Camouflaged Poisoning Attack on Graph Neural Networks

Chao Jiang*
Auburn University
Auburn, Alabama, USA
czj0043@auburn.edu

Yi He†
Old Dominion University
Norfolk, Virginia, USA
yihe@cs.odu.edu

Richard Chapman
Auburn University
Auburn, Alabama, USA
chapmro@auburn.edu

Hongyi Wu†
Old Dominion University
Norfolk, Virginia, USA
h1wu@odu.edu

ABSTRACT

Graph neural networks (GNNs) have enabled the automation of many web applications that entail node classification on graphs, such as scam detection in social media [7, 42] and event prediction in service networks [28]. Nevertheless, recent studies [12, 55, 62, 63] revealed that the GNNs are vulnerable to adversarial attacks, where feeding GNNs with *poisoned* data at training time can lead them to yield catastrophically devastative test accuracy. This finding heats up the frontier of attacks and defenses against GNNs. However, the prior studies mainly posit that the adversaries can enjoy free access to 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 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, such that 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. Code is released in github.com/chao92/GAFNC.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Artificial intelligence**; • **Security and privacy**; • **Applied computing** → *Computer forensics*; • **Information systems** → Web mining;

KEYWORDS

Poisoning Attack, Graph Convolutional Networks, Cybersecurity

*Work done during the internship at ODU.
†Corresponding Authors: Dr. He (yihe@cs.odu.edu) and Dr. Wu (h1wu@odu.edu)

ICMR '22, June 27–30, 2022, Newark, NJ, USA

ACM ISBN 978-1-4503-9238-9/22/06.
https://doi.org/10.1145/3512527.3531373

ACM Reference Format:
Chao Jiang, Yi He, Richard Chapman, and Hongyi Wu. 2022. Camouflaged Poisoning Attack on Graph Neural Networks. In *Proceedings of the 2022 Int'l Conference on Multimedia Retrieval (ICMR '22), June 27–30, 2022, Newark, NJ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3512527.3531373

1 INTRODUCTION

Graph neural networks (GNNs) have empowered many web applications and multimedia systems to envision a higher level of automation in data analytic, *e.g.*, malicious user detection in social media [7, 42], interest group discovery in recommender systems [14, 43, 45, 46], event prediction in service networks [28], 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 effective, this message-passing mechanism gives rise to the awareness on the robustness and trustworthiness of GNNs. Indeed, recent adversaries have reported that the training data of GNNs can be easily *poisoned* through making slight perturbation on the input graph, *e.g.*, manipulating the edge connections [55, 63] or changing contents of existing nodes [12]. Training the GNNs with the poisoned graph data shall lead to diminished even devastative test performance. Spurred by this awareness, several methods turned their eyes on defending GNNs via detecting and correcting the distorted graph information, such as adversarial edge correction [37, 49, 60] and nodal perturbation detection [26, 40].

Although the frontier of GNN attack and defense is heating up, 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 node injection attack [16, 33,

38, 44, 61], where a set of fake nodes (*e.g.*, user accounts) are created and added to poison the graph. Alas, this idea does not work well in practice either, as the fake nodes usually carry vicious contents being so *prominent* that their pattern differs from the benign nodes drastically, where simple classifiers, *e.g.*, an outlier detector [6], can easily spotlight and hence screen out the added fake nodes.

Motivated by this, we in this paper 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 with a novel attacking paradigm on GNNs, termed *Generative Adversarial Fake Node Camouflaging* (GAFNC). The key idea of GAFNC is to leverage the power of Generative Adversarial Networks (GAN) [56, 59] 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, poison the victim GNN for two purposes. First, our GAFNC can perform *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 control the poisoning attack in a finer level of granularity by performing *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 paper 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 shall be discussed in Section 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 3, 4, and 5.
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 6 extrapolates the findings.

2 RELATED WORK

Our work relates to the advancements of graph neural network (GNN) designs, as well as the attacks and defenses on GNNs. This section reviews the prior art in the two threads and discusses the relationship and difference between their methods and ours.

2.1 Graph Neural Networks (GNNs)

GNNs were originated from the idea of extending neural networks from Euclidean spaces to discrete graphs [18], with the key idea lying in the learning of enriched node embeddings that recursively aggregate information from their neighborhood though *message-passing*. Depending on the various realization of aggregation functions, suggested by [3, 52], existing GNN works could be roughly categorized into 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 spectrum [8, 19, 21, 23, 23, 36]. However, these spectral-based GNNs usually require the whole graph as the input, thereby being bottlenecked for graphs at a real-world scale due to the high memory and computational cost.

To improve their efficiency and scalability, subsequent spatial-based methods [2, 15] were proposed, defining the message aggregation operations directly on the graph topological structures with subgraph windows or node sequences. Representative works include the Graph Attention Networks [32, 39], which assigns different weights for neighboring regions in the message-passing process. In such a way, the GNN models are allowed to operate on graphs with stochastic inputs, instead of the entire topology, thereby leading to much improved memory efficiency. Despite their difference, the main idea behind the two threads of GNNs remains the same, where an embedding space *harmonizing* both graph topological structure and nodal features is desired. To respect this main idea and without loss of generality, we in this paper model a generic GNN as the victim model, which shall be discussed in Section **??**. 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. In implementation, attacking on the graph convolutional network (GCN) is exemplified to substantiate the viability of the GAFNC paradigm.

2.2 Adversarial Attack and Defense on Graphs

The robustness and vulnerability of deep neural networks have been extensively studied in the continuous Euclidean domains, including images [9, 25], acoustics [53], and natural languages [57]. This domain was highly motivated by Goodfellow et al.'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, thereby leading to the shift of correct classification boundaries and thus devastative accuracy results. Starting from [63], the security issues of GNNs has been brought into wide attention. Mainstream attacking techniques on GNNs can be categorized into *evasion* attack and *poisoning* attack. Defending techniques to counter against both attacks have been proposed [22, 31, 41, 58]. 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 [12, 34].

Poisoning attack, on the contrary, strives to devastate the performance of GNN models in the *training time*, such that the poisoned models can not give reliable prediction on any test samples [35, 55, 62]. So by this definition, our GAFNC approach falls into the category of poisoning attack. However, prior arts mainly posit that the adversaries are given privilege to manipulate the training data arbitrarily, such as adding or deleting graph edges [55, 62] or changing nodal features [35]. Alas, 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 [16, 38, 61] or poison [33, 44] settings. However, they only consider to devastate the model performance in test or training time, respectively, but fail to consider the *stealthiness* of the injected nodes. That said, their crafted nodes usually carry vicious contents that prominently differs from the original nodes. Our GAFNC approach excels these node injection attackers by leveraging generative adversarial network (GAN) [56, 59] to generate *camouflaged* adversarial nodes, which possess two properties: 1) The generated nodes are enforced to look alike 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 [6] and hence performs poisoning attack in a more stealthy fashion.

3 PROBLEM STATEMENT

Node Classification with GNNs. This paper 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 [3, 52]. 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.

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

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 ϕ^l and ψ^l implement arbitrary learnable transformations. In this work, we implement ϕ^l and ψ^l with a non-linear activation and a linear mapping, for the sake of simplicity and without loss of generality, reducing Eq. (1) to the following format.

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

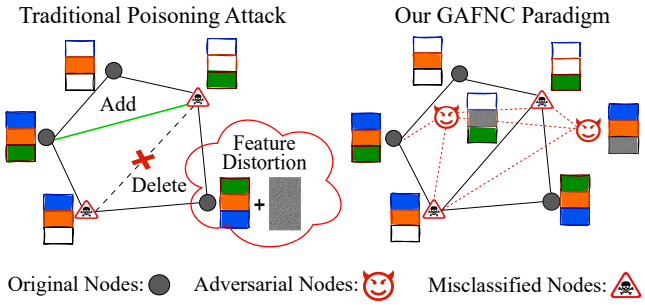

Figure 1: An illustrative comparison of traditional poisoning attacks on graph (left) and our GAFNC paradigm (right).

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. (2), a straightforward and popular idea is to minimize the cross-entropy loss function that leverages the limited labeled nodes, defined as follows.

$$\min_{\mathbf{W}^1,\ldots,\mathbf{W}^M} \sum_{v_i\in\mathbf{V}_L} \mathrm{P}(y_i \neq \hat{y}_i) := -\sum_{v_i\in\mathbf{V}_L} y_i \log \hat{y}_i, \tag{3}$$

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. (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. Such process repeats multiple times until its convergence.

4 THREAT MODEL

In this work, we restrict our interest in attacking GNN models with linear aggregation function, modeled by Eqs. (1) and (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, GNN with linear aggregation is also known as graph convolutional network (GCN), which has manifested its effectiveness in a variety of real-world applications [15, 23, 48], 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. (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 are considered, which take the forms as follows.

$$\text{Global Attack: } \arg\max_{\mathbf{A}',\mathbf{X}'} \sum_{v_i \in \mathbf{V}} \mathrm{P}\left(y_i \neq \hat{y}_i \mid \mathbf{W}^1, \ldots, \mathbf{W}^M\right), \quad (4)$$

$$\text{Target Attack: } \arg\max_{\mathbf{A}',\mathbf{X}'} \sum_{v_i \in \mathbf{V}_\Delta} \mathrm{P}\left(y_\Delta = \hat{y}_i \mid \mathbf{W}^1, \ldots, \mathbf{W}^M\right), \quad (5)$$

$$s.t.\ \|\mathbf{A}' - \mathbf{A}\|_{\mathrm{F}} \leq \epsilon_A, \|\mathbf{X}' - \mathbf{X}\|_{\mathrm{F}} \leq \epsilon_X, \quad (6)$$

where Eq. (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. (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_Δ. The constraints in Eq. (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 ϵ_A and ϵ_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 intuition [49]. Most existing poisoning attack methods can be framed in the modeling of Eqs. (4), (5), and (6).

The Challenge and Limitation. The aforementioned threat model suffer from a prominent drawback – they require access to the original graph data. Obtaining such an access in real-world applications, however, can be very difficult. For example, to devastate 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 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 as *Generative Adversarial Fake Node Camouflaging* (GAFNC). Figure 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 like. 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 in high frequency – such nodes can be so easily detected by naïve network defending system such as an anomaly detector [1]. 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. The details are given in the next section.

5 THE GAFNC PARADIGM

In this section, we elaborate the building blocks of our GAFNC paradigm, with the pipeline delineated in Figure 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 2. We shall scrutinize this block in Section 5.1.

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 the soft mask learning, which was devised for capturing important image regions in computer vision [10, 11], to devise the *edge mask learning* block, as shown in the bottom panel in Figure 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 in high efficacy. The details of the edge mask learning are presented in Section 5.2.

5.1 Adversarial Node Generation

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

$$\min_{D,\Phi} \max_{G} \mathcal{L}_{\mathrm{GNN}}(G, \Phi, \mathcal{G}, \mathcal{G}') + \mathcal{L}_{\mathrm{GAN}}(G, D, \mathcal{G}'), \quad (7)$$

where G, D, and Φ 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

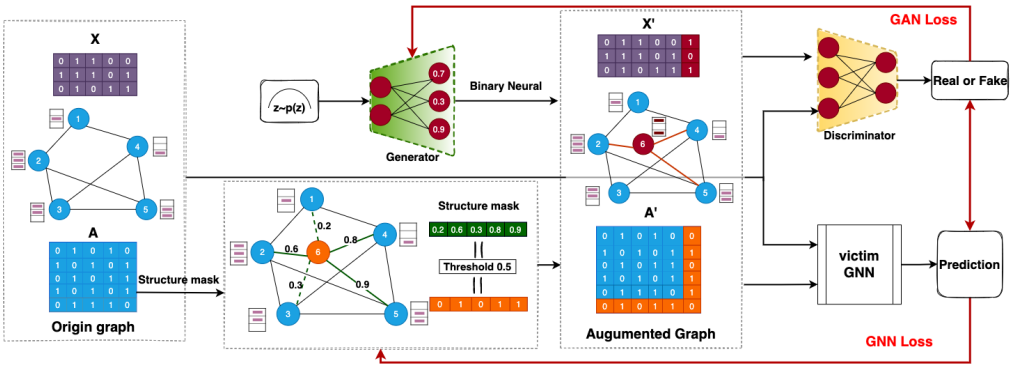

Figure 2: Pipeline of our proposed GAFNC. Blue colored are the original nodes and the corresponding adjacency matrix. Purple colored are the feature matrix of origin graph. Red colored are the new artificial node and its feature contents. Orange colored are the edges between the new node and the original graph and how the adjacency is augmented.

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. (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. (7) is the GAN loss term, which is defined as follows.

$$\mathcal{L}_{\text{GAN}}(G, D, \mathcal{G}') = \mathbb{E}_{v_i \in \mathbf{V}} [1 - \log D(\mathbf{A}, \mathbf{x}_i)] + \mathbb{E}_{v_j \in \{v_{N+1}, \ldots, v_{N+m}\}} \left[\log \left(D(\mathbf{A}', G(\mathbf{z}_j))\right)\right], \quad (8)$$

where $\mathbf{A}'$ and $\mathbf{z}_j$ are the generated connections and nodal features from G. The intuition behind Eq. (8) is that, with D fixed, the second term of Eq. (8) *w.r.t.* 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. (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. 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 [30, 51] 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 [13], 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.

$$DBN(x) = \mu(\sigma(x) - 0.5), \quad (9)$$

$$SBN(x) = \mu(\sigma(x) - v), \quad v \sim U[0, 1], \quad (10)$$

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.

5.2 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. (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 attack has 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 [54]. 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 ceased in their early spreading stage, leaving space for attackers to corrupt democracy in the worst case [4, 24].

To realize the global attack, we define the GNN loss term in Eq. (7) that is to be maximized as follows,

$$\mathcal{L}_{\text{GNN}}(G, \Phi, \mathcal{G}, \mathcal{G}') = \sum_{v_j \in \mathbf{V}'} -y_i \log \hat{y}_i + \lambda_1 \Omega(\mathbf{S}) + \lambda_2 H(\mathbf{S}), \quad (11)$$

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 λ_1 and λ_2 are the regularization

terms, with $\Omega = \sum |(\sigma(\mathbf{S})|$ representing the sum of ℓ_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 Ω. 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. (11) is to maximizing the cross entropy of y and $\hat{y}$. One more attention is that we exclude the output for the new node in the second term of equation Eq. (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. Yet, 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. Thus, we second 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. (7) as follows.

$$\mathcal{L}_{\text{GNN}}(G, \Phi, \mathcal{G}, \mathcal{G}') = \sum_{\substack{v_j \in \mathbf{V}', \\ v_i \in C_t}} -C_k \log \hat{y}_i + \lambda_1 \Omega(\mathbf{S}) + \lambda_2 H(\mathbf{S}), \quad (12)$$

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. (11). The intuition behind Eq. (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.

6 EXPERIMENTS

In this section, we perform empirical study and present the results to verify the effectiveness of our GAFNC attacking approach. Section 6.1 elaborate the benchmark datasets and the evaluation protocol. Section 6.2 presents the experimental results.

6.1 Evaluation Protocol

Datasets. 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 [63] 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 as follows.

Datasets	# Nodes	# Edges	# Features	# Classes
Cora	2,485	5,069	1,433	7
CiteSeer	3,327	4,732	3,703	6
PubMed	19,717	44,338	500	3
Amazon	7487	119043	745	8

Testbed. 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 the 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.

Metrics. 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.

Model Variants. 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. Details defer to Section A.3 due to space limitation.

6.2 Results

Table 1 and Figures 3, 4, and 5 present the experimental results, from which we answer the four research questions as follows.

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

The answer is immediated from the comparison between our GAFNC approach and the baseline in Table 1, 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 20 adversarial nodes only, revealing the vulnerability of GNN models by means that they can be attacked at very low cost in our generative-adversarial poisoning regime.

To further extrapolate this question, we make the following observations from Table 1 and Figure 3. *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 Figure 3 validate this intuition. However, our

Table 1: Comparative results of the victim GNN and our GAFNC approach and its five variants 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.

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

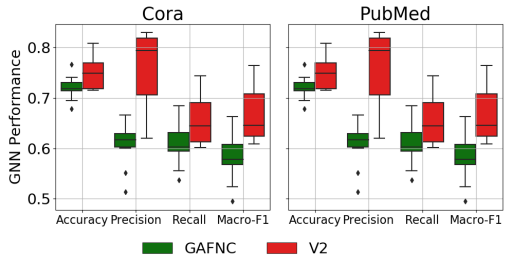

Figure 4: A boxplot comparison between GAFNC and V2 with a 25th – 75th interquartile range.

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 [41], where a jointly learning of benign and malicious nodes can increase the robustness of GNNs. Our GAFNC does not suffer this issue and can 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.

Q2. *How poisonous are the adversarial nodes?*

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 1 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, 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?*

To answer this question, we control the effect of poisonous nodal features by comparing GAFNC to V3 and V2 to V1. We observe from Table 1 that GAFNC and V2 excel in ratios of 18.22% and 21.64% to their competitors, respectively. 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 delivers 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

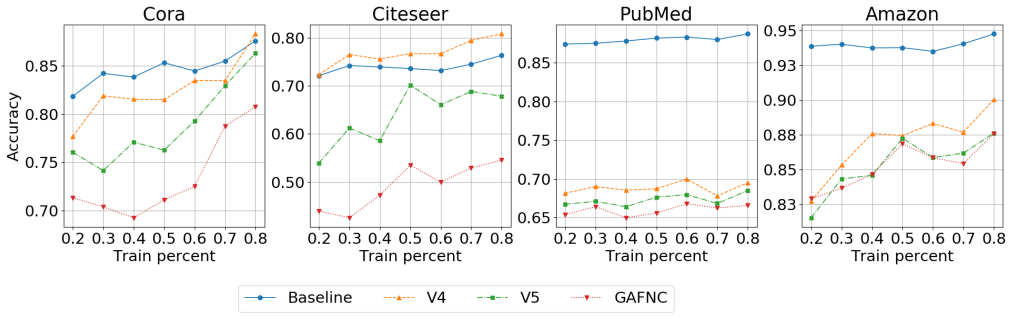

Figure 3: Performance of the classification accuracy of the baseline GNN on the clear datasets and the datasets poisoned by our GAFNC methods and its V4 and V5 variants, where the ratios of training data to the original datasets vary from 20% to 80%. A fixed number of 20 adversarial nodes were added to perform the poisoning attack.

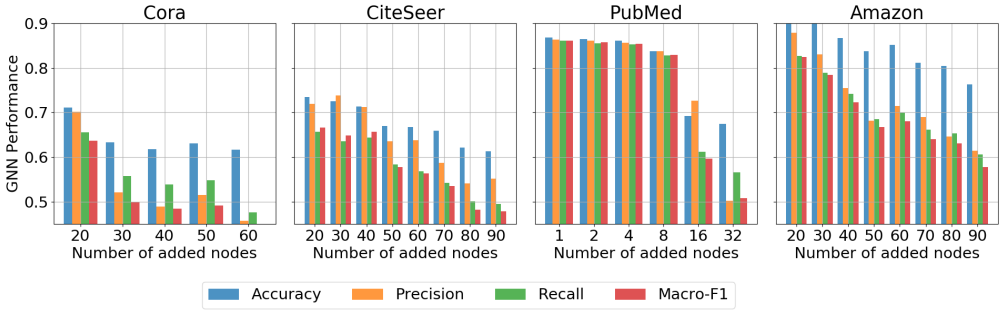

Figure 5: Attacking efficacy in four datasets *w.r.t.* different numbers of added nodes.

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?*

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 5, 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%, 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 [27] 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 Cora dataset is exemplified in Figure 6, where the objective of the target attack is to encourage a misclassification of Node #1336 into a prescribed class. After 1973 epochs of 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.

7 CONCLUSION

This paper 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 crux lying in the craft 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.

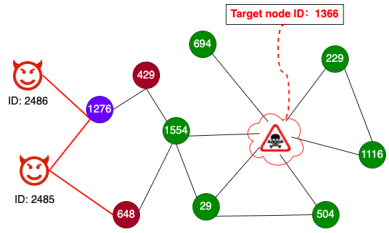

Figure 6: A successful 3-hop target attack in the Cora dataset.

ACKNOWLEDGMENTS

We thank ICMR 2022 reviewers for their constructive feedback. This work was supported in part by the National Science Foundation under Grant CNS-2120279, CNS-1950704, CNS-1828593, and OAC-1829771, Office of Naval Research under grant N00014-20-1-2065, National Security Agency under grant H98230-21-1-0165, H98230-21-1-0278, DoD Center of Excellence in AI and Machine Learning (CoE-AIML) under Contract Number W911NF-20-2-0277, and the Commonwealth Cyber Initiative.

REFERENCES

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

[2] James Atwood and Don Towsley. 2016. Diffusion-Convolutional Neural Networks *(NeurIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 2001–2009.

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

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

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

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

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

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

[9] N. Carlini and D. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 39–57. https://doi.org/10.1109/SP.2017.49

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

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

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

[13] Hao-Wen Dong and Yi-Hsuan Yang. 2018. Training Generative Adversarial Networks with Binary Neurons by End-to-end Backpropagation. arXiv:1810.04714 [cs.LG]

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

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

[16] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. 2021. Robustness of Graph Neural Networks at Scale. *NeurIPS* 34 (2021).

[17] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. *CoRR* abs/1412.6572 (2015).

[18] M. Gori, G. Monfardini, and F. Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 2. 729–734 vol. 2. https://doi.org/10.1109/IJCNN.2005.1555942

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

[20] Yi He, Sheng Chen, Baijun Wu, Xu Yuan, and Xindong Wu. 2021. Unsupervised Lifelong Learning with Curricula. In *WWW*. 3534–3545.

[21] Yi He, Xu Yuan, Nian-Feng Tzeng, and Xindong Wu. 2020. Active Learning with Multi-Granular Graph Auto-Encoder. In *ICDM*. IEEE, 1058–1063.

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

[23] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

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

[25] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. arXiv:1611.01236 [cs.CV]

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

[27] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. 2020. COPOD: Copula-Based Outlier Detection. In *ICDM*. IEEE.

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

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

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

[31] Uday Shankar Shanthamallu, Jayaraman J Thiagarajan, and Andreas Spanias. 2021. Uncertainty-Matching Graph Neural Networks to Defend Against Poisoning Attacks. In *AAAI*, Vol. 35. 9524–9532.

[32] Min Shi, Yu Huang, Xingquan Zhu, Yufei Tang, Yuan Zhuang, and Jianxun Liu. 2021. GAEN: Graph Attention Evolving Networks. In *IJCAI*, Zhi-Hua Zhou (Ed.). ijcai.org, 1541–1547. https://doi.org/10.24963/ijcai.2021/213

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

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

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

[36] Shanshan Tang, Bo Li, and Haijun Yu. 2019. ChebNet: Efficient and Stable Constructions of Deep Neural Networks with Rectified Power Units using Chebyshev Approximations. *CoRR* abs/1911.05467 (2019).

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

[38] Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. 2021. Single Node Injection Attack against Graph Neural Networks. In *CIKM*. 1794–1803.

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *ICLR* (2018). https://openreview.net/forum?id=rJXMpikCZ

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

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

[42] Daixin Wang, Yuan Qi, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, and Shuang Yang. 2019. A Semi-Supervised Graph Attentive Network for Financial Fraud Detection. *2019 IEEE International Conference on Data Mining (ICDM)* (Nov 2019). https://doi.org/10.1109/icdm.2019.00070

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

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

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

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

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

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

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

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

[51] 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. 2015. Citeseerx: Ai in a digital library search engine. *AI Magazine* 36, 3 (2015), 35–48.

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

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

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

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

[56] Chika Yinka-Banjo and Ogban-Asuquo Ugot. 2020. A review of generative adversarial networks and its application in cybersecurity. *Artificial Intelligence*

Review 53, 3 (2020), 1721–1736.
[57] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology* 11, 3 (2020), 1–41.
[58] Xiang Zhang and Marinka Zitnik. 2020. Gnnguard: Defending graph neural networks against adversarial attacks. In *NeurIPS*.
[59] Shuai Zheng, Zhenfeng Zhu, Xingxing Zhang, Zhizhe Liu, Jian Cheng, and Yao Zhao. 2020. Distribution-induced bidirectional generative adversarial network for graph representation learning. In *CVPR*. 7224–7233.
[60] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *KDD*. 1399–1407.
[61] Xu Zou, Qinkai Zheng, Yuxiao Dong, Xinyu Guan, Evgeny Kharlamov, Jialiang Lu, and Jie Tang. 2021. TDGIA: Effective Injection Attacks on Graph Neural Networks. In *KDD*, Vol. abs/2106.06663.
[62] Daniel Zügner and Stephan Günnemann. 2019. Adversarial attacks on graph neural networks via meta learning. In *ICLR*.
[63] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *KDD*. ACM. https://doi.org/10.1145/3219819.3220078

A APPENDIX

A.1 Details of the Studied Datasets

With their statistics given in the main content in Section 6.1, we give a detailed introduction of the studied datasets.

- **Cora** [30] and **CiteSeer** [51] 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** [5] 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× 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 [50] 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** [29] 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 [20], 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 [47], helping the online business users to build and uplift the awareness of analyzing and reinforcing the robustness of their systems.

A.2 Evaluation Metrics: Accuracy, Recall, Precision, and Macro-F1

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 imbalanceness, where the number of samples in the negative class significantly overweighs that in 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 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.

A.3 Details of Ablation Model Variants

The details of the variants proposed in Section 6.1 are summarized in the table below.

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

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 5.