AGNNCert: Defending Graph Neural Networks against Arbitrary Perturbations with Deterministic Certification

Jiate Li[1], Binghui Wang[1,*]
[1]*Illinois Institute of Technology,* [*]*Corresponding Author*

Abstract

Graph neural networks (GNNs) achieve the state-of-the-art on graph-relevant tasks such as node and graph classification. However, recent works show GNNs are vulnerable to adversarial perturbations include the perturbation on edges, nodes, and node features, the three components that form a graph. Empirical defenses against such attacks are soon broken by adaptive ones. While certified defenses offer robustness guarantees, they face several limitations: 1) almost all restrict the adversary's capability to only one type of perturbation, which is impractical; 2) all are designed for a particular GNN task, which limits their applicability; and 3) the robustness guarantees of all methods except one are not 100% accurate.

We address all these limitations by developing AGNNCert, the first certified defense for GNNs against arbitrary (edge, node, and node feature) perturbations with deterministic robustness guarantees, and applicable to the two most common node and graph classification tasks. AGNNCert also encompass existing certified defenses as special cases. Extensive evaluations on multiple benchmark node/graph classification datasets and two real-world graph datasets, and multiple GNNs validate the effectiveness of AGNNCert to provably defend against arbitrary perturbations. AGNNCert also shows its superiority over the state-of-the-art certified defenses against the individual edge perturbation and node perturbation.[1]

1 Introduction

Graph is a natural representation for many real-world data, such as social networks, biological networks, and financial networks. In recent years, there has been a great surge of research interest on graph neural networks (GNNs) [20, 29, 46, 50, 67] for representation learning on graphs, in which each node recursively aggregates representations of its neighbors to update its representation. The learnt representations can be used for various graph-relevant tasks, e.g., node classification [29, 67] and graph classification [18, 20]. For instance, in node classification, GNNs learn a node classifier to predict the label for each node, and learn a graph classifier to predict the label for an entire graph in graph classification. GNNs have achieved outstanding performance on these tasks for various computer security applications, including fraud detection (e.g., detecting fake accounts/users and fake news in social networks [17, 53, 59, 60, 68], fake reviewers and reviews in recommendation systems [13, 55], fraud transactions in e-commerce systems [71], and credit card fraud and money laundering in finance systems [4,61]), intrusion detection [77], and software vulnerability detection [5, 7, 73, 78].

In GNNs, a graph is often represented as three components: nodes, their features, and edges that connect the nodes. However, various works [3, 9, 27, 39, 42, 48, 51, 52, 56–58, 62, 66, 72, 75, 80] have shown that GNNs are vulnerable to *test-time* adversarial attacks, where an adversary can successfully perform the attack by perturbing any individual component or their combinations in the graph. Specifically, given a node/graph classifier and a graph, an attacker could inject a few nodes [27, 48], slightly modify the edges [52, 66, 80] on the graph[2], and/or perturb features of certain nodes [80] such that the classifier makes wrong predictions for the target node/graph. Taking GNN based fake user detection in social networks (e.g., Twitter) as an example. In this context, nodes represent users, edges denote following-follower relationships, and node features capture user profile information. A strategic attacker (i.e., fake users) can manipulate their profiles, modify their connections with other users, and create new fake accounts and connections to evade detection [53].

To mitigate the attacks, two lines of defenses have been proposed. Empirical defenses [14, 49, 62, 66, 76, 79] are developed with heuristic strategies, but were later broken by adaptive/stronger attacks [43]. Certified defenses [1, 25, 26, 30, 54, 64] address the issue by offering robustness guarantees against the worst-case attack scenario. For instance, BiRS [30] achieves the state-of-the-art certified defense perfor-

[1]Source code and the full version are available at: https://github.com/JetRichardLee/AGNNCert.

[2]Edge features are typically incorporated in the edge matrix, whose perturbation can be viewed as a special case of edge perturbation.

mance against the node injection attack, while GNNCert [64] achieves the state-of-the-art against the edge or/and node feature perturbation attack. However, all existing certified defenses face several fundamental limitations shown below (See Table 1 a comprehensive summary).

1. They all restrict the adversary's capability to only *one* type of perturbation, except [64] to edge and node feature perturbation. In practice, however, an attacker could simultaneously manipulate nodes, node features, as well as edges to perform the best-possible attack.
2. They are designed for a particular GNN task, e.g., node classification or graph classification. This naturally limits the applicability of these defenses.
3. Their robustness guarantees are probabilistic (i.e., not 100%), with the exception of [64]. This implies the guarantees could be inaccurate with a certain probability.

Our work: We develop a voting-based defense, called `AGNNCert`, to address all the above limitations. `AGNNCert` is the *first certified defense* for GNNs on the two most common *node and graph classification tasks* against *arbitrary perturbations* with *deterministic* robustness guarantees. Here, an arbitrary perturbation is the perturbation that can arbitrarily manipulate the nodes (i.e., inject new nodes and delete existing nodes), edges (i.e., inject new edges and delete existing edges), and node features on a graph. More specifically, `AGNNCert` can provably predict the same label for a test node/graph with arbitrary perturbation whose perturbation size (i.e., the total number of manipulated nodes, nodes with feature perturbations, and edges) is bounded by a threshold, which we call the *certified perturbation size*.

Generally, given a graph and a GNN node/graph classifier, our voting-based defense includes three steps:

- **Step I: Divide a graph into multiple subgraphs.** We use a hash function [64] to deterministically divide the given graph into multiple subgraphs.
- **Step II: Build a voting node/graph classifier on the subgraphs.** We use the node/graph classifier to predict the label of subgraphs, where each prediction is treated as a vote. We then count the votes for each label, and build a voting classifier that returns the label with the most votes.
- **Step III: Derive the deterministic robustness guarantee.** We derive the certified perturbation size for the voting classifier against arbitrary perturbations on the given graph with deterministic certification.

Under this setup, we first derive the sufficient condition for certified robustness against arbitrary perturbations on GNNs—the number of different predictions on subgraphs generated from the given graph and from the arbitrarily perturbed graph should be bounded (Theorem 1). We then propose two graph division strategies, one is *edge-centric* and the other is *node-centric*, to obtain the upper bounded altered predictions.

Edge-centric graph division: This strategy is inspired by [64], in which we use a hash function to map *edges* from the given graph into multiple subgraphs *such that the edges are disjoint in any two subgraphs* (**Step I**). With it, we show that manipulating any edge in the given graph (via edge injection or deletion) only perturbs one subgraph and hence at most one subgraph prediction is altered (Theorem 2). Further, by leveraging the underlying message-passing mechanism in GNNs and with careful analysis, we prove the generated subgraphs can also bound the different subgraph predictions caused by the node manipulation (Theorem 3) and node feature manipulation (Theorem 4). Together, these theorems ensure the number of subgraph predictions be altered for any node/graph after arbitrary perturbation is bounded (Theorem 5). Further, based on the voting classifier in **Step II** and Theorem 1, we derive in Theorem 6 the certified perturbation size (**Step III**).

Node-centric graph division: The theoretical result under edge-centric graph division reveals the robustness guarantee is largely dominated by the number of edges induced by the manipulated nodes and node features, which could be ineffective in practice. For instance, injected nodes could produce many edges by linking with many nodes in the graph to exceed the certified perturbation size. To mitigate the issue, we propose a *node-centric* graph division method. Our key idea is that if we can ensure all edges of a manipulated node is in a same subgraph, this subgraph is the only one being affected under every node or node feature manipulation. However, naive solutions are ineffective. For instance, we can map nodes into different subgraphs such that they are *non-overlapped*, but it fails for node classification, as every node only appears once in all subgraphs and all target nodes for classification only receive one vote, yielding vacuous robustness.

To address it, we innovatively treat every undirected edge as two directed edges and map each node into a subgraph index only using its outgoing edges (**Step I**). In doing so, all subgraphs are directed and only contain outgoing edges of the nodes with the corresponding index. By leveraging these *directed subgraphs* and the message-passing mechanism in GNNs, we can derive the same bounded number of altered subgraph predictions against edge manipulation (Theorem 7) as in edge-centric graph division. Moreover, this bound against arbitrary node or node feature manipulation is the number of injected/deleted nodes (Theorem 8) or number of nodes whose features can be arbitrarily perturbed (Theorem 9). *This implies the bound is robust to the manipulated node that links with many even infinite number of edges.* Combining them, we derive the total bounded number of altered subgraph predictions against arbitrary perturbation in Theorem 10, and the certified perturbation size in Theorem 11 (**Step III**) with the built voting classifier on the directed subgraphs (**Step II**).

Evaluation: We extensively evaluate `AGNNCert` on multiple graph datasets and multiple node and graph classifiers against arbitrary perturbations. We use the certified node/graph accuracy at perturbation size m as the evaluation metric, which means the fraction of test nodes/graphs that are provably classified as the true label against arbitrary perturbations whose

perturbation size is m. Our results show that: 1) Under edge-centric graph division, AGNNCert can obtain about 70% (or 60%) certified node (or graph) accuracy when the perturbation size is 200 (or 10), i.e., 200 (or 10) edges induced by the edge manipulation, injected/deleted edges associated with the node manipulation, and edges associated with node feature manipulation are arbitrarily perturbed; 2) Under node-centric graph division, AGNNCert can obtain similar certified node (or graph) accuracy when the total number of 200 (or 10) edges and nodes induced by edge, node, and node feature manipulations are arbitrarily perturbed, where the manipulated nodes allow to have infinite number of edges.

As AGNNCert can also defend against fewer manipulations, we further compare it with the state-of-the-art certified defenses of GNNs for node classification against node injection attack [30], and for graph classification against node feature or/and edge manipulation [64]. Our results show AGNNCert significantly outperforms [30] under node-centric graph division, and outperforms [64] under both graph division methods.

We also evaluate AGNNCert on two real-world graph datasets (Amazon co-purchasing dataset [6] with 2M nodes and 51M edges and Big-Vul code vulnerability dataset [15] with 10,900 vulnerable C++ codes) to demonstrate its scalability and practicability. Our results show AGNNCert obtains promising robustness guarantees with an acceptable computational overhead over the undefended GNNs.

Contributions: Our contributions are summarized below:

- We develop the first certified defense to robustify GNNs for node and graph classification against arbitrary perturbations on individual graphs.
- We propose two strategies to realize our defense that leverages the unique message-passing mechanism in GNNs.
- Our robustness guarantee is accurate with probability 1.
- Our defense treat existing certified defenses as special cases, as well as significantly outperforming them.

2 Background and Problem Definition

2.1 Graph Neural Network (GNN)

Let a graph be $G = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, which consists of the nodes $\mathcal{V}$, node features $\mathbf{X}$, and edges $\mathcal{E}$. We denote $u \in \mathcal{V}$ as a node, $e = (u,v) \in \mathcal{E}$ as an edge, and $\mathbf{X}_u$ as node u's feature.

GNNs learn representations for graph data by following the *message passing* strategy with two operations, i.e., the aggregate operation Agg and combine operation Comb. Specifically, Agg iteratively aggregates the representations of all neighbors of a node, while Comb updates the node's representation by combining it with the aggregated neighbors' representations. The two operations are formally defined below:

$$\boldsymbol{l}_v^{(k)} = \texttt{Agg}\big(\{\boldsymbol{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}\big), \boldsymbol{h}_v^{(k)} = \texttt{Comb}\big(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{l}_v^{(k)}\big), \quad (1)$$

where $\boldsymbol{h}_v^{(k)}$ denotes node v's representation in the k-th layer and $\boldsymbol{h}_v^{(0)} = \mathbf{X}_v$. $\mathcal{N}(v)$ denotes the neighbors of v.

Different GNNs use different aggregate and combine operations. For example, in Graph Convolutional Network (GCN) [29], the two operations are integrated as follows:

$$\boldsymbol{h}_v^{(k)} = \texttt{ReLU}\big(\boldsymbol{W}^{(k)} \cdot \texttt{Mean}\{\boldsymbol{h}_u^{(k-1)} : u \in \mathcal{N}(v) \bigcup \boldsymbol{h}_v^{(k-1)}\}\big), \quad (2)$$

where the element-wise mean pooling function Mean acts as the aggregate operation and ReLU the combine operation. $\Theta = \{\boldsymbol{W}^{(1)}, \cdots, \boldsymbol{W}^{(K)}\}$ are all the learned parameters.

A node v's final representation $\boldsymbol{h}_v^{(K)}$ captures structural information within v's K-hop neighbors, which are used for many tasks. In this paper we focus on the two classic classification tasks on graphs: node classification and graph classification. We denote f as the GNN node or graph classifier and $\mathcal{Y}$ as the set of all labels.

Node classification: f takes a graph G as input and predicts each node $v \in G$ a label $y_v \in \mathcal{Y}$ based on v's learnt representation $\boldsymbol{h}_v^{(K)}$. That is, $y_v = f(G)_v = \texttt{softmax}(\boldsymbol{h}_v^{(K)})$.

Graph classification: f takes a graph G as input and predicts a label $y_G \in \mathcal{Y}$ for the whole graph G by using all nodes' representations $\{\boldsymbol{h}_v^{(K)}\}_{v \in G}$. For instance, when averaging all nodes' final representations, we have $y_G = f(G) = \texttt{softmax}(\texttt{Avg}(\{\boldsymbol{h}_v^{(K)}\}_{v \in G}))$.

2.2 Adversarial Attacks on GNNs

In adversarial attacks against GNNs, an attacker can manipulate a graph $G = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ into a perturbed one $G' = \{\mathcal{V}', \mathcal{E}', \mathbf{X}'\}$, where $\mathcal{V}'$, $\mathcal{E}'$, $\mathbf{X}'$ are the perturbed version of $\mathcal{V}$, $\mathcal{E}$, and $\mathbf{X}$, respectively.

Edge manipulation: The attacker can 1) *inject new edges* $\mathcal{E}_+$, and 2) *delete existing edges*, denoted as $\mathcal{E}_- \subset \mathcal{E}$ from G.

Node manipulation: The attacker perturbs G by (1) *injecting new nodes* $\mathcal{V}_+$, whose node feature denoted as $\mathbf{X}'_{\mathcal{V}_+}$ can be arbitrary, together with the arbitrarily injected new edges $\mathcal{E}_{\mathcal{V}_+} \subseteq \{(u,v) \notin \mathcal{E}, \forall u \in \mathcal{V}_+ \vee v \in \mathcal{V}_+\}$ induced by $\mathcal{V}_+$; and (2) *deleting existing nodes* $\mathcal{V}_- \subset \mathcal{V}$. When $\mathcal{V}_-$ are deleted, their features $\mathbf{X}_{\mathcal{V}_-} \subseteq \mathbf{X}$ and all connected edges $\mathcal{E}_{\mathcal{V}_-} = \{(u,v) \in \mathcal{E}, \forall u \in \mathcal{V}_- \vee v \in \mathcal{V}_-\}$ are also removed.

Node feature manipulation: The attacker arbitrarily manipulates features $\mathbf{X}_{\mathcal{V}_r}$ of a set of representative nodes $\mathcal{V}_r$ to be $\mathbf{X}'_{\mathcal{V}_r}$. We also denote the edges connected with nodes $\mathcal{V}_r$ as $\mathcal{E}_{\mathcal{V}_r} = \{(u,v) \in \mathcal{E} : \forall u \in \mathcal{V}_r \vee v \in \mathcal{V}_r\}$.

Arbitrary manipulation: The attacker can manipulate the graph G with an arbitrary combined perturbations on edges, nodes, and node features.

For description simplicity, we will use $\{\mathcal{E}_+, \mathcal{E}_-\}$ to indicate the edge manipulation with arbitrary injected edges $\mathcal{E}_+$ and deleted edges $\mathcal{E}_-$ on G. Similarly, we will use $\{\mathcal{V}_+, \mathcal{E}_{\mathcal{V}_+}, \mathbf{X}'_{\mathcal{V}_+}, \mathcal{V}_-, \mathcal{E}_{\mathcal{V}_-}\}$ to indicate the node manipulation, and $\{\mathcal{V}_r, \mathcal{E}_{\mathcal{V}_r}, \mathbf{X}'_{\mathcal{V}_r}\}$ the node feature manipulation. Any combination of the manipulations is inherently well-defined.

GNN Task	Node Classification						Graph Classification						Certification Type
Attack Type	$\mathcal{E}_{\pm}$	**X**!	$\mathcal{V}_{\pm}$	$\mathcal{E}_{\pm}$&**X**!	$\mathcal{V}_{\pm}$&**X**!	Arbitrary	$\mathcal{E}_{\pm}$	**X**!	$\mathcal{V}_{\pm}$	$\mathcal{E}_{\pm}$&**X**!	$\mathcal{V}_{\pm}$&**X**!	Arbitrary	
RS [54]	✓	✓	×	×	×	×	✓	✓	×	×	×	×	**Probabilistic**
Sparsity-Aware RS [1]	✓	✓	✓	◯	×	×	✓	✓	✓	×	×	×	
Node-Aware Bi-RS [30]	×	×	✓	×	×	×	×	×	×	×	×	×	
GNNCert [64]	◯	◯	×	◯	×	×	✓	✓	×	✓	×	×	**Deterministic**
AGNNCert-E (Ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
AGNNCert-N (Ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

Table 1: Summarizing the existing certified defenses of GNN against adversarial perturbations and their capability against different types of manipulations. $\mathcal{E}_{\pm}$, $\mathcal{V}_{\pm}$, and **X**! represent the edge manipulation (injection/deletion), node manipulation (injection/deletion), and node feature perturbation, respectively. ✓ means the defense is able to defend the respective attack, ◯ means the defense could be adapted to defend the attack, and × means not able to.

2.3 Voting based Certified Defense

Voting-based GNNCert [64] has achieved state-of-the-art certified defense performance against node feature and edge manipulation. Here we review [64] since our defense is also based on voting. GNNCert is only applicable for graph classification and consists of three steps.

Step I: divide a graph into multiple subgraphs. Given a graph $G = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, and a graph classifier f. GNNCert uses a hash function h (e.g., MD5) to generate the subgraphs for G. A hash function takes a bit string as input and outputs an integer (e.g., within a range $[0, 2^{128}-1]$). It uses the string of edge or node index as the input to the hash function. For instance, for a node u, its string is denoted as $\texttt{str}(u)$, while for an edge $e = (u,v)$, its string is $\texttt{str}(u) + \texttt{str}(v)$, where "+" means string concatenation, and $\texttt{str}$ turns the node index into a string and adds "0" prefix to align it into a fixed length.

To defend against edge manipulation, it uses h to map each edge into a subgraph index. Assuming T_e subgraphs in total, the subgraph index i_e of every edge $e = (u,v)$ is defined as[3]

$$i_e = h[\text{str}(u) + \text{str}(v)] \bmod T_e + 1, \tag{3}$$

where $\bmod$ is the module function. Denoting $\mathcal{E}^i$ as the set of edges whose subgraph index is i, i.e., $\mathcal{E}^i = \{\forall e \in \mathcal{E} : i_e = i\}$, T_e subgraphs for G can be built as $\mathcal{G}_T^e = \{G_i = (\mathcal{V}, \mathcal{E}^i, \mathbf{X}) : i = 1, 2, \cdots, T_e\}$, where edges in different subgraphs are disjoint, i.e., $\mathcal{E}^i \cap \mathcal{E}^j = \emptyset, \forall i, j \in \{1, \cdots, T_e\}, i \neq j$.

To defend against node feature manipulation, it uses h to map each node into a subgraph index. Assuming T_n subgraphs in total, the subgraph index i_u of every node u is

$$i_u = h[\text{str}(u)] \bmod T_n + 1, \tag{4}$$

It then uses $\mathbf{X}^i$ to denote the features of nodes whose subgraph index is i. Then, T_n subgraphs can be built as: $\mathcal{G}_T^n = \{G_i = (\mathcal{V}, \mathcal{E}, \mathbf{X}^i) : i = 1, 2, \cdots, T_n\}$,

To defend against both manipulations, it then constructs a total of $T = T_e \cdot T_n$ subgraphs $\mathcal{G}_T = \{G_t = (\mathcal{V}, \mathcal{E}^i, \mathbf{X}^j), t = 1, \cdots, T_e \cdot T_n, i = \lceil t/T_n \rceil, j = t - (i-1) \cdot T_n\}$.

[3] In the undirected graph, we put the node with a smaller index (say u) first and let $h[\text{str}(v) + \text{str}(u)] = h[\text{str}(u) + \text{str}(v)]$.

Step II: build a voting graph classifier on all subgraphs. GNNCert applies the graph classifier f to make predictions on all T subgraphs, and count the vote c_y for every class $y \in \mathcal{Y}$.

$$c_{y_G} = \sum_{i=1}^{T} \mathbb{I}(f(G_i) = y_G), \forall y_G \in \mathcal{Y} \tag{5}$$

It then defines a *voting graph classifier* $\bar{f}$ as returning the class with the most vote:

$$\bar{f}(G) = \arg\max_{y_G \in \mathcal{Y}} c_{y_G} \tag{6}$$

Step III: derive the deterministic robustness guarantee for the voting graph classifier. GNNCert guarantees that T_n (or T_e) subgraphs are corrupted when an attacker injects or deletes an *arbitrary* edge (or *arbitrarily* perturb the features of a node). Then, GNNCert shows the voting classifier $\bar{f}$ tolerates up to $\lfloor M^f/T_e \rfloor$ perturbed edges ***OR*** $\lfloor M^f/T_n \rfloor$ of nodes with adversarially perturbed features, where $M^f \in [0, T_n \cdot T_e/2]$ is a constant depending on the number of votes of f's output.

Limitations of GNNCert: 1) It only derives the robustness guarantee against edge manipulation *OR* node feature manipulation. Under a very special case when $T_n = T_e = T$, we can derive its robustness against both edge *AND* node feature manipulation, where the certified perturbation size is $\lfloor M^f/T \rfloor$. However, its performance is worse than ours (See Figure 9). 2) It is only applicable for graph classification. 3) It cannot defend against the well-known node injection attack.

2.4 Problem Statement

Threat model: Given a GNN node/graph classifier f and a graph G, the adversary can *arbitrarily* manipulate a number of the edges, nodes, and node features in G such that f misclassifies target graphs in graph classification or target nodes in node classification. For instance, when a social network platform deploys a GNN detector to detect fake users (the adversary) [53, 68], the fake users is motivated to evade them [52]: they can modify their profiles, their connections with some users, and create new fake accounts and connections to bypass detection. Since we focus on certified defenses, we consider the strongest attack where the adversary has white-box access to G and f, i.e., it knows all the edges, nodes, and node features in G, and all the model parameters about f.

Defense goal: We aim to build a certifiably robust GNN that:

- has a deterministic robustness guarantee;
- is suitable for both node and graph classification tasks;
- provably predicts the same label against the arbitrary perturbation when the *perturbation size*, i.e., the total number of manipulated nodes, nodes with feature perturbation, and edges, is bounded by a threshold, which we call the *certified perturbation size*.

Our ultimate goal is to obtain the largest-possible certified perturbation size that satisfies all the above conditions.

3 Our Voting-based Defense: AGNNCert

In this section we introduce our voting-based certified defense AGNNCert for GNNs against arbitrary perturbations. We first give an overview of AGNNCert in Section 3.1, which consists of three critical steps, e.g., the first step is to divide a graph into multiple subgraphs with disjoint edges. We then design two distinct graph division strategies (one is edge-centric in Section 3.2 inspired by [64] and the other is node-centric in Section 3.3 by further enhancing the robustness guarantee). Within each strategy, we derive our deterministic certified robustness results, which can treat existing defenses as special cases. Figure 1 briefly illustrates our AGNNCert.

3.1 Overview

Given a graph $G = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, a GNN node/edge classifier f, the set of classes $\mathcal{Y}$, and a target node $v \in \mathcal{V}$ if the task is node classification. At a high level, our defense framework is similar to [64] that consists of three steps below:

Step I: divide the graph into multiple subgraphs. We divide G into a set of T subgraphs $\mathcal{G}_T = \{G_1, G_2, \ldots, G_T\}$ via a hash function and ensure edges in different subgraphs are *disjoint*.

Step II: build a voting-based node/graph classifier: We apply the GNN classifier f to make predictions on all the T subgraphs, and count the vote c_y for every class y in $\mathcal{Y}$.

$$\textbf{Node classifier: } c_{y_v} = \sum_{i=1}^{T} \mathbb{I}(f(G_i)_v = y_v), \forall y_v \in \mathcal{Y} \quad (7)$$

$$\textbf{Graph clasifier: } c_{y_G} = \sum_{i=1}^{T} \mathbb{I}(f(G_i) = y_G), \forall y_G \in \mathcal{Y} \quad (8)$$

We then define our *voting node/graph classifier* $\bar{f}$ as returning the class with the most vote:

$$\textbf{Voting node classifier: } \bar{f}(G)_v = \arg\max_{y_v \in \mathcal{Y}} c_{y_v} \quad (9)$$

$$\textbf{Voting graph classifier: } \bar{f}(G) = \arg\max_{y_G \in \mathcal{Y}} c_{y_G} \quad (10)$$

Step III: derive the deterministic robustness guarantee. We denote y_a and y_b as the class with the most vote c_{y_a} and the second-most vote c_{y_b}, respectively. We pick the class with a smaller index if ties exist. Denote G' as the perturbed graph of G under arbitrary perturbation, and $\mathcal{G}'_T = \{G'_1, G'_2, \ldots, G'_T\}$ be the set of T subgraphs generated for G' under the same graph division strategy. Then we have the below condition for certified robustness against arbitrary attacks on GNNs.

Theorem 1 (Sufficient Condition for Certified Robustness)**.** *Let $y_a, y_b, c_{y_a}, c_{y_b}$ be defined above in node classification or graph classification, and let $M = \lfloor c_{y_a} - c_{y_b} - \mathbb{I}(y_a > y_b) \rfloor / 2$. The voting classifier $\bar{f}$ guarantees the same prediction on both G' and G for the target node v in node classification or the target graph G in graph classification, if the number of subgraphs' predictions on $\{G_i\}$'s and $\{G'_i\}$' that are different under the arbitrary perturbation is bounded by M. I.e.,*

$$\forall G' : \sum_{i=1}^{T} \mathbb{I}(f(G_i)_v \neq f(G'_i)_v) \leq M \implies \bar{f}(G)_v = \bar{f}(G')_v \quad (11)$$

$$\forall G' : \sum_{i=1}^{T} \mathbb{I}(f(G_i) \neq f(G'_i)) \leq M \implies \bar{f}(G) = \bar{f}(G') \quad (12)$$

Proof. See Appendix A in the full version.

The above theorem motivates us to design the graph division method such that: 1) the number of different subgraph predictions on $\mathcal{G}_T$ and $\mathcal{G}'_T$ can be upper bounded (and the smaller the better). 2) the difference between the most vote c_{y_a} and second-most vote c_{y_b} is as large as possible, in order to ensure larger certified perturbation size.

Next, we introduce our two graph division methods. Figure 2 visualizes the divided subgraphs of the two methods without and with the adversarial manipulation.

3.2 Edge-Centric Graph Division

Our first graph division method is edge-centric inspired by [64]. The idea is to divide *edges* in a graph into different subgraphs, such that each edge is deterministically mapped into *only one subgraph*. With this strategy, we can bound the number of altered predictions on these subgraphs before and after the arbitrary perturbation (Theorem 5), which facilitates deriving the certified perturbation size (Theorem 6). Next, we show our edge-centric graph division method in detail.

Generating edge-centric subgraphs: We follow [64] to use the hash function to map edges as shown in Equation 3. We build T subgraphs for G as $\mathcal{G}_T = \{G_i = (\mathcal{V}, \mathbf{X}, \mathcal{E}^i) : i = 1, 2, \cdots, T\}$, where $\mathcal{E}^i \cap \mathcal{E}^j = \emptyset, \forall i, j \in \{1, \cdots, T\}, i \neq j$.

Recall that [64] maps both edges and node features to generate two sets of subgraphs to defend against node feature and edge manipulations. Instead, our method only needs to map edges into a set of subgraphs, which is not only efficient, but also obtains much defense performance.

Bounding the number of different subgraph predictions: For a perturbed graph G', we use the same graph division strategy to generate a set of T subgraphs $\mathcal{G}'_T = \{G'_1, G'_2, \cdots, G'_T\}$. Then, we can upper bound the number of different subgraph predictions on $\mathcal{G}_T$ and $\mathcal{G}'_T$ against any individual perturbation.

Theorem 2. *Assume a graph G is under the edge manipulation $\{\mathcal{E}_+, \mathcal{E}_-\}$, then at most $|\mathcal{E}_+| + |\mathcal{E}_-|$ subgraphs generated by our edge-centric graph division have different predictions between $\mathcal{G}'_T$ and $\mathcal{G}_T$.*

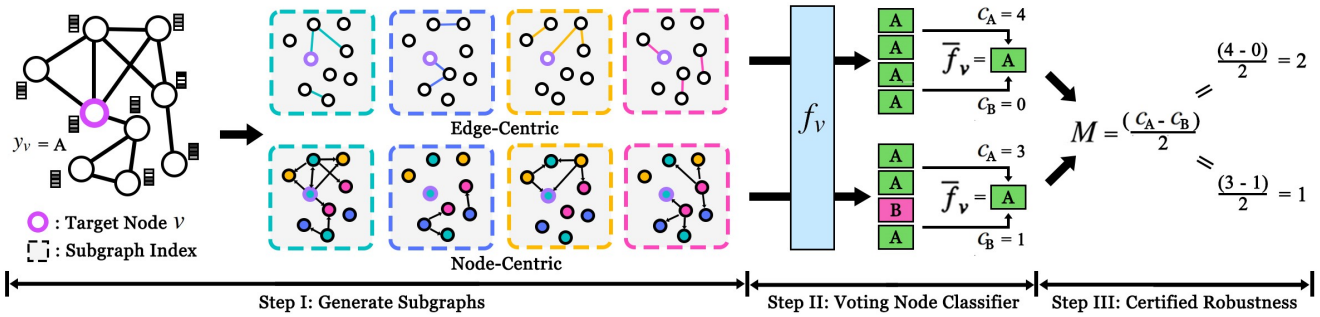


Figure 1: Overview of our AGNNCert (use node classification for illustration), which consists of three steps. Assume we are given an input graph G, a GNN node classifier f, and a target node v with label y_v for classification. **Step I:** it divides G into a set of (e.g., 4) subgraphs via the proposed *Edge-Centric Graph Division* (Section 3.2) or *Node-Centric Graph Division* (Section 3.3) strategy. **Step II:** it builds a voting node classifier $\bar{f}$ based on all the subgraphs. Specifically, the target node's predicted class by f on all subgraphs are treated as votes, and $\bar{f}$ returns the class with the most vote as the final prediction. **Step III:** it derives the certified perturbation size M for $\bar{f}$ against arbitrary perturbations with a deterministic (100%) guarantee.

Proof. Edges in all subgraphs of $\mathcal{G}_T$ are disjoint. Hence, when any edge in G is deleted or added by an adversary, only one subgraph from $\mathcal{G}_T$ is affected. Further, when any $|\mathcal{E}_+|+|\mathcal{E}_-|$ edges in G are perturbed, there are at most $|\mathcal{E}_+|+|\mathcal{E}_-|$ subgraphs between $\mathcal{G}_T$ and $\mathcal{G}'_T$ are different. By applying the node/graph classifier on $\mathcal{G}_T$ and $\mathcal{G}'_T$, there are at most $|\mathcal{E}_+|+|\mathcal{E}_-|$ predictions that are different between them. □

Unlike edge manipulation, both node and node feature manipulations involve all components (i.e., edges, nodes, and node features) in the graph. At first glance, it seems hard to bound the alter subgraph predictions in this case. After careful analysis, we observe the underlying message-passing mechanism in GNNs (Section 2) still facilitates us to obtain the upper bound shown below.

Theorem 3. *Assume a graph G is under the node manipulation $\{\mathcal{V}_+, \mathcal{E}_{\mathcal{V}_+}, \mathbf{X}'_{\mathcal{V}_+}, \mathcal{V}_-, \mathcal{E}_{\mathcal{V}_-}\}$, then at most $|\mathcal{E}_{\mathcal{V}_+}|+|\mathcal{E}_{\mathcal{V}_-}|$ subgraphs generated by our edge-centric graph division have different predictions between $\mathcal{G}'_T$ and $\mathcal{G}_T$.*

Theorem 4. *Assume a graph G is under the node feature manipulation $\{\mathcal{V}_r, \mathcal{E}_{\mathcal{V}_r}, \mathbf{X}'_{\mathcal{V}_r}\}$, then at most $|\mathcal{E}_{\mathcal{V}_r}|$ subgraphs generated by our edge-centric graph division have different predictions between $\mathcal{G}'_T$ and $\mathcal{G}_T$.*

Proof. Our proof for the above two theorems is based on the key observation that manipulations on isolated nodes have no influence on other nodes' representations in GNNs. Take node injection for instance and the proof for other cases are similar. Note that all subgraphs after node injection will contain the newly injected nodes, but they still do not have overlapped edges between each other via the hash mapping. Hence, the edges $E_{\mathcal{V}_+}$ induced by the injected nodes $\mathcal{V}_+$ exist in at most $|E_{\mathcal{V}_+}|$ subgraphs. In other word, the injected nodes $\mathcal{V}_+$ in at least $T-|E_+|$ subgraphs have no edges and are isolated.

Due to the message passing mechanism in GNNs, every node only uses its neighboring nodes' representations to update its own representation. Hence, the isolated injected nodes, whatever their features $\mathbf{X}'_{\mathcal{V}_+}$ are, would have no influence on other nodes' representations, implying at least $T-|E_+|$ subgraphs' predictions maintain the same. □

With above theorems, we can bound the total number of different subgraph predictions with *arbitrary perturbation*.

Theorem 5 (Bounded Number of Edge-Centric Subgraphs with Altered Predictions under Arbitrary Perturbation)**.** *Given any GNN node/graph classifier f, a graph G, and T edge-centric subgraphs $\mathcal{G}_T$ for G. A perturbed graph G' of G is with arbitrary edge manipulation $\{\mathcal{E}_+, \mathcal{E}_-\}$, node manipulation $\{\mathcal{V}_+, \mathcal{E}_{\mathcal{V}_+}, \mathcal{V}_-, \mathcal{E}_{\mathcal{V}_-}\}$, and node feature manipulation $\{\mathbf{X}_{\mathcal{V}_r}, \mathcal{V}_r, \mathcal{E}_{\mathcal{V}_r}\}$. Then at most $m = |\mathcal{E}_+|+|\mathcal{E}_-|+|\mathcal{E}_{\mathcal{V}_+}|+|\mathcal{E}_{\mathcal{V}_-}|+|\mathcal{E}_{\mathcal{V}_r}|$ predictions are different by the node/graph classifier f on the subgraphs $\mathcal{G}'_T$ generated for the perturbed graph G' and on $\mathcal{G}_T$. In other words, $\sum_{i=1}^{T}\mathbb{I}(f(G_i)_v \neq f(G'_i)_v) \leq m$ for any target node $v \in G$ in node classification or $\sum_{i=1}^{T}\mathbb{I}(f(G_i) \neq f(G'_i)) \leq m$ in graph classification.*

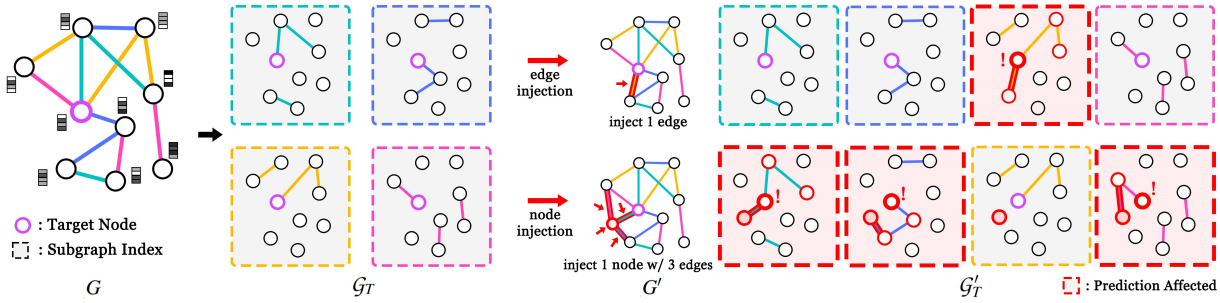
Deriving the robustness guarantee against arbitrary perturbation: Based on Theorem 1 and Theorem 5, we can derive the certified perturbation size as the maximal perturbation such that Equation 11 or Equation 12 is satisfied. Formally,

Theorem 6 (Certified Robustness Guarantee with Edge–Centric Subgraphs against Arbitrary Perturbation)**.** *Let $f, y_a, y_b, c_{y_a}, c_{y_b}$ be defined above for edge-centric subgraphs, and m be the perturbation size induced by an arbitrary perturbed graph G' on G. The voting classifier $\bar{f}$ guarantees the same prediction on both G' and G for the target node v in node classification (i.e., $\bar{f}(G')_v = \bar{f}(G)_v$) or target graph G in graph classification (i.e., $\bar{f}(G') = \bar{f}(G)$), when m satisfies*

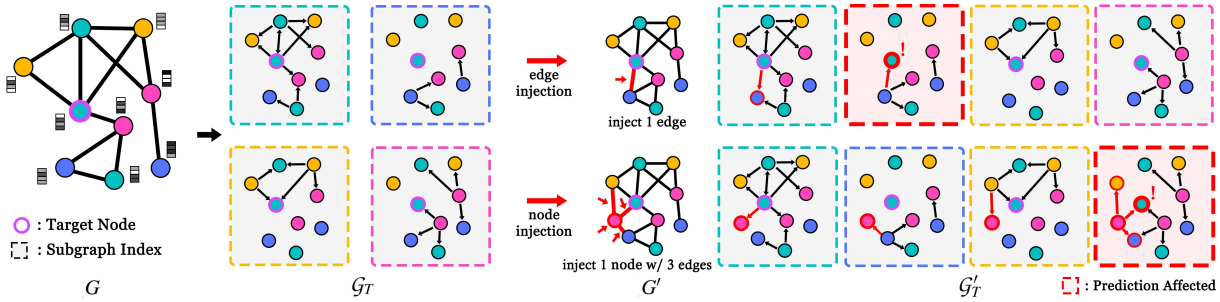
$$m \leq M = \lfloor c_{y_a} - c_{y_b} - \mathbb{I}(y_a > y_b) \rfloor / 2. \tag{13}$$

In other words, the maximum certified perturbation size is M.

Remark: We have the following remarks from our theoretical result in Theorem 6.



(a) Edge-Centric Graph Division against edge injection and node injection attacks



(b) Node-Centric Graph Division against edge injection and node injection attacks

Figure 2: Illustration of our edge-centric and node-centric graph division strategies for node classification. We use edge injection and node injection attacks to show the bounded number of altered predictions on the generated subgraphs after the attack. **To summarize:** 1 injected edge affects at most 1 subgraph prediction in both graph division strategies. In contrast, 1 injected node with, e.g., 3 injected edges can affect (at most) 3 subgraph predictions with edge-centric graph division, but at most 1 subgraph prediction with node-centric graph division. Figures 11-12 in Appendix also show other attacks and on graph classification.

- No (adaptive/unknown) attack can break `AGNNCert` if its perturbation budget is within the derived bound M, regardless of the attack knowledge of ANNCert.
- It can be applied for any GNN node/graph classifier.
- The guarantee is true with a probability 100%.
- It treats existing robustness guarantees as special cases.
  - For edge manipulation $\{\mathcal{E}_+, \mathcal{E}_-\}$ [1, 54, 64], the voting classifier $\bar{f}$ is certified robust if $|\mathcal{E}_+| + |\mathcal{E}_-| \leq M$.
  - For node manipulation $\{\mathcal{V}_+, \mathcal{E}_{\mathcal{V}_+}, \mathbf{X}'_{\mathcal{V}_r}, \mathcal{V}_-, \mathcal{E}_{\mathcal{V}_-}\}$ [30], $\bar{f}$ is certified robust if $|\mathcal{E}_{\mathcal{V}_+}| + |\mathcal{E}_{\mathcal{V}_-}| \leq M$.
  - For node feature manipulation $\{\mathcal{V}_r, \mathcal{E}_{\mathcal{V}_r}, \mathbf{X}_{\mathcal{V}_r}\}$ [26, 64], $\bar{f}$ is certified robust if $|\mathcal{E}_{\mathcal{V}_r}| \leq M$.
  - For both edge and node feature manipulation [64], $\bar{f}$ is certified robust if $|\mathcal{E}_+| + |\mathcal{E}_-| + |\mathcal{E}_{\mathcal{V}_r}| \leq M$.

3.3 Node-Centric Graph Division

We observe the robustness guarantee under edge-centric graph division is largely dominated by the edges (i.e., $\mathcal{E}_{\mathcal{V}_+}, \mathcal{E}_{\mathcal{V}_-}$) induced by the manipulated nodes $\mathcal{V}_+, \mathcal{V}_-$, and edges $\mathcal{E}_{\mathcal{V}_r}$ by the perturbed node features $\mathbf{X}'_{\mathcal{V}_r}$. This guarantee could be weak against node or node feature manipulation, as the number of edges (i.e., $|\mathcal{E}_{\mathcal{V}_+}|, |\mathcal{E}_{\mathcal{V}_-}|, |\mathcal{E}_{\mathcal{V}_r}|$) could be much larger, compared with the number of the nodes (i.e., $|\mathcal{V}_+|, |\mathcal{V}_-|, |\mathcal{V}_r|$). For instance, an injected node could link with many edges to a given graph in practice, and when the number exceeds M in Equation 13, the certified robustness guarantee is ineffective.

This flaw inspires us to generate subgraphs, where we expect at most one subgraph is affected under every node or node feature manipulation (this means all edges of a manipulated node should be in a same subgraph). We design a tailored node-centric graph division strategy to achieve our goal.

Naive solutions are ineffective: A first solution is to map nodes into different subgraphs that are *non-overlapped*, like mapping edges into subgraphs that are non-overlapped in edge-centric method. Though this method may work for graph classification, it completely fails for node classification, as every node only appears once in all subgraphs and all target nodes can only receive one vote, yielding vacuous robustness.

A second solution is to retain all nodes in every subgraph (say G_i), but keep only edges connected to nodes with the index i. However, this idea still does not work, because some nodes not with index i may still connect to nodes with index i, and manipulations on nodes with index i would still influence representations of those nodes with a different index.

Generating node-centric directed subgraphs: We notice the failure of the second solution is because the message passing between two connected nodes u and v is bidirectional. If we decompose an undirected edge into two directed edges, and only use the outgoing edges of nodes, e.g., with index i,

then the message is passed in one direction, i.e., from index i nodes to their connected nodes. Hence, we propose dividing graphs into directed subgraphs.

We use a hash function h to generate directed subgraphs for a given graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. Our node-centric graph division strategy as follow: (1) we treat every undirected edge $e = (u, v) \in G$ as two directed edges for u[4]: the outgoing edge $u \to v$ and incoming edge $v \to u$; (2) for every node u, we compute the subgraph index of its every outgoing edge $u \to v$:

$$i_{u \to v} = h[\text{str}(u)] \bmod T + 1. \tag{14}$$

Note all outgoing edges of u are mapped in the same subgraph.

We use $\vec{\mathcal{E}}_i$ to denote the set of directed edges whose subgraph index is i, i.e., $\vec{\mathcal{E}}_i = \{\forall u \to v \in \mathcal{E} : i_{u \to v} = i\}$. Then, we can construct T *directed* subgraphs for G as $\vec{\mathcal{G}}_T = \{\vec{G}_i = (\mathcal{V}, \vec{\mathcal{E}}_i, \mathbf{X}) : i = 1, 2, \cdots, T\}$. Here, we mention that we need to further postprocess the subgraphs for graph classification, in order to derive the robustness guarantee. Particularly, in each subgraph $\vec{G}_i$, we remove all other nodes whose subgraph index is not i. This is because although they have no influence on other nodes' representation, their information would still be passed to the global graph embedding aggregation. To make up the loss of connectivity between nodes and simulate the aggregation, we add an extra node with a zero feature, and add an outgoing edge from every node with index i to it.

Bounding the number of different subgraph predictions: Similarly, for a perturbed graph G', we use the same graph division strategy to generate a set of T *directed subgraphs* $\vec{\mathcal{G}}'_T = \{\vec{G}'_1, \vec{G}'_2, \cdots, \vec{G}'_T\}$. We first show the theoretical results that can upper bound the number of different subgraph predictions on $\vec{\mathcal{G}}_T$ and $\vec{\mathcal{G}}'_T$ against any individual perturbation.

Theorem 7. *Assume a graph G is under the edge manipulation $\{\mathcal{E}_+, \mathcal{E}_-\}$, then at most $|\mathcal{E}_+| + |\mathcal{E}_-|$ subgraphs generated by our node-centric graph division have different predictions between $\vec{\mathcal{G}}'_T$ and $\vec{\mathcal{G}}_T$.*

Proof. We simply analyze when an arbitrary edge (u, v) is deleted/added from G. It is obvious at most two subgraphs $G_{i_{u \to v}}$ and $G_{i_{v \to u}}$ are perturbed after perturbation, but via detailed analysis, at most one subgraph's prediction is affected.

We consider the following two cases: i) $i_{u \to v} = i_{v \to u}$. This means u and v are in the same subgraph, hence at most one subgraph's prediction is affected. ii) $i_{u \to v} \neq i_{v \to u}$. In subgraph $\vec{G}_{i_{u \to v}}$, v only has incoming edges. Due to the message passing mechanism in GNNs, only the node v's representation $\boldsymbol{h}_v^{(K)}$ is affected. Symmetrically in subgraph $\vec{G}_{i_{v \to u}}$, only node u's representation $\boldsymbol{h}_u^{(K)}$ is affected. Therefore, for node classification on a target node $w \in \mathcal{V}$, there exists at most one subgraph whose prediction is affected (when $w = u$ or $w = v$); for graph classification, since u (or v) is removed in subgraph $\vec{G}_{i_{v \to u}}$ (or $\vec{G}_{i_{u \to v}}$), no prediction is changed on the two subgraphs.

[4]GNNs inherently handles directed graphs with directed message passing. Particularly, each node only uses its incoming neighbors' message for update.

Generalizing the analysis to any $|\mathcal{E}_+| + |\mathcal{E}_-|$ edges in G being perturbed, at most $|\mathcal{E}_+| + |\mathcal{E}_-|$ predictions are different between $\mathcal{G}_T$ and $\mathcal{G}'_T$. □

Theorem 8. *Assume a graph G is under the node manipulation $\{\mathcal{V}_+, \mathcal{E}_{\mathcal{V}_+}, \mathbf{X}'_{\mathcal{V}_+}, \mathcal{V}_-, \mathcal{E}_{\mathcal{V}_-}\}$, then at most $|\mathcal{V}_+| + |\mathcal{V}_-|$ subgraphs generated by our node-centric graph division have different predictions between $\vec{\mathcal{G}}'_T$ and $\vec{\mathcal{G}}_T$.*

Theorem 9. *Assume a graph G is under the node feature manipulation $\{\mathcal{V}_r, \mathcal{E}_{\mathcal{V}_r}, \mathbf{X}'_{\mathcal{V}_r}\}$, then at most $|\mathcal{V}_r|$ subgraphs generated by our edge-centric graph division have different predictions between $\vec{\mathcal{G}}'_T$ and $\vec{\mathcal{G}}_T$.*

Proof. Our proof for the above two theorems is based on the key observation that: *in a directed graph, manipulations on nodes with no outgoing edge have no influence on other nodes' representations in GNNs*. For any node $u \in G$, only one subgraph $\vec{G}_{h[\text{str}(u)] \bmod T + 1}$ has outgoing edges. Take node injection for instance and the proof for other cases are similar. Note that all subgraphs after node injection will contain newly injected nodes V_+, but they still do not have overlapped nodes with outgoing edges between each other via the hashing mapping. Hence, the injected nodes only have outgoing edges in at most $|V_+|$ subgraphs. Due to the directed message passing mechanism in GNNs, every node only uses its incoming neighboring nodes' representation to update its own representation. Hence, the injected nodes with no outgoing edges, whatever their features $\mathbf{X}'_{\mathcal{V}_+}$ are, would have no influence on other nodes' representations, implying at least $T - |V_+|$ subgraphs' predictions maintain the same. □

Remark: With edge manipulation, like Theorem 2, Theorem 7 has the same bounded number of altered subgraph predictions w.r.t. manipulated edges. Unlike Theorems 3 and 4, Theorems 8 and 9 bound the number of altered subgraph predictions w.r.t. manipulated nodes. *Importantly, we highlight these two bounds allow a manipulated node to link with many even infinite number of edges. Hence, these bounds are inherently robust against node inject attacks which often inject few nodes but with moderate number of edges, and node feature perturbations where the perturbed nodes have high degrees.*

With above theorems, the total number of different subgraph predictions between $\vec{\mathcal{G}}'_T$ and $\vec{\mathcal{G}}_T$ with *arbitrary perturbation* can be straightforwardly bounded below.

Theorem 10 (Bounded Number of Node-Centric Subgraphs with Altered Predictions under Arbitrary Perturbation)**.** *Let $f, v, G, \mathcal{E}_+, \mathcal{E}_-, \mathcal{V}_+, \mathcal{V}_-, \mathcal{V}_r$ be defined in Theorem 5, and $\vec{\mathcal{G}}_T, \vec{\mathcal{G}}'_T$ contain directed subgraphs under the node-centric graph division. Then, at most $\bar{m} = |\mathcal{E}_+| + |\mathcal{E}_-| + |\mathcal{V}_+| + |\mathcal{V}_-| + |\mathcal{V}_r|$ predictions are different by the node/graph classifier f on $\vec{\mathcal{G}}'_T$ and on $\vec{\mathcal{G}}_T$. In other words, $\sum_{i=1}^T \mathbb{I}(f(\vec{G}_i)_v \neq f(\vec{G}'_i)_v) \leq \bar{m}$ for any target node $v \in G$ in node classification or $\sum_{i=1}^T \mathbb{I}(f(\vec{G}_i) \neq f(\vec{G}'_i)) \leq \bar{m}$ in graph classification.*

Deriving the robustness guarantee against arbitrary perturbation: Based on Theorem 1 and Theorem 10, we can derive the certified perturbation size formally stated below

Theorem 11 (Certified Robustness Guarantee with Node–Centric Subgraphs against Arbitrary Perturbation)**.** *Let $f, y_a, y_b, c_{y_a}, c_{y_b}$[5] be defined above for node-centric subgraphs, and $\bar{m}$ be the perturbation size induced by an arbitrary perturbed graph G' on G. With a probability 100%, the voting classifier $\bar{f}$ guarantees the same prediction on both G' and G for the target node v in node classification (i.e., $\bar{f}(G')_v = \bar{f}(G)_v$) or the target graph G in graph classification (i.e., $\bar{f}(G') = \bar{f}(G)$), if*

$$\bar{m} \leq M = \lfloor c_{y_a} - c_{y_b} - \mathbb{I}(y_a > y_b) \rfloor / 2. \tag{15}$$

Remark: Similarly, our theoretical result can be applied for any GNN node/graph classifier, is true with probability 100%, and cannot be broken by any attack with perturbation budget $\leq M$. Further, it can treat existing defenses as special cases.

- For edge manipulation $\{\mathcal{E}_+, \mathcal{E}_-\}$ [1, 54, 64], the voting classifier $\bar{f}$ is certified robust if $|\mathcal{E}_+| + |\mathcal{E}_-| \leq M$.
- For node manipulation $\{\mathcal{V}_+, \mathcal{E}_{\mathcal{V}_+}, \mathbf{X}'_{\mathcal{V}_+}, \mathcal{V}_-, \mathcal{E}_{\mathcal{V}_-}\}$ [30], $\bar{f}$ is certified robust if $|\mathcal{V}_+| + |\mathcal{V}_-| \leq M$.
- For node feature manipulation $\{\mathcal{V}_r, \mathcal{E}_{\mathcal{V}_r}, \mathbf{X}'_{\mathcal{V}_r}\}$ [26, 64], $\bar{f}$ is certified robust if $|\mathcal{V}_r| \leq M$.
- For both edge and node feature manipulation [64], $\bar{f}$ is certified robust if $|\mathcal{E}_+| + |\mathcal{E}_-| + |\mathcal{V}_r| \leq M$.

4 Experiments

4.1 Experiment Settings

Datasets: We use four node classification datasets (Cora-ML [41], Citeseer [47], PubMed [47], Amazon-C [69]) and four graph classification datasets (AIDS [45], MUTAG [10], PROTEINS [2], and DD [12]) for evaluation. In each dataset, we take 30% nodes (for node classification) or 50% graphs (for graph classification) as the training set, 10% and 20% as the validation set, and the remaining nodes/graphs as the test set. Table 6 in Appendix shows the basic statistics of them. Our experiments are tested on a machine with NVIDIA RTX-4090 24G GPU, AMD EPYC 7352 CPU, and 60G RAM.

GNN classifiers and `AGNNCert` training: We adopt the three well-known GNNs as the base node/graph classifiers: GCN [29], GSAGE [20] and GAT [50], and use their official source code[6]. To enhance the robustness performance, existing certified defense [30,64] augment the training set with generated subgraphs [64] or noisy graphs [30] to train the GNN classifier. Similarly, `AGNNCert` trains the GNN classifier using both the training nodes/graphs and their generated subgraphs, whose labels are same as the training nodes/graphs'. We denote the two versions of `AGNNCert` under edge-centric graph division and node-centric graph division as `AGNNCert-E` and `AGNNCert-N`, respectively. By default, we use GCN as the node/graph classifier in our experiments.

Evaluation metric: Following existing works [30,54,64], we use the certified node/graph accuracy at perturbation size as the evaluation metric. For arbitrary perturbation, the perturbation size is the total number of manipulated nodes, edges, and nodes whose features can be arbitrarily perturbed. Given a perturbation size m and test nodes/graphs, certified node/graph accuracy at m is the fraction of test nodes/graphs that are accurately classified by the voting node/graph classifier and its certified perturbation size is no smaller than m. Note that the standard node/graph accuracy is under $m = 0$.

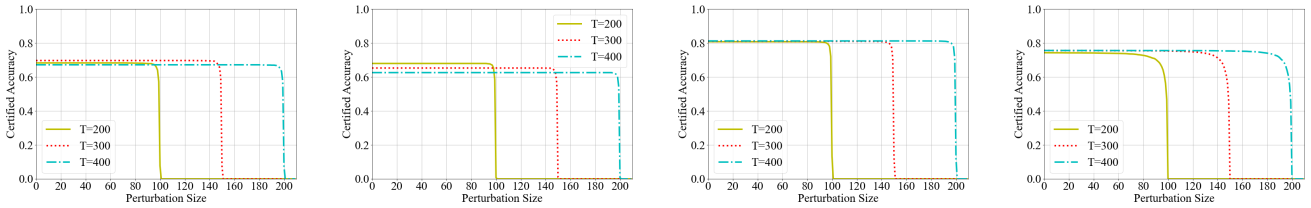
Compared baselines: As `AGNNCert` encompasses existing defenses as special cases, we can compare `AGNNCert` with them against less types of perturbation. Here, we choose the state-of-the-art Bi-RS [30] and GNNCert [64] for comparison.

- **Bi-RS:** It certifies GNN for *node classification* against node inject attacks with a probabilistic guarantee. During training, Bi-RS augments the graph with N_1 noisy graphs from a smoothing distribution (defined in its Eqn.3) and trains the node classifier with both clean graphs and their noisy ones. During certification, Bi-RS utilizes Monte-Carlo sampling to compute the certified perturbation size. Given a graph and the trained node classifier, Bi-RS first generates N_2 noisy graphs for the given graph and then derives the robustness guarantee for each target node on the noisy graphs that is correct with a probability $1 - \alpha$. Note that ensuring a smaller α needs more samples. Bi-RS sets $N_2 = 50,000$ and $\alpha = 0.01$. In our experiment, we also set $N_1 = T$.
- **GNNCert:** It is the state-of-the-art certified defense (with a deterministic guarantee) of GNN for *graph classification* against edge manipulation, and both edge and node feature manipulation (more details see Section 2.3). We denote the two variants as GNNCert-E and GNNCert-EN, respectively. During training, GNNCert-E and GNNCert-EN use the extra T_e and $T_e \cdot T_n$ subgraphs for training the base graph classifier. During certification, GNNCert-E and GNNCert-EN also use the same number of subgraphs. *We highlight that, for edge manipulation, GNNCert-E has the same bound as our `AGNNCert-E` under edge-centric graph division. This is because the generated subgraphs of both defenses are exactly the same, and so does the voting graph classifier when using the same base GNN classifier.*

Parameter setting: `AGNNCert` has two hyperparameters: the hash function h and the number of subgraphs T. By default, we use MD5 as the hash function and set $T = 30,300$ respectively for node and graph classification, considering their different graph sizes. We also study the impact of them.

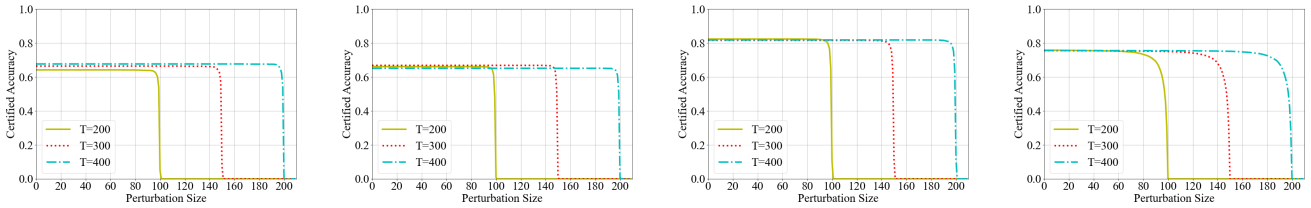
[5] Note that c_{y_a}, c_{y_b} have different values with those in edge-centric graph division, as the generated node-centric subgraphs are different from edge-centric subgraphs. Here we use the same notation for description brevity.

[6] https://github.com/tkipf/gcn; https://github.com/williamleif/GraphSAGE; https://github.com/PetarV-/GAT



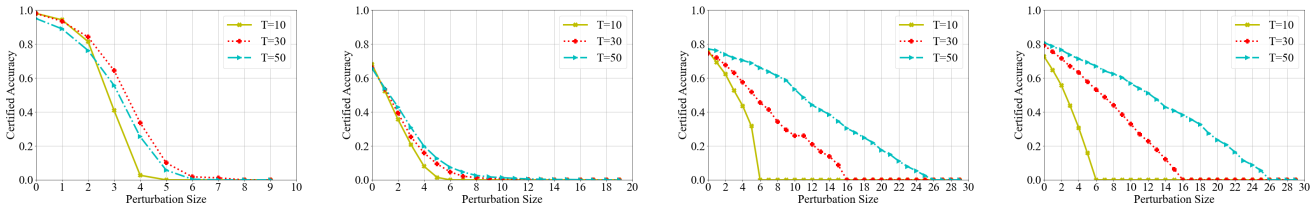
(a) Cora-ML (b) Citeseer (c) Pubmed (d) Amazon-C

Figure 3: Certified node accuracy of our AGNNCert-E w.r.t. the number of subgraphs T.



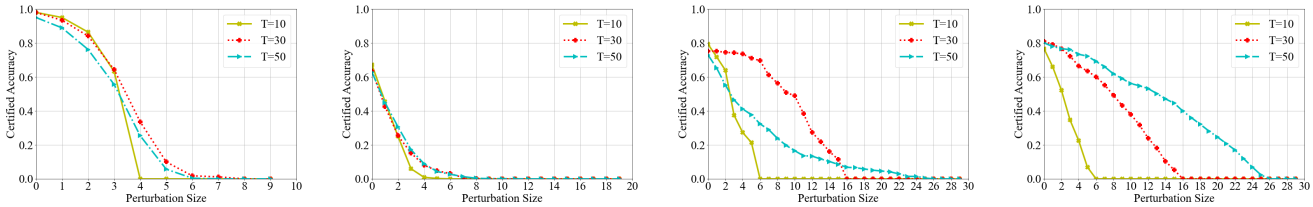
(a) Cora-ML (b) Citeseer (c) Pubmed (d) Amazon-C

Figure 4: Certified node accuracy of our AGNNCert-N w.r.t. the number of subgraphs T.



(a) AIDS (b) MUTAG (c) PROTEINS (d) DD

Figure 5: Certified graph accuracy of our AGNNCert-E w.r.t. the number of subgraphs T.



(a) AIDS (b) MUTAG (c) PROTEINS (d) DD

Figure 6: Certified graph accuracy of our AGNNCert-N w.r.t. the number of subgraphs T.

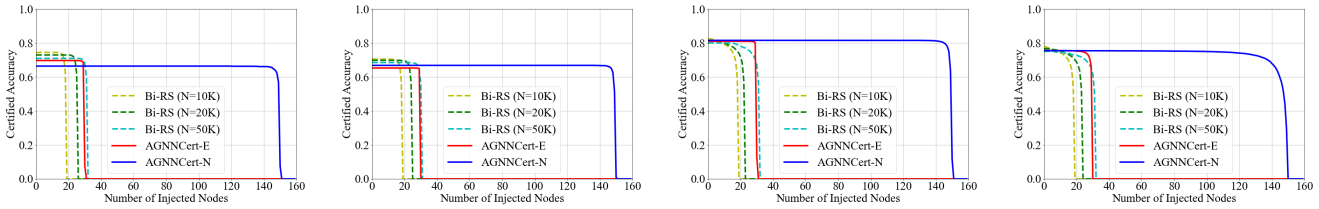
4.2 Experiment Results

4.2.1 AGNNCert against Arbitrary Perturbation

Main results:[7] Figures 3-4 show the certified node accuracy and Figures 5-6 show the certified graph accuracy at perturbation size m w.r.t. T under the two graph division strategies, respectively. We have the following observations.

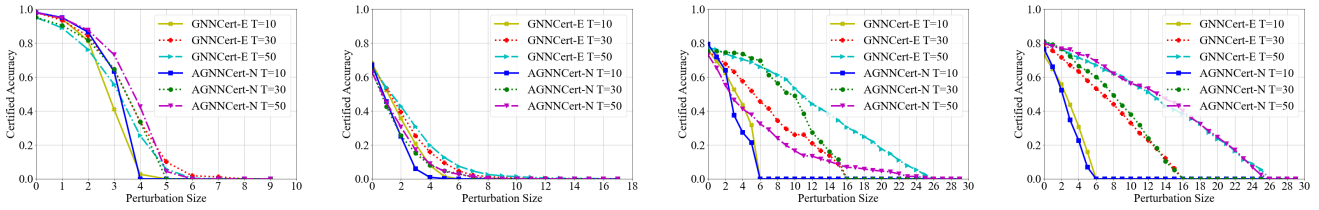
- Both AGNNCert-E and AGNNCert-N can tolerate the perturbation size up to 200 and 25, on the node classification and graph classification datasets, respectively. This means AGNNCert-E can defend against a total of 200 (25) arbitrary edges, while AGNNCert-N against a total of 200 (25) arbitrary edges and nodes caused by the arbitrary perturbation, on the node (graph) classification datasets, respectively. Note that node classification datasets have several orders of more nodes/edges than graph classification datasets, hence AGNNCert can tolerate more perturbations on them.
- T acts as the robustness-accuracy tradeoff. That is, a larger (smaller) T yields a higher (lower) certified perturbation size, but a smaller (higher) normal accuracy ($m = 0$).
- In AGNNCert-N, the guaranteed perturbed nodes can have an infinite number of edges. This thus implies AGNNCert-N produces better robustness than AGNNCert-E against the perturbed edges by node/node feature manipulation.

[7] See more results in the full version.



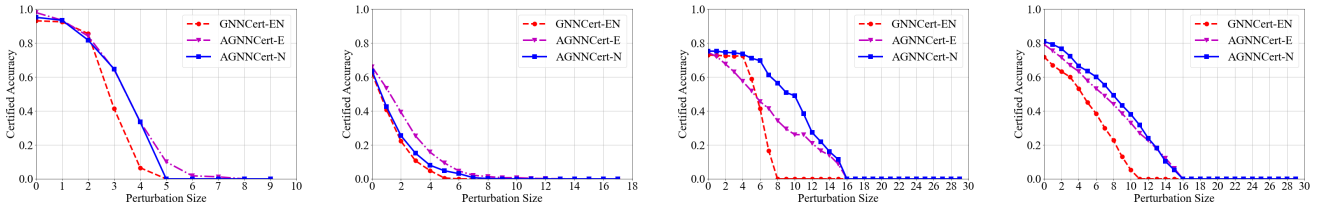
(a) Cora-ML (b) Citeseer (c) Pubmed (d) Amazon-C

Figure 7: Certified node accuracy of AGNNCert and Bi-RS against node inject attacks.



(a) AIDS (b) MUTAG (c) PROTEINS (d) DD

Figure 8: Certified graph accuracy of AGNNCert-N and GNNCert-E against edge manipulation.



(a) AIDS (b) MUTAG (c) PROTEINS (d) DD

Figure 9: Certified graph accuracy of AGNNCert and GNNCert-EN against edge and node feature manipulation.

Impact of hash function: Figure 13- Figure 16 in the full version show the certified node/edge accuracy of AGNNCert-E and AGNNCert-N with different hash functions. We observe that our certified accuracy and certified perturbation size are almost the same in all cases. This reveals AGNNCert is insensitive to hash functions, and [64] draws a similar conclusion.

Impact of base GNN classifiers: Figures 17-20 and Figures 21-24 in the full version show the certified accuracy at perturbation size using GSAGE and GAT as the base classifier, respectively. We have similar observations as those results with GCN. For instance, T trade offs robustness and accuracy.

Impact of subgraphs on the certified accuracy: We test the certified accuracy of (not) using subgraphs to train the GNN classifier. Figures 25-26 in the full version show the comparison results under the default T for node/graph classification. The results show training with subgraphs can enhance the certified robustness of AGNNCert, especially on large datasets. This is because training and certification both involve raw graphs and the subgraphs, making their distributions similar.

Impact of subgraphs on the normal accuracy: We test the normal accuracy of (not) using subgraphs to train the GNN classifier. Table 2 shows the comparison results of the test node/graph accuracy of the normally trained GNN without sbugraphs and AGNNCert with GNN trained on the subgraphs. We observe that the accuracy of AGNNCert is 5% smaller than that of normally trained GNN in almost all cases, and in some cases even larger. This implies the augmented subgraphs for training marginally affects the normal test accuracy.

Table 2: Node/graph accuracy of normally trained GNN and of AGNNCert with GNN trained on the subgraphs.

Dataset	GCN	AGNNCert		GSAGE	AGNNCert		GAT	AGNNCert	
		-E	-N		-E	-N		-E	-N
Cora-ML	0.73	0.70	0.68	0.67	0.67	0.68	0.74	0.68	0.69
Citeseer	0.66	0.65	0.67	0.64	0.63	0.64	0.66	0.65	0.66
Pubmed	0.86	0.81	0.82	0.84	0.84	0.84	0.85	0.84	0.84
Amazon-C	0.81	0.76	0.76	0.80	0.77	0.75	0.78	0.74	0.74
AIDS	0.99	0.98	0.96	0.97	0.96	0.97	0.96	0.98	0.98
MUTAG	0.71	0.66	0.65	0.70	0.66	0.67	0.71	0.67	0.66
Proteins	0.75	0.75	0.75	0.80	0.79	0.77	0.82	0.77	0.77
DD	0.80	0.79	0.81	0.81	0.80	0.81	0.81	0.77	0.80

4.2.2 Comparing AGNNCert with Bi-RS and GNNCert

Comparing AGNNCert with Bi-RS for node classification against node injection attacks: We first add some details

Table 3: Training and test time of provable defenses and undefended GNN on the evaluated datasets.

Datasets		Cora-ML	Citeseer	Pubmed	Amazon-C	Datasets	AIDS	MUTAG	PROT.	DD
Training Time (per epoch)	**GCN**	0.03s	0.03s	0.12s	0.31s	GCN	6.66s	14.82s	3.87s	6.45s
	Bi-RS	16.73s	22.21s	117.57s	98.10s	**GNNCert-E**	114.90s	388.01s	107.72s	171.34s
	AGNNCert-E	17.46s	21.44s	110.58s	102.31s	AGNNCert-E	100.55s	389.08s	95.70s	163.27s
	AGNNCert-N	18.59s	22.47s	102.26s	96.55s	AGNNCert-N	101.94s	400.97s	98.61s	151.18s
Test/Certification Time	**GCN**	0.01s	0.01s	0.02s	0.08s	GCN	1.46s	2.66s	0.70s	1.02s
	Bi-RS	1658s	1943s	60589s	15792s	**GNNCert-E**	22.15s	82.21s	26.38s	32.85s
	AGNNCert-E	7.35s	8.36s	44.91s	35.29s	AGNNCert-E	24.34s	82.68s	23.05s	33.14s
	AGNNCert-N	7.45s	8.40s	42.69s	36.41s	AGNNCert-N	22.57s	86.15s	25.45s	32.85s

of Bi-RS. Bi-RS assumes the number of injected nodes is ρ and each node can connect at most τ edges, so the total perturbed edges is $\rho \cdot \tau$. It also involves two hyperparameters p_e and p_n, which means the probability of deleting an edge and deleting a node (and all its connected edges), respectively. These parameters are used to derive the certified perturbation size (see its Eqn 5). In the experiment, we follow Bi-RS by setting $\tau = 5$ and pick its best result from 9 combinations with $p_e = \{0.7, 0.8, 0.9\}$ and $p_n = \{0.7, 0.8, 0.9\}$. Figure 7 shows the comparison results.

- **AGNNCert-E vs Bi-RS:** We first mention the number of injected nodes in AGNNCert-E is calculated by dividing the bounded number of edges in Equation 13 by τ. We can see the two methods have comparable certified node accuracy w.r.t. the number of injected nodes, which indicates AGNNCert-E is already as effective as Bi-RS. Further, we highlight our AGNNCert-E's theoretical result is *deterministic* and *far more general*—it bounds the total number of perturbed edges induced by the node inject attack, where the combination of the number of injected nodes and the number of incident edges for each injected node is arbitrary.
- **AGNNCert-N vs Bi-RS:** We can see AGNNCert-N has much better certified node accuracy than Bi-RS w.r.t. the number of injected nodes (under $\tau = 5$). Furthermore, we highlight that each bounded node in AGNNCert-N can inject as many (even infinite) edges as possible. Hence, the total number of bounded edges in AGNNCert-N could be infinite, which is infinitely higher than Bi-RS's bound when using the total perturbed edges as the evaluation metric.

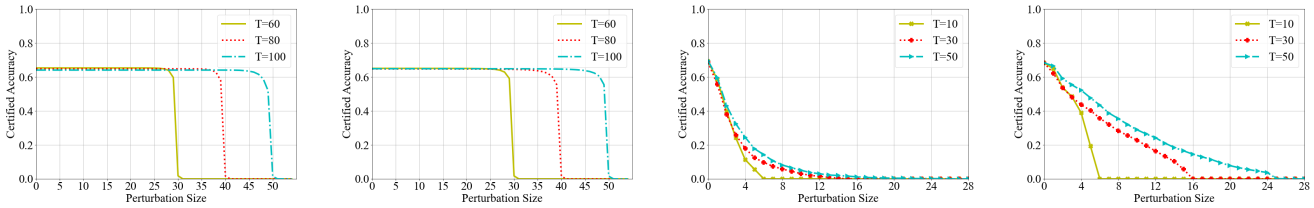
Comparing AGNNCert with GNNCert for graph classification against edge manipulation: Recall that, when using the same hash function and same number of subgraphs in both defenses, AGNNCert-E and GNNCert-E produce the same subgraphs and same voting graph classifier. Hence, their certified graph accuracy/perturbation size are same. Here, we compare AGNNCert-N with GNNCert-E, and results are in Figure 8. We observe both methods have close certified accuracy/perturbation size, implying they have comparable robustness guarantee against edge manipulation.

Comparing AGNNCert with GNNCert against edge AND node feature manipulation: As analyzed in Section 2.3, the initial guarantee of GNNCert is for edge manipulation *or* node feature manipulation. To defend against both manipulations, it requires $T_e = T_n$. Figure 9 shows the comparison results under $T_e = T_n = T$. We can see our AGNNCert performs better than GNNCert-EN. For instance, on PROTEINS, AGNNCert-E can certify a total of 15 perturbed edges by both manipulations, and AGNNCert-N certifies a total of 15 edges and nodes whose features can be arbitrarily perturbed. Instead, GNNCert-EN can only tolerate up to 7 edges and nodes. This may because, compared to AGNNCert, GNNCert-EN generates far more subgraphs (T^2) with each subgraph having less edges and many nodes in subgraphs do not have features (0 values), thus using much less information in the raw graph.

Table 4: Big-O complexity comparison for defense training and certification. We also include the base GNN for completeness. We do not include other complexity factors in training and certification, as they are similar in all defenses. In practice, N_2 can be as large as $100,000$; N_1, T_e, T_n and T have values ≤ 100. Hence $N_2 \gg N_1 \simeq T_e \simeq T_n \simeq T$.

Defenses	Training	Certification
GNN	O(1)	O(1)
Bi-RS	O(N_1)	O(N_2)
GNNCert-E	$O(T_e)$	$O(T_e)$
GNNCert-EN	$O(T_e \cdot T_n)$	$O(T_e \cdot T_n)$
AGNNCert-E	$O(T)$	$O(T)$
AGNNCert-N	$O(T)$	$O(T)$

Comparing the computational complexity and runtime of the defenses: Table 4 shows the Big-O complexity of the compared defenses and the base GNN for training and certification/testing. We only show the factor on the augmented graphs as other factors are similar in all methods. We observe that: 1) As $N_1 \simeq T_e \simeq T_n \simeq T$, all defenses have close Big-O complexity for training (except GNNCert-EN). 2) GNNCert-E has a similar training and certification complexity as ours, but it can only defend against the edge manipulation. 3) Bi-RS is the least efficient for certification due to needing vast samples to ensure high confidence guarantees. 4) All defenses are T slower than the base GNN in training and certification. We also record the runtime in Table 3 and these defenses' runtime matches the observations from the Big-O analysis.



(a) Amazon2M: AGNNCert-N (b) Amazon2M: AGNNCert-E (c) Big-Vul: AGNNCert-N (d) Big-Vul: AGNNCert-E

Figure 10: Certified node/graph accuracy of AGNNCert w.r.t. the number of subgraphs T on Amazon2M and Big-Vul.

5 Evaluations on Real-World Graph Datasets

In this section, we will evaluate AGNNCert on two real-world graph datasets, i.e., Amazon2M co-purchasing dataset [6] for node classification and Big-Vul code vulnerability dataset [15] for graph classification.

5.1 Experimental Settings

Amazon2M dataset is a network representation of products from Amazon, where nodes signify products, and edges indicate two products are frequently purchased together. This dataset consists of 2,449,029 nodes and 61,859,140 edges and is used for node classification — each node has 100 features and is labeled as one of 47 products and the task is to classify products. We divide nodes into 30% for training, 20% for validation, and 50% for testing.

Big-Vul is widely-used code vulnerability dataset, which comprises extensive source code vulnerabilities extracted from 348 open-source C/C++ GitHub projects, spanning from 2002 to 2019. It contains 188,636 C/C++ functions, including 10,900 vulnerable ones (covering 91 vulnerability types), and 7,203 benign ones. Following the recent work [7], we built code graphs by taking code statements as nodes, control-flow or data-flow dependencies as edges and utilizing GraphCodeBERT's [19] token embedding layer to initialize node features. Afterwards, we labeled these code graphs as vulnerable or benign, and formed vulnerability detection problem as a binary graph classification task [7]. We divide the graphs into 80% for training, 10% for validation, and 10% for testing.

We use GCN as the base GNN in AGNNCert (MD5 as the hash function) to train Big-Vul, and cluster-GCN [6] (a more computation- and memory- efficient variant of GCN) as a base GNN in AGNNCert to train the large-scale Amazon2M.

5.2 Experimental Results

Runtime and accuracy: Table 5 shows the training and test time, and test accuracy of AGNNCert and the base GNN on Amazon2M ($T = 80$) and Big-Vul ($T = 30$). We observe that: 1) Test accuracies of AGNNCert and base GNN are close, indicating AGNNCert maintains the utility in these real-world graphs; 2) AGNNCert is about T times slower than the base GNN, again consistent with the Big-O analysis in Table 4.

Table 5: Runtime and test accuracy of AGNNCert and the base undefended GNN on Amazon2M (T=80) and Big-Vul (T=30). As AGNNCert-E and AGNNCert-N have close runtime and test accuracy, we simplicity use AGNNCert for brevity.

Dataset	Method	Train time/epoch	Test time	Test acc.
Amazon2M	**Cluster-GCN**	3.2s	1.1s	0.72
	AGNNCert	287s	107s	0.68
Big-Vul	**GCN**	27.8s	2.3s	0.70
	AGNNCert	827s	65s	0.69

Certified accuracy: Figure 10 reports the certified accuracies of AGNNCert on the two datasets. The results validate that AGNNCert is also an effective defense for safeguarding real-world GNN applications against graph perturbations. For instance, AGNNCert-N can tolerate up to 50 edges and nodes on Amazon2M with arbitrary perturbations; and AGNNCert-E can defend against 24 arbitrarily perturbed edges on Big-Vul.

6 Discussions and Limitations

AGNNCert's performance with larger/powerful GNNs: The certified robustness result is determined by the gap between the most votes (for the correct label) and second-most votes obtained by a GNN on subgraphs. Hence, a GNN making more accurate predictions on subgraphs exhibits better certified robustness. A more powerful/larger GNN may achieve better robustness, as it is expected to provide more accurate predictions. For instance, we test a 6-layer ResGCN [35] on Pubmed, and its certified accuracy is 2% higher than that of the used 3-layer GCN under the same perturbation size.

Node-centric vs. edge-centric AGNNCert: When defending against node perturbations, AGNNCert-N outperforms AGNNCert-E because AGNNCert-N guarantees an infinite number of perturbed edges, whereas AGNNCert-E's guarantee is bounded. However, when defending against edge manipulation attacks, it is hard to say which method is better, as we cannot ascertain which M value (in Equation 13 for AGNNCert-E and Equation 15 for AGNNCert-N) is larger, considering the two methods use distinct graph division strategies.

AGNNCert may be ineffective against training-time attacks on GNNs: The proposed AGNNCert is primarily designed to robustify a *clean* GNN model against *test-time* attacks. Its effectiveness relies on the gap between the most-

votes and second-most-votes be sufficiently large. However, if the GNN model is poisoned [58] or backdoored [63, 70, 75] during training (e.g., a compromised model downloaded from the internet), and our defense is unaware of it, the derived bound may be weakened as the poisoned/backdoored model could reduce this gap. We will leave this as future work.

`AGNNCert` may be ineffective on graph similarity or matching tasks: `AGNNCert` takes a *single* graph as input. However, certain security applications involve a pair of graphs, e.g., GNN-based (binary or source) code similarity analysis [16, 21, 28, 37, 38, 40] takes as input a pair of (e.g., control-flow) graphs generated from the code, and they can be formalized as a graph similarity/matching problem. In this context, an adversary is able to manipulate the source code such that the respective code graph could be largely changed (e.g., many node indexes and edges are changed), while maintaining the code functionality. This attack would make it hard to obtain the one-to-one correspondence between subgraphs generated from the two source graphs. Hence, it is difficult to directly apply `AGNNCert` for certification in this setting.

Inefficiency of `AGNNCert` to large-scale graphs: As shown in Table 4, our `AGNNCert` has a training and certification complexity that is T times of the base GNN's. This overhead becomes significant when applying `AGNNCert` to large graphs (see Table 5). We acknowledge it is important future work to speed up `AGNNCert`, while holding its theoretical results.

7 Related Work

Adversarial attacks on GNNs: Various works [3, 9, 27, 39, 42, 48, 51, 52, 56–58, 62, 66, 72, 75, 80] show GNN classifiers are vulnerable to adversarial perturbations. Given a GNN (node/graph) classifier and a graph, an attacker could inject a few nodes [27, 48], slightly modify the graph structure [9, 66, 80], and/or perturb node features [80] such that the classifier makes wrong predictions for the perturbed graph (in graph classification) or target nodes (in node classification). For instance, [48] utilizes reinforcement learning techniques to design node injection attacks, while [9] designs graph perturbation attacks to both graph and node classification. Most attacks require the attacker fully/partially knows the GNN model (e.g., parameters, architecture), while [42, 56] relaxing this to only have black-box access, i.e., only query the GNN model API. For example, [56] formulates this black-box attack to GNNs as an online optimization with bandit feedback. The original problem is NP-hard and they then propose an online attack based on (relaxed) bandit convex optimization which is proven to be sublinear to the query number.

Defenses against attacks on GNNs: Many empirical defenses [14, 49, 62, 66, 76, 79] were proposed against the adversarial attacks on GNNs. However, these defenses do not have guaranteed performance under the worst-case setting, and were soon broken by adaptive/stronger attacks [43]. Hence, we focus on certified defense in this work.

Certified defenses [1, 25, 26, 30, 54, 64] design robust GNNs that guarantee the same predicted label on clean and perturbed graphs, when the perturbation size (e.g., number of perturbed edges, node features, or injected nodes) on the graph is bounded. [1] and [54] generalized randomized smoothing (RS) [8, 22, 31], the state-of-the-art certified defense against adversarial perturbations on the image domain, to the graph domain and certify any GNN against the edge perturbation. [30] designs a node-aware Bi-RS certified defense against the node injection attack and achieve the state-of-the-art. Further, [64] extended randomized ablation [34], a voting-based defense for image models, to build provably robust graph classifier against the node feature perturbation, edge perturbation, and combined edge and feature perturbations.

However, all existing certified defenses face several limitations. First, except [64] against edge and node feature perturbation, all can only certify one type of perturbation, e.g., edge perturbation. Second, they are only applied for a particular task such as node classification or graph classification, but not both. Adapting these defenses for both tasks would yield unsatisfactory guarantees as shown in our results in Section 4. Third, their robustness guarantees are not 100% (except [64]), implying the guarantee could be inaccurate with certain probability. Our `AGNNCert` addresses all these limitations.

Voting-based certified defenses: Voting is a versatile ensemble method in machine learning (ML) [11] primarily for classification, and each method defines the voter for its own purpose. Recently, voting has been also used to robustify ML models against adversarial attacks, including adversarial image perturbation [33], graph perturbation [36, 64, 70], image patch perturbation [32, 65], text perturbation [44, 74], and data poisoning attacks [23, 24]. The key steps of this type of defense are: divide an input data (e.g., an image, a graph, or a sentence) into a set of sub-data, build a voting classifier to predict all sub-data (each prediction is a vote), and derive the robustness guarantee for the voting classifier. The essential requirement is to ensure only a bounded number of predictions are changed with a bounded adversarial perturbation. The key difference among these defenses is they create problem-dependent sub-data and voters for the majority voting.

8 Conclusion

We study the robustness of GNNs against adversarial attacks. Particularly, we develop `AGNNCert`, the first certified defense for GNNs against arbitrary perturbations (on nodes, edges, and node features) with deterministic guarantees. `AGNNCert` designs novel graph division strategies and leverages the message-passing mechanism in GNNs for deriving the robustness guarantee. The universality of `AGNNCert` makes it encompass existing certified defenses as special cases. Evaluation results validate `AGNNCert`'s effectiveness and efficiency against arbitrary perturbations on GNNs and superiority over the state-of-the-art certified defenses.

9 Acknowledgement

We sincerely thank all the anonymous reviewers and our shepherd for their valuable feedback and constructive comments. We also extend our gratitude to Kexin Pei and Yuede Ji for providing the real-world code vulnerability dataset and conducting the evaluations. This work is partially supported by the National Science Foundation (NSF) under grant Nos. ECCS-2216926, CNS-2241713, CNS-2331302, CNS-2339686, and the Cisco Research Award.

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.

10 Ethics Considerations

This research strictly adheres to ethical guidelines and responsibilities, ensuring compliance with established standards.

1) Identification of Stakeholders

Researchers: Those advancing the field by building upon this work, focusing on both defending GNNs against adversarial attacks and exploring trustworthy GNNs (e.g., against training-time poisoning attacks and both training- and test-time backdoor attacks) in general.
Developers and Practitioners of AI Systems: Individuals and organizations implementing or applying provably robust GNNs in real-world graph-related applications such as fraud detection in social networks, web, online auction networks, intrusion detection, and software vulnerability detection.
End-users: People interacting with GNN-powered systems, including users of social networks, recommender systems, or financial platforms.
Society at Large: Individuals impacted by ethical considerations and risks associated with deploying AI technologies, especially in domains leveraging GNNs (e.g., social networks, healthcare, finance).

2) Potential Risks for Stakeholders and Mitigations

For Researchers. *Potential Risk:* Adversaries may develop novel attacks that surpass the guaranteed bounds of the considered threat model (e.g., perturbations beyond the certified perturbation size). *Mitigation:* With larger perturbations on graph data, those perturbed graphs might have significant differences with normal graphs. Therefore, researchers can leverage detection methods, such as structural-similarity based methods, to identify the perturbed graphs. Researchers can also collaborate with ethics experts to ensure that the research aligns with best practices for responsible AI development.
For Developers and Practitioners. *Potential Risk:* The proposed defense method may not generalize well to other graph learning applications that are different from the considered applications. *Mitigation:* Comprehensive empirical validation across diverse graph datasets and real-world scenarios ensures robustness. Clear communication of limitations will help developers manage risks effectively.
For End-users. *Potential Risk:* Robust GNN mechanisms might inadvertently compromise data privacy or produce biased outcomes. *Mitigation:* Incorporating privacy-preserving (such as differential privacy and cryptographic methods) and fair training techniques enhances data security and fairness.
For Society. *Potential Risk:* Misuse of robust GNNs in critical domains (e.g., healthcare, finance) could exacerbate social inequities, privacy breaches, or manipulation of vulnerable populations. *Mitigation:* Balancing AI security advancements with societal considerations (including fairness, transparency, and accountability) mitigates potential harm. Ethical implications for vulnerable populations will be addressed, prioritizing societal well-being.

3) Considerations Motivating Ethical-Related Decisions

Research Goal: The primary objective is to enhance the robustness of GNNs against adversarial attacks while minimizing potential harm to stakeholders. Defense strategies are designed to be both practical and ethical.
Benefits and Harms: *Benefits:* Improved robustness of GNN systems reduces risks of adversarial manipulation and protecting users. *Harms:* Potential empowerment of malicious actors and overestimating the effectiveness of defense methods.
Rights: We are particularly concerned with privacy rights, as adversarial attacks can sometimes expose sensitive data or violate individuals' privacy. Our defense strategies aim to mitigate such risks, promoting the ethical use of GNNs while safeguarding individuals' rights.

4) Awareness of Ethical Perspectives

We are aware that different members of the research community may hold differing views on the ethical implications of trustworthy AI. Some may prioritize transparency in revealing attack strategies to help build better defenses, while others may argue that such knowledge could be misused. In line with the principles of responsible AI research, we have opted to emphasize defense over offense, focusing on methods that mitigate risk without creating new avenues for harm.

11 Open Science

In compliance with the Open Science Policy, we have made our code, pretrained models, and data openly accessible at https://github.com/JetRichardLee/AGNNCert. Additionally, all artifacts have been published on the Zenodo platform https://zenodo.org/records/14737141 to facilitate the reproduction of the research described in the paper.

Through these efforts, we aim to contribute to the broader scientific community while upholding the highest standards of ethical conduct.

References

[1] Aleksandar Bojchevski, Johannes Gasteiger, and Stephan Günnemann. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *ICML*, 2020.

[2] K.M. Borgwardt, C.S. Ong, S. Schönauer, SVN Vishwanathan, A.J. Smola, and H. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 2005.

[3] Yongqiang Chen, Han Yang, Yonggang Zhang, MA KAILI, Tongliang Liu, Bo Han, and James Cheng. Understanding and improving graph injection attack by promoting unnoticeability. In *ICLR*, 2022.

[4] Dawei Cheng, Xiaoyang Wang, Ying Zhang, and Liqing Zhang. Graph neural network for fraud detection via spatial-temporal attention. *IEEE TKDE*, 2020.

[5] Xiao Cheng, Haoyu Wang, Jiayi Hua, Guoai Xu, and Yulei Sui. Deepwukong: Statically detecting software vulnerabilities using deep graph neural network. *ACM TOSEM*, 2021.

[6] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *KDD*, 2019.

[7] Zhaoyang Chu, Yao Wan, Qian Li, Yang Wu, Hongyu Zhang, Yulei Sui, Guandong Xu, and Hai Jin. Graph neural networks for vulnerability detection: A counterfactual explanation. In *ISTTA*, 2024.

[8] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

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

[10] A.K. Debnath, R.L. Lopez de Compadre, G. Debnath, A.J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

[11] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[12] P.D. Dobson and A.J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *J. of Mol. Bio.*, 330(4):771–783, 2003.

[13] Yingtong Dou, Zhiwei Liu, Li Sun, et al. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *CIKM*, 2020.

[14] Negin Entezari, Saba Al-Sayouri, Amirali Darvishzadeh, and Evangelos Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *WSDM*, 2020.

[15] Jiahao Fan, Yi Li, Shaohua Wang, and Tien N. Nguyen. A c/c++ code vulnerability dataset with code changes and cve summaries. In *ICMSR*, 2020.

[16] Lian Gao, Yu Qu, Sheng Yu, Yue Duan, and Heng Yin. Sigmadiff: Semantics-aware deep graph matching for pseudocode diffing. In *NDSS*, 2024.

[17] Peng Gao, Binghui Wang, Neil Zhenqiang Gong, Sanjeev R Kulkarni, Kurt Thomas, and Prateek Mittal. Sybilfuse: Combining local attributes with global structure to perform robust sybil detection. In *CNS*, 2018.

[18] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[19] Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, et al. Graphcodebert: Pre-training code representations with data flow. In *ICLR*, 2021.

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[21] Haoyu He, Yuede Ji, and H Howie Huang. Illuminati: Towards explaining graph neural networks for cybersecurity analysis. In *EuroS&P*. IEEE, 2022.

[22] Hanbin Hong, Binghui Wang, and Yuan Hong. Unicr: Universally approximated certified robustness via randomized smoothing. In *ECCV*, 2022.

[23] Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *AAAI*, 2021.

[24] Jinyuan Jia, Yupei Liu, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning and backdoor attacks. In *AAAI*, 2022.

[25] Jinyuan Jia, Binghui Wang, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of community detection against adversarial structural perturbation via randomized smoothing. In *WWW*, 2020.

[26] Hongwei Jin, Zhan Shi, Venkata Jaya Shankar Ashish Peruri, and Xinhua Zhang. Certified robustness of graph convolution networks for graph classification under topological attacks. In *NeurIPS*, 2020.

[27] Mingxuan Ju, Yujie Fan, Chuxu Zhang, and Yanfang Ye. Let graph be the go board: gradient-free node injection attack for graph neural networks via reinforcement learning. In *AAAI*, 2023.

[28] Dongkwan Kim, Eunsoo Kim, Sang Kil Cha, Sooel Son, and Yongdae Kim. Revisiting binary code similarity analysis using interpretable feature engineering and lessons learned. *IEEE TSE*, 2022.

[29] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[30] Yuni Lai, Yulin Zhu, Bailin Pan, and Kai Zhou. Node-aware bi-smoothing: Certified robustness against graph injection attacks. In *IEEE SP*, 2024.

[31] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE SP*, 2019.

[32] Alexander Levine and Soheil Feizi. (de) randomized smoothing for certifiable defense against patch attacks. In *NeurIPS*, 2020.

[33] Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defenses against general poisoning attacks. In *ICLR*, 2020.

[34] Alexander Levine and Soheil Feizi. Robustness certificates for sparse adversarial attacks by randomized ablation. In *AAAI*, 2020.

[35] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, 2019.

[36] Jiate Li, Meng Pang, Yun Dong, Jinyuan Jia, and Binghui Wang. Provably robust explainable graph neural networks against graph perturbation attacks. In *ICLR*, 2025.

[37] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *ICML*, 2019.

[38] Jiahao Liu, Jun Zeng, Xiang Wang, and Zhenkai Liang. Learning graph-based code representations for source-level functional similarity detection. In *ICSE*, 2023.

[39] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. Towards more practical adversarial attacks on graph neural networks. In *NeurIPS*, 2020.

[40] Andrea Marcelli, Mariano Graziano, Xabier Ugarte-Pedrero, et al. How machine learning is solving the binary function similarity problem. In *Security*, 2022.

[41] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[42] Jiaming Mu, Binghui Wang, Qi Li, Kun Sun, Mingwei Xu, and Zhuotao Liu. A hard label black-box adversarial attack against graph neural networks. In *CCS*, 2021.

[43] Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? In *NeurIPS*, 2022.

[44] Hengzhi Pei, Jinyuan Jia, Wenbo Guo, Bo Li, and Dawn Song. Textguard: Provable defense against backdoor attacks on text classification. In *NDSS*, 2023.

[45] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–297. Springer, 2008.

[46] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE TNN*, 2008.

[47] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Rad. Collective classification in network data. *AI magazine*, 2008.

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

[49] Shuchang Tao, Huawei Shen, Qi Cao, Liang Hou, and Xueqi Cheng. Adversarial immunization for certifiable robustness on graphs. In *WSDM*, 2021.

[50] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[51] Xingchen Wan, Henry Kenlay, Robin Ru, Arno Blaas, Michael A Osborne, and Xiaowen Dong. Adversarial attacks on graph classifiers via bayesian optimisation. In *NeurIPS*, 2021.

[52] Binghui Wang and Neil Zhenqiang Gong. Attacking graph-based classification via manipulating the graph structure. In *CCS*, 2019.

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

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

[55] Binghui Wang, Jinyuan Jia, and Neil Zhenqiang Gong. Graph-based security and privacy analytics via collective classification with joint weight learning and propagation. In *NDSS*, 2019.

[56] Binghui Wang, Youqi Li, and Pan Zhou. Bandits for structure perturbation-based black-box attacks to graph neural networks with theoretical guarantees. In *CVPR*, 2022.

[57] Binghui Wang, Minhua Lin, Tianxiang Zhou, and more. Efficient, direct, and restricted black-box graph evasion attacks to any-layer graph neural networks via influence function. In *WSDM*, 2024.

[58] Binghui Wang, Meng Pang, and Yun Dong. Turning strengths into weaknesses: A certified robustness inspired attack framework against graph neural networks. In *CVPR*, 2023.

[59] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. Sybilscar: Sybil detection in online social networks via local rule based propagation. In *INFOCOM*, 2017.

[60] Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. Sybilblind: Detecting fake users in online social networks without manual labels. In *RAID*, 2018.

[61] Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, et al. Scalable graph learning for anti-money laundering: A first look. In *NIPSW*, 2018.

[62] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, and more. Adversarial examples on graph data: Deep insights into attack and defense. In *IJCAI*, 2019.

[63] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. Graph backdoor. In *USENIX Security)*, 2021.

[64] Zaishuo Xia, Han Yang, Binghui Wang, and Jinyuan Jia. Deterministic certification of graph neural networks against adversarial perturbations. In *ICLR*, 2024.

[65] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. {PatchGuard}: A provably robust defense against adversarial patches via small receptive fields and masking. In *USENIX Security*, 2021.

[66] Kaidi Xu, Hongge Chen, Sijia Liu, and more. Topology attack and defense for graph neural networks: An optimization perspective. In *IJCAI*, 2019.

[67] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[68] Weizhi Xu, Junfei Wu, Qiang Liu, Shu Wu, and Liang Wang. Evidence-aware fake news detection with graph neural networks. In *WWW*, 2022.

[69] Cheng Yang, Jiawei Liu, and Chuan Shi. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *WWW*, 2021.

[70] Yuxin Yang, Qiang Li, Jinyuan Jia, Yuan Hong, and Binghui Wang. Distributed backdoor attacks on federated graph learning and certified defenses. In *CCS*, 2024.

[71] Ge Zhang, Zhao Li, Jiaming Huang, Jia Wu, Chuan Zhou, Jian Yang, and Jianliang Gao. efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *ACM TOIS*, 2022.

[72] He Zhang, Xingliang Yuan, Chuan Zhou, and Shirui Pan. Projective ranking-based gnn evasion attacks. *IEEE TKDE*, 2022.

[73] Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, Xudong Liu, Chunming Hu, and Yang Liu. Detecting condition-related bugs with control flow graph neural network. In *ISSTA*, 2023.

[74] Xinyu Zhang, Hanbin Hong, Yuan Hong, Peng Huang, Binghui Wang, Zhongjie Ba, and Kui Ren. Text-crs: A generalized certified robustness framework against textual adversarial attacks. In *IEEE SP*, 2024.

[75] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Backdoor attacks to graph neural networks. In *SACMAT*, 2021.

[76] Xin Zhao, Zeru Zhang, Zijie Zhang, Lingfei Wu, Jiayin Jin, Yang Zhou, Dejing Dou, and Da Yan. Expressive 1-lipschitz neural networks for robust multiple graph learning against adversarial attacks. In *ICML*, 2021.

[77] Meihui Zhong, Mingwei Lin, Chao Zhang, and Zeshui Xu. A survey on graph neural networks for intrusion detection systems: Methods, trends and challenges. *Computers & Security*, 2024.

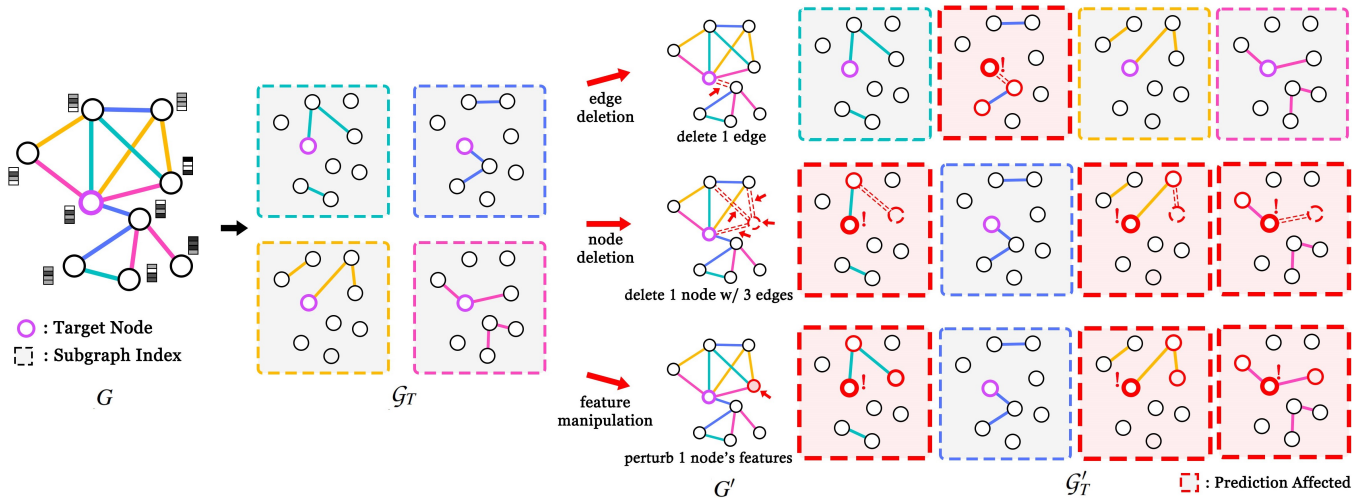
[78] Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In *NeurIPS*, 2019.

[79] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *KDD*, 2019.

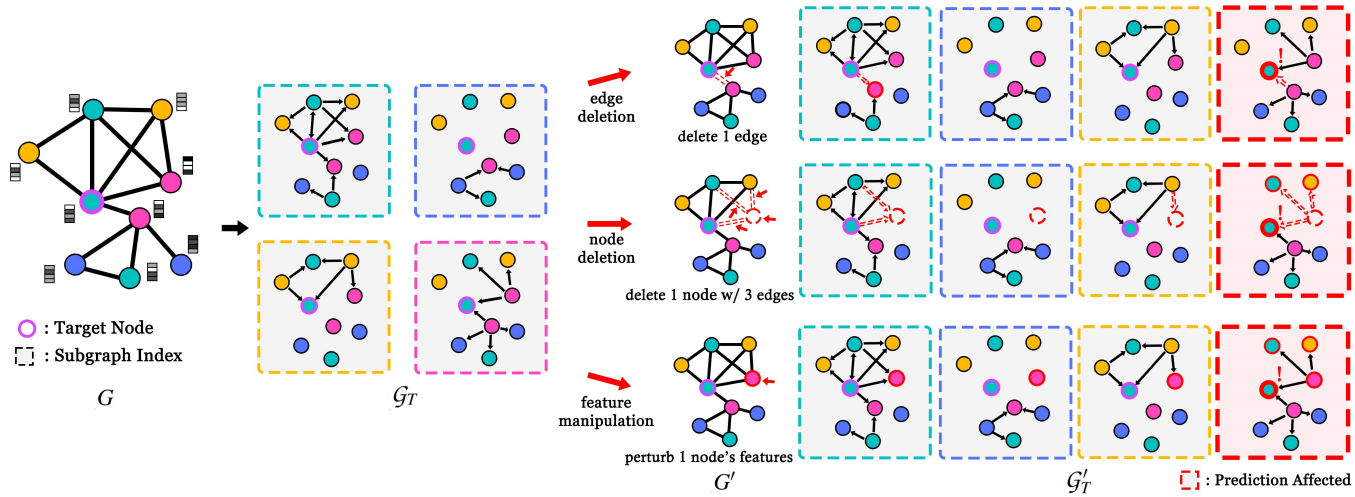
[80] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *KDD*, 2018.

Node Classification					Graph Classification				
Dataset	**Ave Degree**	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{C}\rvert$	**Dataset**	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{V}\rvert_{avg}$	$\lvert\mathcal{E}\rvert_{avg}$	$\lvert\mathcal{C}$
Cora-ML	5.6	2,995	8,416	7	AIDS	2,000	15.7	16.2	2
Citeseer	2.8	3,327	4,732	6	MUTAG	4,337	30.3	30.8	2
Pubmed	4.5	19,717	44,338	3	PROTEINS	1,113	39.1	72.8	2
Amazon-C	71.5	13,752	491,722	10	DD	1,178	284.3	715.7	2
Amazon2M	50.5	2,449,029	61,859,140	47	Big-Vul	18,103	35.5	117.3	2

Table 6: Datasets and their statistics for both Node Classification and Graph Classification.

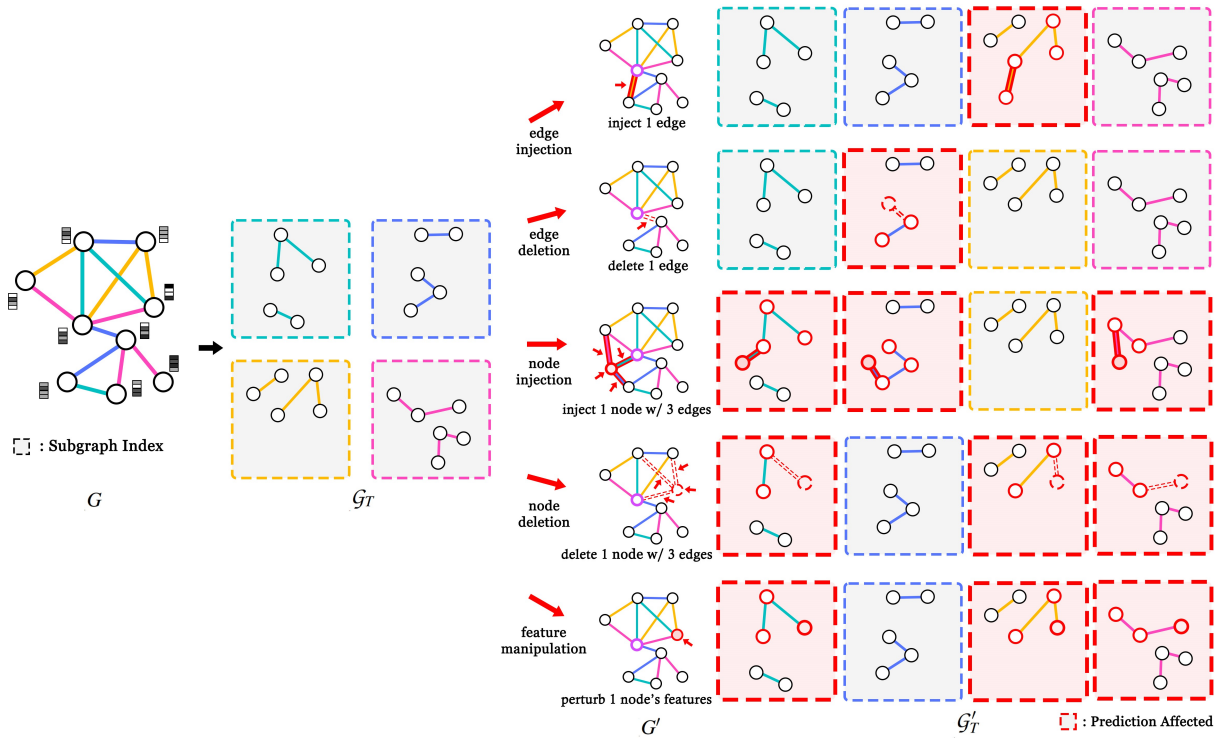


(a) Edge-Centric Graph Division for Node Classification against edge deletion, node deletion and node feature manipulation

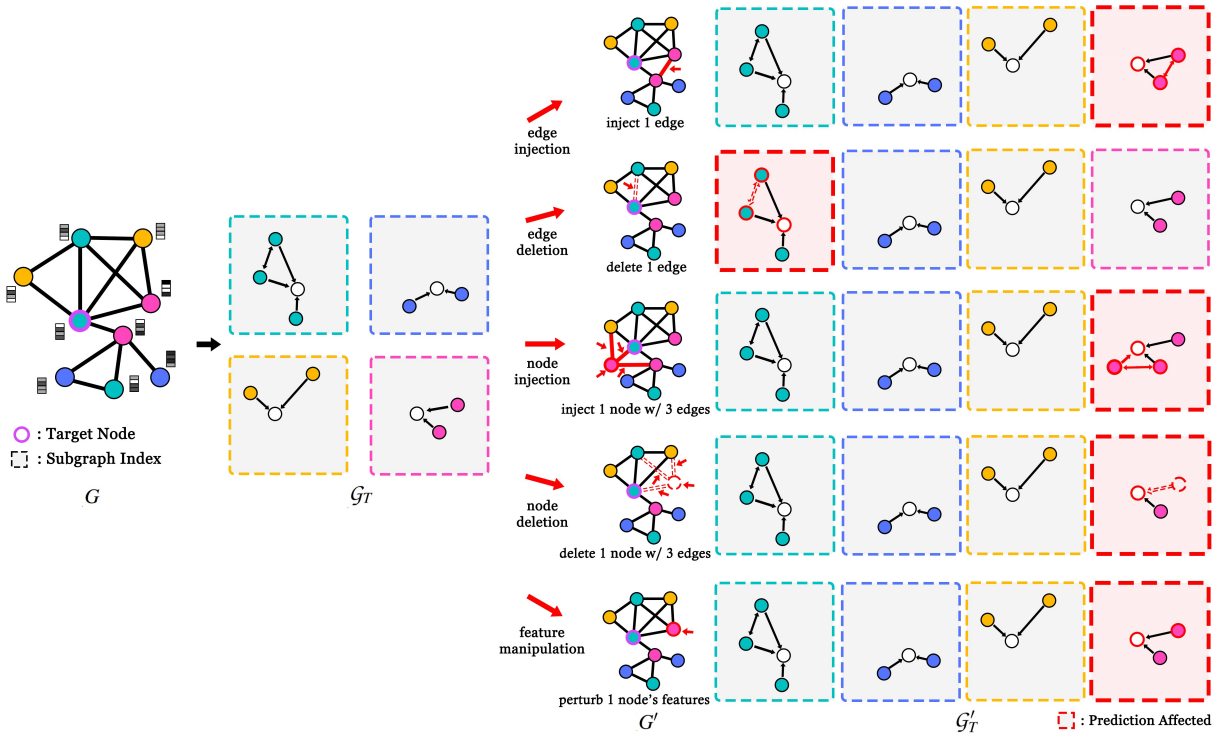


(b) Node-Centric Graph Division for Node Classification against edge deletion, node deletion and node feature manipulation

Figure 11: Illustration of our edge-centric and node-centric graph division strategies for node classification against edge deletion, node deletion, and node feature manipulation. **To summarize:** 1 deleted edge affects at most 1 subgraph prediction in both graph division strategies. In contrast, 1 deleted node with, e.g., 3 incident edges can affect at most 3 subgraph predictions with edge-centric graph division, but at most 1 subgraph prediction with node-centric graph division.



(a) Edge-Centric Graph Division for Graph Classification against edge manipulation, node manipulation and feature manipulation



(b) Node-Centric Graph Division for Graph Classification against edge manipulation, node manipulation and feature manipulation

Figure 12: Illustration of our edge-centric and node-centric graph division strategies for graph classification. The conclusion are similar to those for node classification.