Certified Robustness of Graph Neural Networks against Adversarial Structural Perturbation

Binghui Wang, Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong

ECE Department, Duke University, Durham, NC, USA, 27708
{binghui.wang,jinyuan.jia,xiaoyu.cao,neil.gong}@duke.edu

ABSTRACT

Graph neural networks (GNNs) have recently gained much attention for node and graph classification tasks on graph-structured data. However, multiple recent works showed that an attacker can easily make GNNs predict incorrectly via perturbing the graph structure, i.e., adding or deleting edges in the graph. We aim to defend against such attacks via developing certifiably robust GNNs. Specifically, we prove the certified robustness guarantee of any GNN for both node and graph classifications against structural perturbation. Moreover, we show that our certified robustness guarantee is tight. Our results are based on a recently proposed technique called randomized smoothing, which we extend to graph data. We also empirically evaluate our method for both node and graph classifications on multiple GNNs and multiple benchmark datasets. For instance, on the Cora dataset, Graph Convolutional Network with our randomized smoothing can achieve a certified accuracy of 0.49 when the attacker can arbitrarily add/delete at most 15 edges in the graph.

CCS CONCEPTS

• Security and privacy; • Computing methodologies → Machine learning;

KEYWORDS

Adversarial machine learning, graph neural network, certified robustness, structural perturbation

ACM Reference Format:
Binghui Wang, Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. 2021. Robustness of Graph Neural Networks against Adversarial Structural Perturbation. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467295

1 INTRODUCTION

Graphs are a powerful tool to represent data from diverse domains such as social networks, biology, finance, security, etc. Node classification and graph classification are two basic tasks on graphs. Specifically, given a graph, node classification aims to classify nodes in the graph in a semi-supervised fashion, while graph classification aims to assign a label to the entire graph instead of individual nodes. Node and graph classifications have many applications, including but not limited to, profiling users in social networks [19, 23, 35, 66], classifying proteins [20, 49], and fraud detection [18, 37, 46, 51–55]. Various methods such as label propagation [68], belief propagation [38], iterative classification [43], and graph neural networks (GNNs) [17, 20, 26, 49, 62] have been proposed for node/graph classifications. Among them, GNNs have attracted much attention recently because of their expressiveness and superior performance.

However, several recent works [1, 10, 50, 69, 70] have demonstrated that an attacker can easily fool GNNs to make incorrect predictions via perturbing 1) the node features and/or 2) the structure of the graph (i.e., adding and/or deleting edges in the graph). Therefore, it is of great importance to develop certifiably robust GNNs against such attacks. A few recent works [2, 3, 24, 71, 72] study certified robustness of GNNs against node feature or structural perturbation. However, these methods only focus on certifying robustness for a specific GNN [2, 24, 71, 72] or and did not formally prove tight robustness guarantees [3] in general.

We aim to bridge this gap in this work. Specifically, we aim to provide tight certified robustness guarantees of any GNN classifier against structural perturbations for both node and graph classification. Towards this goal, we leverage a recently developed technique called randomized smoothing [5, 9, 21, 28, 32, 33], which can turn any base classifier (e.g., a GNN classifier in our problem) to a robust one via adding random noise to a testing example (i.e., a graph in our problem). A classifier is certifiably robust if it provably predicts the same label when the attacker adds/deletes at most K edges in the graph, where we call K certified perturbation size.

Graph structures are binary data, i.e., a pair of nodes can be either connected or unconnected. Therefore, we develop randomized smoothing for binary data and leverage it to obtain certified robustness of GNNs against structural perturbation. First, we theoretically derive a certified perturbation size for any GNN classifier with randomized smoothing via addressing several challenges. For instance, we divide the graph structure space into regions in a novel way such that we can apply the Neyman-Pearson Lemma [36] to certify robustness. We also prove that our derived certified perturbation size is tight if no assumptions on the GNN classifier are made.

Second, we design a method to compute our certified perturbation size in practice. It is challenging to compute our certified perturbation size as it involves estimating probability bounds simultaneously and solving an optimization problem. To address the challenge, we first adopt the simultaneous confidence interval estimation method [21] to estimate the probability bounds with probabilistic guarantees. Then, we design an algorithm to solve
optimization problem to obtain our certified perturbation size with the estimated probability bounds.

We also empirically evaluate our method. Specifically, for node classification, we consider Graph Convolutional Network (GCN) [26] and Graph Attention Network (GAT) [49] on several benchmark datasets including Cora, Citeseer, and Pubmed [43]. For graph classification, we consider Graph Isomorphism Network (GIN) [61] on benchmark datasets including MUTAG, PROTEINS, and IMDB [63]. For instance, on the Cora dataset, GCN with our randomized smoothing can achieve certified accuracies of 0.55, 0.50, and 0.49 when the attacker arbitrarily adds/deletes at most 5, 10, and 15 edges, respectively. On the MUTAG dataset, GIN with our randomized smoothing can achieve certified accuracies of 0.45, 0.45, and 0.40 when the attacker arbitrarily adds/deletes at most 5, 10, and 15 edges, respectively.

Our major contributions can be summarized as follows:

- We prove the certified robustness guarantee of any GNN against structural perturbation. Moreover, we show that our certified robustness guarantee is tight.
- Our certified perturbation size is the solution to an optimization problem and we design an algorithm to solve the problem.
- We empirically evaluate our method for both node and graph classifications on multiple benchmark datasets.

2 BACKGROUND

2.1 Node Classification vs. Graph Classification

We consider GNNs for both node classification [20, 26, 49] and graph classification [20, 61]. Suppose we are given an undirected graph $G$ with node features.

- **Node classification.** A node classifier predicts labels for nodes in the graph in a semi-supervised fashion. Specifically, a subset of nodes in $G$ are already labeled, which are called training nodes and denoted as $V_T$. A node classifier $f_u$ takes the graph $G$ and the training nodes $V_T$ as an input and predicts the label for remaining nodes, i.e., $f_u(G, V_T, u)$ is the predicted label for a node $u$.

- **Graph classification.** A graph classifier aims to predict a label for the entire graph instead of individual nodes, i.e., $f(G)$ is the predicted label for the graph $G$. Such a graph classifier can be trained using a set of graphs with ground truth labels.

2.2 Adversarial Structural Perturbation

An attacker aims to fool a node classifier or graph classifier to make predictions as the attacker desires via perturbing the graph structure, i.e., deleting some edges and/or adding some edges in the graph [1, 10, 50, 69, 70]. Since our work focuses on structural perturbation, we treat the node feature matrix and the training nodes as constants. Moreover, we simply write a node classifier $f_u(G, V_T, u)$ or a graph classifier $f(G)$ as $f(s)$, where the binary vector $s$ represents the graph structure. Note that a node classifier should also take a node $u$ as input and predict its label. However, we omit the explicit dependency on a node $u$ for simplicity. We call $s$ a **structure vector**. For instance, $s$ can be the concatenation of the upper triangular part of the adjacency matrix of the graph (excluding the diagonals) when the attacker can modify the connection status of any pair of nodes, i.e., $s$ includes the connection status for each pair of nodes in the graph. When the attacker can only modify the connection status between $u$ and each remaining node, $s$ can also be the $u$th row of the adjacency matrix of the graph (excluding the self-loop $(u, u)$th entry). We assume the structure vector $s$ has $n$ entries. As we will see, such simplification makes it easier to present our certified robustness against structural perturbation.

We denote by vector $δ ∈ \{0, 1\}^n$ the attacker’s perturbation to the graph structure. Specifically, if $δ_u = 1$ if and only if the attacker changes the $u$th entry in the structure vector $s$, i.e., the attacker changes the connection status of the corresponding pair of nodes. Moreover, $s ⊕ δ$ is the perturbed structure vector, which represents the perturbed graph structure, where $⊕$ is the XOR operator between two binary variables. We use $||δ||_0$ to measure the magnitude of the adversarial perturbation as it has semantic meanings. Specifically, $||δ||_0$ is the number of node pairs whose connection statuses are modified by the attacker.

3 CERTIFIED ROBUSTNESS

3.1 Randomized Smoothing with Binary Noise

We first define a noise distribution in the discrete structure vector space $\{0, 1\}^n$. Then, we define a smoothed classifier based on the noise distribution and a node/graph classifier (called base classifier). Specifically, we consider the noise vector $e$ has the following probability distribution in the discrete space $\{0, 1\}^n$:

$$Pr(e_i = 0) = β, \quad Pr(e_i = 1) = 1 - β,$$

where $i = 1, 2, \ldots, n$. When we add a random noise vector $e$ to the structure vector $s$, the $i$th entry of $s$ is preserved with probability $β$ and changed with probability $1 - β$. In other words, our random noise means that, for each pair of nodes in the graph, we keep its connection status with probability $β$ and change its connection status with probability $1 - β$. Based on the noise distribution and a base node/graph classifier $f(s)$, we define a smoothed classifier $g(s)$ as follows:

$$g(s) = \arg\max_{c ∈ C} Pr(f(s ⊕ e) = c),$$

where $C$ is the set of labels, $⊕$ is the XOR operator between two binary variables, $Pr(f(s ⊕ e) = c)$ is the probability that the base classifier $f$ predicts label $c$ when we add random noise $e$ to the structure vector $s$, and $g(s)$ is the label predicted for the perturbed structure vector $s ⊕ δ$. Existing randomized smoothing methods [5, 9, 21, 28, 32, 33] add random continuous noise (e.g., Gaussian noise, Laplacian noise) to a testing example (i.e., the structure vector $s$ in our case). However, such continuous noise is not meaningful for the binary structure vector.

Our goal is to show that a label is provably the predicted label by the smoothed classifier $g$ for the perturbed structure vector $s ⊕ δ$ when the $0$-norm of the adversarial perturbation $δ$, i.e., $||δ||_0$, is bounded. Next, we theoretically derive the certified perturbation size $K$, prove the tightness of the certified perturbation size, and discuss how to compute the certified perturbation size in practice.
3.2 Theoretical Certification

3.2.1 Overview. Let \( X = s \oplus \varepsilon \) and \( Y = s \ominus \delta \oplus \varepsilon \) be two random variables, where \( \varepsilon \) is the random binary noise drawn from the distribution defined in Equation 1. \( X \) and \( Y \) represent random structure vectors obtained by adding random binary noise \( \varepsilon \) to the structure vector \( s \) and its perturbed version \( s \oplus \delta \), respectively.

Suppose, when taking \( X \) as an input, the base GNN classifier \( f \) correctly predicts the label \( c_A \) with the largest probability. Then, our key idea is to guarantee that, when taking \( Y \) as an input, \( f \) still predicts \( c_A \) with the largest probability. Moreover, we denote \( c_B \) as the predicted label by \( f \) with the second largest probability.

Then, our goal is to find the maximum perturbation size such that the following inequality holds:

\[
\Pr(f(Y) = c_A) > \Pr(f(Y) = c_B).
\]  

Note that it is challenging to compute the probabilities \( \Pr(f(Y) = c_A) \) and \( \Pr(f(Y) = c_B) \) exactly because \( f \) is highly nonlinear in practice. To address the challenge, we first derive a lower bound of \( \Pr(f(Y) = c_A) \) and an upper bound of \( \Pr(f(Y) = c_B) \). Then, we require that the lower bound of \( \Pr(f(Y) = c_A) \) is larger than the upper bound of \( \Pr(f(Y) = c_B) \). Specifically, we derive the lower bound and upper bound by constructing certain regions in the graph structure space \( \{0, 1\}^n \) such that the probabilities \( \Pr(Y) \) is in these regions can be efficiently computed for any \( ||\delta||_0 = k \). Then, we iteratively search the maximum \( k \) under the condition that the lower bound is larger than the upper bound. Finally, we treat the maximum \( k \) as the certified perturbation size \( K \).

3.2.2 Deriving the lower and upper bounds. Our idea is to divide the graph structure space \( \{0, 1\}^n \) into regions in a novel way such that we can apply the Neyman-Pearson Lemma [36] to derive the lower bound and upper bound. First, for any data point \( z \in \{0, 1\}^n \), we have the density ratio \( \frac{\Pr(X = z)}{\Pr(Y = z)} = \left( \frac{\beta}{\gamma} \right)^w \) based on the noise distribution defined in Equation 1, where \( w = ||s - z||_0 \) and \( \epsilon = ||s \oplus \delta - z||_0 \) (please refer to Section 7.1 in for details).

Therefore, we have the density ratio \( \frac{\Pr(X = z)}{\Pr(Y = z)} = \left( \frac{\beta}{\gamma} \right)^m \) for any \( z \in \{0, 1\}^n \), where \( m = n - n + 1, \ldots, n - n + 1 \). Furthermore, we define a region \( R(m) \) as the set of data points whose density ratios are \( \left( \frac{\beta}{\gamma} \right)^m \), i.e., \( R(m) = \{ z \in \{0, 1\}^n : \frac{\Pr(X = z)}{\Pr(Y = z)} = \left( \frac{\beta}{\gamma} \right)^m \} \), and we denote by \( r(m) \) the corresponding density ratio, i.e., \( r(m) = \left( \frac{\beta}{\gamma} \right)^m \). Moreover, we rank the \( m + 1 \) regions in a descending order with respect to their density ratios, and denote the ranked regions as \( R_1, R_2, \ldots, R_{m+1} \).

Suppose we have a lower bound of the largest label probability \( \Pr(f(X) = c_A) \) for \( c_A \) and denote it as \( p_A \), and an upper bound of the remaining label probability \( \Pr(f(X) = c) \) for \( c \neq c_A \) and denote it as \( p_B \). Assuming there exist \( c_A \) and \( p_A, p_B \in [0, 1] \) such that

\[
\Pr(f(X) = c_A) \geq p_A \geq p_B \geq \max_{c \neq c_A} \Pr(f(X) = c).
\]  

Next, we construct two regions \( A \) and \( B \) such that \( \Pr(X \in A) = p_A \) and \( \Pr(X \in B) = p_B \), respectively. Specifically, we gradually add the regions \( R_1, R_2, \ldots, R_{m+1} \) to \( A \) until \( \Pr(X \in A) = p_A \). Moreover, we gradually add the regions \( R_{m+1}, R_{m+2}, \ldots, R_1 \) to \( B \) until \( \Pr(X \in B) = p_B \). We construct the regions \( A \) and \( B \) in this way such that we can apply the Neyman-Pearson Lemma [36] for them. Formally, we define the regions \( A \) and \( B \) as follows:

\[
A = \bigcup_{j=1}^{m+1} R_j \cup R_{a^*},
\]

\[
B = \bigcup_{j=b^*+1}^{m+1} R_j \cup R_{b^*},
\]

where

\[
a^* = \arg\min_{a \in \{1, 2, \ldots, m+1\}} \sum_{j=1}^{m+1} \Pr(X \in R_j) \geq p_A,
\]

\[
b^* = \arg\max_{b \in \{1, 2, \ldots, m+1\}} \sum_{j=b}^{m+1} \Pr(X \in R_j) \geq p_B.
\]

\( R_{a^*} \) is any subregion of \( R_{a^*} \) such that \( \Pr(X \in R_{a^*}) = p_A - \Sigma_{j=1}^{m+1} \Pr(X \in R_j) \), and \( R_{b^*} \) is any subregion of \( R_{b^*} \) such that \( \Pr(X \in R_{b^*}) = p_B - \Sigma_{j=b^*+1}^{m+1} \Pr(X \in R_j) \).

Finally, based on the Neyman-Pearson Lemma, we can derive a lower bound of \( \Pr(f(Y) = c_A) \) and an upper bound of \( \Pr(f(Y) = c_B) \). Formally, we have the following lemma:

**Lemma 1.** We have the following bounds:

\[
\Pr(f(Y) = c_A) \geq \Pr(Y \in A),
\]

\[
\Pr(f(Y) = c_B) \leq \Pr(Y \in B).
\]

**Proof.** See Section 7.1.

3.2.3 Deriving the certified perturbation size. Given Lemma 1, we can derive the certified perturbation size \( K \) as the maximum \( k \) such that the following inequality holds for \( ||\delta||_0 = k \):

\[
\Pr(Y \in A) > \Pr(Y \in B).
\]

Formally, we have the following theorem:

**Theorem 1 (Certified Perturbation Size).** Let \( f \) be any base node/graph classifier. The random noise vector \( \epsilon \) is defined in Equation 1. The smoothed classifier \( g \) is defined in Equation 2. Given a structure vector \( s \in \{0, 1\}^n \), suppose there exist \( c_A \) and \( p_A, p_B \in [0, 1] \) that satisfy Equation 4. Then, we have

\[
g(s \oplus \delta) = c_A, \forall ||\delta||_0 \leq K,
\]

where the certified perturbation size \( K \) is the solution to the following optimization problem:

\[
K = \arg\max_k \left( \forall ||\delta||_0 = k \right)
\]

\[
s.t. \Pr(Y \in A) > \Pr(Y \in B), \forall ||\delta||_0 = k.
\]

**Proof.** See Section 7.2.

Next, we show that our certified perturbation size is tight in the following theorem.

**Theorem 2 (Tightness of the Certified Perturbation Size).** Assume \( p_A \geq p_B, p_A + p_B \leq 1 \), and \( p_A + (|C| - 1) \cdot p_B \geq 1 \), where \( |C| \) is the number of labels. For any perturbation \( \delta \) with \( ||\delta||_0 > K \), there exists a base classifier \( g^* \) consistent with Equation 4 such that \( g(s \oplus \delta) \neq c_A \) or there exist ties.

**Proof.** See Section 7.3.
We have several observations on our major theorems.

- Our theorems are applicable to any base node/graph classifier. Moreover, although we focus on classifiers on graphs, our theorems are applicable to any base classifier that takes binary features as input.
- Our certified perturbation size $K$ depends on $p_A$, $p_B$, and $\delta$.
  In particular, when the probability bounds $p_A$ and $p_B$ are tighter, our certified perturbation size $K$ is larger. We use the probability bounds $p_A$ and $p_B$ instead of their exact values, because it is challenging to compute $p_A$ and $p_B$ exactly.
- When no assumptions on the base classifier are made and randomized smoothing with the noise distribution defined in Equation 1 is used, it is impossible to certify a perturbation size larger than $K$.

3.3 Certification in Practice

Computing our certified perturbation size $K$ in practice faces two challenges. The first challenge is to estimate the probability bounds $p_A$ and $p_B$. The second challenge is to solve the optimization problem in Equation 11 to get $K$ with the given $p_A$ and $p_B$. We leverage the method developed in [21] to address the first challenge, and we develop an efficient algorithm to address the second challenge.

Estimating $p_A$ and $p_B$: We view the probabilities $p_1, p_2, \cdots, p_{|C|}$ as a multinomial distribution over the label set $C$. If we sample a noise $\epsilon$ from our noise distribution uniformly at random, then $f(s \oplus \epsilon)$ can be viewed as a sample from the multinomial distribution. Then, estimating $p_A$ and $p_C$ for $c \in C \setminus \{c_A\}$ is essentially a one-sided simultaneous confidence interval estimation problem. In particular, we leverage the simultaneous confidence interval estimation method developed in [21] to estimate these bounds with a confidence level at least $1 - \alpha$. Specifically, we sample a random noise, i.e., $\eps_1, \eps_2, \cdots, \eps_d$, from the noise distribution defined in Equation 1. We denote by $d_c$ the frequency of the label $c$ predicted by the base classifier for the $d$ noisy examples. Formally, we have $d_c = \sum_{j=1}^d I(f(s \oplus \eps_j) = c)$ for each $c \in C$ and $I$ is the indicator function. Moreover, we assume $c_A$ has the largest frequency and the smoothed classifier predicts $c_A$ as the label, i.e., $g(s) = c_A$. According to [21], we have the following probability bounds with a confidence level at least $1 - \alpha$:

$$p_A = B\left(\frac{\alpha}{|C|}; d_{c_A}, d - d_{c_A} + 1\right)$$
$$p_C = B\left(1 - \frac{\alpha}{|C|}; d_c + 1, d - d_c\right), \forall c \neq c_A,$$

where $B(q; u, w)$ is the $q$th quantile of a beta distribution with shape parameters $u$ and $w$. Then, we estimate $p_B$ as follows:

$$p_B = \min\left\{\max_{c \neq c_A} p_C, 1 - p_A\right\}.$$

Computing $K$: After estimating $p_A$ and $p_B$, we solve the optimization problem in Equation 11 to obtain $K$. First, we have:

$$\Pr(Y \in \mathcal{A}) = \sum_{j=1}^{a^* - 1} \Pr(Y \in \mathcal{R}_j) + \left(p_a - \sum_{j=1}^{a^* - 1} \Pr(X \in \mathcal{R}_j)\right)/r_a^*,$$

Table 1: Dataset statistics.

| Dataset      | #Nodes | #Edges | #Classes |
|--------------|--------|--------|----------|
| Node Classification |        |        |          |
| Cora         | 2,708  | 5,429  | 7        |
| Citeseer     | 3,327  | 4,732  | 6        |
| Pubmed       | 19,717 | 44,338 | 3        |
| Graph Classification |        |        |          |
| MUTAG        | 188    | 17.9   | 2        |
| PROTEINS     | 1,113  | 39.1   | 2        |
| IMDB         | 1,500  | 13.0   | 3        |

$$\Pr(Y \in \mathcal{B}) = \sum_{j=b^*+1}^{2n+1} \Pr(Y \in \mathcal{R}_j) + \left(p_B - \sum_{j=b^*+1}^{2n+1} \Pr(X \in \mathcal{R}_j)\right)/r_b^*,$$

where $r_a^*$ and $r_b^*$ are the density ratios in the regions $\mathcal{R}_a^*$ and $\mathcal{R}_b^*$, respectively. Therefore, the key to solve the optimization problem is to compute $\Pr(X \in \mathcal{R}(m))$ and $\Pr(Y \in \mathcal{R}(m))$ for each $m \in \{-n, -n+1, \cdots, n\}$ when $||\delta||_0 = k$. Specifically, we have:

$$\Pr(X \in \mathcal{R}(m)) = \sum_{j=\max\{0,m\}}^{\min\{n,n+m\}} \beta^n(j-m)(1-\beta)^j \cdot t(m, j),$$

$$\Pr(Y \in \mathcal{R}(m)) = \sum_{j=\max\{0,m\}}^{\min\{n,n+m\}} \beta^n(j-\beta)^j \cdot t(m, j),$$

where $t(m, j)$ is defined as follows:

$$t(m, j) = \begin{cases} 0, & \text{if } (m+k) \mod 2 \neq 0, \\ 0, & \text{if } 2j - m < k, \\ \frac{n-k}{n-j} \left(\frac{k}{2}\right)^j, & \text{otherwise}. \end{cases}$$

See Section 7.4 for the details on obtaining Equation 18 and 19. Then, we iteratively find the largest $k$ such that the constraint in Equation 12 is satisfied.

4 EVALUATION

4.1 Experimental Setup

We evaluate our method on multiple GNNs and benchmark datasets for both node classification and graph classification.

**Benchmark datasets and GNNs:** We use benchmark graphs and GNNs for both node and graph classification. Table 1 shows the statistics of our graphs.

- **Node classification:** We consider Graph Convolutional Network (GCN) [26] and Graph Attention Network (GAT) [49] for node classification. Moreover, we use the Cora, Citeseer, and Pubmed datasets [43]. They are citation graphs, where nodes are documents and edges indicate citations between documents. In particular, an undirected edge between two documents is created if one document cites the other. The bag-of-words feature of a document is treated as the node feature. Each document also has a label.
The figure shows the impact of \( \beta \) on the certified accuracy of the smoothed GCN, GAT, and GIN. Each graph represents a different dataset: (a) Cora, (b) Citeseer, and (c) Pubmed. The x-axis represents the perturbation size, while the y-axis shows the certified accuracy. The plots illustrate how the certified accuracy changes as the perturbation size varies, with different \( \beta \) values.

- **Graph classification**: We consider Graph Isomorphism Network (GIN) [61] for graph classification. Moreover, we use the MUTAG, PROTEINS, and IMDB datasets [63]. MUTAG and PROTEINS are bioinformatics datasets. MUTAG contains 188 mutagenic aromatic and heteroaromatic nitro compounds, where each compound represents a graph and each label means whether or not the compound has a mutagenic effect on the Gramnegative bacterium *Salmonella typhimurium*. PROTEINS is a dataset where nodes are secondary structure elements and there is an edge between two nodes if they are neighbors in the amino-acid sequence or in three-dimensional space. Each protein is represented as a graph and is labeled as enzyme or non-enzyme. IMDB is a movie collaboration dataset. Each graph is an ego-network of an actor/actress, where nodes are actors/actresses and an edge between two actors/actresses indicates that they appear in the same movie. Each graph is obtained from a certain genre of movies, and the task is to classify the genre of a graph.

- **Training and testing**: For each node classification dataset, following previous works [26, 49], we sample 20 nodes from each class uniformly at random as the training dataset. Moreover, we randomly sample 100 nodes for testing. For each graph classification dataset, we use 90% of the graphs for training and the remaining 10% for testing, similar to [61].

- **Parameter setting**: We implement our method in pyTorch. To compute the certified perturbation size, our method needs to specify the noise parameter \( \beta \), the confidence level \( 1 - \alpha \), and the number of samples \( d \). Unless otherwise mentioned, we set \( \beta = 0.7 \), \( 1 - \alpha = 99.9\% \), and \( d = 10,000 \). We also explore the impact of each parameter while fixing the other parameters to the default settings in our experiments. When computing the certified perturbation size for a node \( u \) in node classification, we consider an attacker perturbs the connection status between \( u \) and the remaining nodes in the graph. We use the publicly available source code for GCN, GAT, and GIN. We use our randomized smoothing to smooth each classifier.
4.2 Experimental Results

Like Cohen et al. [9], we use certified accuracy as the metric to evaluate our method. Specifically, for a smoothed GNN classifier and a given perturbation size, certified accuracy is the fraction of testing nodes (for node classification) or testing graphs (for graph classification), whose labels are correctly predicted by the smoothed classifier and whose certified perturbation size is no smaller than the given perturbation size.

**Impact of the noise parameter** $\beta$: Figure 1 and Figure 2 respectively show the certified accuracy of the smoothed GCN and smoothed GAT vs. perturbation size for different $\beta$ on the three node classification datasets. Figure 3 shows the certified accuracy of GIN with randomized smoothing vs. perturbation size for different $\beta$ on the three graph classification datasets.

We have two observations. First, when $\beta = 0.5$, the certified accuracy is independent with the perturbation size, which means that an attacker can not attack a smoothed GNN classifier via perturbing the graph structure. This is because $\beta = 0.5$ essentially means that the graph is sampled from the space $\{0, 1\}^d$ uniformly at random. In other words, the graph structure does not contain information about the node features. Second, $\beta$ controls a tradeoff between accuracy under no attacks and robustness. Specifically, when $\beta$ is larger, the accuracy under no attacks (i.e., perturbation size is 0) is larger, but the certified accuracy drops more quickly as the perturbation size increases.

**Impact of the confidence level** $1 - \alpha$: Figure 4a and 4b show the certified accuracy of the smoothed GCN and smoothed GIN vs. perturbation size for different confidence levels, respectively. We observe that as the confidence level $1 - \alpha$ increases, the certified accuracy curve becomes slightly lower. This is because a higher confidence level leads to a looser estimation of the probability bound $p_A$ and $p_B$, which means a smaller certified perturbation size for a testing node/graph. However, the differences of the certified accuracies between different confidence levels are negligible when the confidence levels are large enough.

**Impact of the number of samples** $d$: Figure 5a and 5b show the certified accuracy of the smoothed GCN and smoothed GIN vs. perturbation size for different numbers of samples $d$, respectively. We observe that as the number of samples increases, the certified accuracy curve becomes higher. This is because a larger number of samples makes the estimated probability bound $p_A$ and $p_B$ tighter, which means a larger certified perturbation size for a testing node/graph. On Cora, the smoothed GCN achieves certified accuracies of 0.53, 0.50, and 0.49 when $d = 50K$ and the attacker arbitrarily adds/deletes at most 5, 10, and 15 edges, respectively. On MUTAG, GIN with our randomized smoothing achieves certified accuracies of 0.45, 0.45, and 0.40 when $d = 50K$ and the attacker arbitrarily adds/deletes at most 5, 10, and 15 edges, respectively.

5 RELATED WORK

We review studies on certified robustness for classifiers on both non-graph data and graph data.

5.1 Non-graph Data

Various methods have been tried to certify robustness of classifiers. A classifier is certifiably robust if it predicts the same label for data points in a certain region around an input example. Existing certification methods leverage Satisfiability Modulo Theories [6, 13, 25, 42], mixed integer-linear programming [4, 8, 15], linear programming [57, 58], semidefinite programming [39, 40], dual optimization [11, 12], abstract interpretation [16, 34, 44], and layer-wise relaxation [56, 65]. However, these methods are not scalable to large neural networks and/or are only applicable to specific neural network architectures.

Randomized smoothing [5, 9, 21, 28, 29, 31–33, 41, 64] was originally developed to defend against adversarial examples. It was first proposed as an empirical defense [5, 33]. For instance, Cao and Gong [5] proposed to use uniform random noise from a hypercube centered at an input example to smooth a base classifier. Lecuyer et al. [28] derived a certified robustness guarantee for randomized smoothing with Gaussian noise or Laplacian noise via differential privacy techniques. Li et al. [32] leveraged information theory to derive a tighter certified robustness guarantee. Cohen et al. [9] leveraged the Neyman-Pearson Lemma [36] to obtain a tight certified robustness guarantee under $L_2$-norm for randomized smoothing with Gaussian noise. Specifically, they showed that a smoothed classifier verifiably predicts the same label when the $L_2$-norm of the adversarial perturbation to an input example is less than a threshold. Salman et al. [41] designed an adaptive attack against smoothed classifiers and used the attack to train classifiers in an adversarial training paradigm to enlarge the certified robustness.
under $L_2$-norm. Zhai et al. [64] proposed to explicitly maximize the certified robustness via a new training strategy. All these methods focused on top-1 predictions. Jia et al. [21] extended [9] to derive the first certification of top-$K$ predictions against adversarial examples. Compared to other methods for certified robustness, randomized smoothing has two key advantages: 1) it is scalable to large neural networks, and 2) it is applicable to any base classifier.

5.2 Graph Data

Several studies [1, 7, 10, 45, 50, 59, 60, 69, 70] have shown that attackers can fool GNNs via manipulating the node features and/or graph structure. [14, 47, 48, 59, 60, 67] proposed empirical defenses without certified robustness guarantees. A few recent works [2, 3, 24, 71] study certified robustness of GNNs. Zügner and Günnemann [71] considered node feature perturbation. In particular, they derived certified robustness for a particular type of GNN called graph convolutional network [26] against node feature perturbation. More specifically, they formulated the certified robustness as a linear programming problem, where the objective is to require that a node’s prediction is constant within an allowable node feature perturbation. Then, they derived the robustness guarantee based on the dual form of the optimization problem, which is motivated by [57]. [2, 24, 72] considered structural perturbation for a specific GNN. Specifically, [2] required that GNN prediction is a linear function of (personalized) PageRank [27] and [24, 72] could only certify the robustness of GNNs. For example, [24] derived the certified robustness of GCN for graph classification via utilizing dualization and convex envelope tools.

Two recent works [3, 22] leveraged randomized smoothing to provide a model-agnostic certification against structural perturbation. Jia et al. [22] certified the robustness of community detection against structural perturbation. They essentially model the problem as binary classification, i.e., whether a set of nodes are in the same community or not, and design a randomized smoothing technique to certify its robustness. Bojchevski et al. [3] generalize the randomized smoothing technique developed in [29] to the sparse setting and derive certified robustness for GNNs. They did not formally prove that their multi-class certificate is tight. However, in the special case of equal flip probabilities, which is equivalent to our certificate, our tightness proof (Theorem 2) applies.

6 CONCLUSION AND FUTURE WORK

In this work, we develop the certified robustness guarantee of GNNs for both node and graph classifications against structural perturbation. Our results are applicable to any GNN classifier. Our certification is based on randomized smoothing for binary data which we develop in this work. Moreover, we prove that our certified robustness guarantee is tight when randomized smoothing is used and no assumptions on the GNNs are made. An interesting future work is to incorporate the information of a given GNN to further improve the certified robustness guarantee.

7 PROOFS

7.1 Proof of Lemma 1

We first describe the Neyman-Pearson Lemma for binary random variables, which we use to prove Lemma 1.

Lemma 2 (Neyman-Pearson Lemma for Binary Random Variables). Let $X$ and $Y$ be two random variables in the discrete space $\{0,1\}^n$ with probability distributions $Pr(X)$ and $Pr(Y)$, respectively. Let $h: \{0,1\}^n \to \{0,1\}$ be a random or deterministic function.

- Let $S_1 = \{z \in \{0,1\}^n : Pr(X = z) > r\}$ and $S_2 = \{z \in \{0,1\}^n : Pr(X = z) = r\}$ for some $r > 0$. Assume $S_2 \subseteq S_1$ and $S = S_1 \cup S_2$. If $Pr(h(X) = 1) \geq Pr(X \in S)$, then $Pr(h(Y) = 1) \geq Pr(Y \in S)$.

- Let $S_1 = \{z \in \{0,1\}^n : Pr(X = z) < r\}$ and $S_2 = \{z \in \{0,1\}^n : Pr(X = z) = r\}$ for some $r > 0$. Assume $S_1 \subseteq S_2$ and $S = S_1 \cup S_2$. If $Pr(h(X) = 1) \leq Pr(X \in S)$, then $Pr(h(Y) = 1) \leq Pr(Y \in S)$.

Proof. The proof can be found in a standard statistics textbook, e.g., [30]. For completeness, we include the proof here. Without loss of generality, we assume that $h$ is random and denote $h(0|z)$ (resp. $h(1|z)$) as the probability that $h(z) = 1$ (resp. $h(z) = 0$). We denote $S^c$ as the complement of $S$, i.e., $S^c = \{0,1\}^n \setminus S$. For any $z \in S$, we have $Pr(X = z) \geq r$, and for any $z \in S^c$, we have $Pr(X = z) \leq r$. We prove the first part, and the second part can be proved similarly.

\[
Pr(h(Y) = 1) - Pr(Y \in S) = 
\sum_{z \in \{0,1\}^n} h(1|z)Pr(Y = z) - \sum_{z \in S} Pr(Y = z)
= \left( \sum_{z \in S^c} h(1|z)Pr(Y = z) + \sum_{z \in S} h(1|z)Pr(Y = z) \right)
- \left( \sum_{z \in S^c} h(0|z)Pr(Y = z) + \sum_{z \in S} h(0|z)Pr(Y = z) \right)
= \sum_{z \in S} h(1|z)Pr(Y = z) - \sum_{z \in S} h(0|z)Pr(Y = z)
\geq \frac{1}{r} \left( \sum_{z \in S} h(1|z)Pr(X = z) - \sum_{z \in S^c} h(0|z)Pr(X = z) \right)
= \frac{1}{r} \left( \sum_{z \in S} h(1|z)Pr(X = z) + \sum_{z \in S} h(1|z)Pr(X = z) \right)
- \left( \sum_{z \in S} h(1|z)Pr(X = z) + \sum_{z \in S^c} h(0|z)Pr(X = z) \right)
= \frac{1}{r} \left( \sum_{z \in \{0,1\}^n} h(1|z)Pr(X = z) - \sum_{z \in S} Pr(X = z) \right)
\geq 0.
\]

Next, we restate Lemma 1 and show our proof.

Lemma 1. We have the following bounds:

\[
Pr(f(Y) = c_A) \geq Pr(Y \in \mathcal{A}),
\]

\[
Pr(f(Y) = c_B) \leq Pr(Y \in \mathcal{B}).
\]

Proof. Based on the regions $\mathcal{A}$ and $\mathcal{B}$ defined in Equation 5 and Equation 6, we have $Pr(X \in \mathcal{A}) = p_A$ and $Pr(X \in \mathcal{B}) = p_B$. 
Moreover, based on the conditions in Equation 4, we have:

\[ \Pr(f(X) = c_A) \geq p_A = \Pr(X \in \mathcal{A}); \]

\[ \Pr(f(X) = c_B) \leq p_B = \Pr(X \in \mathcal{B}). \]

Next, we leverage Lemma 2 to derive the condition for \( \Pr(f(Y) = c_A) > \Pr(f(Y) = c_B) \). Specifically, we define \( h(z) = I(f(z) = c_A) \). Then, we have:

\[ \Pr(h(X) = 1) = \Pr(f(X) = c_A) \geq \Pr(X \in \mathcal{A}). \]

Moreover, we have \( \frac{\Pr(X=x)}{\Pr(Y=x)} > r_{A^*} \) for any \( z \in \bigcup_{j=1}^{n-1} \mathcal{R}_j \) and \( \frac{\Pr(X=x)}{\Pr(Y=x)} = r_{A^*} \) for any \( z \in \mathcal{R}_{A^*} \), where \( r_{A^*} \) is the probability density ratio in the region \( \mathcal{R}_{A^*} \). Therefore, according to the first part of Lemma 2, we have \( \Pr(f(Y) = c_A) \geq \Pr(Y \in \mathcal{A}) \). Similarly, based on the second part of Lemma 2, we have \( \Pr(f(Y) = c_B) \leq \Pr(Y \in \mathcal{B}). \)

\[ \blacksquare \]

7.2 Proof of Theorem 1

Recall that our goal is to make \( \Pr(f(Y) = c_A) > \Pr(f(Y) = c_B) \). Based on Lemma 1, it is sufficient to require \( \Pr(Y \in \mathcal{A}) > \Pr(Y \in \mathcal{B}) \). Therefore, we derive the certified perturbation size \( K \) as the maximum \( k \) such that the above inequality holds for \( \forall \|\delta\|_0 = R \).

7.3 Proof of Theorem 2

Our idea is to construct a base classifier \( f^* \) consistent with the conditions in Equation 4, but the smoothed classifier is not guaranteed to predict \( c_A \). Let disjoint regions \( \mathcal{A} \) and \( \mathcal{B} \) be defined as in Equations 5 and 6. We denote \( \Gamma = \{1, 2, \cdots, c\} \setminus \{c_A, c_B\} \). Then, for each label \( c_j \) in \( \Gamma \), we can find a region \( C_j \subseteq \{0, 1\}^n \setminus (\mathcal{A} \cup \mathcal{B}) \) such that \( C_j \cap C_j = \emptyset \), \( j \in \Gamma \), \( i \neq j \), and \( \Pr(X \in C_j) = \frac{1}{2} \). We can construct such regions because \( p_A + (|\mathcal{C}|-1) \cdot p_B \geq 1 \). Given those regions, we construct the following base classifier:

\[ f^*(x) = \begin{cases} 
    c_A & \text{if } z \in \mathcal{A} \\
    c_B & \text{if } z \in \mathcal{B} \\
    c_i & \text{if } z \in C_i
\end{cases} \]

By construction, we have \( \Pr(f^*(X) = c_A) = p_A \), \( \Pr(f^*(X) = c_B) = p_B \), and \( \Pr(f^*(X) = c_i) \leq p_B \) for any \( c_j \in \Gamma \), which are consistent with Equation 4. From our proof of Theorem 1, we know that when \( \|\delta\|_0 > K \), we have:

\[ \Pr(Y \in \mathcal{A}) \leq \Pr(Y \in \mathcal{B}), \]

or equivalently we have:

\[ \Pr(f^*(Y) = c_A) \leq \Pr(f^*(Y) = c_B) \]

Therefore, we have either \( g(s \oplus \delta) \neq c_A \) or there exist ties when \( \|\delta\|_0 > K \).

7.4 Computing \( \Pr(X \in \mathcal{R}(m)) \) and \( \Pr(Y \in \mathcal{R}(m)) \)

We first define the following regions:

\[ \mathcal{R}(w, v) = \{z \in \{0, 1\}^n : |z - s|_0 = w \text{ and } |z - s \oplus \delta|_0 = v \}, \]

for \( w, v \in \{0, 1, \cdots, n\} \). Intuitively, \( \mathcal{R}(w, v) \) includes the binary vectors that are \( w \) bits different from \( s \) and \( v \) bits different from \( s \oplus \delta \). Next, we compute the size of the region \( \mathcal{R}(w, v) \) when \( \|\delta\|_0 = k \). Without loss of generality, we assume \( s = \{0, 0, \cdots, 0\} \) as a zero vector and \( s \oplus \delta = \{1, 1, \cdots, 1, 0, 0, \cdots, 0\} \), where the first \( k \) entries are 1 and the remaining \( n - k \) entries are 0. We construct a binary vector \( z \in \mathcal{R}(w, v) \). Specifically, suppose we flip \( i \) zeros in the last \( n - k \) zeros in both \( s \) and \( s \oplus \delta \). Then, we flip \( w - i \) of the first \( k \) bits of \( s \) and flip the rest \( k - w + i \) bits of the first \( k \) bits of \( s \oplus \delta \). In order to have \( z \in \mathcal{R}(w, v) \), we need \( k - w + i + v = i \), i.e., \( i = (w + v - k)/2 \). Therefore, we have the size \( |\mathcal{R}(w, v)| \) of the region \( \mathcal{R}(w, v) \) as follows:

\[ |\mathcal{R}(w, v)| = \begin{cases} 
    0, & \text{if } (w + v - k) \mod 2 \neq 0, \\
    n - k, & \text{if } w = v < k, \\
    \binom{k}{w - v}, & \text{otherwise}
\end{cases} \]

Moreover, for each \( z \in \mathcal{R}(w, v) \), we have \( \Pr(X = z) = \beta^{n-w}(1-\beta)^w \) and \( \Pr(Y = z) = \beta^{n-v}(1-\beta)^v \). Therefore, we have:

\[ \Pr(X \in \mathcal{R}(w, v)) = \beta^{n-w}(1-\beta)^w \cdot |\mathcal{R}(w, v)|, \]

\[ \Pr(Y \in \mathcal{R}(w, v)) = \beta^{n-v}(1-\beta)^v \cdot |\mathcal{R}(w, v)|. \]

Note that \( \mathcal{R}(m) = \bigcup_{w=v=m} \mathcal{R}(w, v) \). Therefore, we have:

\[ \Pr(X \in \mathcal{R}(m)) = \Pr(X \in \bigcup_{w=v=m} \mathcal{R}(w, v)) = \sum_{j=\max(m,0)}^{\min(n,n+m)} \Pr(X \in \mathcal{R}(j-m, j)) \]

\[ = \sum_{j=\max(m,0)}^{\min(n,n+m)} \beta^{n-(j-m)}(1-\beta)^{j-m} \cdot |\mathcal{R}(j-m, j)| \]

\[ = \sum_{j=\max(m,0)}^{\min(n,n+m)} \beta^{n-(j-m)}(1-\beta)^{j-m} \cdot t(m, j), \]

where \( t(m, j) = |\mathcal{R}(j-m, j)| \).

Similarly, we have:

\[ \Pr(Y \in \mathcal{R}(m)) = \sum_{j=\max(m,0)}^{\min(n,n+m)} \beta^{n-j}(1-\beta)^{j} \cdot t(m, j). \]

Acknowledgements. We thank the anonymous reviewers for their constructive comments. This work is supported by the National Science Foundation under Grants No. 1937786 and 1937787 and the Army Research Office under Grant No. W911NF2110182. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In NeurIPS.
[2] Aleksandar Bojchevski and Stephan Günnemann. 2019. Certifiable Robustness to Graph Perturbations. In NeurIPS.
[3] Aleksandar Bojchevski, Johannes Klicpera, and Stephan Günnemann. 2020. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In ICMRL.
[4] Rudy R Bunel, Ilker Turksan, Philip Torr, Pushmeet Kohli, and Pawan K. Madhavendra. 2018. A unified view of piecewise linear neural network verification. In NeurIPS.
[5] Xiaoyu Cao and Neil Zhongqiang Gong. 2017. Mitigating evasion attacks to deep neural networks via region-based classification. In ACSAC.
[6] Nicholas Carlini, Guy Katz, Clark Barrett, and David L. Dill. 2017. Provably minimally-distorted adversarial examples. arXiv (2017).
