Task and Model Agnostic Adversarial Attack on Graph Neural Networks

Kartik Sharma[1*], **Samidha Verma**[2], **Sourav Medya**[3],
Arnab Bhattacharya[4], **Sayan Ranu**[2]

[1] Georgia Institute of Technology, Atlanta, USA
[2] Indian Institute of Technology, Delhi, India
[3] University of Illinois, Chicago, USA
[4] Indian Institute of Technology, Kanpur, India
ksartik@gatech.edu, csy207575@iitd.ac.in, medya@uic.edu, sayanranu@iitd.ac.in, arnabb@iitk.ac.in

Abstract

Adversarial attacks on Graph Neural Networks (GNNs) reveal their security vulnerabilities, limiting their adoption in safety-critical applications. However, existing attack strategies rely on the knowledge of either the GNN model being used or the predictive task being attacked. *Is this knowledge necessary?* For example, a graph may be used for multiple downstream tasks unknown to a practical attacker. It is thus important to test the vulnerability of GNNs to adversarial perturbations in a model and task agnostic setting. In this work, we study this problem and show that GNNs remain vulnerable even when the downstream task and model are unknown. The proposed algorithm, TANDIS (**T**argeted **A**ttack via **N**eighborhood **DIS**tortion) shows that distortion of node neighborhoods is effective in drastically compromising prediction performance. Although neighborhood distortion is an NP-hard problem, TANDIS designs an effective heuristic through a novel combination of *Graph Isomorphism Network* with *deep Q-learning*. Extensive experiments on real datasets and state-of-the-art models show that, on average, TANDIS is up to 50% more effective than state-of-the-art techniques, while being more than 1000 times faster.

1 Introduction and Related Work

Graph neural networks (GNNs) (Hamilton, Ying, and Leskovec 2017; Kipf and Welling 2017; Velickovic et al. 2018; Nishad et al. 2021; Bhattoo, Ranu, and Krishnan 2022), have received much success in structural prediction tasks (Goyal, Jain, and Ranu 2020; Ranjan et al. 2022; Thangamuthu et al. 2022; Jain et al. 2021; Manchanda et al. 2020). Consequently, GNNs have witnessed significant adoption in the industry (Pal et al. 2020; Bose et al. 2019). It is therefore imperative to ensure that GNNs are secure and robust to adversarial attacks. In this work, we investigate this aspect, identify vulnerabilities, and link them to graph properties that potentially hold the solution towards making GNNs more secure.

An attack may occur either during training (*poisoning*) or testing (*evasion*). The attack strategy depends on the extent of information access. In the literature, primarily, three modes of information access has been studied.

1. **White-box attack (WBA):** In a white-box attack (Liu et al. 2019; Wang and Gong 2019; Wu et al. 2019; Xu et al. 2019a), the attacker has access to all information such as model architecture, model parameters and training hyperparameters.
2. **Black-box attack (BBA):** In a black-box attack (Bojchevski and Günnemann 2019; Chang et al. 2020; Chen et al. 2021; Dai et al. 2018; Gupta and Chakraborty 2021; Li et al. 2020; Ma, Ding, and Mei 2020; Ma et al. 2019; Wang et al. 2020), the attacker does not have access to the training model parameters. The attacker can only pose black-box queries on a test sample and get the model output in return (such as a label or score).
3. **Grey-box attack (GBA):** A grey-box attack is a mix of white-box and black-box attacks (Liu et al. 2019; Sun et al. 2020; Wang and Gong 2019; Zügner, Akbarnejad, and Günnemann 2018; Zügner and Günnemann 2019), where the attacker has partial information of the training model. The extent of partial information depends on the particular attack.

In this work, we focus on *targeted black-box, evasion attacks* (Dai et al. 2018). Specifically, we consider an input test graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{\mathcal{X}})$, a target node $t \in \mathcal{V}$ and a budget $\mathcal{B}$. Our goal is to introduce at most $\mathcal{B}$ edge alterations (additions or deletions) in $\mathcal{G}$ such that the performance of an *unknown* GNN model $\mathcal{M}$ is maximally reduced on node t.

Limitations of Existing Works: Although a black-box attack does not require any information on the model parameters, they operate under three key assumptions:

1. **Task-specific strategy:** Most GNN models are trained for a specific task (e.g., node classification) using an appropriately chosen loss function. Existing techniques for adversarial attacks tailor their strategy towards a specific prediction task under the assumption that this task is known. Hence, they do not generalize to unseen tasks. While this assumption acts as an added layer of security for GNN models, we ask the question: *Is it possible to design task-agnostic adversarial attacks?*
2. **Knowledge of the GNN model:** Although black-box attacks do not need knowledge of the model parameters, they often assume the model type. For example, an attack may be customized for GCN and rendered ineffective if the victim model switches to Locality-aware GNNs (You,

[*] Work done as undergraduate thesis at IIT-Delhi

Algorithm	Task	Model	Label
RL-S2V (Dai et al. 2018)		✓	
GF-ATTACK (Chang et al. 2020)	✓		✓
Wang et al. (Wang et al. 2020)		✓	
TANDIS	✓	✓	✓

Table 1: Characterization of existing targeted, black-box evasion attacks based on agnosticism of the features.

Ying, and Leskovec 2019; Nishad et al. 2021).

3. **Label-dependent:** Several BBA algorithms require knowledge of the ground-truth data. As an example, an algorithm may require knowledge of node labels to adversarially attack node classification. These ground truth data is often not available in public domain. For example, Facebook may tag users based on their primary interest area. But this information is proprietary.

Contributions: Table 1 summarizes the limitations in existing algorithms for targeted BBA. In this work, we bridge this gap. Our key contributions are as follows:

- We formulate the problem of *task, model, and label agnostic black-box evasion attacks* on GNNs. The proposed algorithm, TANDIS (**T**argeted **A**ttack via **N**eighborhood **DIS**tortion), shows that such attacks are indeed possible.
- TANDIS exploits the observation that, regardless of the task or the model-type, if the neighborhood of a node can be *distorted*, the downstream prediction task would be affected. Our analysis shows that budget-constrained neighborhood distortion is NP-hard. This computational bottleneck is overcome, by using a *Graph Isomorphism Network* (GIN) to embed neighborhoods and then using *deep Q-learning* (DQN) to explore this combinatorial space in an effective and efficient manner.
- Extensive experiments on real datasets show that, on average, TANDIS is up to 50% more effective than state-of-the-art BBA evasion attacks and 1000 times faster. More importantly, TANDIS establishes that GNNs are vulnerable despite task and model agnosticism.

2 Problem Formulation

We denote a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{\mathcal{X}})$ where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ the set of edges, and $\boldsymbol{\mathcal{X}}$ is the set of attribute vectors corresponding to each node $v \in \mathcal{V}$. In this paper, we assume $\mathcal{G}$ to be an undirected graph. Nonetheless, all of the proposed methodologies easily extend to directed graphs. The *distance* $sp(v, u)$ between nodes v and u is measured in the terms of the length of the shortest path from v to u. The k-hop neighborhood of a node v is therefore defined as $\mathcal{N}^k_{\mathcal{G}}(v) = \{u \in \mathcal{V} \mid sp(v, u) \leq k\}$.

The adjacency matrix of graph $\mathcal{G}$ is denoted as $\boldsymbol{\mathcal{A}}_{\mathcal{G}}$. The *distance* between two undirected graphs, $\mathcal{G} = (\mathcal{G}, \mathcal{E}, \boldsymbol{\mathcal{X}})$ and $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', \boldsymbol{\mathcal{X}})$, with the same node set but different edge set, is defined to be half of the L_1 distance between their adjacency matrices: $d(\mathcal{G}, \mathcal{G}') = \|\boldsymbol{\mathcal{A}}_{\mathcal{G}} - \boldsymbol{\mathcal{A}}_{\mathcal{G}'}\|/2$.

Given a GNN model $\mathcal{M}$, its performance on graph $\mathcal{G}$ towards a particular predictive task is quantified using a *performance metric* $\mathcal{P}_{\mathcal{M}}(\mathcal{G})$:

$$\mathcal{P}^*_{\mathcal{M}}(\mathcal{G}) = f_1\left(\{f_2(\mathbf{z}_v(\mathcal{G}), \ell_v) \mid v \in \mathcal{V}\}\right), \tag{1}$$

where f_2 calculates the performance on node v using the embedding $\mathbf{z}_v(\mathcal{G})$ and ground truth label or score ℓ_v. The overall performance $\mathcal{P}^*_{\mathcal{M}}(\mathcal{G})$ is some aggregation f_1 over the performance across all nodes.

An adversarial attacker wishes to change $\mathcal{G}$ by performing $\mathcal{B}$ edge deletions and additions so that the performance on a *target node* t is minimized. We assume that the attacker has access to a subset of nodes $\mathcal{C} \subseteq \mathcal{V}$ and edges may be modified only among the set $\mathcal{C} \cup \{t\}$. Semantically, $\mathcal{C}$ may represent colluding bots, or vulnerable users who have public profiles instead of private profiles, etc. Note that we do not explicitly allow node deletions and additions, since these can be modeled through deletions and additions over the edges (Dai et al. 2018).

Problem 1 (Targeted Black-box Evasion Attack) *Given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{\mathcal{X}})$, a target node t, budget $\mathcal{B}$, and a subset of accessible nodes $\mathcal{C} \subseteq \mathcal{V}$, perform at most $\mathcal{B}$ edge additions or deletions from edge space $\{(u, v) \mid u, v \in \mathcal{C} \cup \{t\}\}$ to form graph $\mathcal{G}^*$ such that*

$$\mathcal{G}^* = \arg \min_{\mathcal{G}': d(\mathcal{G}, \mathcal{G}') \leq \mathcal{B}} \mathcal{P}_{\mathcal{M}}(\mathcal{G}', t) \tag{2}$$

Both the GNN model $\mathcal{M}$ and performance metric $\mathcal{P}_{\mathcal{M}}(\mathcal{G}', t)$ are *unknown* to the attacker. Note that our formulation is easily extensible to a set of target nodes, where Eq. 2 is aggregated over all targets.

3 TANDIS: Targeted Attack via Neighborhood DIStortion

Since in the search process for $\mathcal{G}^*$, the attacker does not have access to either the performance metric $\mathcal{P}^*_{\mathcal{M}}(\mathcal{G})$ or the model-type of $\mathcal{M}$, we need a *surrogate* function $\phi(\mathcal{G}, t)$ in the *graph space*, such that if $distance(\phi(\mathcal{G}, t), \phi(\mathcal{G}', t)) \gg 0$ then $\mathcal{P}_{\mathcal{M}}(\mathcal{G}, t)$ is significantly different from $\mathcal{P}_{\mathcal{M}}(\mathcal{G}', t)$.

3.1 Neighbourhood Distortion in Graph Space

To identify $\phi(\mathcal{G}, t)$, we first note that there are primarily two types of GNNs:

- **Neighborhood-convolution:** Examples of neighborhood convolution based architectures include GCN (Kipf and Welling 2017), GraphSage (Hamilton, Ying, and Leskovec 2017), and GAT (Velickovic et al. 2018). Here, the embedding of a node v is computed through a *convolution* operation ψ over its k-hop neighborhood, i,e, $\mathbf{z}_{v,\mathcal{G}} = \psi\left(\mathcal{N}^k_{\mathcal{G}}(v)\right)$. Consequently, nodes with similar neighborhoods have similar embeddings (Xu et al. 2019b; You, Ying, and Leskovec 2019).
- **Locality-aware:** In locality-aware GNNs such as P-GNN (You, Ying, and Leskovec 2019), GRAPHREACH (Nishad et al. 2021), DEEPWALK (Perozzi, Al-Rfou, and Skiena 2014), etc., two nodes have similar embeddings if they are located close to each other in the graph. Many real-world graphs display *small-world* and *scale-free* properties (Albert and Barabási 2002). Both small-world and scale-free graphs usually have high clustering coefficients, which

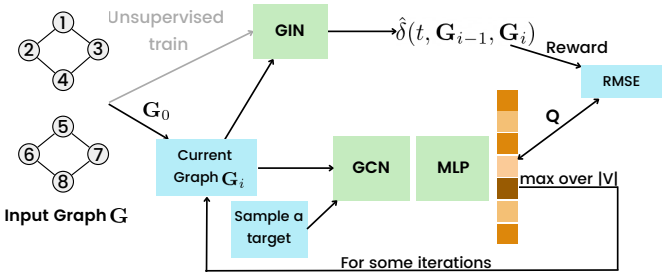


Figure 1: Train pipeline of TANDIS. During testing, we freeze the GCN+MLP and pass the input target t (instead of sampled targets) for a fixed budget $\mathcal{B}$ number of iterations.

indicate that one-hop neighbors share many other neighbors as well. Thus, nearby nodes have a high likelihood of having similar neighborhoods.

The above discussion reveals that node embeddings reflect node neighborhoods in both architectural paradigms. Consequently, the surrogate function $\phi(\mathcal{G}, t)$ may be set to target node t's k-hop neighborhood $\mathcal{N}^k_{\mathcal{G}}(t)$. If $\mathcal{N}^k_{\mathcal{G}}(t)$ can be distorted through adversarial edge edits, then it would also perturb the embedding of t, which in turn would affect the performance metric. Towards that end, we define the notion of *neighborhood distortion* as follows.

Definition 1 (Neighborhood Distortion) *Let $\mathcal{N}^k_{\mathcal{G}}(v)$ and $\mathcal{N}^k_{\mathcal{G}'}(v)$ be the k-hop neighborhoods of v in original graph $\mathcal{G}$ and adversarially perturbed graph $\mathcal{G}'$ respectively. The distortion in $\mathcal{G}'$ with respect to a node v is simply the Jaccard distance of their neighborhoods.*

$$\delta(v, \mathcal{G}, \mathcal{G}') = 1 - \frac{|\mathcal{N}^k_{\mathcal{G}}(v) \cap \mathcal{N}^k_{\mathcal{G}'}(v)|}{|\mathcal{N}^k_{\mathcal{G}}(v) \cup \mathcal{N}^k_{\mathcal{G}'}(v)|} \tag{3}$$

Problem 1 may now be reduced to the following objective:

$$\mathcal{G}^* = \arg \max_{\mathcal{G}' : d(\mathcal{G}, \mathcal{G}') \leq \mathcal{B}} \delta(t, \mathcal{G}, \mathcal{G}') \tag{4}$$

where t is the target node. Unfortunately, maximizing $\delta(t, \mathcal{G}, \mathcal{G}')$ is NP-hard even for $k = 2$.

Theorem 1 *The problem $\max_{\mathcal{G}' : d(\mathcal{G}, \mathcal{G}') \leq \mathcal{B}} \delta(t, \mathcal{G}, \mathcal{G}')$ is NP-hard and the objective is non-monotone and non-submodular.*

PROOF. We reduce the SET COVER problem to our maximization objective. See Appendix B in the full version for more details (Sharma et al. 2021).

Owing to NP-hardness, computing the optimal solution for the problem in Eq. 4 is not feasible. Furthermore, since the objective is neither submodular nor monotone, even the greedy algorithms that provide $1 - 1/e$ approximations are not applicable here (Nemhauser and Wolsey 1978). We, therefore, investigate other optimization strategies.

3.2 A Reinforcement Learning Approach for Neighborhood Distortion

Fig. 1 presents the pipeline of our proposed algorithm TANDIS. It proceeds through three phases.

1. **Training data generation:** We capture a feature-space representation of node neighborhoods using *Graph Isomorphism Network* (GIN) (Xu et al. 2019b). This design choice is motivated by the property that GIN is as powerful as the *Weisfeiler-Lehman graph isomorphism test* (Xu et al. 2019b). On the feature space generated by GIN, we define a neighborhood distortion measure.
2. **Train phase:** We train a deep Q-learning network (DQN), to predict the impact of an edge edit on distortion in the neighborhood embedding space. The proposed DQN framework captures the *combinatorial* nature of the problem, and shares *parameters across budgets*.
3. **Test phase:** Given an unseen graph, we perform $\mathcal{B}$ forward passes through the trained DQN in an iterative manner to form the answer set of $\mathcal{B}$ edge edits.

3.3 Training Data Generation

The input to the training phase is a set of tuples of the form $\langle t, e, \mathscr{M}(\mathcal{G}), \mathscr{M}(\mathcal{G}_e) \rangle$. Here, t is the target node, e denotes an edge perturbation, and $\mathscr{M}(\mathcal{G})$, $\mathscr{M}(\mathcal{G}_e)$ are the GIN embeddings of the original graph $\mathcal{G}$ and the graph formed following edge edit e, $\mathcal{G}_e$, respectively. $\mathscr{M}(\mathcal{G}) = \{\mathbf{Z}^{\mathcal{G}}_v \mid v \in \mathcal{V}\}$ contains the GIN embedding of each node in $\mathcal{G}$. $\mathbf{Z}^{\mathcal{G}}_v$ characterizes the k-hop neighborhood of node v in graph $\mathcal{G}$. t is randomly selected. The edge perturbations are selected using a greedy mechanism, which we detail in Appendix E of the full version (Sharma et al. 2021).

Embeddings via GIN: GIN draws its expressive power by deploying an *injective* aggregation function. These embeddings would thus help us in distinguishing neighborhood structures. We train these embeddings in a task and model-agnostic manner by minimizing the following *unsupervised* loss function.

$$\mathcal{L}(\mathscr{M}(\mathcal{G})) = -\log\left(P\left(\mathcal{N}^k_{\mathcal{G}}(v) \mid \mathbf{Z}^{\mathcal{G}}_v\right)\right), \text{where} \tag{5}$$

$$P\left(\mathcal{N}^k_{\mathcal{G}}(v) \mid \mathbf{Z}^{\mathcal{G}}_v\right) = \prod_{u \in \mathcal{N}^k_{\mathcal{G}}(v)} P\left(u \mid \mathbf{Z}^{\mathcal{G}}_v\right) \text{ and} \tag{6}$$

$$P\left(u \mid \mathbf{Z}^{\mathcal{G}}_v\right) = \frac{exp\left(\mathbf{Z}^{\mathcal{G}}_u \cdot \mathbf{Z}^{\mathcal{G}}_v\right)}{\sum_{\forall u' \in \mathcal{V}} exp\left(\mathbf{Z}^{\mathcal{G}}_{u'} \cdot \mathbf{Z}^{\mathcal{G}}_v\right)} \tag{7}$$

For the pseudocode, please refer to Appendix C of Sharma et al. (2021).

Distortion in embedding space: We define distortion in the perturbed graph $\mathcal{G}'$ with respect to the original graph $\mathcal{G}$ as follows:

$$\widehat{\delta}(t, \mathcal{G}, \mathcal{G}') = d_o(t) - d_p(t) \tag{8}$$

Here, $d_o(t) = \frac{1}{|\mathcal{N}^k_{\mathcal{G}}(t)|} \sum_{u \in \mathcal{N}^k_{\mathcal{G}}(t)} \|\mathbf{Z}^{\mathcal{G}'}_t - \mathbf{Z}^{\mathcal{G}'}_u\|_2$ denotes the *mean ℓ_2 distance* in the *perturbed embedding space* between target node t and its neighbors before perturbation. On the other hand, $d_p(t) = \frac{1}{|\mathcal{N}^k_{\mathcal{G}'}(t)|} \sum_{u \in \mathcal{N}^k_{\mathcal{G}'}(t)} \|\mathbf{Z}^{\mathcal{G}'}_t - \mathbf{Z}^{\mathcal{G}'}_u\|_2$ denotes the distance in the perturbed embedding space between target nodes t and its neighbors after perturbation.

In our formulation, if the neighborhood remains unchanged, then $d_o(t) = d_p(t)$, and hence distortion is 0. The distortion will be maximized if the distance of the target node to the original neighbors become significantly higher

in the perturbed space than the distance to the neighbors in the perturbed graph. With the above definition of distortion in the embedded space, the maximization objective in Eq. 4 is approximated as :

$$\mathcal{G}^* = \arg \max_{\mathcal{G}': d(\mathcal{G},\mathcal{G}') \leq \mathcal{B}} \widehat{\delta}(t, \mathcal{G}, \mathcal{G}') \tag{9}$$

This approximation is required due to two reasons: **(1)** Maximizing Eq. 4 is NP-hard, and **(2)** Eq. 4 is not differentiable.

3.4 Learning Q-function

We optimize Eq. 9 via Q-learning (Sutton and Barto 2018), which inherently captures the combinatorial aspect of the problem in a budget-independent manner. As depicted in Fig. 2, our deep Q-learning framework is an *end-to-end* architecture with a sequence of two separate neural components, a graph neural network (in experiments, we use GCN (Kipf and Welling 2017)) and a separate Multi-layer Perceptron (MLP) with the corresponding learned parameter set Θ_Q. Given a set of edges S and an edge $e \notin S$, we predict the n-step reward, $Q_n(S, e)$, for including e to the solution set S via the surrogate function $Q'_n(S, e; \Theta_Q)$.

Defining the framework: The Q-learning task is defined in terms of state space, action space, reward, policy, and termination condition.

• **State space:** The state space characterizes the state of the system at any time step i. Since our goal is to distort the neighborhood of target node t, we define it in terms of its k-hop neighborhood $\mathcal{N}^k_{\mathcal{G}_i}(t)$. Note that the original graph evolves at each step through edge edits. $\mathcal{G}_i$ denotes the graph at time step i. The representation of state s_i is defined as:

$$\boldsymbol{\mu}_{s_i} = \sum_{v \in \mathcal{N}^k_{\mathcal{G}_i}(t)} \boldsymbol{\mu}_v \tag{10}$$

where $\boldsymbol{\mu}_v$ is the embedding of node v learned using a GNN (GCN in our implementation).

• **Action:** An action, a_i at i-th step corresponds to adding or deleting an edge $e = (v, t)$ to the solution set S_{i-1} where the target node is t. The representation of the action a_i is:

$$\boldsymbol{\mu}_{a_i} = \pm \textsc{Concat}\left(\boldsymbol{\mu}_v, \boldsymbol{\mu}_t\right), \tag{11}$$

where the sign denotes edge addition $(+)$ and deletion $(-)$.

• **Reward:** The immediate (0-step) reward of action a_i is its *marginal gain* in distortion, i.e.,

$$r(s_i, a_i) = \widehat{\delta}(t, \mathcal{G}_i, \mathcal{G}_{i+1}) \tag{12}$$

where $\mathcal{G}_{i+1}$ is the graph formed due to a_i on $\mathcal{G}_i$.

• **Policy and Termination:** At the i-th step, the policy $\pi(e \mid S_{i-1})$ selects the edge with the highest *predicted* n-step reward, i.e., $\arg\max_{e \in C_t} Q'_n(S_i, e; \Theta_Q)$, where

$$Q'_n(S_i, e = (u, t); \Theta_Q) = \mathbf{w}_1^T \cdot \sigma(\boldsymbol{\mu}_{\boldsymbol{s_i}, \boldsymbol{a_i}}), \text{where} \tag{13}$$

$$\boldsymbol{\mu}_{\boldsymbol{s_i}, \boldsymbol{a_i}} = \mathbf{W}_2 \cdot \textsc{Concat}\left(\boldsymbol{\mu}_{s_i}, \boldsymbol{\mu}_{a_i}\right)$$

$\mathbf{w}_1, \mathbf{W}_2$ are learnable weight vector and matrix respectively. We terminate training after T samples.

Learning the Parameter Set Θ_Q: Θ_Q consists of $\mathbf{w}_1$, $\mathbf{W}_2$, and $\{\mathbf{W}^l_{GCN}\}$. While $\mathbf{w}_1$ are $\mathbf{W}_2$ are used to compute $Q'_n(S_i, e)$, $\mathbf{W}^l_{GCN}$ are parameters of the GCN for each hidden layer $l \in [1, k]$. Q-learning updates parameters in a single episode via *Adam optimizer* (Kingma and Ba 2015) minimizing the *squared loss*.

$$J(\Theta_Q) = (y - Q'_n(S_i, e_i; \Theta_Q))^2, \text{where}$$

$$y = \gamma \cdot \max_{e=(u,t),\, u \in C} \{Q'_n(S_{i+n}, e; \Theta_Q)\} + \sum_{j=0}^{n-1} r(S_{i+j}, e_{i+j})$$

The *discount factor* γ balances the importance of immediate reward with the predicted n-step future reward (Sutton and Barto 2018). The pseudocode with additional details is provided in the full version (Sharma et al. 2021).

3.5 Test Phase

Given an unseen graph $\mathcal{G}$ with budget $\mathcal{B}$, we perform $\mathcal{B}$ forward passes through the DQN. Each forward pass returns the edge with the highest predicted long-term reward and updates the state space representations. Note that the GIN is not used in the test phase. It is only used for training the DQN. Furthermore, the proposed inference pipeline is also independent of the test GNN or the loss function since it directly attacks the target node neighborhood.

3.6 Complexity Analysis

Here, we analyze the running time complexity of TANDIS during test phase. The complexity analysis for the train phase is provided in Appendix F in the full version (Sharma et al. 2021) along with a complete proof of the obtained test-time complexity.

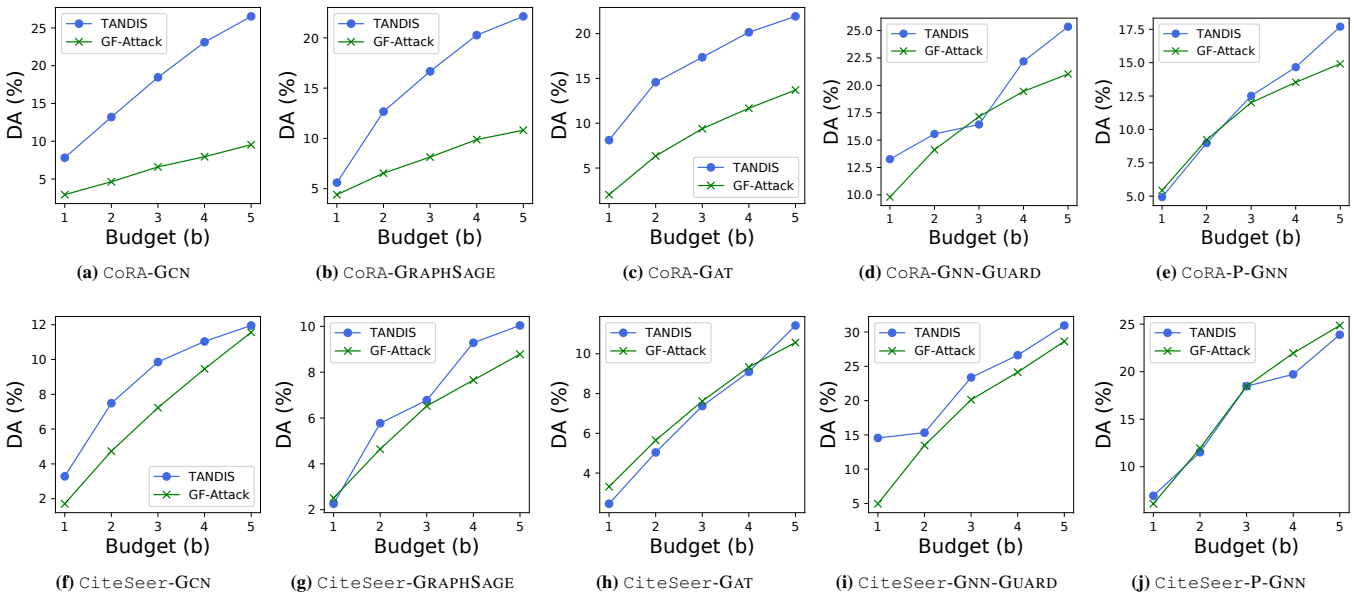
Test phase involves finding $Q'_n(S_i, e)$ by doing a forward pass on a k-layer GCN followed by an L-layer MLP. We ignore the effects of the number of layers k, L. Combining the costs from both GCN and MLP, and the fact that we make $\mathcal{B}$ forward passes, the total computation complexity is $O(\mathcal{B}(|\mathcal{E}|h_g + |\mathcal{V}|h_g^2 + |\mathcal{V}|(h_m(1 + h_g))))$. Since h_g and h_m may be considered as constants, the complexity with respect to the input parameters reduces to $O(\mathcal{B}(|\mathcal{E}| + |\mathcal{V}|))$.

4 Empirical Evaluation

We evaluate TANDIS across *three downstream tasks* of link prediction **(LP)**, node classification **(NC)** and pairwise node classification **(PNC)** (You, Ying, and Leskovec 2019; Nishad et al. 2021). We provide additional details about the experimental setup in Appendix G of the full version (Sharma et al. 2021). Our code base is shared in the supplementary package.

Datasets: We use three standard real-world datasets in our experiments following (Chang et al. 2020): `CoRA` (McCallum et al. 2000), `CiteSeer` and `PubMed` (Sen et al. 2008). For more details, refer Appendix G of the full version (Sharma et al. 2021).

Baselines: We compare with GF-ATTACK (Chang et al. 2020), which is the closest in terms of features except being model-agnostic (for which, we can exploit transferability of attacks) (Recall Table 1). We also compare with RL-S2V (Dai et al. 2018), which is model-agnostic but not task-agnostic (limited to node classification). Comparisons with non-neural baselines such as random, degree, and greedy are presented in the full version.



(a) CoRA-GCN **(b)** CoRA-GRAPHSAGE **(c)** CoRA-GAT **(d)** CoRA-GNN-GUARD **(e)** CoRA-P-GNN

(f) CiteSeer-GCN **(g)** CiteSeer-GRAPHSAGE **(h)** CiteSeer-GAT **(i)** CiteSeer-GNN-GUARD **(j)** CiteSeer-P-GNN

Figure 2: Link prediction: Drop in accuracy (DA%) in (a-d) CoRA and (e-h) CiteSeer. Higher means better.

In this context, Bojchevski and Günnemann (2019) also attacks GNNs through an idea based on neighborhood distortion. However, we do not compare against it since it is a poisoning attack. Poisoning attacks perturb the training data instead of test data as in our case. Furthermore, when this algorithm is adapted for evasion attacks, our main baseline GF-ATTACK (Chang et al. 2020) outperforms significantly.

Target models: We test the efficacy our our attacks on five state-of-the-art GNN models in our experiments: **(1)** GCN (Kipf and Welling 2017), **(2)** GRAPHSAGE (Hamilton, Ying, and Leskovec 2017), **(3)** GAT (Velickovic et al. 2018), **(4)** P-GNN (You, Ying, and Leskovec 2019), and **(5)** GNN-GUARD (Zhang and Zitnik 2020). The first three are based on *neighborhood-convolution*, P-GNN is locality-aware, and GNN-GUARD is a GNN model specifically developed to avert adversarial poisoning attacks.

Parameters: To train the attack models, we use 10% nodes each for training and validation and 80% for test. On the test set, we ensure that the class distribution is balanced by down-sampling the majority class. Other details regarding the hyperparameters are mentioned in Sharma et al. (2021).

Metric: We quantify the attack performance using the percentage change in accuracy before and after attack, i.e. *Drop-in-Accuracy (DA%)*, defined as

$$DA(\%) = \frac{\text{Original Accuracy} - \text{Accuracy after attack}}{\text{Original Accuracy}} \times 100. \tag{14}$$

4.1 Impact of Perturbations

In this section, we compare TANDIS with relevant baselines on CiteSeer and CoRA. Since both GF-ATTACK and RL-S2V fail to scale for higher budgets on PubMed, we defer the results on PubMed to Section 4.2.

Link Prediction (LP): Fig. 2 evaluates the performance on LP. TANDIS consistently produces better results i.e., higher drop in accuracy, in most cases. The only exception are GAT and P-GNN in CiteSeer (Figures 2h, 2j). In these cases, the drop in accuracy is similar for TANDIS and GF-ATTACK. Another interesting observation that emerges from this experiment is that the drop in accuracy in GNN-GUARD is higher than the standard GNN models. This is unexpected since GNN-GUARD is specifically designed to avert adversarial attacks. However, it must be noted that GNN-GUARD was designed to avert poisoning attacks, whereas TANDIS launches an evasion attack. This indicates that defence mechanisms for poisoning attacks might not transfer to evasion attacks.

Pairwise Node Classification (PNC): Here, the task is to predict if two nodes belong to the same class or not. Figure 3 shows that TANDIS is better across all models except in GAT on CiteSeer, where both obtain similar performance.

Node Classification (NC): Table 2 presents the results on NC. Since RL-S2V is prohibitively slow in inference, we report results till $\mathcal{B} = 5$ for RL-S2V. TANDIS outperforms GF-ATTACK and RL-S2V in majority of the cases. However, the results in NC are more competitive. Specifically, GF-ATTACK outperforms TANDIS in a larger number of cases than in LP and PNC. A deeper analysis reveals that attacking NC is an easier task than LP and PNC. This is evident from the observed drop in accuracy in NC, which is significantly higher than in LP and PNC. Consequently, it is harder for one technique to significantly outshine the other.

Summary: Overall, three key observations emerge from these experiments. First, it is indeed possible to launch task and model agnostic attacks on GNN models. Second, TANDIS, on average, produces a drop in accuracy that is more than 50% (i.e., 1.5 times) higher than GF-ATTACK. Third,

Budget	Dataset	CoRA					CiteSeer				
	Models	GCN	SAGE	GAT	Guard	P-GNN	GCN	SAGE	GAT	Guard	P-GNN
(unattk)		81.6	80.8	83.9	83.4	78.3	73.9	76.1	74.9	75.2	74.6
$\mathcal{B}=1$	RL-S2V	3.4	**3.8**	3.8	7.4	**7.9**	4.6	5.7	4.0	**9.9**	3.3
	GF-ATTACK	5.0	2.9	1.7	2.5	3.3	4.9	4.9	**20.8**	4.3	3.7
	TANDIS	**19.2**	2.6	**8.0**	**12.6**	4.9	**33.2**	**10.8**	6.9	7.5	**4.1**
$\mathcal{B}=5$	RL-S2V	7.6	7.9	8.8	15.3	16.6	10.2	13.8	11.1	**28.6**	8.0
	GF-ATTACK	22.6	**16.8**	**21.5**	20.6	15.1	35.5	22.0	**34.7**	18.0	**14.8**
	TANDIS	**42.5**	14.5	**21.5**	**26.7**	**29.5**	**44.9**	**23.1**	26.4	13.6	10.7
$\mathcal{B}=10$	GF-ATTACK	36.3	**25.2**	34.6	30.0	25.0	52.8	30.0	44.8	**28.4**	**24.9**
	TANDIS	**47.2**	23.0	**35.2**	**31.9**	**60.0**	**54.4**	**31.1**	**44.9**	22.7	19.6
$\mathcal{B}=20$	GF-ATTACK	53.6	38.0	48.3	**42**	45.8	**67.5**	38.1	61.4	**39.2**	37.8
	TANDIS	**58.1**	**40.5**	**53.4**	**42**	**81.1**	61.2	**40.6**	**65.4**	37.1	**38.7**

Table 2: Node Classification: Drop in DA against budget. We denote GRAPHSAGE and GNN-GUARD as SAGE and Guard respectively. "unattk" represents the accuracy before attack.

	Link Prediction			Pairwise NC		
Models	GCN	SAGE	GAT	GCN	SAGE	GAT
GF-ATTACK	**5.6**	**4.4**	6.4	0.2	0.6	0.4
TANDIS	3.5	3.4	**7.4**	**0.8**	**1.8**	**1.2**

(a) LP and PNC

Models	GCN	SAGE	GAT	P-GNN	GNN-GUARD
(unattacked)	86.2	86.5	87.6	86.3	83.6
RL-S2V	2.8	2.3	2.7	2.1	5.2
GF-ATTACK	36.9	3.8	**36.5**	3.4	11.7
TANDIS	**41.6**	**9.7**	20.9	**3.5**	**13.7**

(b) NC

Table 3: Drop in accuracy (DA) in PubMed at budget $= 1$.

neighborhood distortion is an effective mechanism to perturb graphs and, hence, potential solutions to defence strategies may lie in recognizing and deactivating graph links that significantly distort node neighborhoods.

4.2 Impact on PubMed

Efficacy: Consistent with previous results, the impact of TANDIS on NC is higher than in LP and PNC (See Figs. 4a-4c). Furthermore, we note that P-GNN and GRAPHSAGE are more robust than other models. We are unable to include GF-ATTACK in these plots due to its high inference times. Nonetheless, we compare with GF-ATTACK and RL-S2V for budget $= 1$ in Table 3. As visible, TANDIS obtains the best performance in most of the cases.

4.3 Running Time

Figure 4d compares the inference times of TANDIS and the second-best baseline GF-ATTACK, i.e. the running time of finding black-box perturbations. We do not compare with RL-S2V, since it fails to complete running for any $\mathcal{B} > 1$ even after 10 hours. TANDIS (represented by solid lines) is more than 3 orders of magnitude faster than GF-ATTACK for the largest dataset, PubMed. Also, time taken by TANDIS grows at a much lower rate with the increase in budget than GF-ATTACK. Furthermore, TANDIS requires training only once per dataset and the entire training procedure takes upto 3 hours even for the largest dataset, i.e. PubMed.

4.4 Potential Detection Mechanisms

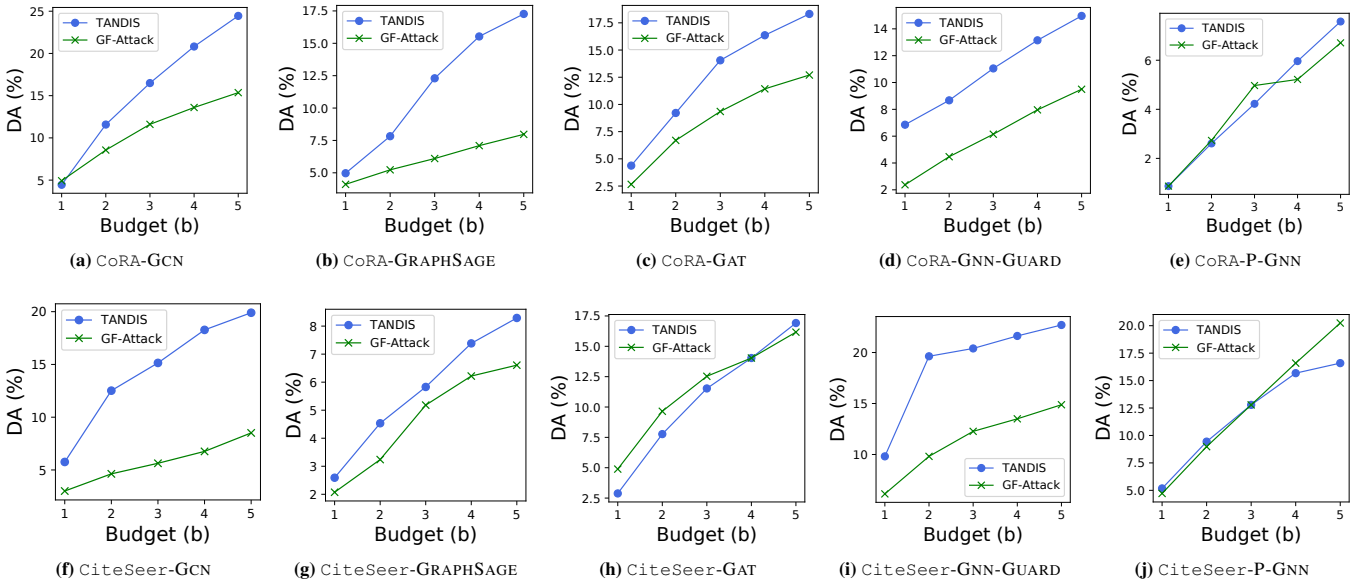
Here, we conduct an experiment on CoRA dataset to find correlations between perturbations chosen by our attacks and the ones chosen according to common network properties. The aim is to **(1)** interpret our attacks in graph space and **(2)** identify potential ways to catch our evading attack.

In particular, we consider all node pairs of the form (t, v), where t is the target and v is the node to which an edge may be added or removed. We characterize each v with several graph level properties such as clustering coefficient, conductance, etc (See Sharma et al. (2021) for the complete list). Corresponding to each property, we create a sorted list P of nodes in descending order of their property values. Similarly, we create another list D with nodes that are sorted based on the distortion produced due to removal/addition of edge (t, v). We next compute the *Spearman's rank correlation* coefficient between P and D and its p-value. Properties with statistically significant correlations may indicate graph-level features that govern the selection of perturbations by TANDIS and thus, effective in detecting these attacks.

Table 4 presents the properties that are statistically significant. These results indicate that edges to nodes of high *clustering coefficient (cc)* lead to large distortion. This result is not surprising since a node with high cc short-circuits

Feature	Correlation (p-value)
Feature Similarity (fs)	$\mathbf{-0.20^* \pm 0.196}$
Degree (dg)	$\mathbf{-0.15^* \pm 0.008}$
Local Clustering Coeff. (cc)	$\mathbf{+0.25^* \pm 0.004}$

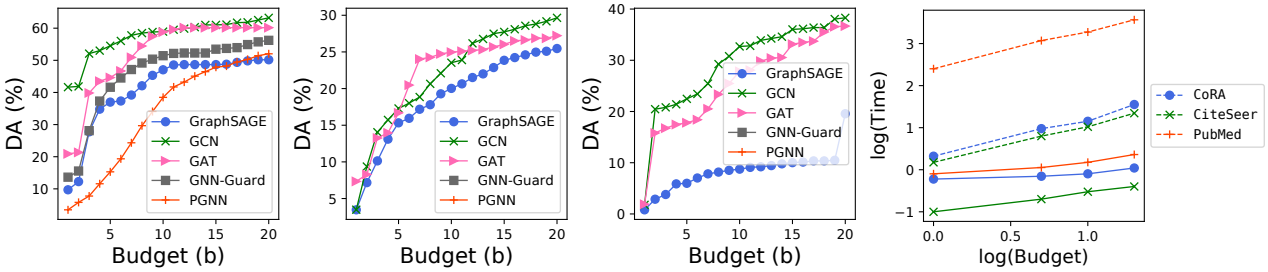
Table 4: Correlation of node properties with neighborhood distortion (* indicates $p < 0.001$).



(a) CoRA-GCN **(b)** CoRA-GRAPHSAGE **(c)** CoRA-GAT **(d)** CoRA-GNN-GUARD **(e)** CoRA-P-GNN

(f) CiteSeer-GCN **(g)** CiteSeer-GRAPHSAGE **(h)** CiteSeer-GAT **(i)** CiteSeer-GNN-GUARD **(j)** CiteSeer-P-GNN

Figure 3: Pairwise node classification: Drop in accuracy (DA%) in (a-d) CoRA and (e-h) CiteSeer. Higher means better.



(a) NC (b) LP (c) PNC (d) Inference Time

Figure 4: Performance of TANDIS in the PubMed dataset (a-c), (d) Inference times of TANDIS (solid lines) and GF-ATTACK (dashed lines) against budget.

the target node to a large number of nodes (or removes this short-circuit in case of edge deletion). The negative correlation with *fs* is also intuitive, since low *fs* with a node indicates that an edge to it injects a large amount of heterophily. The results for *dg* is rather surprising, where a negative correlation indicates that edges to low-degree nodes are more effective. A deeper analysis reveals that *dg* is negatively correlated with *cc*, which may explain the negative correlation with distortion as well. Note that, in GF-ATTACK, it has been shown that perturbing edges with high degree nodes is not an effective attack strategy. Our study is consistent with that result, while also indicating that *cc* is a more effective graph-level parameter.

5 Conclusions

GNNs have witnessed widespread usage for several tasks such as node classification and link prediction. Hence, assessing its robustness to practical adversarial attacks is important to test their applicability in security-critical domains. While the literature on adversarial attacks for GNNs is rich, they are built under the assumption that the attacker has knowledge of the specific GNN model used to train and/or the prediction task being attacked. The key insight in our proposed work is that simply distorting the neighborhood of the target node leads to an effective attack regardless of the underlying GNN model or the prediction task. Hence, hiding the model or the task information from the attacker is not enough. We also find that such perturbations are not correlated with simple network properties and may be hard to detect. Our work thus opens new avenues of research that can focus on defending and detecting such practical attacks to analyze why such perturbations should transfer across different tasks and victim models.

Acknowledgements

We thank the HPC facility of IIT Delhi for computational resources and the reviewers for their constructive feedback.

References

Albert, R.; and Barabási, A.-L. 2002. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1): 47.

Bhattoo, R.; Ranu, S.; and Krishnan, N. M. A. 2022. Learning Articulated Rigid Body Dynamics with Lagrangian Graph Neural Network. In *Advances in Neural Information Processing Systems*.

Bojchevski, A.; and Günnemann, S. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *Proceedings of the 36th International Conference on Machine Learning, ICML*.

Bose, A. J.; Jain, A.; Molino, P.; and Hamilton, W. L. 2019. Meta-graph: Few shot link prediction via meta learning. *arXiv preprint arXiv:1912.09867*.

Chang, H.; Rong, Y.; Xu, T.; Huang, W.; Zhang, H.; Cui, P.; Zhu, W.; and Huang, J. 2020. A Restricted Black-Box Adversarial Framework Towards Attacking Graph Embedding Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Chen, J.; Zhang, D.; Ming, Z.; and Huang, K. 2021. GraphAttacker: A General Multi-Task GraphAttack Framework. *arXiv preprint arXiv:2101.06855*.

Dai, H.; Li, H.; Tian, T.; Huang, X.; Wang, L.; Zhu, J.; and Song, L. 2018. Adversarial attack on graph structured data. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, 1115–1124.

Goyal, N.; Jain, H. V.; and Ranu, S. 2020. GraphGen: A Scalable Approach to Domain-agnostic Labeled Graph Generation. In *Proceedings of The Web Conference 2020*, 1253–1263.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–864.

Gupta, V.; and Chakraborty, T. 2021. Adversarial Attack on Network Embeddings via Supervised Network Poisoning. *arXiv preprint arXiv:2102.07164*.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 1024–1034.

Jain, J.; Bagadia, V.; Manchanda, S.; and Ranu, S. 2021. NeuroMLR: Robust & Reliable Route Recommendation on Road Networks. *Advances in Neural Information Processing Systems*, 34: 22070–22082.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Li, J.; Zhang, H.; Han, Z.; Rong, Y.; Cheng, H.; and Huang, J. 2020. Adversarial attack on community detection by hiding individuals. In *Proceedings of the Web Conference 2020*.

Liu, X.; Si, S.; Zhu, J.; Li, Y.; and Hsieh, C.-J. 2019. A Unified Framework for Data Poisoning Attack to Graph-based Semi-supervised Learning. In *Advances in Neural Information Processing Systems*.

Ma, J.; Ding, S.; and Mei, Q. 2020. Towards More Practical Adversarial Attacks on Graph Neural Networks. *Advances in neural information processing systems*.

Ma, Y.; Wang, S.; Derr, T.; Wu, L.; and Tang, J. 2019. Attacking graph convolutional networks via rewiring. *arXiv preprint arXiv:1906.03750*.

Manchanda, S.; Mittal, A.; Dhawan, A.; Medya, S.; Ranu, S.; and Singh, A. 2020. GCOMB: Learning Budget-constrained Combinatorial Algorithms over Billion-sized Graphs. *Advances in Neural Information Processing Systems*, 33.

McCallum, A. K.; Nigam, K.; Rennie, J.; and Seymore, K. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2): 127–163.

Nemhauser, G. L.; and Wolsey, L. A. 1978. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3): 177–188.

Nishad, S.; Agarwal, S.; Bhattacharya, A.; and Ranu, S. 2021. GraphReach: Locality-Aware Graph Neural Networks using Reachability Estimations. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*.

Pal, A.; Eksombatchai, C.; Zhou, Y.; Zhao, B.; Rosenberg, C.; and Leskovec, J. 2020. PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710.

Ranjan, R.; Grover, S.; Medya, S.; Chakaravarthy, V.; Sabharwal, Y.; and Ranu, S. 2022. GREED: A Neural Framework for Learning Graph Distance Functions. In *Advances in Neural Information Processing Systems*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Sharma, K.; Verma, S.; Medya, S.; Ranu, S.; and Bhattacharya, A. 2021. Task and Model Agnostic Adversarial Attack on Graph Neural Networks. *arXiv preprint arXiv:2112.13267*.

Sun, L.; Wang, J.; Yu, P. S.; and Li, B. 2018. Adversarial Attack and Defense on Graph Data: A Survey. *CoRR*, abs/1812.10528.

Sun, Y.; Wang, S.; Tang, X.; Hsieh, T.-Y.; and Honavar, V. 2020. Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach. In *Proceedings of the Web Conference 2020*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Thangamuthu, A.; Kumar, G.; Bishnoi, S.; Bhattoo, R.; Krishnan, N. M. A.; and Ranu, S. 2022. Unravelling the Performance of Physics-informed Graph Neural Networks for Dynamical Systems. In *Thirty-sixth Conference on Neural*

Information Processing Systems Datasets and Benchmarks Track.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

Wang, B.; and Gong, N. Z. 2019. Attacking Graph-Based Classification via Manipulating the Graph Structure. In *SIGSAC*, 2023–2040.

Wang, B.; Zhou, T.; Lin, M.; Zhou, P.; Li, A.; Pang, M.; Fu, C.; Li, H.; and Chen, Y. 2020. Efficient Evasion Attacks to Graph Neural Networks via Influence Function. *arXiv preprint arXiv:2009.00203*.

Wu, H.; Wang, C.; Tyshetskiy, Y.; Docherty, A.; Lu, K.; and Zhu, L. 2019. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*.

Xu, K.; Chen, H.; Liu, S.; Chen, P.-Y.; Weng, T.-W.; Hong, M.; and Lin, X. 2019a. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019b. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations (ICLR)*.

You, J.; Ying, R.; and Leskovec, J. 2019. Position-aware Graph Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, 7134–7143.

Zhang, H.; Zheng, T.; Gao, J.; Miao, C.; Su, L.; Li, Y.; and Ren, K. 2019. Data Poisoning Attack against Knowledge Graph Embedding. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*.

Zhang, X.; and Zitnik, M. 2020. GNNGuard: Defending Graph Neural Networks against Adversarial Attacks. In *Advances in Neural Information Processing Systems*.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2847–2856.

Zügner, D.; and Günnemann, S. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations (ICLR)*.