarly in evaluating academic writing: A narrative research on efl students' experience. *Premise: Journal of English Education and Applied Linguistics*, 7(1):80–96, 2018.

[7] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. How good is my gan? In *Proceedings of the European conference on computer vision (ECCV)*, pages 213–229, 2018.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[9] Veit Sandfort, Ke Yan, Perry J Pickhardt, and Ronald M Summers. Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. *Scientific reports*, 9(1):1–9, 2019.

[10] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[11] Yizhe Zhang, Zhe Gan, and Lawrence Carin. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*, volume 21, pages 21–32, 2016.

[12] Dawei Zhou, Lecheng Zheng, Jiejun Xu, and Jingrui He. Misc-gan: A multi-scale generative model for graphs. *Frontiers in big Data*, 2:3, 2019.

[13] Zachary Papanastasopoulos, Ravi K Samala, Heang-Ping Chan, Lubomir Hadjiiski, Chintana Paramagul, Mark A Helvie, and Colleen H Neal. Explainable ai for medical imaging: deep-learning cnn ensemble for classification of estrogen receptor status from breast mri. In *Medical imaging 2020: Computer-aided diagnosis*, volume 11314, pages 228–235. SPIE, 2020.

[14] Yang Zhou, Amnay Amimeur, Chao Jiang, Dejing Dou, Ruoming Jin, and Pengwei Wang. Density-aware local siamese autoencoder network embedding with autoencoder graph clustering. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1162–1167. IEEE, 2018.

[15] Yang Zhou, Sixing Wu, Chao Jiang, Zijie Zhang, Dejing Dou, Ruoming Jin, and Pengwei Wang. Density-adaptive local edge representation learning with generative adversarial network multi-label edge classification. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1464–1469, 2018.

[16] Yang Zhou, Chao Jiang, Zijie Zhang, Dejing Dou, Ruoming Jin, and Pengwei Wang. Integrating local vertex/edge embedding via deep matrix fusion and siamese multi-label classification. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1018–1027. IEEE, 2019.

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *stat*, 1050:20, 2017.

[19] Wikipedia. DeepScale-free network, 2022.

[20] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *International Conference on Machine Learning*, pages 610–619. PMLR, 2018.

[21] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[22] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[23] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[24] Lillian J Ratliff, Samuel A Burden, and S Shankar Sastry. Characterization and computation of local nash equilibria in continuous games. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 917–924. IEEE, 2013.

[25] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling. *Likelihood, Adversary*, 11, 2015.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[27] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[28] Philip Bachman and Doina Precup. Data generation as sequential decision making. *Advances in Neural Information Processing Systems*, 28, 2015.

[29] Philip S Thomas and Emma Brunskill. Policy gradient methods for reinforcement learning with function approximation and action-dependent baselines. *arXiv preprint arXiv:1706.06643*, 2017.

[30] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[31] Xu-Wen Wang, Yize Chen, and Yang-Yu Liu. Link prediction through deep generative model. *Iscience*, 23(10):101626, 2020.

[32] Luca Rendsburg, Holger Heidrich, and Ulrike Von Luxburg. Netgan without gan: From random walks to low-rank approximations. In *International Conference on Machine Learning*, pages 8073–8082. PMLR, 2020.

[33] Hao-Wen Dong and Yi-Hsuan Yang. Training generative adversarial networks with binary neurons by end-to-end backpropagation, 2018.

[34] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005.

[35] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[36] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[37] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 2020.

[38] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[39] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *ICLR2014, CBLS, April 2014*, 2014.

[40] Shanshan Tang, Bo Li, and Haijun Yu. Chebnet: Efficient and stable constructions of deep neural networks with rectified power units using chebyshev approximations. *CoRR*, abs/1911.05467, 2019.

[41] Yi He, Sheng Chen, Thu Nguyen, Bruce A Wade, and Xindong Wu. Deep matrix tri-factorization: Mining vertex-wise interactions in multi-space attributed graphs. In *SDM*, pages 334–342. SIAM, 2020.

[42] Yi He, Xu Yuan, Nian-Feng Tzeng, and Xindong Wu. Active learning with multi-granular graph auto-encoder. In *ICDM*, pages 1058–1063. IEEE, 2020.

[43] James Atwood and Don Towsley. Diffusion-convolutional neural networks. NeurIPS'16, page 2001–2009, Red Hook, NY, USA, 2016. Curran Associates Inc.

[44] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *KDD*, pages 1416–1424, 2018.

[45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018.

[46] Min Shi, Yu Huang, Xingquan Zhu, Yufei Tang, Yuan Zhuang, and Jianxun Liu. GAEN: graph attention evolving networks. In Zhi-Hua Zhou, editor, *IJCAI*, pages 1541–1547. ijcai.org, 2021.

[47] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale, 2017.

[48] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, Los Alamitos, CA, USA, may 2017. IEEE Computer Society.

[49] Yi Xie, Cong Shi, Zhuohang Li, Jian Liu, Yingying Chen, and Bo Yuan. Real-time, universal, and robust adversarial attacks against speaker recognition systems. In *ICASSP*, pages 1738–1742. IEEE, 2020.

[50] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology*, 11(3):1–41, 2020.

[51] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[52] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[53] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *KDD*. ACM, 2018.

[54] Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. In *NeurIPS*, 2020.

[55] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *KDD*, pages 66–74, 2020.

[56] Uday Shankar Shanthamallu, Jayaraman J Thiagarajan, and Andreas Spanias. Uncertainty-matching graph neural networks to defend against poisoning attacks. In *AAAI*, volume 35, pages 9524–9532, 2021.

[57] Binghui Wang, Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of graph neural networks against adversarial structural perturbation. In *KDD*, pages 1645–1653, 2021.

[58] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *ICML*, pages 1115–1124. PMLR, 2018.

[59] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Non-target-specific node injection attacks on graph neural networks: A hierarchical reinforcement learning approach. In *WWW*, volume 3, 2020.

[60] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *IJCAI*, 2019.

[61] Tsubasa Takahashi. Indirect adversarial attacks via poisoning neighbors for graph convolutional networks. In *2019 IEEE International Conference on Big Data*, pages 1395–1400. IEEE, 2019.

[62] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019.

[63] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2847–2856, 2018.

[64] Xu Zou, Qinkai Zheng, Yuxiao Dong, Xinyu Guan, Evgeny Kharlamov, Jialiang Lu, and Jie Tang. TDGIA: effective injection attacks on graph neural networks. In *KDD*, volume abs/2106.06663, 2021.

[65] Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. Single node injection attack against graph neural networks. In *CIKM*, pages 1794–1803, 2021.

[66] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. *NeurIPS*, 34, 2021.

[67] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *WWW*, pages 673–683, 2020.

[68] Jihong Wang, Minnan Luo, Fnu Suya, Jundong Li, Zijiang Yang, and Qinghua Zheng. Scalable attack on graph data by injecting vicious nodes. *Data Mining and Knowledge Discovery*, 34(5):1363–1389, 2020.

[69] Shuai Zheng, Zhenfeng Zhu, Xingxing Zhang, Zhizhe Liu, Jian Cheng, and Yao Zhao. Distribution-induced bidirectional generative adversarial network for graph representation learning. In *CVPR*, pages 7224–7233, 2020.

[70] Chika Yinka-Banjo and Ogban-Asuquo Ugot. A review of generative adversarial networks and its application in cybersecurity. *Artificial Intelligence Review*, 53(3):1721–1736, 2020.

[71] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. Outlier detection: Methods, models, and classification. *ACM Computing Surveys (CSUR)*, 53(3):1–37, 2020.

[72] World Health Organization. Coronavirus disease (COVID-19), mar 2021.

[73] Qingyu Chen, Alexis Allot, and Zhiyong Lu. Keep up with the latest coronavirus research. *Nature*, 579(7798):193–194, 2020.

[74] Zakariyaa Ait El Mouden, Rachida Moulay Taj, Abdeslam Jakimi, and Moha Hajar. Towards using graph analytics for tracking covid-19. *Procedia Computer Science*, 177:204–211, 2020.

[75] Adrian Groza. Detecting fake news for the new coronavirus by reasoning on the Covid-19 ontology. *arXiv preprint arXiv:2004.12330*, 2020.

[76] Jiang Guoqian and Solbrig Harold. CORD-19-on-FHIR, apr 2020.

[77] David Oniani, Guoqian Jiang, Hongfang Liu, and Feichen Shen. Constructing co-occurrence network embeddings to assist association extraction for COVID-19 and other coronavirus infectious diseases. *Journal of the American Medical Informatics Association*, 27(8):1259–1267, 2020.

[78] Nansu Zong, Rachael Sze Nga Wong, Yue Yu, Andrew Wen, Ming Huang, and Ning Li. Drug–target prediction utilizing heterogeneous bio-linked network embeddings. *Briefings in bioinformatics*, 22(1):568–580, 2021.

[79] Yang Zhou, Amnay Amimeur, Chao Jiang, Dejing Dou, Ruoming Jin, and Pengwei Wang. Density-aware local Siamese autoencoder network embedding with autoencoder graph clustering. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1162–1167, 2018.

[80] Yang Zhou, Chao Jiang, Zijie Zhang, Dejing Dou, Ruoming Jin, and Pengwei Wang. Integrating local vertex/edge embedding via deep matrix fusion and siamese multi-label classification. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1018–1027, 2019.

[81] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[82] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In *International semantic web conference*, pages 542–557. Springer, 2013.

[83] Dominik Wienand and Heiko Paulheim. Detecting incorrect numerical data in dbpedia. In *European Semantic Web Conference*, pages 504–518. Springer, 2014.

[84] Yahui Long, Min Wu, Yong Liu, Yuan Fang, Chee Keong Kwoh, Jinmiao Chen, Jiawei Luo, and Xiaoli Li. Pre-training graph neural networks for link prediction in biomedical networks. *Bioinformatics*, 38(8):2254–2262, 2022.

[85] Qingsong Yang, Pingkun Yan, Yanbo Zhang, Hengyong Yu, Yongyi Shi, Xuanqin Mou, Mannudeep K Kalra, Yi Zhang, Ling Sun, and Ge Wang. Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss. *IEEE transactions on medical imaging*, 37(6):1348–1357, 2018.

[86] Zixia Zhou, Yuanyuan Wang, Yi Guo, Yanxing Qi, and Jinhua Yu. Image quality improvement of hand-held ultrasound devices with a two-stage generative adversarial network. *IEEE Transactions on Biomedical Engineering*, 67(1):298–311, 2019.

[87] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR, 2017.

[88] Wenyuan Li, Yunlong Wang, Yong Cai, Corey Arnold, Emily Zhao, and Yilian Yuan. Semi-supervised rare disease detection using generative adversarial network. *arXiv preprint arXiv:1812.00547*, 2018.

[89] Luca Rendsburg, Holger Heidrich, and Ulrike Von Luxburg. NetGAN without GAN: From Random Walks to Low-Rank Approximations. In *International Conference on Machine Learning*, pages 8073–8082, 2020.

[90] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *International Conference on Machine Learning*, pages 610–619, 2018.

[91] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[92] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[93] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223, 2017.

[94] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[95] Qingyu Chen, Alexis Allot, and Zhiyong Lu. LitCovid: an open database of COVID-19 literature. *Nucleic Acids Research*, 49(D1):D1534–D1540, 11 2020.

[96] Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic acids research*, 41(W1):W518—-W522, 2013.

[97] Daniel Zügner. Implementation of the paper "NetGAN: Generating Graphs via Random Walks", 2018.

[98] Holger Heidrich. This is the official repository of the ICML 2020 paper "NetGAN without GAN: From Random Walks to Low-Rank Approximations"., 2020.

[99] Scikit-learn.org. scikit-learn: machine learning in Python, 2007.

[100] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

[101] bIomedical Knowledge Empowered Ai lab. Deep Denoising of Raw Biomedical Knowledge Graph from COVID-19 Literature, LitCovid and Pubtator, 2021.

[102] Hao-Wen Dong and Yi-Hsuan Yang. Training Generative Adversarial Networks with Binary Neurons by End-to-end Backpropagation. oct 2018.

[103] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[104] Anderson Rossanez, Julio Cesar Dos Reis, Ricardo da Silva Torres, and Hélène de Ribaupierre. KGen: a knowledge graph generator from biomedical scientific literature. *BMC medical informatics and decision making*, 20(4):1–24, 2020.

[105] Daixin Wang, Yuan Qi, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, and Shuang Yang. A semi-supervised graph attentive network for financial fraud detection. *2019 IEEE International Conference on Data Mining (ICDM)*, Nov 2019.

[106] Adam Breuer, Roee Eilat, and Udi Weinsberg. Friend or faux: Graph-based early detection of fake accounts on social networks. In *WWW*, pages 1287–1297, 2020.

[107] Di Wu, Qiang He, Xin Luo, Mingsheng Shang, Yi He, and Guoyin Wang. A posterior-neighborhood-regularized latent factor model for highly accurate web service qos prediction. *IEEE Transactions on Services Computing*, 2019.

[108] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *KDD*, pages 968–977, 2019.

[109] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *WWW*, pages 417–426, 2019.

[110] Di Wu, Xin Luo, Mingsheng Shang, Yi He, Guoyin Wang, and Xindong Wu. A data-characteristic-aware latent factor model for web services qos prediction. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[111] Wenjuan Luo, Han Zhang, Xiaodi Yang, Lin Bo, Xiaoqing Yang, Zang Li, Xiaohu Qie, and Jieping Ye. Dynamic heterogeneous graph neural network for real-time event prediction. In *KDD*, pages 3213–3223, 2020.

[112] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. In *IJCAI*, 2019.

[113] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *KDD*, pages 1399–1407, 2019.

[114] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. Transferring robustness for graph neural network against poisoning attacks. In *WSDM*, pages 600–608, 2020.

[115] Binghui Wang, Neil Zhenqiang Gong, and Hao Fu. Gang: Detecting fraudulent users in online social networks via guilt-by-association on directed graphs. In *ICDM*, pages 465–474. IEEE, 2017.

[116] Jia Li, Honglei Zhang, Zhichao Han, Yu Rong, Hong Cheng, and Junzhou Huang. Adversarial attack on community detection by hiding individuals. In *WWW*, pages 917–927, 2020.

[117] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, pages 6861–6871. PMLR, 2019.

[118] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[119] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *NeurIPS*, 2017.

[120] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *ICML*, pages 883–892. PMLR, 2018.

[121] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[122] Jian Wu, Kyle Mark Williams, Hung-Hsuan Chen, Madian Khabsa, Cornelia Caragea, Suppawong Tuarob, Alexander G Ororbia, Douglas Jordan, Prasenjit Mitra, and C Lee Giles. Citeseerx: Ai in a digital library search engine. *AI Magazine*, 36(3):35–48, 2015.

[123] Haitao Xu, Daiping Liu, Haining Wang, and Angelos Stavrou. E-commerce reputation manipulation: The emergence of reputation-escalation-as-a-service. In *WWW*, pages 1296–1306, 2015.

[124] Yochai Benkler, Robert Faris, and Hal Roberts. *Network propaganda: Manipulation, disinformation, and radicalization in American politics*. Oxford University Press, 2018.

[125] Sneha Kudugunta and Emilio Ferrara. Deep neural networks for bot detection. *Information Sciences*, 467:312–322, 2018.

[126] Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active learning for networked data. In *ICML*, pages 79–86, 2010.

[127] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):1–37, 2008.

[128] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52, 2015.

[129] Yi He, Sheng Chen, Baijun Wu, Xu Yuan, and Xindong Wu. Unsupervised lifelong learning with curricula. In *WWW*, pages 3534–3545, 2021.

[130] Di Wu, Xin Luo, Mingsheng Shang, Yi He, Guoyin Wang, and MengChu Zhou. A deep latent factor model for high-dimensional and sparse matrices in recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[131] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.

[132] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. COPOD: copula-based outlier detection. In *ICDM*. IEEE, 2020.