*electronics*

MDPI

*Article*

# A Secure GNN Training Framework for Partially Observable Graph

**Dongdong An [1], Yi Yang [1], Wenyan Liu [2,3], Qin Zhao [1], Jing Liu [4], Hongda Qi [1] and Jie Lian [1,*]**

[1] Shanghai Engineering Research Center of Intelligent Education and Bigdata, Shanghai Normal University, Shanghai 200234, China; andongdong@shnu.edu.cn (D.A.); 1000468002@smail.shnu.edu.cn (Y.Y.); q_zhao@shnu.edu.cn (Q.Z.); hongda_qi@shnu.edu.cn (H.Q.)
[2] Ant Group, Hangzhou 310023, China; enxuan.lwy@antgroup.com
[3] College of Computer Science and Technology, Zhejiang University, Hangzhou 310058, China
[4] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China; jliu@sei.ecnu.edu.cn
* Correspondence: lianjie@shnu.edu.cn

**Abstract:** Graph Neural Networks (GNNs) are susceptible to adversarial injection attacks, potentially compromising the model integrity, reducing accuracy, and posing security risks. However, most of the current countermeasures focus on enhancing the robustness of GNNs rather than directly addressing these specific attacks. The challenge stems from the difficulty of protecting all nodes in the entire graph and the agnostic of the attackers. Therefore, we propose a secure training strategy for GNNs that counters the vulnerability to adversarial injection attacks and overcomes the obstacle of partial observability in existing defense mechanisms—where defenders are only aware of the graph's post-attack structure and node attributes, without the identification of compromised nodes. Our strategy not only protects specific nodes but also extends security to all nodes in the graph. We model the graph security issues as a Partially Observable Markov Decision Process (POMDP) and use Graph Convolutional Memory (GCM) to transform the observations of a POMDP into states with temporal memory proceeding to use reinforcement learning to solve for the optimal defensive strategy. Finally, we prevent learning from malicious nodes by limiting the convolutional scope, thus defending against adversarial injection attacks. Our defense method is evaluated on five datasets, achieving an accuracy range of 74% to 86.7%, which represents an enhancement of approximately 5.09% to 100.26% over post-attack accuracies. Compared with various traditional experimental models, our method shows an accuracy improvement ranging from 0.82% to 100.26%.

**Keywords:** Graph Neural Networks; adversarial injection attack; partial observability; reinforcement learning

check for updates

**Citation:** An , D.; Yang, Y.; Liu, W.; Zhao, Q.; Liu, J.; Qi, H.; Lian, J. A Secure GNN Training Framework for Partially Observable Graph. *Electronics* **2024**, *13*, 2721. https://doi.org/10.3390/electronics13142721

Academic Editor: Gemma Piella

Received: 4 June 2024
Revised: 25 June 2024
Accepted: 4 July 2024
Published: 11 July 2024

**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

Graph Neural Networks (GNNs) have revolutionized the analysis of graph-based data. They outperform traditional neural network techniques in fields such as social network analytics [1], recommender systems [2,3], and bioinformatics [4]. The emergence of variants such as graph convolutional networks and graph attention networks has both deepened the understanding of graph data processing and spawned novel applications, enhancing the status of GNNs as a cornerstone in deep learning.

However, GNNs have been proven to be susceptible to adversarial injection attacks [5], which aim to deceive the model during the training phase by injecting maliciously crafted data to mislead the learning trajectory. These attacks have revealed GNN vulnerabilities, especially when facing carefully designed malicious modifications [6]. A common threat posed by adversarial injection is the ease with which such models can be misled by nuanced alterations to the graph data, such as the addition of nodes or edges, potentially leading to misclassification. In practical scenarios, a key challenge faced by defenders is the inability

to fully observe the overall state of the graph. These malicious attacks may be difficult to detect directly because the defenders cannot know exactly which nodes or edges have been compromised. This partial observability makes designing effective defense strategies more complex. For example, a financial system that incorrectly categorizes high-risk individuals as low risk due to such manipulations may fall prey to fraudulent activities. As shown in Figure 1, prior to any attack, the risk assessment system is capable of accurately evaluating user risk levels, effectively distinguishing between high-risk and low-risk users. However, following an adversarial injection attack, wherein attackers manipulate data by transacting with or establishing associations with other users, the model becomes predisposed to erroneously assessing the user's risk level, and it is difficult to detect the attackers, thus successfully compromising the integrity of the assessment.

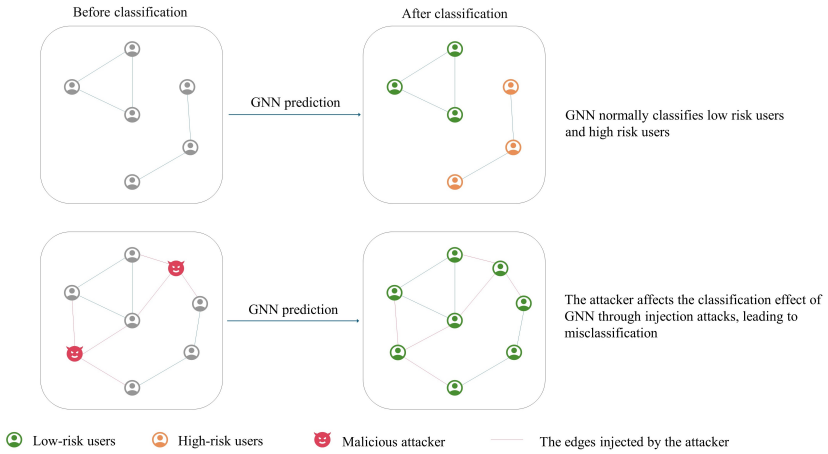



**Figure 1.** Adversarial injection attacks: an example of risk assessment in the financial sector.

There are also some studies that address this issue. For instance, Santos et al. [7] primarily focus on graph structure recognition, but by handling noise and feature normalization, they enhance robustness and stability under partially observable conditions. Similarly, Machado et al. [8] utilize causal inference through training convolutional neural networks, demonstrating strong generalization capabilities in both dense and sparse networks and at different noise levels, thereby enhancing model robustness. However, these methods do not specifically propose solutions for this particular problem.

Our work centers on the challenges of adversarial defense within semi-supervised node classification tasks, with the goal of bolstering the model's security across the entire graph. The main challenges are as follows: (1) Obtaining a secure training model in the face of adversarial injection attacks on GNNs is a challenge. Many current methods focus on their own robustness, which may not be effective against adversarial injection attacks. Therefore, a secure GNN training method is crucial to address this issue. (2) In real-world scenarios, defenders cannot protect individual user nodes, nor can they know which nodes in the graph are maliciously attacked. Usually, defenders can only observe the graph post-attack. Thus, designing effective defense strategies under partial observability conditions is challenging.

To address these challenges, we propose a secure training approach for GNNs based on reinforcement learning. Figure 2 is an illustrative example of our secure training process. First, we cast graph security issues as a Partially Observable Markov Decision Process (POMDP). POMDPs facilitate the management of the indeterminacies that stem from attacker actions and system state ambiguities. We then employ Graph Convolutional Memory (GCM) to restructure POMDP observations into states with embedded temporal memory, setting the stage for reinforcement learning algorithms to derive an optimal defensive strategy. Our solution features a custom convolutional scope controlled by a unique mask layer preceding convolutional layers, fostering a more secure GNN framework. Experimental results show the efficacy of our method via evaluation on four citation

networks and one social network dataset, with accuracy improvements of approximately 0.82% to 100.26% over comparable adversarial defense models. Furthermore, precision-based assessment showcases both the accuracy and security topologies facilitated by our defense mechanism on original, post-attack, and post-defense data.

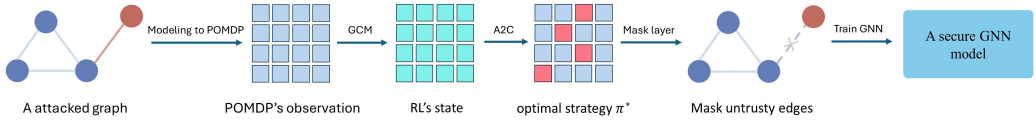



**Figure 2.** An illustrative example of our secure training process.

In summary, the main contributions of this paper are as follows:

1. To address the challenges posed by unknown attackers and the need to defend all nodes, we model graph security as a POMDP to aptly manage the unpredictable nature of adversarial threats and uncertain system conditions, offering a robust defense mechanism for all nodes.
2. To address the difficulties in solving a POMDP, we utilize Graph Convolutional Memory (GCM) to transform the observations of a POMDP into states with temporal memory and ultimately use reinforcement learning algorithms to solve for the optimal strategy.
3. We introduce a mask layer positioned before the convolutional layers, which offers a list of masks capable of setting mask values for all edges. Guided by an optimal strategy, this allows for controlling the scope of convolution, thereby achieving a secure GNN.
4. We evaluate our method on four citations and one social network dataset and compare it with adversarial defense methods such as MedianGCN, RobustGCN, and AirGCN. The experimental results show that our method can effectively improve the security of GNN models against adversarial injection attacks. After employing our defense method, an accuracy of 74% to 86.7% was achieved, which is an improvement of approximately 5.09% to 100.26% compared with the accuracy after the attack.

## 2. Related Works

### *2.1. Adversarial Graph Neural Networks*

The powerful graph data analysis capabilities of Graph Neural Networks (GNNs) originate from the mutual transfer and aggregation of information between entities, and they have been widely applied in many real-world applications. Vulnerabilities in GNN algorithms, when applied in real-world scenarios, could lead to severe consequences and pose security risks. Currently, the security training methods for GNNs can be divided into two categories: based on model data security and training method security.

Adversarial attacks on GNNs can be categorized into node-level attacks and graph-level attacks. Node-level attacks are adversarial attacks targeting node-level classification tasks, aiming to mislead the classifier by modifying the features or connections of specific nodes or their neighboring nodes, resulting in the misclassification of specific nodes. For example, in a social network, an attacker modifies a user's interest tags or friend relationships, causing them to be misclassified into a different interest category. In contrast, graph-level attacks are adversarial attacks targeting graph-level classification tasks, aiming to mislead the classifier by perturbing the features or structure of the entire graph or its subgraphs, resulting in the misclassification of the entire graph. For example, in a drug molecular graph, an attacker modifies the features of certain atoms or the bonds between them, causing the entire molecule to be misclassified into a different pharmacological category. Given the focus of this paper, we will primarily discuss node-level attacks and the corresponding defensive mechanisms.

Based on Model Data Security: Model data security primarily involves adversarial disturbance cleaning, injecting adversarial training examples, etc., by approaching decision

boundaries and adversarial boundaries to avoid the impact of adversarial injection attacks and improve security. Ioannidis et al. [9] introduced the Pruned Graph Scattering Transform (pGST), which utilizes a graph spectrum-based criterion to guide the pruning algorithm, preserving informative scattering features in real time while avoiding the exponential complexity associated with traditional Graph Scattering Transforms (GSTs). This pruning strategy reduces computational complexity, thus lowering potential security risks during data processing. Ioannidis et al. [10] introduced a Tensor Graph Convolutional Network (TGCN). TGCN, by incorporating learnable weights and graph-based regularizers, adapts to different relationships in the graph. These designs help the model better process and recognize data structures and associations in multi-relational graphs, thereby improving robustness and data security in adversarial attack scenarios. Dai et al. [11] proposed adversarial training methods in graph domains, creating disturbances by randomly adding or removing edges in the adjacency matrix and updating model parameters to minimize prediction loss, thus enhancing the robustness of graph models. Similarly, Xu et al. [12] improved adversarial examples by computing projection gradients for discrete graph data, adding disturbances through node similarity. Abusnaina et al. [13] used methods such as anomaly detection, exploring the intrinsic differences between adversarial and safe edges and nodes to protect graph neural models. However, the process of securing the data is still affected by the potential flaws in the algorithm, so the data security depends on the effectiveness of the algorithm.

Based on Model Training Method Security: This mainly targets verification during the training process, using methods such as SAT verification and interval verification to ensure that the training is secure. For example, Angne et al. [14] proposed a model-checking method that performs exhaustive searches on networks under limited states to verify satisfiability problems of dependent attributes, studying simulated GNNs to solve NP-complete problems, using SAT formulas to visualize GNNs and describe corresponding properties. Bunnel et al. [15] utilized branching strategies to recursively divide the search space into smaller spaces, using boundary methods to calculate boundaries for each divided space and tightening the boundaries of the final objective function over the entire search space to ensure the verified model's security. However, this method is beset with efficiency challenges due to the sheer number of states requiring exploration, leading to substantial time and resource utilization during the search process, particularly for large-scale GNNs. Wang B. et al. [16] extended a technique known as randomized smoothing to graph data. By adding random noise to the test samples (i.e., graphs), any base classifier (such as the GNN classifier in our problem) can be transformed into a robust classifier. A classifier that can prove to predict the same label when an attacker adds/removes up to K edges is provably robust. Zhang et al. [17] proposed GNNGuard, a universal defense method against various training-time attacks that disturb the discrete graph structure. GNNGuard can be directly integrated into any GNN. Its core principle involves detecting and quantifying the relationship between a graph structure and node features, and then leveraging this relationship to mitigate the negative impacts of attacks.

The advantage of model data security lies in achieving better data security effects at a lower cost, thus ensuring the safe operation of GNN models, but there may be flaws in the algorithm that lead to security vulnerabilities. Therefore, this paper chooses to focus on the model's training methods.

### *2.2. Partially Observable Models*

Solving POMDPs is usually difficult due to several key factors. First, the complexity of the state space and the uncertainty from incomplete observations make it hard for agents to make decisions. Agents cannot see the complete state of the environment and must rely on partial information, which requires estimating and maintaining a belief state—a guess of the possible true states. Additionally, POMDPs often involve continuous state and action spaces, adding to the computational complexity. Finding optimal strategies requires advanced algorithms and approximation methods to manage the high computational de-

mands. These challenges make solving POMDPs a complex task. Marcus Hoerger et al. [18] introduced a new online solver for POMDPs called Lazy Belief Extraction for Continuous Observation POMDPs (LABECOP). LABECOP lazily extracts sequences of beliefs and their corresponding action values and action choice strategies, meaning they only extract for the action-observation sequences of events currently sampled. This enables LABECOP to consider every sampled observation encountered during the planning process, and thus consider every belief sequence without discretizing the observation space. This method is more effective in computing robust strategies for problems with continuous observation spaces. Robert K. Helmeczi et al. [19] proposed a solution based on Linear Programming (LP), which introduces a grid-based approximation method and LP model to generate approximate strategies for Constrained Partially Observable Markov Decision Processes (CPOMDPs), covering both finite and infinite time horizons. This method can effectively generate approximate strategies and provides a flexible way to incorporate various constraints into the model. Steven D. Morad et al. [20] introduced Graph Convolutional Memory (GCM), the first Reinforcement Learning (RL) memory framework with switchable task-specific priors. GCM uses human-defined topological priors to form graph neighborhoods and combines them into a larger network topology. By querying the graph with graph convolution, it merges relevant memories into past context-related summaries, and GCM outperforms the current state-of-the-art methods in control, memory, and navigation tasks with fewer parameters.

### *2.3. Safety of Neural Network Training Processes*

Currently, the safety of neural network training processes has become a new method for defending and verifying model security, ensuring that the model is safety oriented during training through modifying the model structure, auxiliary verification, and other means. Kissel et al. [21] proposed Safe Regularization, combining the training of neural networks with auxiliary optimization objectives, such as obtaining structured weight matrices, to form a dictionary sorting optimization problem, achieving better robustness of hyperparameters through this method. Zhao et al. [22] proposed a safe neural network controller, whereby, verifying safety properties, we build barrier neural networks with barrier functions, train both the controller neural network and the barrier neural network, and realize a cycle of verification and synthesis, thereby obtaining a safe network model. Pauli et al. [23] presented an optimization scheme based on the Alternating Direction Method of Multipliers (ADMM), which minimizes the training loss of neural networks, thus forming a semi-definite programming-based training process and improving robustness.

## 3. Preliminary

This section introduces the basic knowledge of theories involved in this paper, mainly including concepts related to adversarial graph neural networks and reinforcement learning strategies.

### *3.1. Adversarial Graph Neural Networks*

Graph Neural Networks (GNNs), as a deep learning algorithm specifically for modeling and learning graph-structured data, have demonstrated outstanding performance in the fields of graph data analysis, node classification, and graph classification [24,25]. As GNNs are widely deployed in various application areas, the problem of adversarial attacks they face is becoming increasingly prominent. Adversarial attacks aim to induce incorrect predictions or classifications by making precise and minimal modifications to the model's inputs. In this context, research on adversarial graph neural networks is divided into two main categories: adversarial attacks [5,6] and their defensive mechanisms [26].

1. Adversarial Attacks: Adversarial injection attacks on graph neural networks mainly include two types: node-level attacks and graph-level attacks. Node-level attacks aim to perturb the features of one or multiple nodes, affecting the model's node classification or link prediction performance. Graph-level attacks focus on modifying

the structure or attributes of the entire graph to mislead the model's judgment in tasks such as graph classification.

2. Adversarial Defense: In response to adversarial attacks, researchers have proposed various defensive strategies to enhance the robustness of GNNs, including adversarial training, defense based on graph structure, preprocessing of graph data, and detection and repair of adversarial samples. Adversarial training involves introducing adversarial samples into the training process to improve the model's resistance. Defense strategies based on a graph structure use the inherent structural properties of graphs to block attacks, while preprocessing methods clean input data to reduce adversarial disturbances. In addition, adversarial sample detection strives to identify and handle malicious disturbances in the model's inputs.

### *3.2. Reinforcement Learning*

Reinforcement Learning (RL) is a machine learning paradigm centered on interactive learning. Its goal is to optimize the behavior decision-making strategy of an agent within a certain environment to maximize the cumulative reward received. Under the reinforcement learning framework, an agent explores different action choices under various states and adjusts its behavior strategy based on the reward signal fed back by the environment, gradually improving its strategy. As shown in Figure 3, a reinforcement learning system is mainly composed of the following elements:

1. Environment: The external world where the agent learns and makes decisions, with interactions between the agent and the environment involving information such as state, action, and reward.
2. Agent: The main body of learning and decision making, choosing the appropriate action based on the current state and obtaining reward signals through interactions with the environment.
3. State: Describes the observational information of the environment and is the basis for the agent's decision making.
4. Action: The operations or decisions that an agent can perform in a specific state.
5. Reward: The immediate feedback signal given by the environment, used to evaluate the quality of an action.
6. Policy: The goal of the agent's learning, which is the mapping rule from state to action, determining the actions the agent should take in different states.

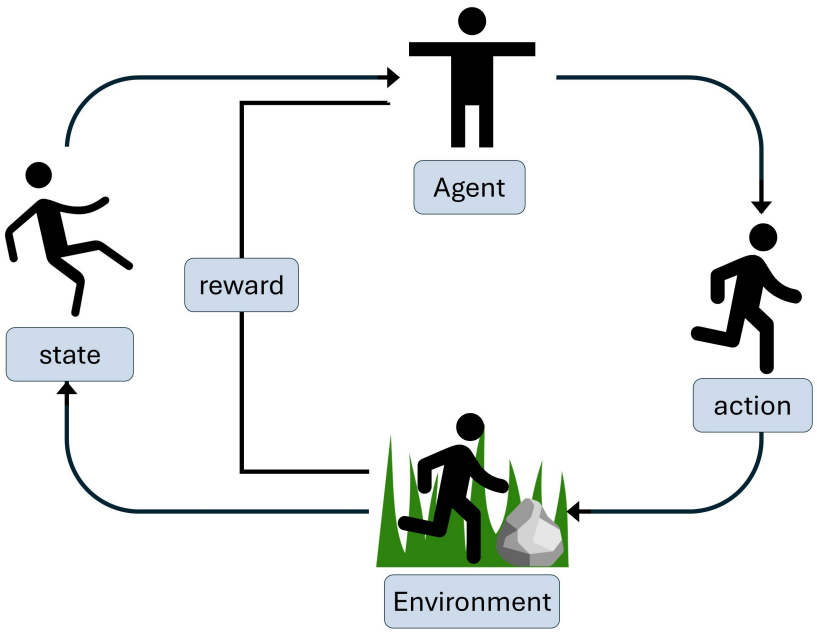



**Figure 3.** Reinforcement learning model.

Through the delineation of the state space, action space, transition probabilities, and reward function, we construct the framework for a POMDP. Leveraging reinforcement learning algorithms, we conduct a policy, which determines the actions taken, evaluation, and optimization based on environmental feedback (rewards) and state transition probabilities, aiming to identify the optimal strategy. In adversarial reinforcement learning scenarios, agents must acquire adaptive strategies amidst interference from adversaries to maintain robustness and sustain optimal performance in adversarial environments.

### *3.3. Partially Observable Markov Decision Process*

A Partially Observable Markov Decision Process (POMDP) is used to describe decision-making processes where the state of the environment is not fully observable. In POMDPs, agents cannot directly observe the entire state of the environment but instead make decisions based on partially observed information. This characteristic of incomplete observation requires agents to make decisions by inferring the state of the environment, which usually involves the use of a belief or hidden state to represent the estimated probability distribution over the environmental states. It is defined by the seven-tuple $(S, A, T, R, \Omega, O, \gamma)$, where $S$ is the state space, representing the set of all possible states the system might be in; $A$ is the action space, representing the set of all actions that the agent can take; $T : S \times A \times S \to [0, 1]$ is the state transition function, describing the probability of the system transitioning from state $s \in S$ to state $s' \in S$ after taking an action $a \in A$; $R : S \times A \to \mathbb{R}$ is the reward function, describing the immediate reward received by the agent after taking action $a \in A$ in state $s \in S$; $\Omega$ is the observation space, representing the set of all possible observations the agent can make; $O : S \times A \times \Omega \to [0, 1]$ is the observation function, describing the probability of the agent observing $o \in \Omega$ in state $s \in S$ after taking an action $a \in A$; and $\gamma \in [0, 1]$ is the discount factor, used to balance the importance of immediate versus future rewards. POMDPs are often used to model decision-making problems with incomplete information, such as in areas like robotic navigation, resource management, and game strategies.

## 4. Method

Our primary research addresses the security concerns of semi-supervised node classification tasks in GNNs. Given that GNNs are vulnerable to adversarial injection attacks and considering the partial observability of adversarial defense methods—where defenders are only aware of the graph structure and node attributes after the attack, without knowledge of which nodes were adversarially injected—we propose a GNN secure training strategy operable under a solely observable graph structure and node attributes. This strategy does not just protect a specific node; instead, it takes into account the security of all nodes across the entire graph. The steps are shown in Figure 4 and can be mainly divided into three parts: (1) model the problem as a POMDP model; (2) use reinforcement learning to solve the optimal strategy of the POMDP; and (3) control the convolutional scope based on the optimal strategy, thereby obtaining a secure training process. Specifically, first, the graph security issues are converted into a POMDP. Then, using Graph Convolutional Memory (GCM), the observation values $O$ of a POMDP are transformed into a state with temporal memory, denoted by $\widehat{s_t}$. After that, $\widehat{s_t}$ is taken as the state in reinforcement learning to solve for the optimal strategy of the POMDP problem. Finally, the convolutional scope is controlled by the optimal strategy to avoid learning from malicious nodes.

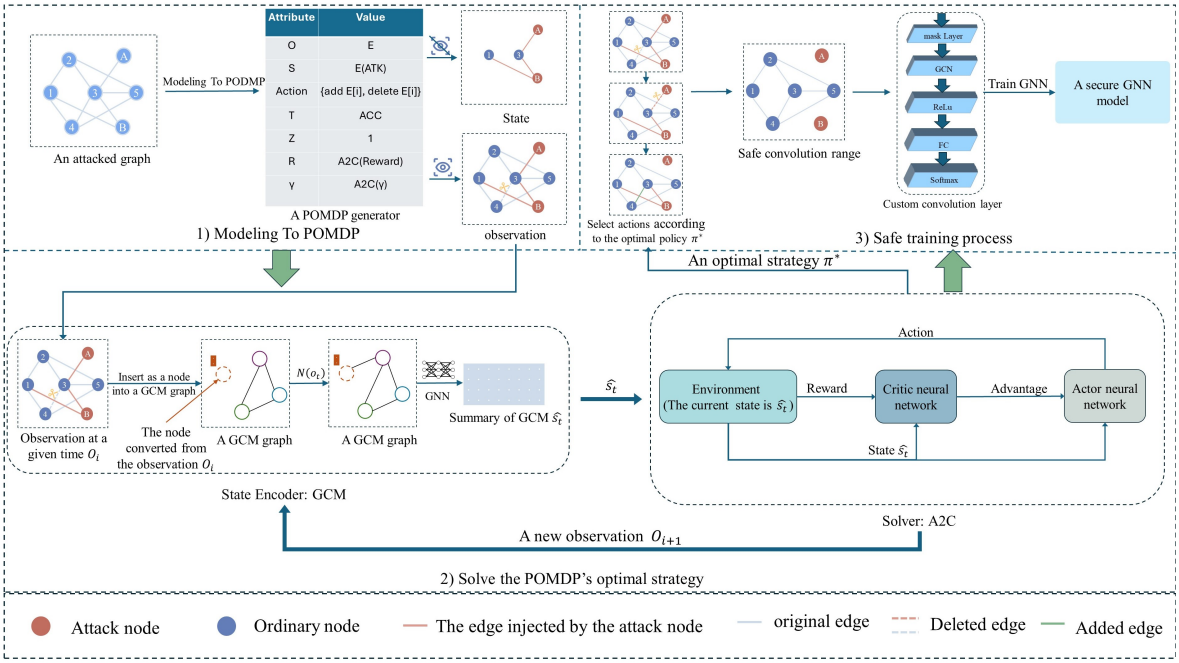



**Figure 4.** Secure GNN training process overview.

### *4.1. Definition of Problem*

We focus on the semi-supervised learning task for undirected graph classification. We define an undirected graph $G = (V, E)$, where $V = v_1, v_2, \ldots, v_n$ is the set of nodes, and $E = e_1, e_2, \ldots, e_n$ is the set of edges. The graph $G$ has a total of $N$ nodes. Moreover, to capture the structure of the graph, we define $G = (A, X)$, where $A \in \{0,1\}^{N \times N}$ is a binary adjacency matrix. $A_{ij} = 1$ indicates the existence of an undirected edge between the nodes $v_i$ and $v_j$, while $A_{ij} = 0$ indicates that there is no edge between the nodes. $X \in \mathbb{R}^{N \times F}$ is the feature matrix that represents the features of all nodes, where the i-th row of the feature matrix $X$ represents the $F$-dimensional feature vector of the node $v_i$. Each node has a label, $y_i \in Y = \{c_1, c_2, \ldots, c_k\}$, where $k$ is the number of classes. In summary, we define the dataset $D = (A, X, Y)$, where $Y = y_1, y_2, \ldots, y_N$. GNNs are trained to obtain a classifier $f_\theta : V \to Y$ based on labeled and unlabeled nodes. Given that GNNs may be vulnerable to poisoning attacks, attackers introduce a small perturbation $\Delta$ to the original structure $A$ to create $A' = A + \Delta$. The resulting dataset becomes $D' = (A', X, Y)$. In this case, a classifier $f'_\theta$ trained on the perturbed dataset $D'$ fails to correctly predict the node classification $y$ for the node $v$. Using the message-passing mechanism of GNNs and existing GNN models as input, our goal is to learn a structure for message passing that mitigates the negative impact of the attack. Based on the above symbolic definition, we define the problem of defending against graph poisoning attacks: In the face of poisoning attacks, attackers apply a perturbation $\Delta$ to the original graph $G$, and the perturbation $\Delta$ is unknown. This leads to a perturbed graph $G'$, which reduces the performance of the GNN model. Therefore, we need to find a new GNN $f'_\theta$ that considers the safety design of all nodes in the graph.

Based on the above symbolic definition, we define the problem of defending against graph poisoning attacks: In the face of poisoning attacks, attackers apply an unknown perturbation $\Delta$ to the original graph $G$, resulting in a perturbed graph $G'$ that reduces the performance of the GNN model. We need to find a new GNN $f'_\theta$ that considers the safety design of all nodes in the graph.

$$\min \left[ f_{\theta'}(G') - f_\theta(G) \right] \tag{1}$$

### *4.2. POMDP Modeling*

In fact, adversarial injection attacks might inject multiple attack nodes simultaneously, necessitating defense mechanisms for all nodes. A black-box state with multiple attack nodes fits better as the state space of a POMDP since it is directly unobservable. Moreover, as the evaluation of transitioning from the state $s_1$ to $s_2$ relies on accuracy, the model's accuracy serves as an indirect observation of the system state rather than providing direct state information. In this context, accuracy acts as an observation outcome, reflecting potential changes in state. Thus, the collection of edges between nodes is more suitable as the observation space for a POMDP, offering directly obtainable information about the current state. Given the problem's indirect observation characteristic and inherent uncertainty, a POMDP offers a more fitting framework to model and address this issue.

To address the above issues, we model graph security issues as a POMDP. As shown in Table 1, in our POMDP, the state space $S = s_1, s_2, s_3, \ldots, s_n$ represents the attacks by malicious nodes, the action space Action $= \{\text{add edge}, \text{remove edge}\}$, and the observation space $O = o_1, o_2, o_3, \ldots, o_n$, where $o_i$ is the set of edges between nodes in the graph. We take actions $a_t \in$ Action from policy $\pi$, transition to the next state $s_{t+1}$ according to the transition probability $T(s_t, a_t) : S \times \text{Action} \to S$, and receive a reward $R(s_t, a_t) : S \times \text{Action} \to \mathbb{R}$. In general, the transition process of states and observations in a POMDP can be described as follows:

**Table 1.** POMDP definition.

| Attribute | Value |
|---|---|
| O | E |
| S | E(ATK) |
| Action | {add E[i], remove E[i]} |
| T | ACC |
| Z | 1 |
| R | A2C(Reward) |
| $\gamma$ | A2C($\gamma$) |

In order to address the challenges presented, we propose a framework that formulates the graph security issues as a POMDP. The proposed POMDP framework, delineated in Algorithm 1, encompasses the following components:

- An observation space $O = \{o_0, o_1, o_2, \ldots, o_n\}$, where each $o_i$ signifies the set of inter-node edges within the graph;
- A state space $S = \{s_0, s_1, s_2, \ldots, s_n\}$, representing the subset of edges clandestinely injected by adversarial nodes, which remain unobservable;
- An action space Action $= \{\text{add edge}, \text{remove edge}\}$, delineating the permissible modifications to the graph's topology;
- A transition probability $T(s', s, a)$, which is instantiated by the classification accuracy ACC, corresponding to the graph's current state, postulating that actions leading to an enhancement in ACC are associated with a higher transition probability to the resultant state;
- An observation probability $Z$ is set unequivocally to 1, ensuring that any action taken results in a deterministic observational outcome;
- A reward function $R$ that aligns with the reward parameter derived from the Actor–Critic (A2C) reinforcement learning algorithm, quantifying the reward obtained upon transitioning from the state $s$ to the state $s'$ via the action $a$;
- A discount factor $\gamma$, ascribed from the A2C algorithm, employed to calibrate the significance accorded to immediate versus prospective rewards. A value of $\gamma$ proximal

to 1 ($\gamma < 1$) signifies a predilection for long-term strategies and the consequential future rewards.

The operational mechanism of the POMDP model engages in selecting an action $a_t \in$ Action predicated on the policy $\pi$, ensuing a state transition to $s_{t+1}$ according to the defined transition probability $T(s_t, a_t) : S \times \text{Action} \rightarrow S$, and acquiring a reward $R(s_t, a_t) : S \times \text{Action} \rightarrow \mathbb{R}$. This is succeeded by an update to the policy $\pi$ predicated on the obtained reward. Concisely, the state and observation transition dynamics within the POMDP framework are articulated as follows:

For a current state $s_t$, executing an action $a_t$ leads to a transition to the next state $s_{t+1}$ according to the following transition probability $T$:

$$s_t \times (\pi(a_t|s_t) \times a_t) \times T(s_{t+1}|s_t, a_t) \rightarrow s_{t+1} \tag{2}$$

For a current observation $o_t$, executing an action $a_t$ results in a change in the observation to $o_{t+1}$, as follows:

$$o_t \times (\pi(a_t|o_t) \times a_t) \rightarrow o_{t+1} \tag{3}$$

This means that a state corresponds to multiple observations. For a state at a certain moment, executing an action may or may not change the state, but the corresponding observation will definitely change.

As shown in Figure 5, we present a simple example to introduce our POMDP model for the graph security problem: For the initial state $s_0$, the state $s_0$ is defined by the set of edges injected by the attacked nodes A and B, and its corresponding observation is $o_0$, where $o_0$ is the set of all edges in the graph. Now, executing an action "remove e(3,4)" and then evaluating that this edge may not be one injected by the attacker, our state $s_0$ does not change. However, removing an edge changes the observation $o_0$ to $o_1$. Similarly, at this moment, executing the action "remove e(A,3)" and evaluating this edge as potentially dangerous and injected by the attacker, the state $s_0$ will change to $s_1$, and the observation $o_1$ will change to $o_2$. If we obtain the optimal strategy for the POMDP, then, as illustrated, we will end up with a secure graph that has removed the edges maliciously injected by the attack nodes.

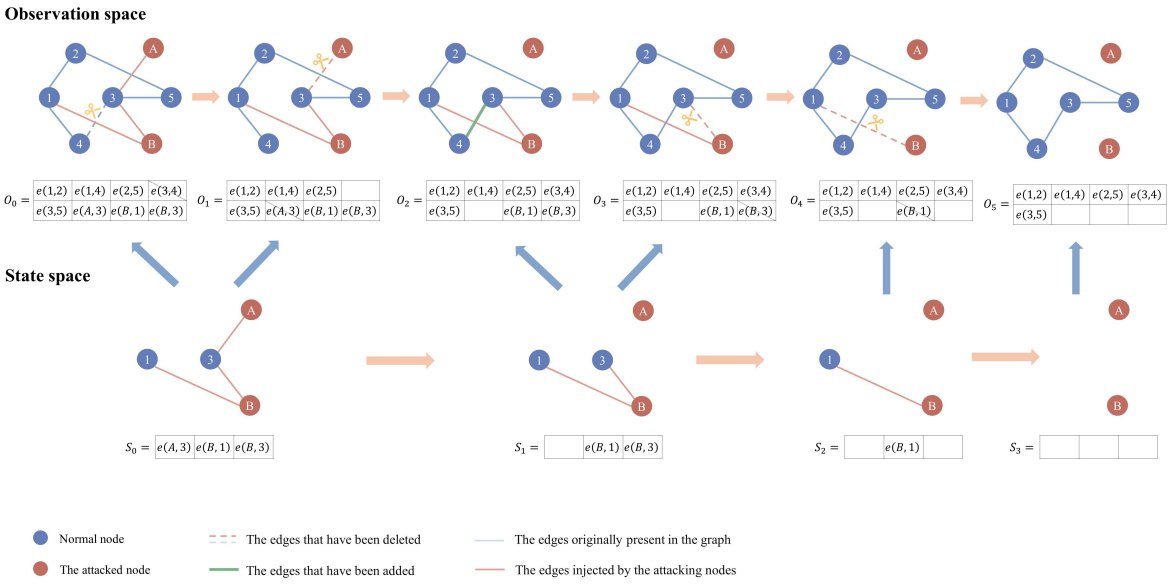



**Figure 5.** Modeling example of graph security issues based on a POMDP.

It should be noted that the state space is not directly observable to us, meaning the attack nodes A and B, and in reality, we do not know if they are attack nodes and what

their attack actions are. To facilitate the understanding of how we model the graph security problem as a POMDP model and how our POMDP model operates, we explicitly represent the attack nodes and the edges they inject.

Specifically, we first model the graph security issues as a POMDP model. For a graph $G'$, after being attacked, the initial state $s_0$ is the disturbance $\Delta$ added by the attack node $x$, but both the attack node and the disturbance $\Delta$ are unknown. The observation $o$ is known; it is the set of edges between nodes in the graph, and the initial observation $o_0$ includes the edges injected by the attack nodes. We attempt to change the state $s$ by taking the action $a_t$ to add or remove an edge in the graph. If the operated edge is one injected by an attack node, then the state $s$ will change. Moreover, each action taken will lead to a transition from the observation $o_t$ to $o_{t+1}$.

In brief, in our defined POMDP model, the agent's current observation and the required action will result in a transition to the next observation. This action may or may not change the state, depending on the defined reward function (Equation (4)).

### *4.3. Determining the Optimal Strategy*

In the previous section, we modeled graph security issues as a POMDP. In MDP, the policy uses the real state $\pi(s_t) : S \rightarrow$ Action, but in a POMDP, due to its partially observable nature, solving the POMDP is challenging, so we rely on Graph Convolutional Memory (GCM) [20] to solve the POMDP. The GCM's goal is to construct an approximation of $s_t$ from the observation sequence $\tau = o_1, \ldots, o_t$. The GCM reconstructs a graph; for an input observation $o_t$, the GCM treats $o_t$ as a node placed within the graph. It then computes its neighbor nodes $N(o_t)$ using topological priors and updates its edges, and finally, the GNN summarizes this graph to generate the approximate state $\widehat{s_t}$. This approximate state $\widehat{s_t}$ is then used for decision making in the policy $\pi$.

To solve the POMDP, we use the A2C (Advantage Actor–Critic) algorithm [27] as the reinforcement learning method to find the optimal strategy. The A2C (Advantage Actor–Critic) algorithm uses two networks: one decides actions (Actor), and the other evaluates the quality of these actions (Critic), updating both simultaneously to learn tasks faster and more stably. In the A2C algorithm, the Actor generates actions based on the current policy, while the Critic evaluates the resulting state to provide feedback on the effectiveness of these actions. The Actor selects actions according to the current policy $\pi$, while the Critic evaluates the quality of the current policy, i.e., the state-value function $V(s)$ or the action-value function $Q(s, a)$. Specifically, the Actor component proposes actions based on the current approximate state, aiming to change the graph's structure to mitigate or avoid the impact of attacks. The Critic component then evaluates the expected effect of these actions, providing feedback on the current strategy's effectiveness.

The core of the A2C algorithm lies in how it uses feedback (i.e., reward values) obtained from the environment to update the policy. By calculating the advantage function $A(s, a)$, A2C can assess the additional value of taking a specific action $a$ in a given state $s$ compared with following the average policy. The advantage function, defined as the difference between the action-value function $Q(s, a)$ and the state-value function $V(s)$, assesses the relative value of taking a particular action compared with the average policy. The essence of this process is the reward value—the environment's immediate feedback on the agent's actions, which directly affects the calculation of $Q(s, a)$ and $V(s)$. The definition of the reward value is as follows:

$$\text{reward} = \text{acc}_t - \alpha \times \frac{\sum_{i=1}^{n} a_i}{n + 0.1} \tag{4}$$

At the current time $t$, $acc_t$ signifies the model's accuracy. We define the baseline as the set of accuracy rates from the preceding $n$ moments, which is expressed as baseline $= a_1, a_2, \ldots, a_n$, where $n \leq b$ and $b$ is the maximum number of entries that the baseline can contain. The sum $\sum_{i=1}^{n} a_i$ represents the aggregate of the accuracy rates over these $n$ moments in the baseline. When a new accuracy value $acc_{\text{new}}$ is to be included in the

baseline, and the condition $n = b$ holds true, the earliest element $acc_{t-b-1}$ should be excised from the baseline before the inclusion of $acc_{\text{new}}$. This method is delineated by the following rule of update:

$$\begin{cases} \text{baseline} \cup \text{acc}_{\text{new}}, & \text{if } n < b \\ (\text{baseline} \setminus \text{acc}_{t-b-1}) \cup \text{acc}_{\text{new}}, & \text{if } n = b \end{cases} \tag{5}$$

Within this context, "acc" stands for the model's accuracy. After each policy update step, we compute the model's current accuracy at time $t$, denoted as $acc_t$. The accuracy we define is shown as follows:

$$\text{acc} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\Big(\text{argmax}_j \, O_{ij} = y_i\Big) \tag{6}$$

In this context, $N$ denotes the total number of samples; $O_{ij}$ represents the score that the model predicts for the $i$-th sample to be of the class $j$; $y_i$ is the actual class label of the $i$-th sample; $\text{argmax}_j \, O_{ij}$ indicates the model's prediction for the $i$-th sample, specifically, the class $j$ with the highest score; and $\mathbb{I}(\cdot)$ is the indicator function, which outputs 1 if the condition within the parentheses is true—that is, if the predicted class matches the true class—and 0 otherwise.

An agent, upon executing an action within the environment, receives a reward signal, which serves as an immediate indication of the action's utility. The A2C algorithm employs these reward signals to refine the agent's behavioral strategy. More precisely, by optimizing the advantage function $A(s, a)$, the Actor is directed to select actions that are anticipated to provide superior rewards. Concurrently, the Critic, by assessing the disparity between the actual rewards and those predicted based on the extant policy—the temporal-difference (TD) error—systematically calibrates its assessment of the state value function $V(s)$. This error signal, rooted in rewards, is instrumental in the strategic adjustment process as it furnishes an explicit evaluation of the policy's efficacy, thereby enabling the Critic to render more accurate feedback to the Actor and enhancing the strategy optimization process.

With this iterative process of interaction and refinement, the A2C algorithm strikes a balance between exploring new actions to uncover more efficacious strategies and selecting the most appropriate action predicated on accumulated knowledge. As depicted in Algorithm 1, the reinforcement learning training protocol incrementally acquires an optimal strategy that upholds safety within a specified graph structure and under potential attack scenarios, effectively neutralizing the threat of graph poisoning attacks. This methodology constitutes a secure convolution process, accounting for not only the immediate impact of the current state of the graph and its actions but also, through the synergy of GCM and A2C, the overarching structure and enduring safety of the entire graph, thereby offering a holistic and dynamic defense mechanism for graph data.

**Algorithm 1:** Optimal strategy solving algorithm

**Input:** POMDP model $M$
**Output:** The optimal policy $\pi^*$
$o_0 \leftarrow \text{M}(O)$ `// Obtain the observations O`
$t \leftarrow 0$
$s_t, m_t \leftarrow \text{GCM}(o_t, \text{None})$ `// Encoding state` $s$ `by GCM`
**while** *policy not in expect* **do**
  $a_t \leftarrow \text{GetActionByPolicy}(s_t)$ `// Select actions according to the policy`
  $s_t, m_t \leftarrow \text{GCM}(o_t, m_t)$ `// Encoding state` $s$ `by GCM`
  $s_{t+1} \leftarrow S(s_t, a_t)$ `// Obtain the next state based on the action and the state`
  $r(s_t, a_t) \leftarrow$ Compute with Equation (4) `// Calculating the reward`
  $\pi_t \leftarrow \text{updateByA2C}(r, a_t, s_t)$ `// Update strategy`
**end**
**return** $\pi^*$

### *4.4. Mask Layer*

In the previous section, we have obtained an optimal strategy through reinforcement learning algorithms. Now we need to use this strategy to train the GNN network. As shown in Figure 6, we have added a custom mask layer before the GCN convolutional layers. The mask layer provides a mask value for each edge, which allows us to control the convolutional range of the GNN by changing the mask values. In the process of solving the optimal strategy previously, the modification of edges was achieved by modifying the mask value. Therefore, changes to the edges must go through the mask layer in order to be effective. For example, if the mask value corresponding to an edge is 0, then the GNN will not convolve this edge, effectively deleting it. If the mask value is 1, the convolution process will operate on this edge as usual. As illustrated in Algorithm 2, with the optimal strategy, we can obtain a set of trustworthy edges, meaning that when the connections between the nodes in the graph are all secure, we can set the values of the mask to control the convolutional range of the convolutional layer. By preventing the convolutional layer from learning dangerous edges, we achieve a safe GNN.

Following the GCN, we also employ ReLU, FC (Fully Connected), and Softmax layers. ReLU introduces non-linearity, enhancing the model's learning capability; the FC layer provides a mechanism to integrate features in preparation for the final task; and the Softmax layer ensures that the model's outputs can be interpreted as probabilities. This combination enables the model to learn complex patterns within the graph and perform an accurate classification of nodes or the entire graph.

**Algorithm 2:** Convolutional range control algorithm

**Input:** Optimal policy $\pi^*$, original edge set OriginalE, graph $G$
**Output:** Safe node classification Outcome $Y^*$

**if** *policy is optimal* **then**
    // If the optimal policy is obtained
    ConfidenceE $\leftarrow \pi^*$ // Record the set of trusted edges in the optimal policy
    **foreach** $i \in$ *ConfidenceE* **do**
        // Trusted edges have the mask value set to True
        mask$[i] \leftarrow$ True
    **end**
    **foreach** $i \in$ *OriginalE* $\setminus$ *ConfidenceE* **do**
        // Untrusted edges have the mask value set to False
        mask$[i] \leftarrow$ False
    **end**
    $X_1 \leftarrow \mathrm{GCN}(G, \mathrm{mask})$ // Control the convolution scope through the value of the mask
    $X_2 \leftarrow \mathrm{ReLu}(X_1)$ // ReLU functions processing
    $X_3 \leftarrow \mathrm{FC}(X_2)$ // Fully connected layer processing
    $X_4 \leftarrow \mathrm{Softmax}(X_3)$ // Softmax functions processing
**end**
$Y^* \leftarrow X_4$
**return** $Y^*$

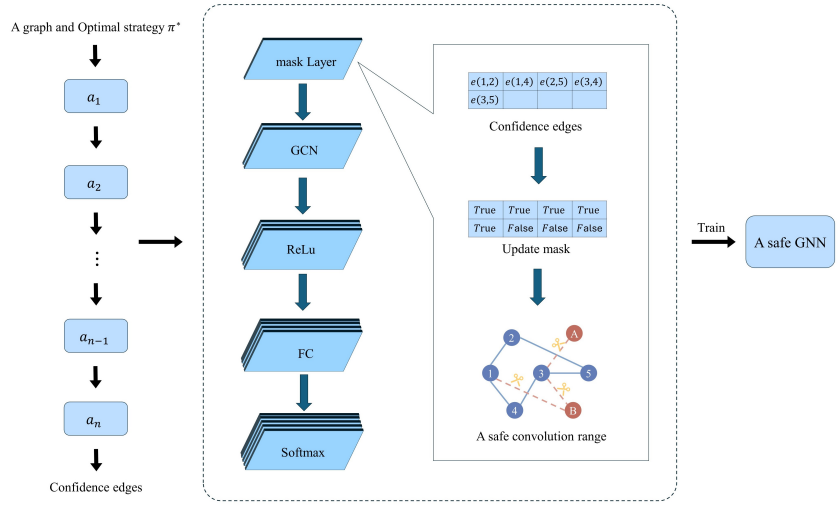



**Figure 6.** GNN training framework with a mask layer.

## 5. Experiments

In this section, we provide a detailed description of the specific experimental procedures, focusing on the effectiveness of our method. This section is divided into two parts. The first part details the experimental preparations, including datasets, experimental methods, and evaluation metrics. The second part discusses whether our method outperforms existing research in defending against attacks and analyzes data to assess the model's effectiveness, with particular emphasis on verifying the model's defensive capabilities.

1. Data Preparation: We evaluated our experiments on the following seven datasets: CiteSeer, Cora, Cora_ML, PubMed, football, Dolphins, and the Karate Club dataset. The information of the dataset is shown in the Table 2. CiteSeer is a computer science literature database; Cora is a dataset for literature classification, containing information on machine learning papers; Cora_ML is a literature classification dataset that is smaller than Cora; PubMed is a biomedical literature database covering literature in biomedical and life sciences; the football dataset is a network of American football games between Division IA colleges during the 2000 fall regular season. The Dolphins dataset represents the social interactions among bottlenose dolphins in Doubtful Sound, New Zealand, with nodes indicating individual dolphins and edges representing their observed associations. The Karate Club dataset details the social network of a university karate club, including friendships between members and the eventual split into two factions due to a conflict. Due to the large volume of data, we extracted subgraphs for the 800 most important nodes in Cora and PubMed; for CiteSeer, Cora_ML, football, Dolphins, and the Karate Club dataset, we used all the data.
2. Model Preparation: We divided the models into three categories: original models, attack models, and defense models.
    (a) For the original model, we used the most common GCN to measure the performance of the original model on the clean dataset. The GCN (Graph Convolutional Network) is one of the most representative graph neural networks, learning hidden representations by encoding local graph structures and node features.
    (b) For the attack model, we used a commonly used adversarial injection attack method, Adv Injection (ADV), which is a gradient-based injection attack method with good attack performance for adding disturbances to the dataset.
    (c) For the defense model, we used Robust GCN [28], Median GCN [29], and AirGNN [30] as comparative experiments to contrast with our method.

- Robust GCN is a defense method based on adversarial training, which defends against adversarial injection attacks by injecting adversarial examples into the model and learning from them.
- Median GCN enhances the robustness of the GCN against structural attacks by adopting an aggregation scheme with high breakpoints (median or trimmed mean).
- AirGNN proposes and derives a simple, efficient, interpretable, and adaptive message passing scheme, resulting in a novel GNN with adaptive residuals that has stronger recovery capabilities for anomalous features.

3. Evaluation Metrics: For each dataset, we randomly select 60% of the nodes for the training set, 20% for the validation set, and the remaining nodes for the test set. To assess the effectiveness of the models, we statistically evaluate them based on model accuracy. Additionally, to compare the effectiveness of different defense methods, we calculate the accuracy before and after defense, as follows:

$$R = ACC_{\text{defense}} - ACC_{\text{attack}} \tag{7}$$

where $ACC_{\text{defense}}$ is the accuracy of the defense model after the attack, and $ACC_{\text{attack}}$ is the model accuracy of the original model after the attack.
In addition, to compare the performance losses of the model on the original (unattacked) dataset after defense, we calculate the difference between the accuracy of the clean dataset and the accuracy after defense, as follows:

$$C = ACC_{\text{clean}} - ACC_{\text{defense}} \tag{8}$$

where $ACC_{\text{clean}}$ is the accuracy of the model when it is not attacked, and$ACC_{\text{defense}}$ is the accuracy of the defense model after the attack.

In order to compare and analyze the detection capabilities of different methods against adversarial injection attacks, the proposed method was compared with RobustGCN, MedianGCN, and AirGNN, respectively, to compare their performances on Adv Injection (ADV). Specific comparison tests are shown in Table 3.

**Table 2.** Summary of graph datasets for node classification.

| Dataset | Number of Nodes | Number of Edges | Number of Classes | Number of Features | Node Features | Description |
|---|---|---|---|---|---|---|
| CiteSeer | 3327 | 4732 | 6 | 3703 | 3703-dimensional sparse vector | A scientific publication citation network with papers categorized into 6 fields. |
| PubMed | 19,717 | 44,338 | 3 | 500 | 500-dimensional sparse vector | A scientific publication citation network with papers categorized into 3 fields. |
| Cora | 2708 | 5429 | 7 | 1433 | 1433-dimensional sparse vector | A scientific publication citation network with papers categorized into 7 fields. |
| Football | 115 | 613 | 11 | 1 | 1-dimensional feature | A network representing football matches between teams. |
| Karate Club | 34 | 78 | 2 | 34 | 34-dimensional feature | A social network of a karate club, categorized into 2 groups. |
| Dolphins | 62 | 159 | 2 | 0 | 0-dimensional feature | A social network of dolphins, with interaction data. |

**Table 3.** Comprehensive results including model accuracy, defense effectiveness, and net accuracy loss.

| Model/Data | CiteSeer | Cora_ML | Football | Cora | PubMed | Dolphins | Karate Club |
|---|---|---|---|---|---|---|---|
| **ACC (%)** | | | | | | | |
| CLN | 0.75 | 0.875 | 0.87 | 0.835 | 0.9 | 1 | 0.886 |
| ATK | 0.669 | 0.806 | 0.391 | 0.681 | 0.825 | 0.692 | 0.629 |
| RobustGCN | 0.732 | 0.85 | 0.696 | 0.781 | 0.853 | 0.938 | 0.771 |
| MedianGCN | 0.72 | 0.812 | 0.478 | 0.763 | 0.834 | 0.831 | 0.714 |
| AirGNN | 0.729 | 0.856 | 0.391 | 0.798 | 0.838 | 0.692 | - |
| Ours | **0.74** | **0.863** | **0.783** | **0.827** | **0.867** | **0.978** | **0.857** |
| **R (%)** | | | | | | | |
| RobustGCN | 0.063 | 0.044 | 0.305 | 0.100 | 0.028 | 0.246 | 0.142 |
| MedianGCN | 0.051 | 0.006 | 0.087 | 0.082 | 0.009 | 0.139 | 0.085 |
| AirGNN | 0.060 | 0.050 | 0.000 | 0.117 | 0.013 | 0 | - |
| Ours | **0.071** | **0.057** | **0.392** | **0.146** | **0.042** | **0.286** | **0.228** |
| **C (%)** | | | | | | | |
| RobustGCN | 0.018 | 0.025 | 0.174 | 0.054 | 0.047 | 0.062 | 0.115 |
| MedianGCN | 0.030 | 0.063 | 0.392 | 0.072 | 0.066 | 0.169 | 0.172 |
| AirGNN | 0.021 | 0.019 | 0.479 | 0.037 | 0.062 | 0.308 | - |
| Ours | **0.010** | **0.012** | **0.087** | **0.008** | **0.033** | **0.022** | **0.029** |

Table 3 shows the model results in different degrees of performance degradation after Adv Injection attacks the original model. The bold text indicates the best-performing methods among all the defense methods compared. Among them, Adv Injection constructs appropriate nodes and edges to attack the model through gradient calculation, resulting in 7–82% performance degradation. Additionally, Table 3 also shows that our method is significantly superior to the other three models in the effectiveness evaluation of the defense, and compared with the original model, all defense methods will produce a certain accuracy loss.

As shown in Figure 7, CLN is the original model accuracy, ATK is the post-attack accuracy, and the other four are the accuracies of different defense methods. It can be found that, after defense, the performance is improved to different degrees compared with the attack, but the accuracy is slightly lower than that of the original model. This may be because the model cannot fully consider all adversarial examples and still lacks the ability to detect some adversarial examples with high concealment. For Adv Injection attack methods, Robust GCN, Median GCN, and AirGNN all provide relatively strong protection capabilities. Comparatively speaking, our method can effectively improve the security of the adversarial injection attacks of the GNN model. Our defense approach achieved 74% to 86.7% accuracy, an improvement of about 5.09% to 100.26% compared with post-attack accuracy, and about 0.82% to 100.26% higher accuracy than several other models.

In this study, we found that AirGNN performance on the Karate Club dataset was significantly below expectations, with an accuracy of only 0.286. This result suggests that the Karate Club dataset may not be suitable for AirGNN design and training methods. The dataset consists of a small number of nodes (34 nodes) and a relatively simple graph structure, making it difficult for the model to fully leverage its adversarial training mechanism. AirGNN relies on generating adversarial samples to enhance the model's robustness, but on smaller and simpler datasets, generating effective adversarial samples may be insufficient, thereby affecting the model's performance. Future research needs to explore testing AirGNN on more diverse and complex datasets to evaluate its true performance advantages.

In addition to the aforementioned points, we also evaluated the computational resources and time required for our method. We used an Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30 GHz and an RTX-3090 GPU for computations. As shown in Table 4, training and inference times vary across different datasets. Training times range from 2 min and 11 s for the karate dataset to 1 h, 21 min, and 58 s for the PubMed dataset, while inference times range from 30 s for the football dataset to 6 min and 14 s for the CiteSeer dataset. Overall, inference times are significantly shorter than training times, indicating that the models have a high level of practicality for real-world applications due to their efficient

inference performance. These experimental results provide a strong basis for evaluating and comparing the computational efficiencies of different datasets and models, highlighting their potential value in practical use cases.

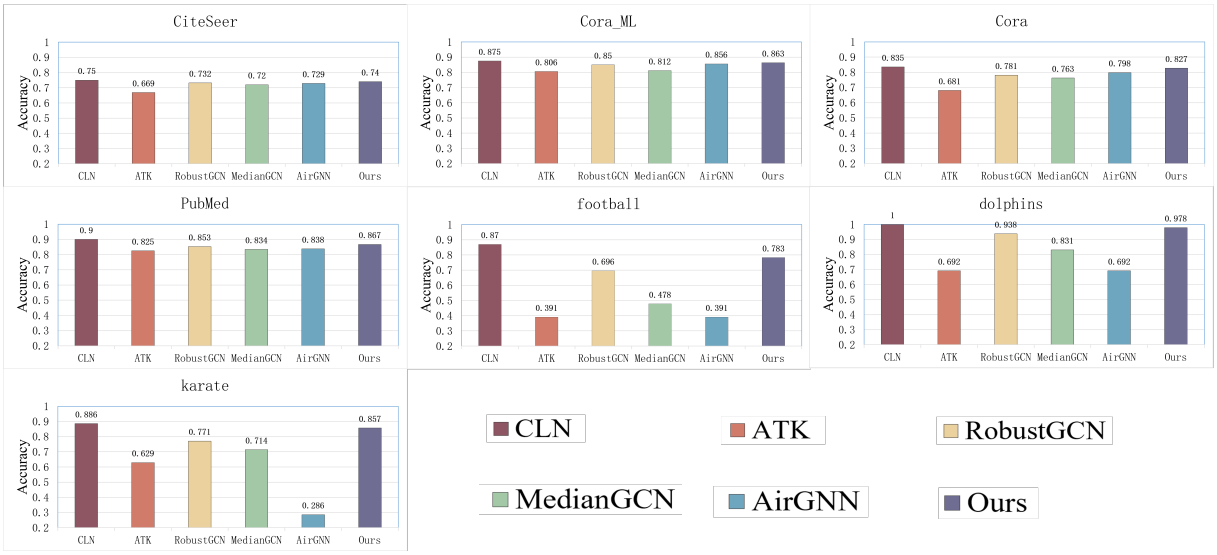



**Figure 7.** Accuracy of different models in the dataset.

**Table 4.** Train and inference times for different datasets.

| Data | Train Time | Inference Time |
|---|---|---|
| CiteSeer | 0 h 59 m 37 s | 0 h 6 m 14 s |
| Cora_ML | 0 h 44 m 12 s | 0 h 2 m 40 s |
| football | 0 h 9 m 25 s | 0 h 0 m 30 s |
| Cora | 0 h 30 m 23 s | 0 h 0 m 57 s |
| PubMed | 1 h 21 m 58 s | 0 h 1 m 10 s |
| dolphins | 0 h 4 m 12 s | 0 h 0 m 11 s |
| karate | 0 h 2 m 11 s | 0 h 0 m 16 s |

In general, in terms of the effectiveness of the model's defense capability, it benefits from the control of the message passing of the graph structure, and the method modeled as POMDP in this paper can more comprehensively defend against the influence of adversarial injection attacks, which is superior to the existing work in various evaluation indicators. The significant accuracy improvements observed in the experiments underscore the profound impact of our research results. Specifically, the increase in accuracy strongly validates the effectiveness of the proposed secure training framework for GNNs. This validation demonstrates that the new approach significantly outperforms traditional methods in enhancing model performance. Additionally, the improvement in accuracy indicates that the method provides higher robustness against adversarial injection attacks, effectively mitigating their impact, protecting model integrity, and ensuring reliable operation in the presence of adversarial threats.

## 6. Conclusions

In this paper, we proposed a novel defense strategy against adversarial injection attacks for Graph Neural Networks (GNNs), leveraging reinforcement learning to address the security challenges under conditions of partial observability. Given the extensive deployment of GNNs across varied domains, addressing their security vulnerabilities, particularly against adversarial injection attacks, becomes crucial. These attacks significantly impact the robustness and accuracy of GNN models. Our principal contribution is the pioneering modeling of the security training process for GNNs as a Partially Observable Markov Decision Process (POMDP), coupled with the utilization of Graph Convolutional Memory (GCM) to convert observations into temporally informed states. This foundation paves

the way for deploying a reinforcement learning algorithm to identify and implement an optimal defense strategy effectively. Our methodology extends beyond the protection of individual nodes, aiming for the security of the entire graph structure. This holistic approach holds substantial significance for real-world applications. Empirical validation on several public datasets underscored the efficacy of our method in enhancing GNN resilience against adversarial injection attacks, with observed accuracy improvements ranging from 5.09% to 100.26%. These results affirm the effectiveness and superior performance of our approach in advancing the security mechanisms within the realm of GNNs. Future research should focus on optimizing computational efficiency to enhance the scalability and real-time application capability of the proposed method. Additionally, exploring the applicability of this method to different types of graph data, such as dynamic graphs and heterogeneous graphs, and evaluating its performance in these environments is crucial. By expanding the range of applicability and validating effectiveness across various types of graph data, we aim to further improve the practical applicability and generalization ability of this method. These efforts will help address the limitations of current methods and enable superior performance in a broader range of application scenarios.

**Author Contributions:** Conceptualization, D.A.; Methodology, D.A.; Software, Y.Y.; Validation, D.A.; Formal analysis, Y.Y.; Writing—original draft, Y.Y.; Writing—review & editing, W.L., Q.Z., J.L. (Jing Liu) and H.Q.; Project administration, J.L. (Jie Lian) All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was sponsored in part by the National Natural Science Foundation Youth Fund under Grant No. 62302308; in part by the National Key Research and Development Program of China under Grant Nos. 2022YFC3302600 and 2022YFB4501704; in part by the National Natural Science Foundation of China under Grant Nos. 61972150, 62132014, 62302308, U2142206, 62372300, and 61702333; in part by the Shanghai Engineering Research Center of Intelligent Education and Big Data; and in part by the Research Base of Online Education for Shanghai Middle and Primary Schools.

**Data Availability Statement:** https://github.com/yy995132670/A-secure-GNN-training-framework-for-parcially-observable-graph, accessed on 25 June 2024.

**Conflicts of Interest:** Authors Dongdong An, Yi Yang, Qin Zhao, Hongda Qi, and Jie Lian were employed by Shanghai Engineering Research Center of Intelligent Education and Bigdata, Shanghai Normal University. Author Wenyan Liu was employed by Ant Group and College of Computer Science and Technology, Zhejiang University. Author Jing Liu was employed by Shanghai Key Laboratory of Trustworthy Computing, East China Normal University. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## References

1. Li, X.; Sun, L.; Ling, M.; Peng, Y. A survey of graph neural network based recommendation in social networks. *Neurocomputing* **2023**, *549*, 126441. [CrossRef]
2. Gao, C.; Zheng, Y.; Li, N.; Li, Y.; Qin, Y.; Piao, J.; Quan, Y.; Chang, J.; Jin, D.; He, X.; et al. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Trans. Recomm. Syst.* **2023**, *1*, 1–51. [CrossRef]
3. Gao, C.; Wang, X.; He, X.; Li, Y. Graph neural networks for recommender system. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, Tempe, AZ, USA, 21–25 February 2022; pp. 1623–1625.
4. Li, R.; Yuan, X.; Radfar, M.; Marendy, P.; Ni, W.; O'Brien, T.J.; Casillas-Espinosa, P.M. Graph signal processing, graph neural network and graph learning on biological data: A systematic review. *IEEE Rev. Biomed. Eng.* **2021**, *16*, 109–135. [CrossRef] [PubMed]
5. Zügner, D.; Akbarnejad, A.; Günnemann, S. Adversarial attacks on neural networks for graph data. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 2847–2856.
6. Sun, Y.; Wang, S.; Tang, X.; Hsieh, T.Y.; Honavar, V. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 673–683.

7. Santos, A.; Rente, D.; Seabra, R.; Moura, J.M. Inferring the Graph of Networked Dynamical Systems under Partial Observability and Spatially Colored Noise. In Proceedings of the ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, Republic of Korea, 14–19 April 2024; pp. 13156–13160.
8. Machado, S.; Sridhar, A.; Gil, P.; Henriques, J.; Moura, J.M.; Santos, A. Recovering the graph underlying networked dynamical systems under partial observability: A deep learning approach. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2023; Volume 37, pp. 9038–9046.
9. Ioannidis, V.N.; Chen, S.; Giannakis, G.B. Efficient and stable graph scattering transforms via pruning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 1232–1246. [CrossRef] [PubMed]
10. Ioannidis, V.N.; Marques, A.G.; Giannakis, G.B. Tensor graph convolutional networks for multi-relational and robust learning. *IEEE Trans. Signal Process.* **2020**, *68*, 6535–6546. [CrossRef]
11. Dai, H.; Li, H.; Tian, T.; Huang, X.; Wang, L.; Zhu, J.; Song, L. Adversarial attack on graph structured data. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1115–1124.
12. Xu, K.; Chen, H.; Liu, S.; Chen, P.Y.; Weng, T.W.; Hong, M.; Lin, X. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv* **2019**, arXiv:1906.04214.
13. Abusnaina, A.; Wu, Y.; Arora, S.; Wang, Y.; Wang, F.; Yang, H.; Mohaisen, D. Adversarial example detection using latent neighborhood graph. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual, 11–17 October 2021; pp. 7687–7696.
14. Angne, H.; Atkari, A.; Dhargalkar, N.; Kale, D. Unravelling SAT: Discussion on the Suitability and Implementation of Graph Convolutional Networks for Solving SAT. In *Proceedings of the Information and Communication Technology for Intelligent Systems: Proceedings of ICTIS 2020*; Springer: Berlin/Heidelberg, Germany, 2021; Volume 2; pp. 251–258.
15. Bunel, R.R.; Turkaslan, I.; Torr, P.; Kohli, P.; Mudigonda, P.K. A unified view of piecewise linear neural network verification. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 4795–4804.
16. Wang, B.; Jia, J.; Cao, X.; Gong, N.Z. Certified robustness of graph neural networks against adversarial structural perturbation. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 1645–1653.
17. Zhang, X.; Zitnik, M. Gnnguard: Defending graph neural networks against adversarial attacks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9263–9275.
18. Hoerger, M.; Kurniawati, H. An on-line POMDP solver for continuous observation spaces. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xian, China, 30 May–5 June 2021; pp. 7643–7649.
19. Helmeczi, R.K.; Kavaklioglu, C.; Cevik, M. Linear programming-based solution methods for constrained partially observable Markov decision processes. *Appl. Intell.* **2023**, *53*, 21743–21769. [CrossRef]
20. Morad, S.; Liwicki, S.; Kortvelesy, R.; Mecca, R.; Prorok, A. Modeling Partially Observable Systems using Graph-Based Memory and Topological Priors. In Proceedings of the Learning for Dynamics and Control Conference. PMLR, Stanford, CA, USA, 23–24 June 2022; pp. 59–73.
21. Kissel, M.; Gottwald, M.; Diepold, K. Neural network training with safe regularization in the null space of batch activations. In Proceedings of the Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, 15–18 September 2020; Proceedings, Part II 29; Springer: Berlin/Heidelberg, Germany, 2020; pp. 217–228.
22. Zhao, H.; Zeng, X.; Chen, T.; Liu, Z.; Woodcock, J. Learning safe neural network controllers with barrier certificates. *Form. Asp. Comput.* **2021**, *33*, 437–455. [CrossRef]
23. Pauli, P.; Koch, A.; Berberich, J.; Kohler, P.; Allgöwer, F. Training robust neural networks using Lipschitz bounds. *IEEE Control Syst. Lett.* **2021**, *6*, 121–126. [CrossRef]
24. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef]
25. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
26. Kurakin, A.; Goodfellow, I.; Bengio, S.; Dong, Y.; Liao, F.; Liang, M.; Pang, T.; Zhu, J.; Hu, X.; Xie, C.; et al. Adversarial attacks and defences competition. In *Proceedings of the The NIPS'17 Competition: Building Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 195–231.
27. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning. PMLR, New York, NY, USA, 20–22 June 2016; pp. 1928–1937.
28. Zhu, D.; Zhang, Z.; Cui, P.; Zhu, W. Robust graph convolutional networks against adversarial attacks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1399–1407.

29. Chen, L.; Li, J.; Peng, Q.; Liu, Y.; Zheng, Z.; Yang, C. Understanding structural vulnerability in graph convolutional networks. *arXiv* **2021**, arXiv:2108.06280.
30. Liu, X.; Ding, J.; Jin, W.; Xu, H.; Ma, Y.; Liu, Z.; Tang, J. Graph neural networks with adaptive residual. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 9720–9733.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.