Towards Robust Graph Neural Networks via Adversarial Contrastive Learning

Shen Wang[1,*], Zhengzhang Chen[2,†], Jingchao Ni[1], Haifeng Chen[2], and Philip S. Yu[3]
[1]*Amazon*, USA
{shenwa, jingchni}@amazon.com
[2]*NEC Laboratories America*, USA
{zchen, haifeng}@nec-labs.com
[3]*Department of Computer Science, University of Illinois at Chicago*, USA
psyu@uic.edu

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10021051

***Abstract*—Graph Neural Network (GNN), as a powerful representation learning model on graph data, attracts much attention across various disciplines. However, recent studies show that GNN is vulnerable to adversarial attacks. How to make GNN more robust? What are the key vulnerabilities in GNN? How to address the vulnerabilities and defend GNN against the adversarial attacks? Adversarial training has shown to be effective in improving the robustness of traditional Deep Neural Networks (DNNs). However, existing adversarial training works mainly focus on the image data, which consists of continuous features, while the features and structures of graph data are often discrete. Moreover, rather than assuming each sample is independent and identically distributed as in DNN, GNN leverages the contextual information across the graph (*e.g.*, neighborhoods of a node). Thus, existing adversarial training techniques cannot be directly applied to defend GNN.**

In this paper, we propose ContrastNet, an effective adversarial defense framework for GNN. In particular, we propose an adversarial contrastive learning method to train the GNN over the adversarial space. To further improve the robustness of GNN, we investigate the latent vulnerabilities in every component of a GNN encoder and propose corresponding refining strategies. Extensive experiments on three public datasets demonstrate the effectiveness of ContrastNet in improving the robustness of popular GNN variants, such as *Graph Convolutional Network* and *GraphSage*, under various types of adversarial attacks.

***Index Terms*—graph neural network, adversarial defense, adversarial training, contrastive learning**

I. Introduction

Graph Neural Network (GNN) has received wide attention [5], [9], [11], [13], [18], [22], [39], [40], [42], [44], [49] in the past few years. It extends the traditional neural networks to graph structured data. The goal of GNN is to learn the representation of the graph, in the node level or graph level, via a neural network consisting of a neural graph encoding function. Because of its remarkable graph representation learning ability, GNN has been explored in various real-world applications, such as physics system [1], [15], healthcare [6], [27], [34], chemistry [10], [52], recommender system [2], [29], [43], [48] and security system [4], [24], [41], [47].

∗ Work was done during an internship at NEC Laboratories America.
† Corresponding author.

However, recent studies [8], [53], [54] have found that GNN can easily be compromised by adversarial attacks. These adversarial attacks are often stealthy and only require small perturbations (*e.g.*, by adding or dropping edges) of the graph structure and/or node features to induce GNN to make incorrect predictions for some specific nodes with high confidence. This makes it highly risky for those aforementioned domains that GNN applies, especially security and financial applications, because the vulnerabilities of GNN could potentially open the backdoor for the attackers. For example, fraudsters could try to disguise themselves by connecting other normal users.

Despite recent advances in adversarial attacks [8], [53], [54], the question of how to build a robust GNN against the adversarial attacks has not been satisfyingly addressed yet. Nonetheless, some methods have been proposed to improve the robustness of traditional Deep Neural Networks (DNNs) [7], [12], [17], [23], [26], [31]–[33], [35], [36], [45]. They fall into three main categories: (1) adversarial training methods [12], [26], [31] , (2) adversarial perturbation removing methods [35], and (3) smoothness enforced methods [32]. However, these defense work only focus on either image data [7], [12], [23], [26], [31], [33], [35], [36], in which the data are independently and identically distributed in the continuous feature space. In contrast, the structure and the node feature of graph data are often discrete and each node is related to its neighborhood. Thus, directly adopting the existing defense methods for defending GNN will not work. Different from defending the traditional DNNs, defending the GNN has two major challenges: (1) the discrete unknown adversarial space, and (2) the latent vulnerabilities from every component of GNN.

Adversarial training, which augments the training data with adversarial examples during the training stage [12], [26], [31] has shown to be effective in defending traditional deep learning methods against adversarial attacks. But as aforementioned, the graphs are discrete and the combinatorial nature of the graph structures makes it much more difficult to create adversarial samples than the image. Thus, how to generate good adversarial graph samples to augment the training data for GNN is a non-trivial problem. Simply generating adversarial

samples from a noise distribution will not be able to tailor toward the graph data. And the generated ineffectual adversarial samples could even weaken the robustness of the model against various adversarial attacks.

In addition to the discrete unknown adversarial space, the vulnerabilities of GNN can come from components of its unique architecture: the aggregator and the updater [50]. As the main strength of GNN, the aggregator computes the new node representation by leveraging context information to cover the graph structure besides the node attributes. However, the aggregator can be vulnerable because the node representation depends not only on its feature but also on its neighborhood [18]. [53] has shown that the attack can be conducted without touching the target node, since the attack on the neighbors may propagate to the target node that is not attacked. Another vulnerability can be related to the updater, which is employed to connect the current layer of the network to its previous layers, so that more structural information can be covered. Recently, GNNs, such as GraphSage, leverage the skip-connection [13] to aggregate the representation from the predecessor layer and drop the rest of intermediate representations during the information propagation within k-hop neighbors. However, stacking multiple such layers could also propagate the noisy or adversarial information from an exponentially increasing number of expanded neighboring member [18]. Only considering the last layer of representation may be vulnerable to the potential noise. Another vulnerability is related to the input dimension. As shown in [37], there is a one-to-one relationship between the adversarial vulnerability and the input dimension, such that the adversarial vulnerability increases with the input dimension. By noticing the vulnerabilities caused by different components, it needs to improve their robustness to adversarial attacks.

To tackle the aforementioned two challenges, in this paper, we propose **ContrastNet**, an effective contrastive learning based framework for defending Graph Neural Network (GNN) against adversarial attacks. In particular, to train a robust GNN, we propose **ACL**, an adversarial contrastive learning method instead of traditional supervised training. **ACL** models the graph adversarial learning as a min-max optimization to overcome the discrete unknown adversarial space problem, and leverages the high-level graph representation as auxiliary information to regularize the node representation learning. Moreover, we investigate the vulnerabilities in the aggregator and the updater of a GNN encoder, and propose graph encoder refining strategies including aggregator refining, updater refining, and bottleneck refining to address those model vulnerabilities. To evaluate the performance of **ContrastNet**, we perform an extensive set of experiments on three public graph datasets. The results demonstrate the effectiveness of our proposed framework on defending the popular GNN variants, such as Graph Convolutional Network (GCN) and GraphSage, against various types of adversarial attacks.

In summary, this paper makes the following contributions:

- We propose ContrastNet, an effective contrastive learning based framework for defending Graph Neural Network (GNN) against adversarial attacks.
- We design ACL, a new training strategy to train GNNs in an adversarial contrastive way to address the discrete unknown adversarial space problem. We adopt a min-max optimization to explicitly explore the adversarial space and exploit the high-level graph representation as auxiliary information to regularize the node representation learning.
- We investigate the latent vulnerabilities in each component of a GNN encoder and propose three corresponding refining strategies.
- The proposed framework is extensively evaluated on three real-world graph datasets. The experimental results show that our framework can effectively defend popular GNNs against various types of adversarial attacks.

The rest of the paper is organized as follows. We introduce some preliminaries and problem definition in Section 2. Section 3 discusses our proposed adversarial defense framework in detail. Section 4 provides the experimental results. Section 5 discusses the related work. Finally, Section 6 concludes the paper.

II. Preliminaries and Problem Formulation

In this section, we present the preliminaries and problem definition of our work.

A. Notations

We use bold lowercase for vectors (*e.g.*, $\mathbf{a}$), bold capital for matrices (*e.g.*, $\mathbf{A}$), and calligraphic letters for sets (*e.g.*, $\mathcal{A}$). A graph is represented by a triplet $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a matrix. Its v-th row, $\mathbf{x}_v$, representing the d-dimensional feature vector of node v. Also, $|\mathcal{V}| = n$ and $|\mathcal{E}| = m$ are the number of nodes and edges in G, respectively. Following the existing work on adversarial attacks [53], in this paper, we consider graphs that are undirected and discretely attributed and the Graph Neural Network (GNN) is convolutional aggregator based[1].

B. Graph Neural Networks

GNN is a neural network parameterized function f_θ, which leverages both graph structure and node feature to learn the representation of node/graph for different tasks. Here θ is the set of all parameters. Typically, f_θ has a graph encoding function $ENC(\cdot)$ as follows to generate the node embedding:

$$\mathbf{h}_v^{(final)} = ENC(\mathbf{h}_v^{(0)}, \{\mathbf{h}_u^{(0)}\}_{u \in \mathcal{N}_v}) \tag{1}$$

where $\mathbf{h}_v^{(final)}$ is the embedding vector of node v after K iterations/layers, $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ is the original feature of node v, and $\mathcal{N}_v$ is the set of neighboring nodes of v in graph G, *i.e.*, $\mathcal{N}_v = \{u | (u, v) \in \mathcal{E}, (v, u) \in \mathcal{E}\}$.

The encoding function is characterized by an *aggregator* function $Aggregator()$ and an *updater* function

[1]Attention based GNNs like Graph Attention Network or Gate updater based GNNs like Graph LSTM won't be covered by this work.

$Updater()$ [50]. The aggregator aims to gather the neighborhood information for each node, which encourages message propagation within the graph. Formally, as shown in the updater part of Fig.1, the k-th layer/iteration of a GNN is described as follows:

$$\mathbf{a}_v^{(k)} = Aggregator^{(k)}(\{\mathbf{h}_u^{(k-1)}\}_{u\in\mathcal{N}_v}) \tag{2}$$

$$\mathbf{h}_v^{(k)} = Updater^{(k)}(\mathbf{a}_v^{(k)}) \tag{3}$$

where $\mathbf{a}_v^{(k)}$ is the aggregated representation of the node representation $\mathbf{h}_v^{(k-1)}$ (at the $(k-1)$-th layer) of node v.

Traditionally, the parameters of GNN f_θ are trained in a supervised manner. For example, considering node classification, each node v has a label $y_v \in \mathcal{Y} = \{1, 2, ..., Y\}$. Then, the goal is to learn a new representation of node v, which can be used to predict the node label. In this case, the GNN is define as $f_\theta^t(G) = f_\theta^t(X, A) \to y \in \mathbb{R}$, where t is the target node indicator.

C. Adversarial Attack on GNNs

Given a graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ and a GNN model f_θ, f_θ is expected to perform good predictions on G (*e.g.*, transductive classification) and on other graphs with a similar distribution of G (*e.g.*, inductive classification). The goal of an *adversarial attack* is to maliciously transform G through a perturbation function $P()$ to a perturbed graph $G' = P(G) = P(\mathcal{V}', \mathcal{E}', \mathbf{X}')$, such that the performance of the original model f_θ on the perturbed graph G' drops dramatically. More precisely, in this work, we focus on *direct targeted poisoning attacks on both node feature and graph structure*. Formally, the adversarial attack is defined as follows:

$$\begin{aligned} &\arg\max_{G'=P(G))} max_{c\neq c_{old}} \ln Z^*_{t,c} - \ln Z^*_{t,c_{old}} \\ &\text{s.t.} \quad Z^*_t = f_{\theta^*}(G') \quad \text{with} \quad \theta^* = \arg\min_\theta \mathcal{L}(\theta; G') \end{aligned} \tag{4}$$

The target is to find a perturbed graph G' that classifiers classify the target node as the new class c and has maximal distance to ground truth class c_{old} in terms of logits.

D. Problem Statement

In this work, we study the problem of *adversarial defense* for GNN, that is to defend a GNN against the direct targeted poisoning attacks. More formally, given a graph G and an original GNN model f_θ, our goal is to construct a robust model $f_\theta^{(robust)}$ during the training phase, such that in the testing phase, the $f_\theta^{(robust)}$ preserves its good performance on the unseen perturbed graph G'.

III. Proposed Method

As discussed in the introduction, addressing the vulnerabilities and defending GNN against the adversarial attacks is challenging due to discrete unknown adversarial space and unique GNN architecture. To overcome these challenges, we propose an adversarial defense framework **ContrastNet** with two critical steps: Adversarial Contrastive Learning (**ACL**) and Graph Encoder Refining (**GER**) as illustrated in Fig. 1. In the **ACL**, our goal is to train GNN to be more distinguishable between real benign samples and adversarial samples and enhance its defensibility against adversarial attacks. And in the **GER**, we aim to find the vulnerabilities in every component of a GNN encoder, and propose corresponding strategies to address these vulnerabilities. These two steps collectively address the challenges of defending GNN against adversarial attacks.

A. ACL: Adversarial Contrastive Learning

Adversarial training has shown to be effective in defending traditional deep neural networks against adversarial attacks. But as aforementioned, how to generate good adversarial graph samples is still a very challenging problem. In this section, we propose a novel adversarial training method for GNN.

Contrastive learning has been widely found useful for representation learning of graph data [14]. Its objective can be written as:

$$\mathcal{L} = \mathbb{E}_{(x^+,y^+,y^-)}[\ell(x^+, y^+, y^-)] \tag{5}$$

where (x^+, y^+) indicates a positive data pair and (x^+, y^-) indicates a negative data pair. $\ell(x^+, y^+, y^-)$ scores a positive tuple (x^+, y^+) against a negative one (x^+, y^-), and $\mathbb{E}_{(x^+,y^+,y^-)}$ represents the expectation w.r.t. the joint distribution over the positive and negative samples. Since negative sampling is independent of the positive label, Eq. 5 can be rewritten as:

$$\mathcal{L} = \mathbb{E}_{p(x^+)}\mathbb{E}_{p(y^+|x^+)p(y^-|x^+)}[\ell(x^+, y^+, y^-)] \tag{6}$$

$$= \mathbb{E}_{p(x^+)}[\mathbb{E}_{p(y^+|x^+)}s(x^+, y^+) - \mathbb{E}_{p(y^-|x^+)}s(x^+, y^-)] \tag{7}$$

where $s(x, y)$ measures the correlation between x and y. $p(y^+|x^+)$ denotes the probability of a positive tuple while $p(y^-|x^+)$ denotes the probability of a negative tuple.

Traditional graph embedding methods adopt a noise contrastive estimation approach and approximate $p(y^-|x^+)$ with a noise distribution probability $p_n(y)$, such that:

$$\mathcal{L} = \mathbb{E}_{p(x^+)}[\mathbb{E}_{p(y^+|x^+)}s(x^+, y^+) - \mathbb{E}_{p_n}s(x^+, y^-)] \tag{8}$$

However, using p_n sacrifices the performance of learning as the negative samples are produced based on a noise distribution. What is worse, using p_n also harms the robustness of the learning as the negative samples are produced without considering the adversarial samples.

To address this problem, we model the negative sampler by a generator under a conditional distribution $g(y|x)$. Formally, it is defined as follows:

$$\mathcal{L} = \mathbb{E}_{p(x^+)}[\mathbb{E}_{p(y^+|x^+)}s(x^+, y^+) - \mathbb{E}_{g(y|x)}s(x^+, y^-)]. \tag{9}$$

We optimize the above objective function in a minimax adversarial learning manner as follows:

$$\begin{aligned} &\min_S \max_G \{\mathbb{E}_{p(y^+|x^+)}[\log S(x, y)] \\ &+\mathbb{E}_{g(y|x)}[\log(1 - S(G(x^+, y^-)))]\}. \end{aligned} \tag{10}$$

This formulation includes a discriminator S and an adversarial sample generator G. Different from GAN, which generates samples close to the given data, G generates adversarial

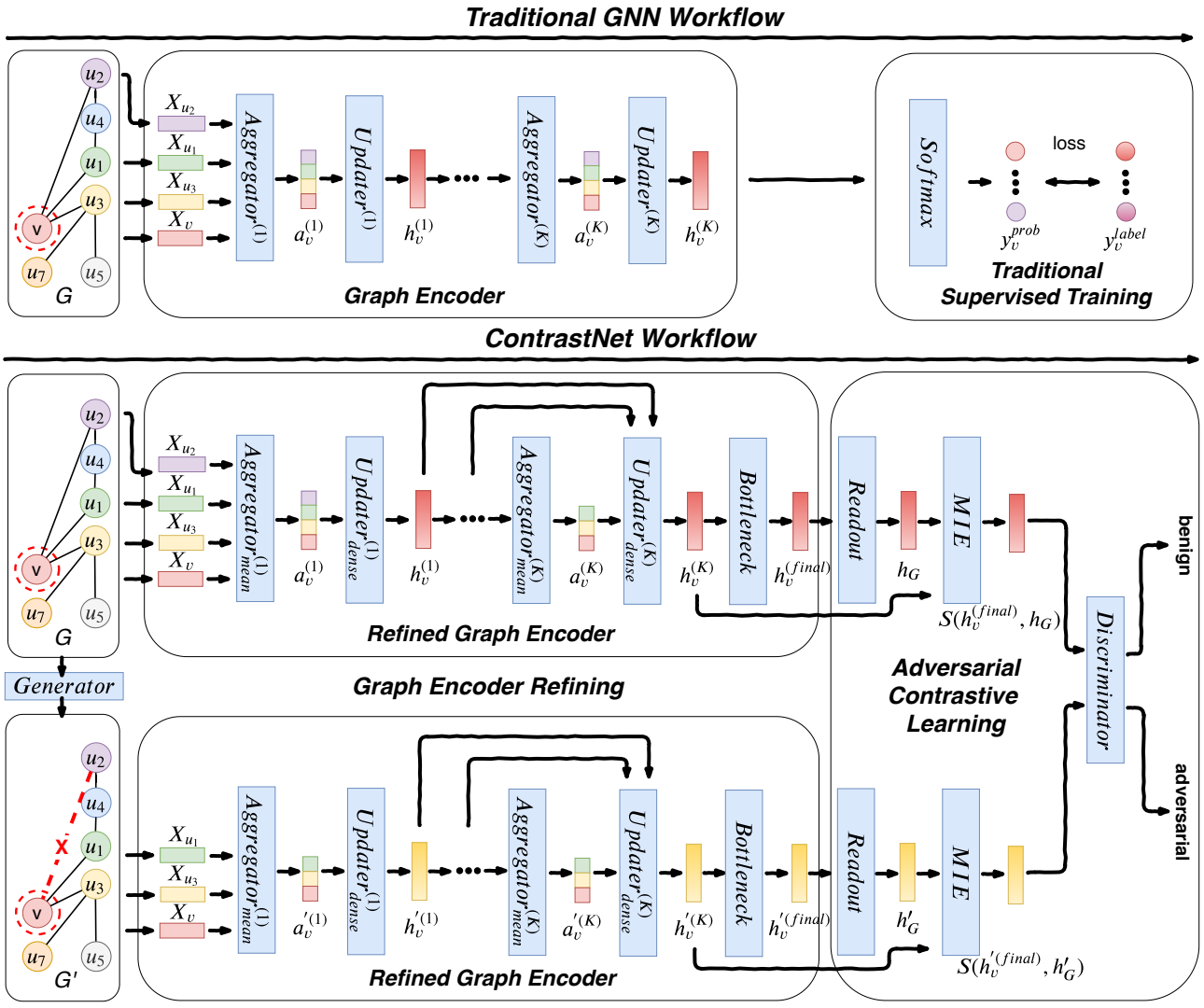

Fig. 1: Comparison between a traditional GNN workflow and **ContrastNet** workflow. **ContrastNet** contains two modules: Adversarial Contrastive Learning (**ACL**) and Graph Encoder Refining (**GER**). The **ACL** takes the benign graph samples (*e.g.*, G) and the adversarial graph samples (*e.g.*, G') as the input to train the refined GNN over contrastive adversarial space. And the **GER** polishes the GNN with mean-pooling aggregator (*i.e.*, $Aggregator_{mean}()$), dense-connected updater (*i.e.*, $Updater_{dense}()$), and bottleneck layer (*i.e.*, *Bottleneck()*).

samples as hard negative samples. The hard negative samples can help to learn a better discriminator for distinguishing the positive and negative pairs. The S and G are trained in a joint way: we adjust the parameters of G to maximize $\log(1 - S(G(x^+, y^-)))$ and adjust the parameters of S to minimize $\log S(x, y)$, as if they follow the two-player minimax game.

However, there is no constraint for the generator G, and G is not tailored to the graph data. This may cause the suboptimal quality of the generated adversarial samples since the adversarial latent space cannot be effectively explored. This limitation may affect the quality of the learned representation. To overcome it, we adopt a conditional generator configuration [28], which extends a conditional model where the generator and discriminator are both conditioned on some extra information. The extra information could be any kind of auxiliary information, such as class labels or data from other modalities.

The condition can be realized by feeding extra information into both the discriminator and generator as an additional input layer. As shown in Fig. 1, we consider the high-level graph representation (*e.g.*, graph embedding $\mathbf{h}_G$) as a global constraint to regularize the node representation learning. The motivation is based on the observation that global-level representation such as graph embedding is less sensitive to the adversarial perturbation than the local-level representation such as node embedding [8]. Thus, the high-level graph representation can be used to guide the robust learning of the local node representation. Accordingly, the objective of our adversarial contrastive learning becomes:

$$\min_{S}\max_{G}\{\mathbb{E}_{p(y^+|x^+)}[\log S(\mathbf{h}_v^{(final)}|\mathbf{h}_G)] + \mathbb{E}_{g(y|x))}[\log(1 - S(G(\mathbf{h}_v^{'(final)}|\mathbf{h}'_G)))]\} \tag{11}$$

where $\mathbf{h}_v^{'(final)}$ and $\mathbf{h}'_G$ are the node and graph embeddings from the generated adversarial samples $G' = (\mathcal{V}', \mathcal{E}', \mathbf{X}') = P(G)$. Here, the perturbation function $P()$ can be defined similarly to [53].

To obtain the graph embedding, we employ a *Readout* function, $Readout()$. Given the node representations $\{\mathbf{h}_v^{(final)}\}_{v\in\mathcal{V}}$ of graph G, the $Readout()$ performs a mean pooling followed by a nonlinear mapping as follows:

$$\mathbf{h}_G = Readout(\mathbf{h}_v^{(final)}|v \in \mathcal{V}) = \sigma(\mathbf{W}_r(\frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_i^{(final)})) \tag{12}$$

where σ represents the sigmoid function, $\mathbf{W}_r$ represents a trainable linear mapping matrix, n indicates the number of nodes within the graph. This module follows the idea of using the pooling function to compute the high-level representation in CNN. The reason of using a mean pooling is that representation generated by the mean operation is more stable and less sensitive to the small-scale adversarial attacks.

For the discriminator, we employ a mutual information estimator, $MIE()$, to jointly model the local and global graph representations via a bilinear scoring function:

$$S(\mathbf{h}_v^{(final)}|\mathbf{h}_G) = MIE(\mathbf{h}_v^{(final)}, \mathbf{h}_G) = \sigma(\mathbf{h}_v^{(final)}\mathbf{W}_D\mathbf{h}_G) \tag{13}$$

where $MIE(\mathbf{h}_v^{(K)}, \mathbf{h}_G)$ represents the mutual information between the node embedding and graph embedding. And $\mathbf{W}_D$ is a trainable scoring matrix and σ denotes the sigmoid function.

B. GER: Graph Encoder Refining

To address the challenge of the intrinsic architecture of GNN, we examine the vulnerabilities of every component of a GNN encoder, and propose corresponding refining strategies as shown in the lower part of Fig.1.

1) Aggregator Refining: As the first component of GNN, the aggregator leverages the graph structure (*i.e.*, the local context of each node) to improve the GNN performance by aggregating neighborhood. However, due to graph properties, such as homophily [25], the representation of a single node can easily be affected by its local context. Consequently, attackers can compromise a node's prediction without directly changing its features and/or edges [53].

Typical neighborhood aggregators of GNN include sum [46], max [13], and mean [18]. The sum aggregator sums up the features within the neighborhood $\mathcal{N}_v$ to captures the full neighborhood. The max aggregator generates the aggregated representation by element-wise max-pooling. It captures neither the exact structure nor the distribution of the neighborhood $\mathcal{N}_v$. Note as the size of the neighborhood $\mathcal{N}_v$ can be different for different v, the mean aggregator averages individual element features out. Different from the sum and max aggregators, it can capture the distribution of the features in the neighborhood $\mathcal{N}_v$.

The adversarial attacks are usually stealthy and can only perform small-scale modifications to the graph data. An ideal aggregator should be robust to such subtle perturbations in two folds: value and expressive power [46]. In the view of the value, among the three aggregators, the max aggregator is very sensitive to the distinct value modification (*e.g.*, by adding a new neighbor of a different class with a big value). In contrast, the sum and mean aggregator are less sensitive to the small-scale value modifications and thus is more robust to adversarial attacks. Different neighborhood sizes of each node will result in different denominators for the mean aggregator, but the same one for the sum aggregator. In this case, the mean aggregator is more robust. In the view of expressive power, the sum aggregator has been proven to have more expressive power than the mean aggregator and thus more sensitive to the structure change.

Therefore, in our **ContrastNet**, for the GNN that uses the sum or max aggregator, we refine its aggregator $Aggregator()$ by aggregating the neighborhood in a **mean pooling** manner, instead of original **sum** or **max pooling** as shown in the middle of Fig.1. Formally, it is defined as follows:

$$\mathbf{a}_v^{(k)} = Aggregator_{mean}^{(k)}(\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}_v\}) \tag{14}$$

$$= ReLU(\mathbf{W}^{(k)}\frac{1}{|\mathcal{N}_v|}\sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{(k-1)}) \tag{15}$$

where k is the index of iteration/layer, $|\mathcal{N}_v|$ is the cardinality of the neighborhood of node v, and $\mathbf{a}_v^{(k)}$ is the result of the mean aggregation. $\mathbf{W}^{(k)}$ is the trainable mapping matrix and $ReLU$ represents the rectified linear unit nonlinear gating function.

2) Updater Refining: The second important component of GNN is updater, which helps to compute the new representation of each node.

Some GNN methods, such as GCN [18], simply use the aggregated representation as the new representation for each node. In this case, each node is updated as follows:

$$\mathbf{h}_v^{(k)} = Updater^{(k)}(\mathbf{a}_v^{(k)}) = \mathbf{a}_v^{(k)} \tag{16}$$

Some other GNNs, such as GraphSage [13], leverage the **skip-connection** to combine the current aggregated representation with the updated representation from the last iteration, such that each node is updated as:

$$\mathbf{h}_v^{(k)} = Updater^{(k)}(\mathbf{a}_v^{(k)}) = [\mathbf{a}_v^{k}; \mathbf{h}_v^{(k-1)}] \tag{17}$$

where $[\cdot;\cdot]$ represents the feature concatenation operation. This **skip-connection** method gathers the representation from the predecessor layer and drops the rest of intermediate representations during the information propagation within k-hop neighbors. However, stacking multiple such layers could also propagate noisy or adversarial information from an exponentially increasing number of expanded neighboring members [18].

To address this problem, we propose to refine the updater in a **dense-connected** style, as inspired by the DenseNet [16]. Our method keeps all intermediate representations and fuse

them together to compute the recent-layer representation. In this way, the recent layer is connected with all previous representations, allowing the subsequent layer to selectively but adaptively fuse structure information from different hops. Consequently, the robustness can be improved for deep GNN. Formally, the **dense-connected** updater is constructed as follows:

$$\mathbf{h}_v^{(k)} = Updater^{(k)}(\mathbf{a}_v^{(k)}) = [\mathbf{a}_v^k; \mathbf{h}_v^{(k-1)}; \mathbf{h}_v^{(k-2)}; ...; \mathbf{h}_v^{(1)}]. \tag{18}$$

3) Bottleneck Refining: In this section, we propose the Bottleneck Refining to enforce the GNN to map the node representation in a non-linearly low-dimension space. Particularly, it consists of a bottleneck mapping followed by a nonlinear mapping as shown in Fig.1.

Recent studies have empirically studied the one-to-one relationship between the adversarial vulnerability and the input dimensionality, and theoretically proved that the adversarial vulnerability of neural networks deteriorates as the input dimensionality increases [37]. On the other hand, it is common that real data are not truly high-dimensional [20]. The data in a high-dimensional space can be well embedded into a lower dimensional space for both better effectiveness and efficiency. Therefore, it is important to perform dimensionality reduction on the input to improve the robustness of a GNN .

To this purpose, we propose a bottleneck refining strategy to add a bottleneck perceptron after the last $Updater^{(K)}$ in order to enforce the output dimensionality of GNN being much lower than that of the input. Formally, the bottleneck perceptron $BPercep()$ is defined as follows:

$$\mathbf{h}_v^{(final)} = BPercep(\mathbf{h}_v^{(K)}) = ReLU((\mathbf{W}_p)\mathbf{h}_v^{(K)}) \tag{19}$$

where $\mathbf{W}_p$ is the trainable mapping matrix map the final layer of node representation into the lower dimensional space. And $ReLU$ represents the rectified linear unit nonlinear gating function.

IV. Experiments

In this section, we evaluate our proposed **ContrastNet** framework on three real-world graph datasets. We first introduce the datsets used in the experiments, experiment setup with attack models, compared methods and configuration. Next, we show the effectiveness of the proposed **ContrastNet** . We further conduct several experiments to show the effectiveness of the adversarial contrastive learning and graph encoder refining.

A. Datasets

TABLE I: The statistics of the datasets.

Dataset	#Node	#Edge	#Feature	#Class
Cora	2,810	7,981	1,433	7
Citeseer	2,110	3,757	3,703	6
PolBlogs	1,222	16,714	1,222	2

We use three benchmark datasets including two academic networks (Cora and Citeseer) and a social network (PolBlogs) for node classification tasks. We select the largest connected component in each dataset for our experiments, and we split each dataset randomly into a labeled set (80%) and an unlabeled set (20%). We then further divide the labeled set into a training set (50%) and a validate set (50%). Table I shows the statistics of the prepossessed datasets.

B. Experiment Setup

1) Attack Models: We conduct three types of popular adversarial attacks:

- **RAND** (a random perturbation attack): Given the target node, it randomly adds edges to the pairs of nodes that belong to different classes and/or deletes edges that connect the pairs of nodes within the same class.
- **FGSM** [12] (a gradient based attack): It generates adversarial examples based on the sign of gradient.
- **NETTACK** (a state-of-the-art optimization based attack [53]): It generates adversarial perturbations by searching the perturbation space.

The attack procedure is similar to [53]. We select 200 nodes as the targets: 50 correctly classified nodes with high confidence; 50 correctly classified nodes but with low confidence; and 100 random nodes.

*2) **Compared Methods**:* To show the effectiveness of the proposed **ContrastNet** framework on defending the adversarial attacks, we compare it with a number of baselines including two adversarial training methods (**RD** [8] and **NCL**), a smoothness enforced method **DIT** [32], and variants of the proposed **ContrastNet**.

RD [8] generates random samples from a noise distribution and trains the GNN in a traditional supervised way. **NCL** also generates random samples from a noise distribution, but follows the way of Adversarial Contrastive Learning (**ACL**) to train the GNN in a contrastive learning way. In contrast, **DIT** [32] leverages model-specific strategies to enforce the model smoothness. **DIT** is originally designed for the image classification via CNN, but we extend it to solve the node classification problem.

We evaluate our defense framework **ContrastNet** on two most popular GNN variants: GCN [18] and GraphSage [13]. To evaluate **ContrastNet** and **ACL**, we use the original GNN (O or O_{XXX}) and the refined GNN after Graph Encoder Refining **GER** (R or R_{XXX}) as graph encoder, and consider their variants in terms of adversarial learning methods: **RD**, **NCL**, **DIT**, and **ACL**. To evaluate **GER**, we compare different versions of GCN and GraphSage in terms of aggregator strategies: XXX_{mean}, XXX_{sum}, and XXX_{max}; updater strategies: XXX_{none} (simple updater as in Eq. 16), XXX_{skip} (skip-connected updater as in Eq. 17), and XXX_{dense} (dense-connected updater); bottleneck strategies: XXX_{low} (final layer dimension is only half of the original dimension), XXX_{mid} (final layer dimension is exactly the same as the original dimension), and XXX_{high} (final layer dimension is twice of the original dimension).

TABLE II: Performance comparison on **GCN** under various attacks.

Data	Attack	O	O_{RD}	O_{DIT}	O_{NCL}	O_{ACL}	R	R_{RD}	R_{DIT}	R_{NCL}	R_{ACL}
Cora	Clean	0.90	0.85	0.85	0.88	0.89	0.88	0.84	0.85	0.88	**0.92**
	RAND	0.60	0.71	0.75	0.72	0.81	0.73	0.72	0.76	0.80	**0.85**
	FGSM	0.03	0.17	0.21	0.39	0.53	0.18	0.20	0.45	0.42	**0.65**
	NETTACK	0.01	0.16	0.19	0.27	0.32	0.15	0.17	0.43	0.38	**0.60**
Citeseer	Clean	0.88	0.82	0.83	0.83	0.84	**0.90**	0.88	0.86	0.86	**0.90**
	RAND	0.60	0.72	0.70	0.75	0.79	0.72	0.75	0.75	0.79	**0.85**
	FGSM	0.07	0.22	0.25	0.44	0.62	0.20	0.25	0.27	0.52	**0.70**
	NETTACK	0.02	0.05	0.19	0.38	0.52	0.16	0.20	0.22	0.45	**0.65**
PolBlogs	Clean	0.93	0.90	0.88	0.90	0.92	0.95	0.95	0.92	0.87	**0.95**
	RAND	0.36	0.80	0.78	0.80	0.79	0.40	0.42	0.40	**0.82**	**0.82**
	FGSM	0.41	0.46	0.43	0.49	0.60	0.50	0.52	0.48	0.58	**0.74**
	NETTACK	0.06	0.10	0.12	0.43	0.58	0.18	0.22	0.20	0.52	**0.65**

TABLE III: Performance comparison on **GraphSage** under various attacks.

Data	Attack	O	O_{RD}	O_{DIT}	O_{NCL}	O_{ACL}	R	R_{RD}	R_{DIT}	R_{NCL}	R_{ACL}
Cora	Clean	0.85	0.84	0.80	0.82	0.81	0.88	0.80	0.83	0.84	**0.88**
	RAND	0.70	0.69	0.72	0.78	0.80	0.78	0.79	0.81	0.82	**0.84**
	FGSM	0.18	0.19	0.29	0.40	0.61	0.25	0.26	0.28	0.48	**0.67**
	NETTACK	0.16	0.17	0.25	0.38	0.57	0.22	0.23	0.22	0.42	**0.60**
Citeseer	Clean	0.83	0.81	0.80	0.80	0.80	0.86	0.84	083.	0.82	**0.90**
	RAND	0.70	0.72	0.68	0.78	0.82	0.80	0.80	0.82	0.82	**0.85**
	FGSM	0.10	0.14	0.18	0.32	0.52	0.22	0.24	0.26	0.50	**0.70**
	NETTACK	0.04	0.08	0.16	0.30	0.48	0.18	0.18	0.28	0.48	**0.65**
PolBlogs	Clean	0.86	0.83	0.82	0.84	0.85	**0.90**	0.86	0.85	0.82	0.89
	RAND	0.43	0.69	0.65	0.62	0.70	0.60	0.62	0.60	0.68	**0.78**
	FGSM	0.40	0.43	0.40	0.48	0.50	0.48	0.46	0.50	0.59	**0.72**
	NETTACK	0.14	0.16	0.20	0.45	0.56	0.20	0.25	0.28	0.55	**0.68**

*3) **Configuration**:* We repeat all the experiments over five different splits of labeled/unlabeled nodes to compute the average results. For GNNs used in the experiments, we set the default number of layers as 4. We also adopt the Adam optimizer with an initial learning rate of 0.01, and decay the learning rate by 0.5 for every 50 epochs.

C. Evaluation of *ContrastNet* Framework and *ACL*

In this experiment, we show the effectiveness of the proposed **ContrastNet** framework and the **ACL** module under various attacks. To guarantee the powerfulness of the adversarial attacks, we set the number of the perturbations as: $n_p = d_v + 2$ with d_v to be the degree of the target node. We report the fraction of the target nodes that get correctly classified as accuracy.

The results in Table II and Table III show that the proposed **ACL** method outperforms all the baselines including the single GER, RD, DIT and NCL, in terms of datasets, GNN variants, and attack models. We also observe that the original GCN and GraphSage are both very vulernable to the attacks, but after applying our **ContrastNet** framework, both of them become way more robust. For example, GCN can achieve an accuracy of 0.90 without any attack on Cora data, but the performance significantly drops to 0.03 under the **FGSM** attack, while R_{ACL} (*i.e.*, GCN with **ContrastNet**) can still achieve an accuracy of 0.65 as shown in Table II. Similarly, for Citeseer dataset, under the **NETTACK** attack, the accuracy of GraphSage drops from 0.83 to 0.04, while R_{ACL} (*i.e.*, GraphSage with **ContrastNet**) can still maintain an accuracy of 0.65 as shown in Table III.

D. Evaluation of *GER*

In this experiment, we show the vulnerabilities of the GNN in terms of its aggregator, updater, and output. We also demonstrate the effectiveness of **GER** in addressing these vulnerabilities.

We test the robustness of different methods under the powerful NETTACK attack[2] with various adversarial perturbations. We use the classification margin score s as the metric for the classifier performance:

$$s = Z^*_{v,c_v^{(gt)}} - max_{c \neq c_v^{(gt)}} Z*_{v,c} \tag{20}$$

where $Z^*_{v,c_v^{(gt)}}$ is the predicted probability for the ground-truth label, while $max_{c \neq c_v^{(gt)}} Z*_{v,c}$ is the incorrect prediction probability with the largest confidence. The larger s is, the higher confidence the classifier has in making a correct prediction.

From the experimental results shown in Fig. 2, we can see that the GCN and GraphSAGE with our proposed **GER** perform better than the original version and the ones using other aggregator or updater strategies. We can also observe that: (1) The mean aggregator outperforms the sum and max ones; (2) The dense-connected updater outperforms the case without the layer-wise connection and the skip-connected one; (3) The lower dimension of the final layer beats both the high and medium ones, since the adversarial vulnerability of neural networks deteriorates as the input dimensionality increases; (4) Comparing GCN with GraphSage, the performance of the latter one is less stable but sometimes better, due to the random

[2]NETTACK is more powerful than RAND and FGSM, thus due to the space limitation, we only show the results of NETTACK in this experiment.

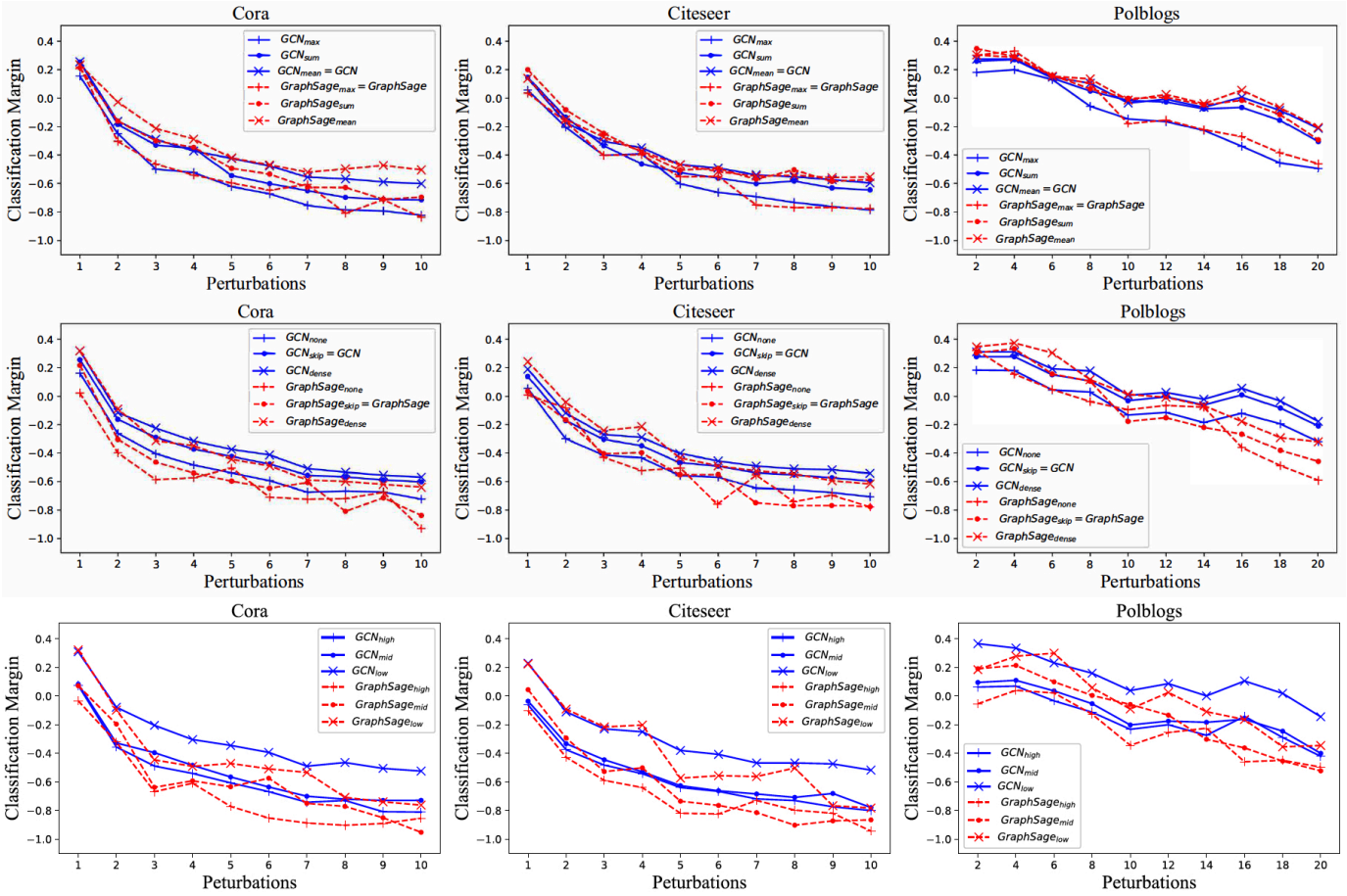

Fig. 2: Evaluation of the **GER**. This experiment is conducted with traditional supervised training.

neighborhood sampling; and (5) It is also worth noticing that PolBlogs is difficult to attack, since the average degree of the node is high.

V. Related Work

In line with the focus of this work, we briefly review the recent work on adversarial attack and defense on traditional machine learning, and adversarial attack and defense on graph neural networks.

A. Attack and Defense on Traditional Machine Learning

Recently, deep neural networks (DNNs) have been proven to be highly sensitive to small adversarial perturbations [12], [38]. Based on the attacker's goals, the adversarial attack can be categorized into poisoning attack and evasion attack, where the poisoning attack's target is the training data and evasion attack's target is the test data. Most of the previously introduced work belong to the evasion attack. [19], [21], [30] studied the problem of poisoning attack and proposed different bi-level optimization solutions.

To defense against the adversarial attack and improve the robustness of the traditional machine learning models, especially DNNs, a number of effects have been made. They fall into three main categories: (1) adversarial training methods [12], [26], [31], (2) adversarial perturbation removing methods [35], and (3) smoothness enforced methods [32]. However, these defense work mainly focus on either image data [7], [12], [23], [26], [31], [33], [35], [36], which assumes data are independently and identically distributed in the continuous feature space and can not be directly applied to GNNs.

B. Attack and Defense on *GNNs*

Recently, a few attempts have been made to study adversarial attacks on **GNNs**. [8] proposed a non-target evasion attack on node classification and graph classification. A reinforcement learning method was used to learn the attack policy that applies small-scale modification to the graph structure. [53] introduced a poisoning attack on node classification. This work adopted a substitute model attack and formulated the attack problem as a bi-level optimization. Different from [8], it attacked both the graph structure and node attributes. Furthermore, it considered both direct attacks and influence attacks. [54] proposed a meta-learning based non-targeted poison attack method. [3] proposed a targeted poison attack on a random walk based unsupervised node embedding.

To defend GNN against adversarial attacks, few initial attempts have been made to study the robustness of GNN. For

example, Zügner and Günnemann [55] proposed a certification model and robust training specifically for perturbation of the discrete node attributes. [51] exploited the Gaussian distribution as the node representation and proposed a variance-based attention mechanism for neighborhood aggregation. However, there is still limited understanding of why GNNs are vulnerable to adversarial attacks and how to defend GNNs against various adversarial attacks. Different from existing methods, this work focuses on defending graph convolutional based GNN against *directed targeted poisoning attacks on both discrete node feature and graph structure*.

VI. Conclusion

In this paper, we presented **ContrastNet**, an adversarial defense framework for Graph Neural Network (GNN). To train a robust GNN, we proposed an adversarial contrastive learning technique instead of traditional supervised training. We also leveraged the high-level graph representation to constrain the generation of the adversarial samples. To address the vulnerabilities in the GNN architecture, we proposed the mean aggregation, dense connection, and bottleneck strategies. We evaluated the proposed framework using extensive experiments on three real-world graph datasets. The experimental results convince us of the effectiveness of our framework on defending the popular GNN variants against various types of adversarial attacks.

VII. Acknowledgements

Shen Wang and Philip S. Yu are supported in part by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941.

References

[1] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *NIPS*, pages 4509–4517, 2016.

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

[3] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. *arXiv preprint arXiv:1809.01093*, 2018.

[4] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM international conference on Information & Knowledge Management*, pages 3747–3756, 2021.

[5] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

[6] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. Gram: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 787–795. ACM, 2017.

[7] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. *arXiv preprint arXiv:1810.07481*, 2018.

[8] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[10] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *NIPS*, pages 6530–6539, 2017.

[11] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.

[14] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[15] Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. In *NIPS*, pages 2701–2711, 2017.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[17] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

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

[19] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

[20] Elizaveta Levina and Peter J Bickel. Maximum likelihood estimation of intrinsic dimension. In *NIPS*, pages 777–784, 2005.

[21] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in neural information processing systems*, pages 1885–1893, 2016.

[22] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[23] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *CVPR*, pages 1778–1787, 2018.

[24] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. Heterogeneous graph neural networks for malicious account detection. In *CIKM*, pages 2077–2085, 2018.

[25] Ben London and Lise Getoor. Collective classification of network data. *Data Classification: Algorithms and Applications*, 399, 2014.

[26] 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.

[27] Chengsheng Mao, Liang Yao, and Yuan Luo. Medgcn: Graph convolutional networks for multiple medical tasks. *arXiv preprint arXiv:1904.00326*, 2019.

[28] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[29] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707, 2017.

[30] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38. ACM, 2017.

[31] Taesik Na, Jong Hwan Ko, and Saibal Mukhopadhyay. Cascade adversarial machine learning regularized with a unified embedding. *arXiv preprint arXiv:1708.02582*, 2017.

[32] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *SP*, pages 582–597, 2016.

[33] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *CVPR*, pages 8571–8580, 2018.

[34] Sungmin Rhee, Seokjun Seo, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *arXiv preprint arXiv:1711.05859*, 2017.

[35] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[36] Lukas Schott, Jonas Rauber, Wieland Brendel, and Matthias Bethge. Robust perception through analysis by synthesis. *CoRR*, abs/1805.09190, 2018.

[37] Carl-Johann Simon-Gabriel, Yann Ollivier, Bernhard Schölkopf, Léon Bottou, and David Lopez-Paz. Adversarial vulnerability of neural networks increases with input dimension. *arXiv preprint arXiv:1802.01421*, 2018.

[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[39] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[40] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[41] Shen Wang, Zhengzhang Chen, Ding Li, Zhichun Li, Lu-An Tang, Jingchao Ni, Junghwan Rhee, Haifeng Chen, and Philip S Yu. Attentional heterogeneous graph neural network: Application to program reidentification. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 693–701. SIAM, 2019.

[42] Shen Wang, Zhengzhang Chen, Jingchao Ni, Xiao Yu, Zhichun Li, Haifeng Chen, and Philip S Yu. Adversarial defense framework for graph neural network. *arXiv preprint arXiv:1905.03679*, 2019.

[43] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. *arXiv preprint arXiv:1811.00855*, 2018.

[44] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[45] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[47] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. Neural network-based graph embedding for cross-platform binary code similarity detection. In *CCS*, pages 363–376, 2017.

[48] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. *arXiv preprint arXiv:1806.01973*, 2018.

[49] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning withdifferentiable pooling. *arXiv preprint arXiv:1806.08804*, 2018.

[50] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[51] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1399–1407, 2019.

[52] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

[53] Daniel Zugner, Amir Akbarnejad, and Stephan Gunnemann. Adversarial attacks on neural networks for graph data. In *SIGKDD*, pages 2847–2856, 2018.

[54] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412*, 2019.

[55] 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 & Data Mining*, pages 246–256, 2019.