GRAPH ADVERSARIAL NETWORKS: PROTECTING INFORMATION AGAINST ADVERSARIAL ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the problem of protecting information when learning with graph structured data. While the advent of Graph Neural Networks (GNNs) has greatly improved node and graph representational learning in many applications, the neighborhood aggregation paradigm exposes additional vulnerabilities to attackers seeking to extract node-level information about sensitive attributes. To counter this, we propose a minimax game between the desired GNN encoder and the worst-case attacker. The resulting adversarial training creates a strong defense against inference attacks, while only suffering small loss in task performance. We analyze the effectiveness of our framework against a worst-case adversary, and characterize the trade-off between predictive accuracy and adversarial defense. Experiments across multiple datasets from recommender systems, knowledge graphs and quantum chemistry demonstrate that the proposed approach provides a robust defense across various graph structures and tasks, while producing competitive GNN encoders.

1 INTRODUCTION

Graph neural networks (GNNs) have brought about performance gains in various tasks involving graph-structured data (Scarselli et al., 2009; Battaglia et al., 2018). A typical example includes movie recommendation on social networks (Ying et al., 2018). Ideally, the recommender system makes a recommendation not just based on the description of an end user herself, but also those of her close friends in the social network. By taking the structured information of friendship in social network into consideration, more accurate prediction is often achieved (Xu et al., 2018; Hamilton et al., 2017). However, with better utility comes more vulnerability to adversarial attacks. To gain sensitive information about a specific user in the network, malicious adversaries could try to infer sensitive information not just only based on the information of the user of interest, but also information of her friends in the network. Such scenarios are increasingly ubiquitous with the rapid growth of users in common social network platforms. The above problem poses the following challenge:

> *How could we protect sensitive information of users in the network from malicious inference attacks while maintaining the utility of service? Furthermore, can we quantify the trade-off between these two goals?*

In this paper, we provide answers to both questions. We propose a simple yet effective algorithm to achieve the first goal through adversarial learning of GNNs, a general framework which we term as **G**raph **A**dversaria**L** Networks (GAL). In a nutshell, the proposed algorithm learns node representations in a graph by simultaneously preserving rich information about the target task and filtering information from the representations that is related to the sensitive attribute via a min-max game (Figure 1). While the saddle point optimization formulation is not new and has been applied broadly (Goh & Sim, 2010; Goodfellow et al., 2014; Madry et al., 2017; Ben-David et al., 2010), we are the first to formulate the attribute inference attack problem on graphs, and to demonstrate min-max optimization is effective for GNNs in these settings, theoretically and empirically.

We provide theoretical guarantees for our algorithm, and quantify the inherent trade-off between GNN predictive accuracy and adversarial defense. First, we prove a *lower bound* for the inference error over the sensitive attribute that any worst-case attacker has to incur under our algorithm. Second, we quantify how much one has to pay in terms of predictive accuracy for protecting sensitive information from adversarial attacks. Specifically, we prove that the loss in terms of predictive accuracy is proportional to how the target task is correlated with the sensitive attribute in input node features.

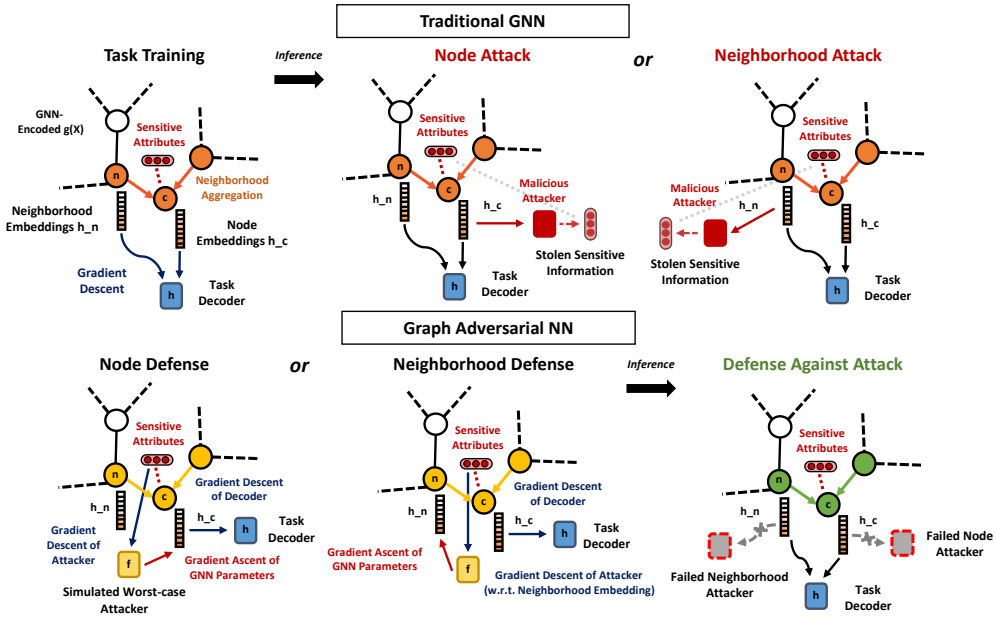


Figure 1: **G**raph **A**dversaria**L** Networks (GAL). GAL defends node and neighborhood inference attacks via a min-max game between the task decoder (blue) and a simulated worst-case attacker (yellow) on both the embedding (descent) and the attributes (ascent). Malicious attackers will have difficulties extracting sensitive attributes at inference time from GNN embeddings trained with our framework.

Empirically, we corroborate our theory and the effectiveness of the proposed framework on 6 graph benchmark datasets. We show that our framework can both train a *competitive* GNN encoder and perform *adequate defense*. For instance, our algorithm successfully decreases the AUC of a gender attacker by 10% on the Movielens dataset while only suffering 3% in task performance. Furthermore, our framework is robust against a new set of attacks we term "neighborhood attacks" or "n-hop attacks", where attackers can infer node-level sensitive attributes from embeddings of 1- and n-distant neighbors. We verify that in these new settings, our algorithm remains effective. Finally, our theoretical bounds on the trade-off between accuracy and defense agree with experimental results.

In summary, we formulate the attribute inference attack problem on GNNs. In this new setting, we show that GNNs trained with a min-max game framework GAL achieve both good predictive power and defense, theoretically (Theorem 3) and empirically (Table 2). Our theory also quantifies the trade-off between accuracy and privacy (Theorem 1), and is supported by experiments (Figure 4).

1.1 Related Work

Adversarial Attack on Graphs. The main difference between our work and previous works in adversarial attacks on graphs (Bojchevski & Günnemann, 2019; Ma et al., 2019; Xu et al., 2019a; Dai et al., 2018; Chang et al., 2019) is the inherent difference in formulation. Prior works focus on perturbations of a graph, e.g., by adding or removing edges, that maximize the loss of "victim" nodes for GNNs. In contrast, attackers in our framework do not alter graph structure; instead, they seek to learn a parameterized network that extracts critical information from the GNN embeddings.

Differential Privacy. We tackle a privacy problem where the adversary's goal is to infer some attribute of a node in a graph. One related but different notion of privacy is known as differential privacy (Dwork et al., 2014), which aims at defending the so-called membership inference attack (Shokri et al., 2017; Nasr et al., 2018). Roughly speaking, here the goal is to design randomized algorithms so that an adversary cannot guess whether an individual appears in the training data or not by looking at the output of the algorithm. As a comparison, in this work we are concerned about attribute inference attacks (Gong & Liu, 2018; Jia & Gong, 2018), and our goal is to find transformations of the data, so that the adversary cannot accurately infer a specific attribute value of the data.

Information Bottleneck. Our work is also closely related to the *information bottleneck* method, which seeks to simultaneously compress data while preserving enough information for target

tasks (Tishby et al., 2000; Alemi et al., 2016; Tishby & Zaslavsky, 2015). Here we provide a brief discussion to contrast our method vs. the information bottleneck method. At a high level, the information bottleneck framework could be understood as maximizing the following objective: $I(Y;Z) - \beta I(X;Z)$ with $\beta > 0$. Specifically, there is no sensitive attribute A appearing in the formulation. On the other hand, the minimax framework in this work is about a trade-off problem, i.e., finding Z that balances between containing information about Y and A, respectively. In particular, the original input X does not appear in our formulation: $I(Y;Z) - \beta I(A;Z)$. Overall, despite their surface similarity, our adversarial defense method is significantly different from that of the information bottleneck method in terms of both formulation and goals.

2 PRELIMINARIES

We begin by introducing our notation. Let $\mathcal{D}$ be a distribution over a sample space $\mathcal{X}$. For an event $E \subseteq \mathcal{X}$, we use $\mathcal{D}(E)$ to denote the probability of E under $\mathcal{D}$. Given a feature transformation function $g : \mathcal{X} \to \mathcal{Z}$ that maps instances from the input space $\mathcal{X}$ to feature space $\mathcal{Z}$, we define $g_\sharp \mathcal{D} := \mathcal{D} \circ g^{-1}$ to be the pushforward distribution of $\mathcal{D}$ under g, i.e., for any event $E' \subseteq \mathcal{Z}$, $g_\sharp \mathcal{D}(E') := \mathcal{D}(\{x \in \mathcal{X} \mid g(x) \in E'\})$. For two distributions $\mathcal{D}$ and $\mathcal{D}'$ over the same sample space Ω, let $d_{\text{TV}}(\mathcal{D}, \mathcal{D}')$ be the total variation (TV) distance between them: $d_{\text{TV}}(\mathcal{D}, \mathcal{D}') := \sup_{E \subseteq \Omega} |\mathcal{D}(E) - \mathcal{D}'(E)|$.

Graph Neural Networks. GNNs learn node and graph representations for predictions on nodes, relations, or graphs (Scarselli et al., 2009). The input is a graph $G = (V,E)$. Each node $u \in V$ has a feature vector X_u, and each edge $(u, v) \in E$ has a feature vector $X_{(u,v)}$. GNNs iteratively compute a representation for each node. Initially, the node representations are the node features: $X_u^{(0)} = X_u$. In iteration $k = 1, \ldots, K$, a GNN updates the node representations $X_u^{(k)}$ by aggregating the neighboring nodes' representations (Gilmer et al., 2017).

$$X_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{\left(X_u^{(k-1)}, X_v^{(k-1)}, X_{(u,v)}\right) : u \in \mathcal{N}(v)\right\}\right), \quad \forall k \in [K]. \tag{1}$$

We can compute a graph representation X_G by pooling the final node representations. Many GNNs, with different aggregation and graph pooling functions, have been proposed (Du et al., 2019; Fey, 2019; Duvenaud et al., 2015; Defferrard et al., 2016; Xu et al., 2020; Velickovic et al., 2018).

Attribute Inference Attack on GNNs. While being a powerful paradigm for learning node and graph representations for downstream tasks, GNNs also expose huge vulnerability to attackers whose goal is to infer sensitive attributes of individual nodes from the learned representations. The goal of an attacker $\mathcal{A}$ is to reconstruct a sensitive attribute A_v of user v by looking at its node representation $X_v^{(K)}$. Here, $A_v \in \{0,1\}$[1] could be the user's age, gender, or income etc. We emphasize that simply removing the sensitive attribute A_v from the embedding X_v is not sufficient, because A_v may be inferred from some different but correlated features.

We study the problem above from an information-theoretic perspective. Let $\mathcal{F}_A := \{f : \mathbb{R}^{d_K} \to \{0,1\}\}$ denote the set of adversarial attackers who have access to the GNN's representation before the last layer as input, and then output a guess of the sensitive attribute. Let $\mathcal{D}$ be a joint distribution over the node input X, the sensitive attribute A, as well as the target variable Y. We define the *advantage* (Goldwasser & Bellare, 1996) of adversarial attackers as

$$\text{Adv}_{\mathcal{D}}(\mathcal{F}_A) := \sup_{f \in \mathcal{F}_A} \left| \Pr_{\mathcal{D}}(f(Z) = 1 \mid A = 1) - \Pr_{\mathcal{D}}(f(Z) = 1 \mid A = 0) \right|, \tag{2}$$

where Z is the random variable that denotes *node representations* after applying a sequence of GNN layer transformations to input X. Here, f represents a particular attacker and the supresum in (2) corresponds to the *worst-case attacker* from a class $\mathcal{F}_A$. If $\text{Adv}_{\mathcal{D}}(\mathcal{F}_A) = 1$, then there exists an attacker who can almost surely guess the sensitive attribute A by looking at the GNN representations Z. Hence, our goal is to design an algorithm that learns GNN representations Z such that $\text{Adv}_{\mathcal{D}}(\mathcal{F}_A)$ is small, which implies successful defense against adversaries. Throughout the paper, we assume that $\mathcal{F}_A$ is symmetric, i.e., if $f \in \mathcal{F}_A$, then $1 - f \in \mathcal{F}_A$ as well.

[1] We assume binary attributes for ease of exposition. Extension to categorical attributes is straightforward.

3 ALGORITHM AND ANALYSIS

In this section, we first relate the aforementioned *advantage* of the adversarial attackers to a quantity that measures the *ability of attackers* in predicting a sensitive attribute A by looking at the node representations $Z = X^{(K)}$. Inspired by this relationship, we proceed to introduce a min-max game between the GNN feature encoder and the worst-case attacker. We analyze the trade-off between accuracy and defense, and provide theoretical guarantees for our algorithm.

Given a symmetric set of attackers $\mathcal{F}_A$, we observe that $1 - \text{Adv}_{\mathcal{D}}(\mathcal{F}_A)$ corresponds to the sum of best Type-I and Type-II *inference error* any attacker from $\mathcal{F}_A$ could hope for:

$$1 - \text{Adv}_{\mathcal{D}}(\mathcal{F}_A) = \inf_{f \in \mathcal{F}_A} \left(\Pr_{\mathcal{D}}(f(Z) = 1 \mid A = 0) + \Pr_{\mathcal{D}}(f(Z) = 0 \mid A = 1) \right). \tag{3}$$

Hence, to minimize $\text{Adv}_{\mathcal{D}}(\mathcal{F}_A)$, a natural strategy is to learn parameters of the GNN so that it filters out the sensitive information in the node representation while still tries to preserve relevant information w.r.t. the target task of inferring Y. This translates into the following unconstrained optimization problem with a trade-off parameter $\lambda > 0$:

$$\min_{h,g} \max_{f \in \mathcal{F}_A} \quad \varepsilon_Y(h \circ g) - \lambda \cdot \varepsilon_A(f \circ g) \tag{4}$$

Here we use $\varepsilon_Y(\cdot)$ and $\varepsilon_A(\cdot)$ to denote the cross-entropy error on predicting target task Y and sensitive attribute A respectively. g corresponds to the GNN feature encoder, h is the last layer of GNN for downstream task, and f is the adversarial attacker. Note that the above formulation is different from the poisoning attacks on graphs literature (Zügner & Günnemann, 2019), where their goal is to learn a robust classifier under graph perturbation. It is also worth pointing out that the optimization formulation in (4) admits an interesting game-theoretic interpretation, where two agents f and g play a game whose score is defined by the objective function in (4). Intuitively, the attacker f seeks to minimize the sum of Type-I and Type-II inference error while the GNN g plays against f by learning transformation to removing information about the sensitive attribute A. Clearly, the hyperparameter λ controls the trade-off between accuracy and defense. On one hand, if $\lambda \to 0$, we barely care about the defense of A and devote all the focus to minimize the predictive error. On the other extreme, if $\lambda \to \infty$, we are only interested in defending against the potential attacks.

The first term in the objective function of (4) acts as an incentive to encourage GNN to preserve task-related information. But will this incentive compromise the information of A? As an extreme case if the target variable Y and the sensitive attribute A are perfectly correlated, then it should be clear that there is a trade-off in achieving accuracy and preventing information leakage of the attribute. Next, we provide an analysis to quantify this inherent trade-off.

3.1 TRADE-OFF BETWEEN PREDICTIVE ACCURACY AND ADVERSARIAL ADVANTAGE

As we discussed above, in general it is impossible to simultaneously achieve the goal of defense and accuracy maximization. Nevertheless, can we quantify such trade-off? Theorem 1 characterizes a trade-off between the cross-entropy error of task predictor and the advantage of the adversarial attackers:

Theorem 1. Let Z be the node representations produced by a GNN g and $\mathcal{F}_A$ be the set of all binary predictors. Define $\delta_{Y|A} := |\Pr_{\mathcal{D}}(Y = 1 \mid A = 0) - \Pr_{\mathcal{D}}(Y = 1 \mid A = 1)|$. Then $\forall h \in \mathcal{H}$,

$$\varepsilon_{Y|A=0}(h \circ g) + \varepsilon_{Y|A=1}(h \circ g) \geq \delta_{Y|A} - \text{Adv}_{\mathcal{D}}(\mathcal{F}_A). \tag{5}$$

Remark. First, $\varepsilon_{Y|A=a}(h \circ g)$ denotes the conditional cross-entropy error of predicting Y given $A = a \in \{0, 1\}$. Hence the above theorem says that, up to a certain threshold given by $\delta_{Y|A}$ (which is a task-specific constant), any target predictor based on the features given by GNN g has to incur a large error on at least one of the sensitive groups. Specifically, the smaller the adversarial advantage $\text{Adv}_{\mathcal{D}}(\mathcal{F}_A)$, the larger the error lower bound. The lower bound in Theorem 1 is *algorithm-independent*, and considers the strongest possible adversarial attacker, hence it reflects an *inherent* trade-off between task utility and defense to attacks. Moreover, Theorem 1 does not depend on the marginal distribution of the sensitive attribute A. Instead, all the terms in our result only depend on the conditional distributions given $A = 0$ and $A = 1$. This is often more desirable than bounds involving mutual information, e.g., $I(A, Y)$, because $I(A, Y)$ is close to 0 if the marginal distribution of A is highly imbalanced. The overall error also admits a lower bound:

Corollary 2. Assume the conditions in Theorem 1 hold, then

$$\varepsilon_Y(h \circ g) \geq \min\{\Pr_{\mathcal{D}}(A=0), \Pr_{\mathcal{D}}(A=1)\} \cdot \big(\delta_{Y|A} - \text{Adv}_{\mathcal{D}}(\mathcal{F}_A)\big).$$

Our lower bounds in Theorem 1 and Corollary 2 capture the price we have to pay for defense. Taking a closer look at the constant term $\delta_{Y|A}$ that appears in the lower bound, realize that

- If the target variable Y is *statistically independent* of the sensitive attribute A, then $\delta_{Y|A} = 0$, hence the first term in the lower bound gracefully reduces to 0, meaning that there will be *no loss* of accuracy even if we perfectly minimize the adversarial advantage.
- If the target variable Y is a *bijective encoding* of the sensitive attribute A, i.e., $Y = A$ or $Y = 1 - A$, then $\delta_{Y|A} = 1$, hence the first term in the lower bound achieves the maximum value of 1, meaning that in this case, if the adversarial advantage $\text{Adv}_{\mathcal{D}}(\mathcal{F}_A)\big) = 0$, then no matter what predictor h we use, it has to incur a joint error of at least $\min\{\Pr_{\mathcal{D}}(A = 0), \Pr_{\mathcal{D}}(A = 1)\}$. Note that this is the error rate we achieve by using constant prediction.

3.2 Guarantees Against Adversarial Attacks

Next, we provide guarantee for GAL defense in (4). The analysis on the optimization trajectory of a general non-convex-concave game (4) is still an active area of research (Daskalakis & Panageas, 2018; Nouiehed et al., 2019; Lin et al., 2019) and hence beyond the scope of this paper. Therefore, we assume that we have access to the min-max stationary point solution of (4), and focus on understanding how the solution of (4) affects the effectiveness of our defense.

In what follows we analyze the true error that a worst-case adversary has to incur in the limit, when both the task classifier and the adversary have unlimited capacity, i.e., they can be any randomized functions from $\mathcal{Z}$ to $\{0, 1\}$. We use the population loss rather than the empirical loss in our objective function. Under such assumptions, given any node embedding Z from a GNN g, the worst-case adversary is the *conditional distribution*:

$$\min_{f \in \mathcal{F}_A} \varepsilon_A(f \circ g) = H(A \mid Z), \qquad \arg\min_{f \in \mathcal{F}_A} \varepsilon_A(f \circ g) = \Pr(A = 1 \mid Z).$$

It follows from a symmetric argument that $\min_{h \in \mathcal{H}} \varepsilon_Y(h \circ g) = H(Y \mid Z)$. Hence we can simplify the GAL objective (4) to the following form where the only variable is the GNN embedding Z:

$$\min_{Z} \quad H(Y \mid Z) - \lambda \cdot H(A \mid Z) \tag{6}$$

We can now analyze the error that has to be incurred by the worst-case adversary:

Theorem 3. Let Z^* be the optimal GNN node embedding of (6) and define $H^* := H(A \mid Z^*)$. Then for any adversary $f : \mathcal{Z} \to \{0, 1\}$, $\Pr(f(Z) \neq A) \geq H^*/2 \lg(6/H^*)$.

Theorem 3 shows that whenever the conditional entropy $H^* = H(A \mid Z^*)$ is large, the inference error incurred by any (randomized) adversary has to be at least $\Omega(H^*/\log(1/H^*))$. In practice, H^* could be adjusted by tuning the trade-off parameter λ. Theorem 3 also shows that GAL helps defense since we always have $H(A \mid Z) \geq H(A \mid X)$ for any GNN features Z created from input X.

As a final note, recall that the representations Z appearing in the bounds above depend on the *graph structure* (as evident from Eq. (1)), and the inference error in Theorem 3 is over the representations Z, this means that the defense could potentially be applied against *neighborhood attacks*, which we provide in-depth empirical validation in Section 4.3.

4 Experiments

In this section, we demonstrate the effectiveness of GAL in defending attribute inference attacks on graphs. Specifically, we address the following three questions:

- ➔ 4.1: Is GAL effective across different tasks and GNN architectures?
- ➔ 4.2: What is the landscape of the tradeoff with respect to the hyperparameter λ?
- ➔ 4.3: Can GAL also defend neighborhood and n-hop attacks?

Table 1: **Summary of benchmark dataset statistics**, including number of nodes $|V|$, number of nodes with sensitive attributes $|S|$, number of edges $|E|$, node-level non-sensitive features d, target task and adversarial task, and whether the experiment concerns with multi-set defense.

DATASET	$\lvert V\rvert$	$\lvert S\rvert$	$\lvert E\rvert$	d	Multi?	METRICS: TASK	ADVERSARY
CITESEER	3,327	3,327	4,552	3,703	✗	AUC	Macro-F1
PUBMED	19,717	19,717	44,324	500	✗	AUC	Macro-F1
QM9	2,383,055	2,383,055	2,461,144	13	✗	MAE	MAE
ML-1M	9,940	6,040	1,000,209	1 (id)	✗	RMSE	Macro-F1/AUC
FB15K-237	14,940	14,940	168,618	1 (id)	✓	MRR/Hits@10	Macro-F1
WN18RR	40,943	40,943	173,670	1 (id)	✗	MRR/Hits@10	Macro-F1

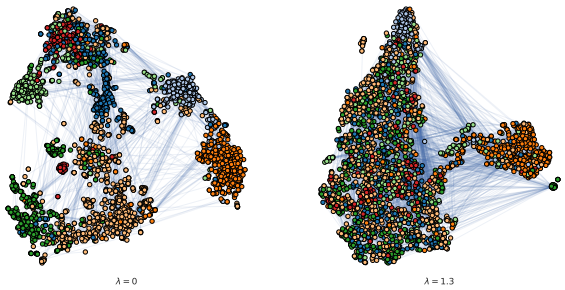


Figure 2: **GAL effectively protects sensitive information**. Both panels show t-SNE plots of feature representations of a graph under different defense strengths λ (0 vs. 1.3). Node colors represent node classes.

We empirically confirm all three questions. To stress test the robustness of GAL, we consider a variety of tasks and GNN encoder architectures. Specifically, we experiment on 5 link-prediction benchmarks (Movielens-1M, FB15k-237, WN18RR, CiteSeer, Pubmed) and 1 graph regression benchmark (QM9), which covers both defense of *single* and *multiple* attributes. More importantly, our goal is not to challenge state-of-the-art training schemes, but to observe the effect in reducing attackers' accuracies while maintaining good performance of the downstream task. Our experiments can be classified into three major categories.

Robustness. We run GAL on all six datasets under a variety of GNN architectures, random seeds and defense strength λ. We report average performance over five runs. We perform ablation study on widely used GNN architectures (Velickovic et al., 2018; Kipf & Welling, 2017; Vashishth et al., 2019; Defferrard et al., 2016; Gilmer et al., 2017; Xu et al., 2019b), and select the best-performing GNNs in the task: e.g., CompGCN is specifically designed for knowledge-graph-related applications, etc.

Trade-off. We compare performance under a wide range of defense strength λ on Movielens-1M. We show that GAL defends the adversarial attackers to a great extent while only suffering minor losses to downstream task performance.

Neighborhood Attacks. In this setting, an attacker also has access to the embeddings of *neighbors* of a node, e.g. the attacker can infer sensitive attribute A_v from $X_w^{(K)}$ (instead of $X_v^{(K)}$) such that there is a path between v and w. Since GNN's neighborhood-aggregation paradigm may introduce such information leakage, attacks shall achieve nontrivial performance. We further generalize this to an n-hop scenario.

In all experiments, the attackers only have access to the training set labels along with embeddings from the GNN, and the performance is measured on the held-out test set. A summary of the datasets, including graph attributes, task, and adversary metrics, is in Table 1. Detailed experimental setups may be found in Appendix B. Overall, our results have successfully addressed all three questions, demonstrating that our framework is attractive for node- and neighborhood-level attribute defense across downstream tasks, GNN architectures, and defense strength λ.

4.1 ROBUST NODE-LEVEL ATTRIBUTE DEFENSE

We first demonstrate that our framework is effective across a wide variety of GNN architectures and tasks. Quantitatively, we report task performances $T_{1/2/3}$ as well as adversary performances $A_{1/2/3}$, both specific to respective datasets in Table 2. Note that fixed-embedding (Bose & Hamilton, 2019)

Table 2: **Performance of predictions and defense on benchmark datasets**. Best-performing GNNs and best defense results are highlighted in bold. The "—" mark stands for "same as above." We also implemented (Bose & Hamilton, 2019)'s fixed embedding results and tested under the same setting, and, since the former work does not allow tradeoff tuning, incompatible fields are marked with "N/A".

	Encoder	Adversary	λ_1	λ_2	λ_3	T_1	T_2	T_3	A_1	A_2	A_3
ML-1M	ChebConv	Gender (F1)	0	0.5	10	**0.852**	0.880	1.338	0.693	0.594	0.501
		Gender (AUC)	—	—	—	—	—	—	0.708	0.598	0.537
		Age	—	—	—	—	0.871	1.337	0.289	0.145	0.100
		Occupation	—	—	—	—	0.874	1.356	0.087	0.041	0.024
	GCN	Gender (F1)	0	0.5	10	0.931	0.999	1.256	0.699	0.584	**0.436**
		Gender (AUC)	—	—	—	—	—	—	0.711	0.606	**0.504**
		Age	—	—	—	—	0.999	1.211	0.238	0.186	**0.074**
		Occupation	—	—	—	—	0.997	1.241	0.057	0.051	**0.013**
	GAT	Gender (F1)	0	0.5	10	0.903	0.976	1.205	0.669	0.504	0.441
		Gender (AUC)	—	—	—	—	—	—	0.685	0.526	0.505
		Age	—	—	—	—	0.966	1.195	0.219	0.104	0.075
		Occupation	—	—	—	—	0.975	1.192	0.058	0.033	**0.013**
	(Bose & Hamilton, 2019)	Gender (F1)	N/A	N/A	N/A	0.874	N/A	N/A	0.667	N/A	N/A
		Gender (AUC)	N/A	N/A	N/A	—	N/A	N/A	0.678	N/A	N/A
		Age	N/A	N/A	N/A	—	N/A	N/A	0.188	N/A	N/A
		Occupation	N/A	N/A	N/A	—	N/A	N/A	0.051	N/A	N/A
CITESEER	GIN	Doc. Class	0	1.25	1.75	0.815	0.731	0.726	0.450	0.318	**0.292**
	GCN	Doc. Class	—	—	—	**0.890**	0.849	0.832	0.601	0.335	0.317
	GAT	Doc. Class	—	—	—	0.889	0.865	0.853	0.604	0.509	0.406
	ChebConv	Doc. Class	—	—	—	0.877	0.833	0.822	0.571	0.405	0.336
PUBMED	GIN	Doc. Class	0	1.15	1.25	0.817	0.782	0.799	0.720	0.403	0.474
	GCN	Doc. Class	—	—	—	**0.964**	0.912	0.895	0.776	0.528	**0.374**
	GAT	Doc. Class	—	—	—	0.930	0.862	0.780	0.768	0.680	0.536
	ChebConv	Doc. Class	—	—	—	0.926	0.891	0.890	0.778	0.592	0.341
QM9	NNConv	Polarizability	0	0.05	0.5	**0.121**	0.132	0.641	1.054	1.359	**3.100**
WN18RR	CompGCN	Word Sense	0	1.0	1.5	**0.462/0.530**	0.437/0.515	0.403/0.492	0.208	**0.131**	0.187
		POS tag	—	—	—	—	0.430/0.510	0.395/0.490	0.822	**0.607**	0.705
FB15K-237	CompGCN	Ent. Attr.	0	1.0	1.5	**0.351/0.529**	0.320/0.488	0.319/0.487	0.682	0.641	**0.630**

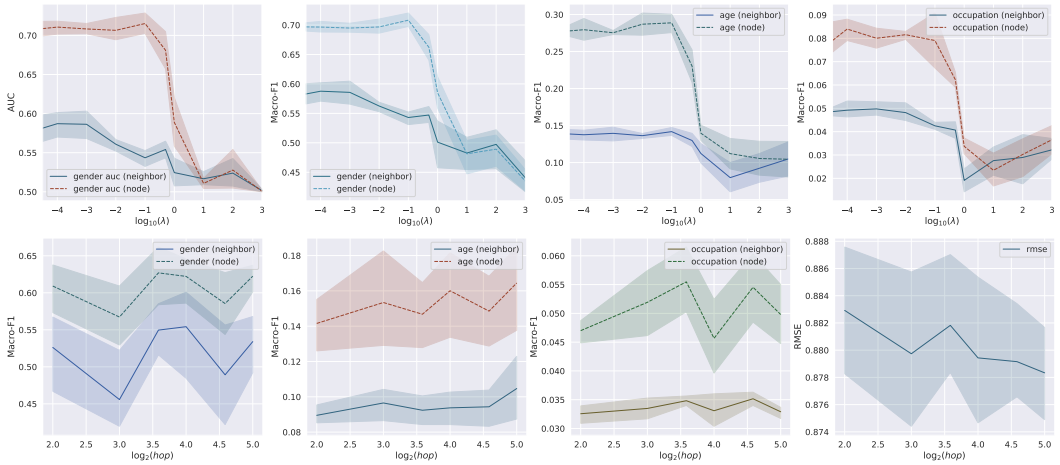


Figure 3: **Performance of neighborhood-level (top) and n-hop (bottom) attribute defense** on a 3-layer/2-layer ChebConv-GNN with Movielens-1M dataset under different λ/"hop" respectively. Band represents 95% confidence interval over five runs. For n-hop experiments, λ is fixed to be 0.8.

results are only reported for Movielens-1M dataset because on other datasets, the experimental setups are not suitable for comparison. When defense strength $\lambda = 0$, the defense has *no effect*, so it works as a baseline result. Across all datasets, we witness a significant drop in attacker's performance with a rather minor decrease in the downstream task performance.

From a qualitative perspective, the effect of adversarial training is also apparent: in Figure 2, we visualized the t-SNE (van der Maaten & Hinton, 2008)-transformed representations of graph embeddings under different regularization strengths on the Cora dataset. We observe that under a high regularization strength, node-level attributes are better mixed in the sense that nodes belonging to different classes are harder to separate from each other on a graph-structural level.

4.2 FINE-GRAINED TRADE-OFF TUNING WITH λ

Second, we show that the hyper-parameter λ in GAL is indeed a good trade-off hyper-parameter that well balances downstream task performance and defense effectiveness. We perform experiments with ChebConv-GNN and the Movielens-1M dataset, and test λ values ranging from 0 to 10^3. Our results in Figure 4 show that as λ increases, it is indeed more difficult for the attacker to extract sensitive information from GNN embeddings; the downstream task performance decreases steadily but with a small rate. When λ is larger than 10, training destabilizes further due to inherent difficulties in optimization, which is demonstrated by the higher variance in task performance. However, this trade-off is less visible for larger graphs like FB15k-237 and WN18RR, where the effects of lambda tends to be less monotonic due to difficulties in optimization. The detailed results are in Table 2.

4.3 NEIGHBORHOOD ATTACK AND n-HOP GENREALIZATION

Finally, we consider the scenario where the attacker seeks to extract node-level information from embeddings of *neighbors*. This is, in particular, a serious concern in GNNs where the message passing layers incorporate neighborhood-level information and hence the information leakage. Our results in Figure 3 verify that, without any form of defense ($\lambda = 0$), the attacker indeed achieves nontrivial performance by accessing the embeddings of neighbors on the Movielens-1M dataset with a simple 2-layer ChebConv-GNN. We further run comprehensive experiments where the defense targets neighborhood-level embeddings, and we observe the degradation of the attacker's performance (both on the node- and neighbor-level embeddings) as λ increases. The results demonstrate that under low values of λ, the neighborhood-level defense has more effects on neighborhood attackers than node attackers, suggesting that node-level embeddings are less protected. However, as λ continues to increase, the degradation in the performance of node attackers is more visible. An extension of this setup is to consider neighborhood-level attacks as "single-hop", where we can naturally define "n-hop" as attackers having accesses to only embeddings of nodes that are n-distant away from the target node.

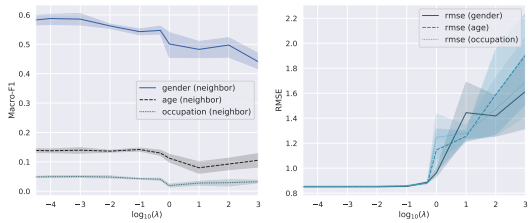


Figure 4: **Performance of node-level attribute defense** on Movielens-1M dataset under different λ, with 95% confidence interval over five runs.

Since finding n-distant neighbors for arbitrary nodes in a large graph on-the-fly is computationally inefficient during training (in the sense that the complexity bound involves $|E|$ and $|V|$), we propose a Monte-Carlo algorithm that probabilistically finds such neighbor in $O(n^2)$ time, the details of which can be found in appendix. We report results under different "hops" with the same encoder-λ setting on Movielens-1M, shown in Figure 3. In general, we observe that as "hop" increases, the retrieved embedding contains less information about the target node. Therefore, adversarial training's effect will have less effect on the degradation of the target task, which is demonstrated by the steady decrease of RMSE. The fluctuations in the trend of node- and neighborhood-level attackers are due to the probabilistic nature of the Monte-Carlo algorithm used to sample neighbors, where it may end up finding a much closer neighbor than intended, destabilizing the training process. This is another trade-off due to limited training time, yet the general trend is still visible, certifying our assumptions.

5 CONCLUSION

In this work, we address the problem of attribute inference attacks on graphs with GNNs. Our proposed framework, termed GAL, introduces a minimax game between the desired GNN encoder and the worst-case attacker. The resulting adversarial training creates a strong defense against inference in terms of a provable lower bound on the error rate, while only suffering a marginal loss in task performance. We also show an information-theoretic bound to characterize the inherent trade-off between accuracy and defense. Experiments on several benchmark datasets show that our method can readily complement existing algorithms deployed in downstream tasks to address information security concerns on graph-structured data.

REFERENCES

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

Pablo Barceló, Egor V Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2019.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *ICML*, 2019.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 2013. to appear.

Avishek Bose and William Hamilton. Compositional fairness constraints for graph embeddings. In *International Conference on Machine Learning*, pp. 715–724, 2019.

Chris Calabro. *The exponential complexity of satisfiability problems*. PhD thesis, UC San Diego, 2009.

Heng Chang, Y. Rong, Tingyang Xu, W. Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, and Junzhou Huang. The general black-box attack method for graph neural networks. *ArXiv*, abs/1908.01297, 2019.

Dexiong Chen, Laurent Jacob, and Julien Mairal. Convolutional kernel networks for graph-structured data. *arXiv preprint arXiv:2003.05189*, 2020.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems*, pp. 15868–15876, 2019.

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

Hanjun Dai, Hui Li, Tian Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial attack on graph structured data. In *ICML*, 2018.

Constantinos Daskalakis and Ioannis Panageas. The limit points of (optimistic) gradient descent in min-max optimization. In *Advances in Neural Information Processing Systems*, pp. 9236–9246, 2018.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. pp. 3844–3852, 2016.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems*, pp. 5724–5734, 2019.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. pp. 2224–2232, 2015.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, pp. 417–426, 2019.

Matthias Fey. Just jump: Dynamic neighborhood aggregation in graph neural networks. *arXiv preprint arXiv:1904.04849*, 2019.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *arXiv preprint arXiv:1905.04497*, 2019.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

Vikas K Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. *arXiv preprint arXiv:2002.06157*, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1273–1272, 2017.

Joel Goh and Melvyn Sim. Distributionally robust optimization and its tractable approximations. *Operations research*, 58(4-part-1):902–917, 2010.

Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. *Summer course "Cryptography and computer security" at MIT*, 1999:1999, 1996.

Neil Zhenqiang Gong and Bin Liu. Attribute inference attacks in online social networks. *ACM Transactions on Privacy and Security (TOPS)*, 21(1):1–30, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. pp. 1025–1035, 2017.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. 2015.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Jinyuan Jia and Neil Zhenqiang Gong. Attriguard: A practical defense against attribute inference attacks via adversarial machine learning. In *27th {USENIX} Security Symposium*, pp. 513–529, 2018.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 7090–7099, 2019.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.

Greg Landrum. Rdkit: Open-source cheminformatics. URL `http://www.rdkit.org`.

Tianyi Lin, Chi Jin, and Michael I Jordan. On gradient descent ascent for nonconvex-concave minimax problems. *arXiv preprint arXiv:1906.00331*, 2019.

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. In *Advances in neural information processing systems*, pp. 7795–7804, 2018.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*, 2019.

Y. Ma, Suhang Wang, Lingfei Wu, and Jiliang Tang. Attacking graph convolutional networks via rewiring. *ArXiv*, abs/1906.03750, 2019.

Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, pp. 2156–2167, 2019.

Changsung Moon, Paul Jones, and Nagiza F Samatova. Learning entity type embeddings for knowledge graph completion. In *Proceedings of the 2017 ACM on conference on information and knowledge management*, pp. 2215–2218, 2017.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. *arXiv preprint arXiv:1903.02541*, 2019.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 634–646, 2018.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D Lee, and Meisam Razaviyayn. Solving a class of non-convex min-max games using iterative first order methods. In *Advances in Neural Information Processing Systems*, pp. 14905–14916, 2019.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint cs.LG/1905.10947*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

Adam Santoro, Felix Hill, David Barrett, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning*, pp. 4477–4486, 2018.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. In *Advances in Neural Information Processing Systems*, pp. 4083–4092, 2019.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2017.

Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackerman, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5. IEEE, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne, 2008.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*, 2019.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. 2018.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 165–174, 2019.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande. Moleculenet: A benchmark for molecular machine learning. 2017.

Kaidi Xu, H. Chen, S. Liu, Pin-Yu Chen, Tsui-Wei Weng, M. Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *IJCAI*, 2019a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019b.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rJxbJeHFPS`.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 974–983, 2018.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-saint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Can graph neural networks help logic reasoning? *arXiv preprint arXiv:1906.02111*, 2019.

Daniel Zügner and Stephan Günnemann. Certifiable robustness and robust training for graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 246–256, 2019.

APPENDIX

A MISSING PROOFS

In this section we provide the detailed proofs of both theorems in the main text. We first rigoriously show the relationship between the adversarial advantage and the inference error made by a worst-case adversarial:

Claim 4. $1 - \mathrm{Adv}_{\mathcal{D}}(\mathcal{F}_A) = \inf_{f \in \mathcal{F}_A} \left(\Pr_{\mathcal{D}}(f(Z) = 1 \mid A = 0) + \Pr_{\mathcal{D}}(f(Z) = 0 \mid A = 1)\right)$.

Proof. Recall that $\mathcal{F}_A$ is symmetric, hence $\forall f \in \mathcal{F}_A$, $1 - f \in \mathcal{F}_A$ as well:

$$\begin{aligned}
1 - \mathrm{Adv}_{\mathcal{D}}(\mathcal{F}_A) &= 1 - \sup_{f \in \mathcal{F}_A} \left| \Pr_{\mathcal{D}}(f(Z) = 0 \mid A = 0) - \Pr_{\mathcal{D}}(f(Z) = 0 \mid A = 1) \right| \\
&= 1 - \sup_{f \in \mathcal{F}_A} \left(\Pr_{\mathcal{D}}(f(Z) = 0 \mid A = 0) - \Pr_{\mathcal{D}}(f(Z) = 0 \mid A = 1) \right) \\
&= \inf_{f \in \mathcal{F}_A} \left(\Pr_{\mathcal{D}}(f(Z) = 1 \mid A = 0) + \Pr_{\mathcal{D}}(f(Z) = 0 \mid A = 1) \right),
\end{aligned}$$

where the second equality above is because

$$\sup_{f \in \mathcal{F}_A} \left(\Pr_{\mathcal{D}}(f(Z) = 0 \mid A = 0) - \Pr_{\mathcal{D}}(f(Z) = 0 \mid A = 1) \right)$$

is always non-negative due to the symmetric assumption of $\mathcal{F}_A$. ■

Before we prove the lower bound in Theorem 1, we first need to introduce the following lemma, which is known as the data-processing inequality of the TV distance.

Lemma 5 (Data-processing)**.** Let $\mathcal{D}$ and $\mathcal{D}'$ be two distributions over the same sample space and g be a Markov kernel of the same space, then $d_{\mathrm{TV}}(g_\sharp \mathcal{D}, g_\sharp \mathcal{D}') \leq d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}')$, where $g_\sharp \mathcal{D}(g_\sharp \mathcal{D}')$ is the pushforward of $\mathcal{D}(\mathcal{D}')$.

Theorem 1. Let Z be the node representations produced by a GNN g and $\mathcal{F}_A$ be the set of all binary predictors. Define $\delta_{Y|A} := |\Pr_{\mathcal{D}}(Y = 1 \mid A = 0) - \Pr_{\mathcal{D}}(Y = 1 \mid A = 1)|$. Then $\forall h \in \mathcal{H}$,

$$\varepsilon_{Y|A=0}(h \circ g) + \varepsilon_{Y|A=1}(h \circ g) \geq \delta_{Y|A} - \mathrm{Adv}_{\mathcal{D}}(\mathcal{F}_A). \tag{5}$$

Proof. Let $g_\sharp \mathcal{D}$ be the induced (pushforward) distribution of $\mathcal{D}$ under the GNN feature encoder g. To simplify the notation, we also use $\mathcal{D}_0$ and $\mathcal{D}_1$ to denote the conditional distribution of $\mathcal{D}$ given $A = 0$ and $A = 1$, respectively. Since $h : \mathcal{Z} \to \{0, 1\}$ is the task predictor, it follows that $(h \circ g)_\sharp \mathcal{D}_0$ and $(h \circ g)_\sharp \mathcal{D}_1$ induce two distributions over $\{0, 1\}$. Recall that $d_{\mathrm{TV}}(\cdot, \cdot)$ is a distance metric over the space of probability distributions, by a chain of triangle inequalities, we have:

$$\begin{aligned}
d_{\mathrm{TV}}(\mathcal{D}(Y \mid A = 0), \mathcal{D}(Y \mid A = 1)) \leq\ & d_{\mathrm{TV}}(\mathcal{D}(Y \mid A = 0), (h \circ g)_\sharp \mathcal{D}_0) \\
& + d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_0, (h \circ g)_\sharp \mathcal{D}_1) + d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_1, \mathcal{D}(Y \mid A = 1)).
\end{aligned}$$

Now by Lemma 5, we have

$$d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_0, (h \circ g)_\sharp \mathcal{D}_1) \leq d_{\mathrm{TV}}(g_\sharp \mathcal{D}_0, g_\sharp \mathcal{D}_1).$$

On the other hand, since $\mathcal{F}_A$ contains all the binary predictors:

$$\begin{aligned}
d_{\mathrm{TV}}(g_\sharp \mathcal{D}_0, g_\sharp \mathcal{D}_1) &= \sup_{E \text{ is measurable}} \left| \Pr_{g_\sharp \mathcal{D}_0}(E) - \Pr_{g_\sharp \mathcal{D}_1}(E) \right| \\
&= \sup_{f_E \in \mathcal{F}_A} \left| \Pr_{g_\sharp \mathcal{D}_0}(f_E(Z) = 1) - \Pr_{g_\sharp \mathcal{D}_1}(f_E(Z) = 1) \right| \\
&= \sup_{f_E \in \mathcal{F}_A} \left| \Pr_{g_\sharp \mathcal{D}}(f_E(Z) = 1 \mid A = 0) - \Pr_{g_\sharp \mathcal{D}}(f_E(Z) = 1 \mid A = 1) \right| \\
&= \mathrm{Adv}_{\mathcal{D}}(\mathcal{F}_A),
\end{aligned}$$

where in the second equation above $f_E(\cdot)$ is the characteristic function of the event E. Now combine the above two inequalities together, we have:

$$d_{\text{TV}}(\mathcal{D}(Y \mid A=0), \mathcal{D}(Y \mid A=1)) - \text{Adv}_{\mathcal{D}}(\mathcal{F}_A) \leq d_{\text{TV}}(\mathcal{D}(Y \mid A=0), (h \circ g)_\sharp \mathcal{D}_0) + d_{\text{TV}}(\mathcal{D}(Y \mid A=1), (h \circ g)_\sharp \mathcal{D}_1).$$

Next we bound $d_{\text{TV}}(\mathcal{D}(Y \mid A=a), (h \circ g)_\sharp \mathcal{D}_a), \forall a \in \{0,1\}$:

$$\begin{aligned}
&d_{\text{TV}}(\mathcal{D}(Y \mid A=a), (h \circ g)_\sharp \mathcal{D}_a) \\
&= \frac{1}{2} \| \mathcal{D}(Y \mid A=a) - (h \circ g)_\sharp \mathcal{D}_a \|_1 \\
&= |\Pr_{\mathcal{D}}(Y=1 \mid A=a) - \Pr_{\mathcal{D}}((h \circ g)(X)=1 \mid A=a)| \quad \text{(Both } Y \text{ and } h(g(X)) \text{ are binary)} \\
&= |\mathbb{E}_{\mathcal{D}}[Y \mid A=a] - \mathbb{E}_{\mathcal{D}}[(h \circ g)(X) \mid A=a]| \\
&\leq \mathbb{E}_{\mathcal{D}}[|Y - (h \circ g)(X)| \mid A=a] \quad \text{(Triangle inequality)} \\
&= \Pr_{\mathcal{D}}(Y \neq (h \circ g)(X) \mid A=a) \\
&\leq \varepsilon_{Y|A=a}(h \circ g),
\end{aligned}$$

where the last inequality is due to the fact that the cross-entropy loss is an upper bound of the 0-1 binary loss. To complete the proof, realize that for binary prediction problems, the total variation term $d_{\text{TV}}(\mathcal{D}(Y \mid A=0), \mathcal{D}(Y \mid A=1))$ admits the following simplification:

$$\begin{aligned}
&d_{\text{TV}}(\mathcal{D}(Y \mid A=0), \mathcal{D}(Y \mid A=1)) \\
&= \frac{1}{2} \left(|\Pr(Y=1 \mid A=0) - \Pr(Y=1 \mid A=1)| + |\Pr(Y=0 \mid A=0) - \Pr(Y=0 \mid A=1)| \right) \\
&= |\Pr(Y=1 \mid A=0) - \Pr(Y=1 \mid A=1)| \\
&= \delta_{Y|A}.
\end{aligned}$$

This gives us:

$$\delta_{Y|A} - \text{Adv}_{\mathcal{D}}(\mathcal{F}_A) \leq \varepsilon_{Y|A=0}(h \circ g) + \varepsilon_{Y|A=1}(h \circ g). \tag{7}$$

■

Corollary 2 then follows directly from Theorem 1:

Corollary 2. Assume the conditions in Theorem 1 hold, then

$$\varepsilon_Y(h \circ g) \geq \min\{\Pr_{\mathcal{D}}(A=0), \Pr_{\mathcal{D}}(A=1)\} \cdot \left(\delta_{Y|A} - \text{Adv}_{\mathcal{D}}(\mathcal{F}_A)\right).$$

Proof. Realize that

$$\begin{aligned}
\varepsilon_Y(h \circ g) &= \Pr_{\mathcal{D}}(A=0) \cdot \varepsilon_{Y|A=0}(h \circ g) + \Pr_{\mathcal{D}}(A=1) \cdot \varepsilon_{Y|A=1}(h \circ g) \\
&\geq \min\{\Pr_{\mathcal{D}}(A=0), \Pr_{\mathcal{D}}(A=1)\} \cdot \left(\varepsilon_{Y|A=0}(h \circ g) + \varepsilon_{Y|A=1}(h \circ g)\right).
\end{aligned}$$

Applying the lower bound in Theorem 1 then completes the proof. ■

The following lemma about the inverse binary entropy will be useful in the proof of Theorem 3:

Lemma 6 (Calabro (2009))**.** Let $H_2^{-1}(s)$ be the inverse binary entropy function for $s \in [0,1]$, then $H_2^{-1}(s) \geq s/2 \lg(6/s)$.

With the above lemma, we are ready to prove Theorem 3.

Theorem 3. Let Z^* be the optimal GNN node embedding of (6) and define $H^* := H(A \mid Z^*)$. Then for any adversary $f : \mathcal{Z} \to \{0,1\}$, $\Pr(f(Z) \neq A) \geq H^*/2 \lg(6/H^*)$.

Proof. To ease the presentation we define $Z = Z^*$. To prove this theorem, let E be the binary random variable that takes value 1 iff $A \neq f(Z)$, i.e., $E = \mathbb{I}(A \neq f(Z))$. Now consider the joint entropy of $A, f(Z)$ and E. On one hand, we have:

$$H(A, f(Z), E) = H(A, f(Z)) + H(E \mid A, f(Z)) = H(A, f(Z)) + 0 = H(A \mid f(Z)) + H(f(Z)).$$

Note that the second equation holds because E is a deterministic function of A and $f(Z)$, that is, once A and $f(Z)$ are known, E is also known, hence $H(E \mid A, f(Z)) = 0$. On the other hand, we can also decompose $H(A, f(Z), E)$ as follows:

$$H(A, f(Z), E) = H(E) + H(A \mid E) + H(f(Z) \mid A, E).$$

Combining the above two equalities yields

$$H(E, A \mid f(Z)) = H(A \mid f(Z)).$$

On the other hand, we can also decompose $H(E, A \mid f(Z))$ as

$$H(E, A \mid f(Z)) = H(E \mid f(Z)) + H(A \mid E, f(Z)).$$

Furthermore, since conditioning cannot increase entropy, we have $H(E \mid f(Z)) \leq H(E)$, which further implies

$$H(A \mid f(Z)) \leq H(E) + H(A \mid E, f(Z)).$$

Now consider $H(A \mid E, f(Z))$. Since $A \in \{0, 1\}$, by definition of the conditional entropy, we have:

$$H(A \mid E, f(Z)) = \Pr(E = 1)H(A \mid E = 1, f(Z)) + \Pr(E = 0)H(A \mid E = 0, f(Z)) = 0 + 0 = 0.$$

To lower bound $H(A \mid f(Z))$, realize that

$$I(A; f(Z)) + H(A \mid f(Z)) = H(A) = I(A; Z) + H(A \mid Z).$$

Since $f(Z)$ is a randomized function of Z such that $A \perp f(Z) \mid Z$, due to the celebrated data-processing inequality, we have $I(A; f(Z)) \leq I(A; Z)$, which implies

$$H(A \mid f(Z)) \geq H(A \mid Z).$$

Combine everything above, we have the following chain of inequalities hold:

$$H(A \mid Z) \leq H(A \mid f(Z)) \leq H(E) + H(A \mid E, f(Z)) = H(E),$$

which implies

$$\Pr(A \neq f(Z)) = \Pr(E = 1) \geq H_2^{-1}(H(A \mid Z)),$$

where $H_2^{-1}(\cdot)$ denotes the inverse function of the binary entropy $H(t) := -t \log t - (1-t) \log(1-t)$ when $t \in [0, 1]$. To conclude the proof, we apply Lemma 6 to further lower bound the inverse binary entropy function by

$$\Pr(A \neq f(Z)) \geq H_2^{-1}(H(A \mid Z)) \geq H(A \mid Z)/2 \lg(6/H(A \mid Z)),$$

completing the proof. ■

B Experimental Setup Details

Our code can be found at the following link:

`https://github.com/iclr2021-gal/Graph-Adversarial-Networks`.

Optimization For the objective function, we selected block gradient descent-ascent to optimize our models. In particular, we took advantage of the optim module in PyTorch (Paszke et al., 2019) by designing a custom gradient-reversal layer, first introduced by (Ganin et al., 2016), to be placed between the attacker and the GNN layer we seek to defend. The implementation of the graident-reversal layer can be found in the Appendix. During training, we would designate two Optimizer instances, one having access to only task-related parameters, and the other having access to attack-related parameters and parameters associated with GNN defense. We could then call the .step() method on the optimizers in an alternating fashion to train the entire network, where the gradient-reversal layer would carry out both gradient descent (of the attacker) and ascent (of protected layers) as expected. Tradeoff control via λ is achieved through multiplying the initial learning rate of the adversarial learner by the desired factor. For graphs that are harder to optimize, we introduce pre-training as the first step in the pipeline, where we train the encoder and the task decoder for a few epochs before introducing the adversarial learner.

Movielens 1M The main dataset of interest for this work is Movielens-1M [2], a benchmarking dataset in evaluating recommender systems, developed by (Harper & Konstan, 2015). In this dataset, nodes are either users or movies, and the type of edge represents the rating the user assigns to a movie. Adapting the formulation of (Bose & Hamilton, 2019), we designate the main task as edge prediction and designate the adversarial task as extracting user-related information from the GNN embedding using multi-layer perceptrons with LeakyReLU functions (Maas, 2013) as nonlinearities. Training/test splits are creating using a random 90/10 shuffle. The network encoder consists of a trainable embedding layer followed by neighborhood aggregation layers. Node-level embeddings have a dimension of 20, and the decoder is a naive bilinear decoder, introduced in (Berg et al., 2017). Both the adversarial trainers and the main task predictors are trained with separate Adam optimizers with learning rate set to 0.01. Worst-case attackers are trained for 30 epochs with a batch-size 256 nodes before the original model is trained for 25 epochs with a batch-size of $8{,}192$ edges.

Planetoid Planetoid [3] is the common name for three datasets (Cora, CiteSeer, Pubmed) used in benchmarks of graph neural networks in the literature, introduced by (Yang et al., 2016). Nodes in these datasets represent academic publications, and edges represent citation links. Since the Cora dataset is considered to be small to have any practical implications in the performance of our algorithm, we report only the results of CiteSeer and Pubmed. Similar to Movielens, the main task is edge prediction, and the attacker will seek to predict node attributes from GNN-processed embeddings. The network architecture is message-passing layers connected with ReLU nonlinearities, and both the decoder and attacker are also single-layer message-passing modules. Regarding training/valid/test splits, we adopt the default split used in the original paper, maintained by (Fey & Lenssen, 2019). The network encoder consists of a trainable embedding layer followed by neighborhood aggregation layers. Node-level embeddings have a dimension of 64, and both the adversarial trainers and the main task predictors are trained with separate Adam optimizers with learning rate set to 0.01. Worst-case attackers are trained for 80 epochs with before the original model is trained for 150 epochs, and the entire graph is fed into the network at once during each epoch.

QM9 QM9 [4] is a dataset used to benchmark machine learning algorithms in quantum chemistry (Wu et al., 2017), consisting of around $130{,}000$ molecules (represented in their spatial information of all component atoms) and 19 regression targets. The main task would be to predict the dipole moment μ for a molecule graph, while the attacker will seek to extract its isotropic polarizability α from the embeddings. The encoder is a recurrent architecture consisting of a NNConv (Gilmer et al., 2017) unit, a GRU (Cho et al., 2014) unit and a Set2Set (Vinyals et al., 2015) unit, with both the decoder and the attacker (as regressors) 2-layer multi-layer perceptrons with ReLU nonlinearities. The training/valid/test is selected in the following manner: the order of samples is randomly shuffled at first, then the first 10,000 and 10,000 - 20,000 samples are selected for testing and validation respectively, and the remaining samples are used for training. Preprocessing is done with scripts provided by (Fey & Lenssen, 2019) [5] , using functions from (Landrum). Node-level embeddings have a dimension of 64, and both the adversarial trainers and the main task predictors are trained with separate Adam optimizers with learning rate set to 0.001. Worst-case attackers are trained for 30 epochs with before the original model is trained for 40 epochs with a batch-size of 128 molecular graphs.

FB15k-237/WN18RR These two datasets are benchmarks for knowledge base completion: while FB15k-237 [6] is semi-synthetic with nodes as common entities, WN18RR [7] is made by words found in the thesaurus. Our formulation is as follows: while the main task from both datasets is edge prediction, the attackers' goals are different:

[2] `https://grouplens.org/datasets/movielens/1m/`

[3] Raw data available at `https://github.com/kimiyoung/planetoid/tree/master/data`. For this work, we used the wrapper provided by `https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/datasets/planetoid.html`.

[4] Raw data available at `https://s3-us-west-1.amazonaws.com/deepchem.io/datasets/molnet_publish/qm9.zip` and `https://ndownloader.figshare.com/files/3195404`

[5] Available at `https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/datasets/qm9.html`

[6] `https://www.microsoft.com/en-us/download/details.aspx?id=52312`

[7] `https://github.com/TimDettmers/ConvE`

- For FB15k-237, we took node-level attributes from (Moon et al., 2017) [8], and task the attacker with predicting the 50-most frequent labels. Since a node in FB15k-237 may have multiple labels associated with it, adversarial defense on this may be seen as protecting sets of node-level attributes, in contrast to single-attribute defense in other experimental settings.
- For WN18RR, we consider two attributes for a node (as a word): its word sense (sense greater than 20 are considered as the same heterogeneous class), and part-of-speech tag. The labels are obtained from (Bordes et al., 2013) [9].

As for the architecture, we used a modified version of the CompGCN paper (Vashishth et al., 2019), where the attacker has access to the output of the CompGCN layer (of dimension 200), and the original task utilizes the ConvE model for the decoder. The training/valid/test split also aligns with the one used in the CompGCN paper. On both datasets, the adversarial trainers and main task predictors are trained with separate Adam optimizers with learning rate set to 0.001. Worst-case attackers are trained for 30 epochs with a batch-size of 128 nodes before the original model is trained for 120 epochs after 35 epochs of pre-training, with a batch-size of 128 nodes.

Computing Infrastructure All models are trained with NVIDIA GeForce® RTX 2080 Ti graphics processing units (GPU) with 11.0 GB GDDR6 memory on each card, and non-training-related operations are performed using Intel® Xeon® Processor E5-2670 (20M Cache, 2.60 GHz).

Estimated Average Runtime Below are the averge training time per epoch for each models used in the main text, when the training is performed on the computing infrastructure mentioned above:

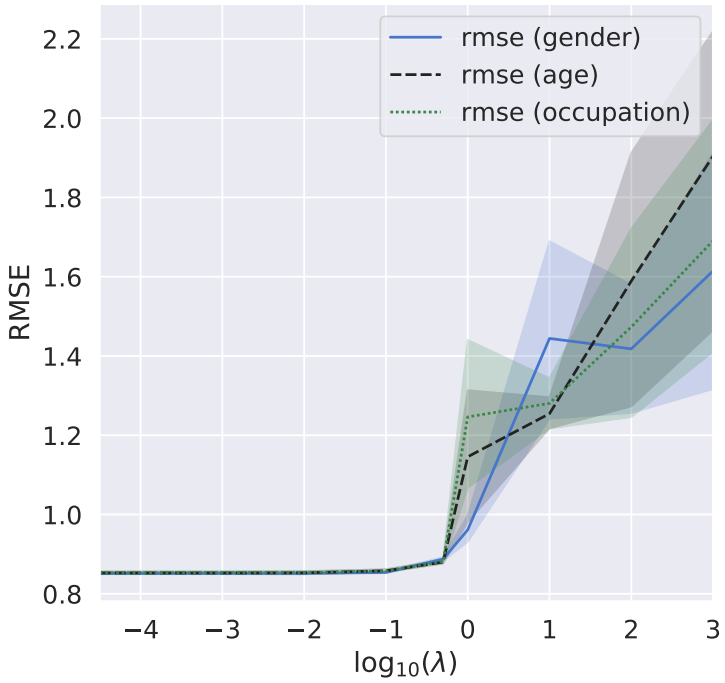
DATASET	Encoder	t
CITESEER	ChebConv	0.0232s
	GCN	0.0149s
	GAT	0.0282s
PUBMED	ChebConv	0.0920s
	GCN	0.0824s
	GAT	0.129s
QM9	NNConv	199.25s
MOVIELENS-1M	ChebConv	16.89s
	GCN	12.05s
	GAT	45.86s
FB15K-237	CompGCN	463.39s
WN18RR	CompGCN	181.55s

[8] https://github.com/cmoon2/knowledge_graph

[9] https://everest.hds.utc.fr/doku.php?id=en:smemlj12

C DEGREDATION OF RMSE ON MOVIELENS-1M DATASET REGARDING NEIGHBORHOOD ATTACK

This is a supplementary figure for the neighborhood attack experiments introduced in the main section. Band represents 95% confidence interval over five runs.



D N-HOP ALGORITHM FOR NEIGHBORHOOD DEFENSE

Intuitively, this algorithm greedily constructs a path of length n by uniformly picking a neighbor from the current end of the path and checking if the node has existed previously in the path, avoiding formation of cycles. Worst-case running time of this algorithm is $O(n^2)$, because in each step of the main loop, the algorithm performs $O(n)$ checks in the worst case scenario.

Algorithm 1 Monte-Carlo Probabilistic N-Hop

Input: $G = (V,E)$: undirected graph (via adjacency list); $v \in V$: starting node; $n \geq 1$: hop
Output: On success: $v' \in V$ such that $d(v, v') = n$ or NO if such vertex doesn't exist; On failure: $v' \in V$ such that $1 \leq d(v, v') \leq n$ or NO if such vertex doesn't exist

procedure NHOP(G, v, n) ▷ Time complexity is $O(n^2)$
  $V \leftarrow \emptyset$ ▷ Initial path is empty
  $t \leftarrow 0$
  $v' = v$
  while $t < n$ **do**
    $S \leftarrow [\mathcal{N}(v')]$ ▷ $O(1)$ time by adjacency list
    $i \leftarrow$ RandInt$(0, |S|)$ ▷ $O(1)$ uniform random sample (without replacement)
    $e \leftarrow S$.pop(i)
    while $e \in V$ and $S \neq []$ **do** ▷ Loop runs at most $O(n)$ times
      $i \leftarrow$ RandInt$(0, |S|)$
      $e \leftarrow S$.pop(i)
    if $e \notin V$ **then**
      $V \leftarrow V \cap \{e\}$
      $v' = e$
    else ▷ Current path not satisfiable, reject
      reject with NO
    $t \leftarrow t + 1$
  accept with v'

E OTHER RELATED WORK

Graph Neural Networks GNNs have been successfully applied in a range of applications, e.g., recommender systems (Ying et al., 2018; Wang et al., 2019; Fan et al., 2019), drug discovery (Klicpera et al., 2020; Stokes et al., 2020; Sanchez-Lengeling & Aspuru-Guzik, 2018; Liu et al., 2018), relational reasoning (Xu et al., 2020; Santoro et al., 2018; Saxton et al., 2019), and knowledge graphs (Nickel et al., 2015; Vashishth et al., 2019). Effective engineering strategies such as big dataset (Hu et al., 2020) and sampling (Zeng et al., 2019) have further promoted scaling of GNNs to large graphs in real business.

Theoretically, recent works study the expressive power for GNNs, and show that GNNs can be as powerful as practical graph isomorphism tests (Xu et al., 2019b; Chen et al., 2019). An exciting line of works aim to design more powerful GNNs, e.g., by further incorporating graph properties in specific applications (Loukas, 2019; Klicpera et al., 2020; Sato et al., 2019; Murphy et al., 2019; Maron et al., 2018; Morris et al., 2019; Garg et al., 2020; Keriven & Peyré, 2019; Zhang et al., 2019; Barceló et al., 2019; Oono & Suzuki, 2019; Maron et al., 2019). Generalization of GNNs relies on the *implicit biases* of gradient descent, due to GNN's rich expressive power. Du et al. (2019) and Xu et al. (2020) study generalization properties of GNNs. They analyze the gradient descent training dynamics of GNNs to characterize what functions GNNs can sample-efficiently learn. Gama et al. (2019) investigate the good performance of GNNs compared with linear graph filters, showing that GNNs can be both discriminative and stable. Recent works also study GNNs from the kernel perspective (Chen et al., 2020; Du et al., 2019).

Our work builds upon both theoretical and application works of GNNs. To the best of our knowledge, this is the first work that characterizes the inherent tradeoffs between GNN predictive accuracy and adversarial defense. Furthermore, we also give a lower bound on the inference error that any

adversary has to make, and we empirically demonstrate the effectiveness of the proposed adversarial defense approach on both node attacks and neighborhood attacks.