COST AWARE POISONING ATTACK AGAINST GRAPH NEURAL NETWORKS

HAN YUWEI

MPhil

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University

Department of Computing

Cost Aware Poisoning Attack against Graph Neural Networks

HAN Yuwei

A thesis submitted in partial fulfillment of the requirements

for the degree of Master of Philosophy

May 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

(Signed)

Yuwei Han (Name of student)

Abstract

Graph Neural Networks (GNNs) have achieved remarkable success in various tasks such as node classification, link prediction, and anomaly detection. However, these applications are vulnerable to adversarial attacks, especially poisoning attacks, where the attacker can modify the graph's structure and features at the model training stage to degrade the model's performance. Despite the existence of such attacks, the efficient utilization of the attacker's budget, in terms of the number and type of modifications allowed, remains an open challenge. This thesis aims to address this challenge by developing cost-aware poisoning attack strategies against GNNs that maximize the degradation of the model's performance while adhering to a constrained attack budget.

We begin by identifying the key factors that contribute to the effectiveness of poisoning attacks on GNNs, focusing on the strategic modification of graph structure. We then propose a set of novel attack methodologies that are designed to exploit these factors efficiently, ensuring that each modification contributes significantly to the overall impact on the GNN's performance. Our approaches are validated through extensive empirical evaluations on standard benchmarks for node classification, link prediction and anomaly detection tasks, demonstrating their superiority over existing attack strategies in terms of cost-effectiveness and impact.

Building on our empirical findings, we formalize the problem of cost-aware adversarial attacks on GNNs, deriving theoretical bounds on the minimum number of modifications required to achieve a desired level of performance degradation. This

formalization not only provides a theoretical foundation for our empirical strategies but also offers insights into the inherent vulnerabilities of GNNs to poisoning attacks.

In summary, this thesis contributes to the field of adversarial machine learning by introducing a comprehensive framework for cost-aware poisoning attacks against GNNs. Our work not only advances the understanding of GNN vulnerabilities but also provides practical tools and theoretical insights to guide the development of more robust GNN models in the face of poisoning threats.

Publications arising from the thesis

Han, Yuwei et al. (2024). "Cost Aware Untargeted Poisoning Attack Against Graph Neural Networks". In: IEEE International Conference on Acoustics, Speech, and Signal Processing.

Acknowledgements

First and foremost, I am deeply grateful to my advisor, Dr. Kai Zhou, for his invaluable advice, inspiration, encouragement, and support throughout my MPhil journey. Joining Dr. Zhou's research lab has been an invaluable experience, and I consider myself fortunate for the decision I made two years ago.

I am grateful to have had the pleasure and fortune of having supportive and encouraging colleagues during my MPhil. I am thankful to all my colleagues from the Secure and Trustworthy Intelligience Laboratory: YulinZhu, Yuni Lai, Xing Ai, YuBu, and Jialong Zhou. I look forward to continued collaboration with you.

I would also like to extend my heartfelt appreciation to my university and department. The exceptional facilities and resources they provided were fundamental to my research. The availability of cutting-edge equipment, a state-of-the-art laboratory, and an extensive collection of journals were pivotal in my academic journey. These resources not only facilitated my research but also enriched my learning experience, culminating in the successful completion of my thesis. The unparalleled academic environment and resources at my disposal were crucial in bringing my ideas to fruition and achieving the desired academic outcomes.

Finally, I would like to thank my family, for their unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivations

Graphs are a ubiquitous data structure employed to represent a wide array of data from diverse real-world applications. They are highly effective in modeling both objects and the intricate interactions between them. For instance, consider a graph representing friendships among Facebook users: such a graph can encompass 2.9 billion nodes and over 490 billion edges.

Graph Neural Networks (GNNs) represent a novel and increasingly relevant approach to the application of machine learning on structured data. Rooted in the concept of generalizing neural network methodologies to effectively handle data structured as graphs, GNNs have gained significant traction in the last decade, owing to their unique capabilities in managing complex, irregularly structured data. In consumer applications, Graph Neural Networks (GNNs) enhance personalized search and recommendations for customers on e-commerce platforms such as Alibaba and social media platforms like Pinterest by analyzing user-user and user-item interactions. In advanced sciences, researchers utilize graphs to model complex systems, such as physics simulations, and employ GNNs to uncover the fundamental laws governing celestial motion. GNNs also contribute to societal well-being, with appli-

cations ranging from detecting fake news to discovering drugs for treating COVID-19.

The primary characteristic of GNNs lies in their ability to leverage both node and edge information in graphs. By applying transformational functions and aggregating feature information from neighboring nodes, GNNs can generate a rich and comprehensive representation of nodes and edges, effectively capturing the topological intricacies inherent in graph structures. Numerous architectures and methods have been proposed to enhance the performance of GNNs from various angles (Y. Liu, K. Ding, et al. 2023), such as boosting their expressive capabilities (Xu et al. 2019), addressing over-smoothing problems (D. Chen et al. 2020) and deepening their architecture (G. Li et al. 2021), among others. Nonetheless, in domains where accuracy alone is not the sole objective, other considerations come into play. For instance, in GNN-based anomaly detection systems, it's vital to ensure robustness against adversarial attacks (K. Zhao et al. 2021). Robustness refers to the ability of systems to perform reliably under various conditions. In the context of Graph Neural Networks (GNNs), a robust GNN maintains model accuracy even when faced with perturbations, such as malicious modifications to the graph structure, like adding or deleting edges. Recent studies (Ju et al. 2023) have shown that GNNs may produce suboptimal results when the graph structure is altered. For example, attackers targeting a GNN used for node classification tasks can insert or delete nodes to their original graphs, thereby changing the classification labels output by the GNNs (B. Wang, Pang, et al. 2023). These security risks are increasingly significant as GNNs are applied to critical tasks. Therefore, it is crucial to study adversarial attacks to fully understand the vulnerabilities of existing GNN systems and develop effective defense strategies to enhance their robustness.

Adversarial attacks encompass various types, such as poisoning attacks, evasion attacks, and backdoor attacks. In practical scenarios, an attacker may gather different kinds of information about the target GNN model, including model parameters,

architectures, and training or testing data. Based on the type and amount of information obtained, attacks are classified into three categories: white-box attacks, black-box attacks, and grey-box attacks. In a white-box attack, attackers have full access to the data and the entire target GNN, including its architecture, parameters, and gradients. Conversely, in a black-box attack, attackers know nothing about the target GNN model and can only send queries to it. In a grey-box attack, the attacker has no knowledge of the classification model and its trained weights but possesses the same knowledge about the data as the classifier.

An attackers capability is also defined within the threat model. Most existing attacks propose injecting perturbations into graph data. Specifically, there are several types of perturbations, including altering the graph structure (adding/deleting/rewiring edges), modifying node attributes, and injecting nodes.

In this thesis, we primarily focus on grey-box poisoning structure attacks for node classification, link prediction and anomaly detection tasks on benchmark datasets. Poisoning attacks target the training phase of GNN model development. Attackers attempt to modify the training graphs of a target GNN, leading to the creation of a compromised model with reduced performance. Poisoning attack against GNNs has been one of the common considerations (Y. Sun et al. 2020; Tian et al. 2022). Even only slight, deliberate perturbations of an instance, also known as adversarial perturbations/examples, can lead to wrong predictions. Such negative results significantly hinder the applicability of these models, leading to unintuitive and unreliable results, and they additionally open the door for attackers that can exploit these vulnerabilities. For example, in anomaly detection applications, network traffic can be represented as graph data (i.e., network traffic graphs). GNNs have become a popular tool for analyzing traffic behavior and identifying abnormal data. However, an attacker can easily modify these traffic graphs, raising concerns about the GNNs robustness in anomaly detection. Specifically, attackers can execute various types of adversarial attacks targeting different phases of GNN development. For instance,

poisoning attackers can manipulate the exploitation code embedded in the traffic samples and poison the training data for the anomaly detector if their samples are gathered and used by the detector developer.

Currently, several studies explore potential poisoning attacks on applications like node classification (Daniel et al. 2019) and link prediction (Waniek et al. 2018). For instance, Metattack (Daniel et al. 2019) targets graph neural networks during training for node classification by perturbing the discrete graph structure. It employs meta-gradients to address the bilevel problem inherent in training-time attacks, treating the graph as a hyperparameter to optimize. However, Metattack does not account for the differences between nodes, assigning equal weights to all selected nodes' connected edges. This results in some modified edges failing to significantly degrade the performance of the graph neural networks, thus only reflecting limited vulnerabilities of current GNNs. This inefficiency, termed as the wastage of attack budget, arises when an attacker alters an edge connecting nodes that are either already misclassified or highly resistant to misclassification. Consequently, no additional nodes are misclassified due to the negligible impact of the edge modification on already misclassified nodes or the robust nature of certain nodes.

While some studies have attempted to address this issue, challenges remain. For example, Liu et al. (Zihan Liu, Luo, Lirong Wu, Zicheng Liu, et al. 2022) demonstrated that cross-entropy loss tends to generate higher gradients for edges connected to nodes with low ground truth confidence, indicating these nodes may have been misclassified. They proposed the GraD method, which redesigns the cross-entropy loss to generate low gradients for nodes with low confidence. However, this method is only effective for cross-entropy loss, limiting its applicability if the attack loss function changes. Wang et al. (B. Wang, Pang, et al. 2023) explored giving different weights to nodes based on their certified robustness size, where larger sizes indicate more robust nodes. They assigned higher weights to nodes with smaller certified sizes, ensuring these nodes are prioritized in attacks. Although this

method improves the attack success rate by reallocating the attack budget, it is time-consuming to compute the certified size, making it inefficient for practical use.

IGA (J. Chen et al. 2020) is a link prediction adversarial attack that uses gradient information to implement an iterative attack strategy. Unfortunately, this method faces similar issues as Metattack, with some edges failing to effectively degrade link prediction performance. To date, no studies have successfully addressed these issues in existing link prediction attack methods. Furthermore, there has been a lack of sufficient research investigating poisoning attacks on graph neural networks for anomaly detection tasks. Current studies only investigate poisoning attacks on graphs against conventional anomaly detector (Zhu, Lai, et al. 2022; Lai et al. 2023), or focus on black-box settings (X. Zhou et al. 2022) or evasion attacks (Venturi et al. 2024). Our work is the first to design poisoning attack algorithms specifically for graph neural network-based anomaly detection under a gray-box setting, marking a significant advancement in this field.

Our motivation is to build a general budget-aware poisoning attack framework for node classification, link prediction, and anomaly detection applications. This framework should prioritize the allocation of the budget to important nodes and edges that significantly impact the prediction accuracy of GNNs while seamlessly integrating with existing attack algorithms and maintaining attack efficiency. However, we need to address two main challenges:

- During the attack process, the graph structure dynamically changes. This means the importance of different nodes and edges for the graph neural network's predictions (e.g., degree) is constantly shifting. For link prediction tasks, the significance of edges also fluctuates. Therefore, we need to dynamically adjust the weights for different nodes and edges throughout the attack process.

- The framework must be adaptable to these three applications and various primary attack losses while preserving the effectiveness of the attack algorithm.

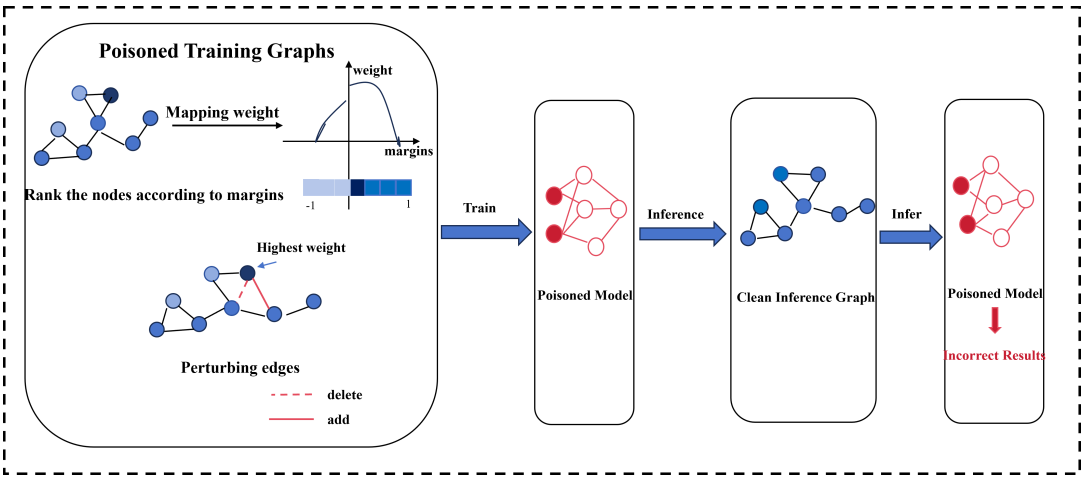


Figure 1.1: The overview of the proposed Cost Aware Poisoning Attack Framework

Our ultimate goal is to develop a cost-aware framework that can be applied to existing structure poisoning attacks. This will enable more efficient use of the attack budget and enhance the overall performance of these poisoning attacks.

1.2 Thesis Contributions Overview

We establish our study by investigating a cost-aware poisoning framework to allocate budgets for nodes and edges in node classification, link prediction, and anomaly detection applications with different priorities. This approach ensures that the budgets are effectively used to identify the vulnerabilities of GNNs. Figure 1.1 shows the pipeline of our approach. Firstly, we design the framework for node classification. We build upon the classical node classification poisoning attack algorithms (Daniel et al. 2019) and integrate our framework to improve their performance. We rank the importance of nodes for graph neural network predictions according to their distance from the decision boundary (margins). As attackers, we first allocate the attack budget to nodes near the decision boundary, which are easier to attack and can significantly degrade the GNNs' performance under a limited budget. This approach simulates the attacker's objectives in real scenarios. We achieve this by assigning different weights to nodes based on their margins. To address the dynamic nature of

the graph structure, we map the margins to weights using a designed function and incorporate it into the attack objective. The margins of the nodes are recalculated in each attack iteration, and the weights for different nodes are updated accordingly. Nodes with higher weights are allocated the budget first. Our framework is effective for various attack objectives in our experiments. Since margin computation is fast, this framework does not affect the attack efficiency of the algorithms.

Next, we extend our cost-aware framework to the application of link prediction to formulate a budget-efficient poisoning attack, named MetaLinkAttack. MetaLinkAttack assigns different weights to edges based on their margins and modifies the edges that generate the largest gradient during the training stage to reduce the link prediction accuracy of GNNs to the maximum extent, based on the IGA algorithm (J. Chen et al. 2020). MetaLinkAttack, with the cost-aware poisoning framework, further improves IGA's performance.

Finally, we design a poisoning attack for graph neural networks-based anomaly detection tasks called MetaAD. This method aims to allow anomaly nodes to bypass detection using the least budget. Specifically, we first allocate the budget to anomaly nodes with small margins (near the decision boundary) and then distribute the remaining budget to the other anomaly nodes. Our method demonstrates better attack performance compared to the Nettack framework (Zügner et al. 2018) in anomaly detection scenarios.

By implementing our cost-aware framework, we ensure that the attack budgets are allocated efficiently, enhancing the overall effectiveness of poisoning attacks across different applications.

The key contributions of this dissertation are summarized as follows:

- We introduce a Cost-Aware Poisoning Attack Loss Framework (CA-attack) (Han et al. 2024) to improve the allocation of the attack budget and maximize the impact of the poisoning attack against GNNs for node classification tasks. Specifically, we dynamically reweight the nodes according to their classifica-

tion margins in the attack loss. This means that the weights of the nodes are adjusted during the optimization process of the attack objective.

- We generalize our cost-aware poisoning attack loss framework to the link prediction task and propose MetaLinkAttack, a budget-efficient poisoning attack for graph neural network-based link prediction.

- We apply our cost-aware poisoning attack loss framework to anomaly detection task and propose the MetaAD, a budget-efficient poisoning attack for graph neural network based anomaly detection.

- Through rigorous empirical assessments on three datasets, we demonstrate that CA-attack, MetaLinkAttack and MetaAD improve existing methods, highlighting its potential as a plug-and-play solution for various graph poisoning attacks.

1.3 Thesis Structure

The rest of this thesis is structured as follows:

Chapter 2 provides a comprehensive literature review of graph neural networks and related applications of poisoning attacks against them. It covers the latest studies on poisoning attacks targeting node classification, link prediction, and anomaly detection. We offer a detailed analysis to highlight the development of current research and identify their shortcomings.

Chapter 3 introduces the Cost-Aware Poisoning Attack (CA-Attack) framework, demonstrating its integration with existing attack strategies and its enhancement of poisoning attack performance in node classification tasks.

Chapter 4 extends the theoretical analysis of our Cost-Aware Poisoning Attack framework to link prediction and anomaly detection. We explore the application of our framework to improve poisoning attack performance in these areas, showing

that it enhances attack effectiveness for both link prediction and anomaly detection across various attack losses.

Chapter 5 presents experimental results on the performance of our framework in node classification, link prediction, and anomaly detection. The experiments demonstrate that our framework can be effectively used in these applications, establishing it as a general framework for gradient-based poisoning attacks against graph neural networks.

Chapter 6 summarizes our contributions and the limitations of this thesis.

Chapter 2

Literature Review

This thesis aims to investigate a budget-efficient framework for implementing cost-aware poisoning attacks against graph neural networks. To this end, we design our framework based on current classical gradient-based grey-box attack methods, creating a cost-aware approach by mapping nodes' or edges' margins to weights, thereby assigning different priorities to various nodes and edges. This chapter presents a review of the development of current graph neural network algorithms and poisoning attack methods for node classification, link prediction, and anomaly detection.

2.1 Graph Neural Networks

Graph neural networks (GNNs) have emerged as a powerful paradigm for analyzing data with inherent graph structures, where the examination of individual nodes is greatly enhanced by considering the information from their neighbors (T. Kipf et al. 2016). This framework has been successfully applied across various domains, demonstrating its versatility and effectiveness. For instance, in recommender systems (W. Fan, Y. Ma, Q. Li, Y. He, Y. E. Zhao, et al. 2019), GNNs have significantly improved the personalization of recommendations by capturing the intricate user-item interaction networks. In computer vision (G. Li et al. 2021), they have facilitated advancements in image recognition by interpreting contextual relationships within

visual data. Moreover, in drug discovery (Duvenaud et al. 2015), GNNs have proven instrumental in identifying novel therapeutic molecules by decoding complex molecular structures.

Among the various GNN architectures, the Graph Attention Network (GAT) (Velickovic et al. 2017), also referred to as GAN, represents a significant advancement in processing graph-structured data. GATs leverage masked self-attentional layers to overcome the limitations of previous methods that relied on graph convolutions or their approximations. By stacking these layers, GATs can implicitly assign different weights to nodes within a neighborhood, enabling a focus on relevant local features without the need for costly matrix operations or prior knowledge of the graph's structure. This capability addresses several significant limitations of spectral-based graph neural networks and renders GATs suitable for both inductive and transductive learning tasks.

GraphSAGE (Hamilton et al. 2017) is another notable GNN framework that exemplifies an inductive learning approach. It considers only training samples linked to the training set's edges during the training process. GraphSAGE operates through two main steps: "Sampling" and "Aggregation." First, node representations are paired with aggregated vectors and passed through a fully connected layer with a non-linear activation function. Each network layer shares a standard aggregator and weight matrix. Finally, a normalization step is applied to the layer's output. The process involves initializing eigenvectors for all nodes, sampling neighbor nodes for each node, aggregating neighbor information, and updating embeddings through a non-linear transformation. The development of GNNs can be traced back to the work of Sperduti et al. (Sperduti et al. 1997) in 1997, which applied neural networks to directed acyclic graphs and inspired subsequent research in the field. The concept of GNNs was formally introduced by Gori et al. (Gori et al. 2005) in 2005 and further developed by Scarselli et al. (Scarselli et al. 2009) in 2009. These early models, known as recurrent graph neural networks (RecGNNs), learned node

representations through iterative propagation of neighbor information until reaching a stable state. However, this approach was computationally demanding, leading to the exploration of more efficient methods.

The success of Convolutional Neural Networks (CNNs) in computer vision inspired the development of convolutional graph neural networks (ConvGNNs), which redefine the notion of convolution for graph data. ConvGNNs are categorized into spectral-based and spatial-based approaches. Spectral-based ConvGNNs, initiated by Bruna et al. (Bruna et al. 2014) in 2013, are based on spectral graph theory, while spatial-based ConvGNNs, which predate their spectral counterparts, focus on graph mutual dependencies and message passing. Despite the early start of spatial-based ConvGNNs, their significance was not fully recognized until more recently, when a variety of spatial-based ConvGNNs emerged.

In addition to RecGNNs and ConvGNNs, the field has seen the introduction of other GNN frameworks, such as graph autoencoders (GAEs) (T. Kipf et al. 2016) and spatial-temporal graph neural networks (STGNNs) (Jiabin Tang et al. 2023). These frameworks can be built upon existing GNN architectures and are used for tasks like network embedding and graph generation.

Graph Convolutional Networks (GCNs) (T. N. Kipf et al. 2017), developed by Thomas Kipf and Max Welling, are a fundamental variant of GNNs that perform convolution operations on graph data. Similar to CNNs, GCNs apply filters or kernels to input neurons, but unlike CNNs, which operate on regular, ordered data, GCNs handle irregular, non-Euclidean data structures. GCNs have been applied to a wide range of problems, including image classification, traffic forecasting, recommendation systems, scene graph generation, and visual question answering. By iteratively applying convolution and aggregation layers, GCNs can capture complex patterns and dependencies within graph data.

In summary, GNNs represent a significant advancement in the analysis of graph-structured data. They provide a flexible and powerful tool for capturing the intricate

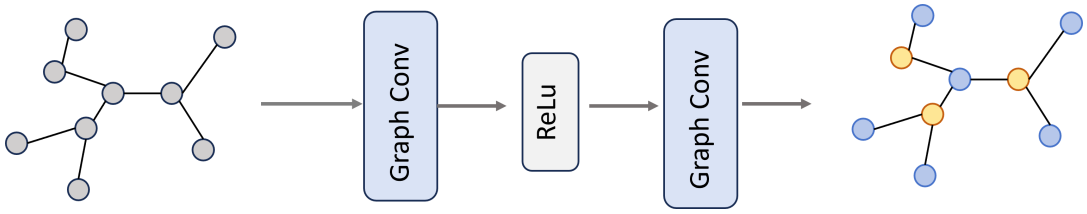


Figure 2.1: Node classification using GCN

relationships and patterns within graphs, making them applicable to a wide array of tasks in various domains.

2.2 Application of Graph Neural Networks

2.2.1 Node Classification

Node classification is a fundamental research direction in graph analysis due to its extensive application scenarios. The goal of node classification is to predict the class of each unlabeled node in the graph based on the available graph information. For example, in citation networks, node classification can determine the research topic of each article. In protein-protein interaction networks, it can assign multiple gene ontology types to each node. A fundamental approach to node classification is the use of Graph Convolutional Networks (GCNs). GCNs encapsulate the hidden representation of each target node by aggregating the feature information from its first-order neighbors. This aggregation process involves summing, averaging, or applying a more complex function to the features of neighboring nodes.

The process begins by initializing each node with its feature vector. During the convolution operation, each node updates its representation by aggregating information from its neighbors and combining it with its own features. This step can be mathematically expressed as:

$$h_v^{(k)} = \sigma(\sum_{\mu \in N(v)} \frac{1}{c_{vu}} W^{(k-1)} h_u^{(k-1)}), \tag{2.1}$$

Here, $h_v^{(k)}$ represents the hidden state of node v at the kth layer, $N(v)$ denotes the set of neighbors of node v, c_{vu} is a normalization constant, $W^{(k-1)}$ is a trainable weight matrix, and σ is a non-linear activation function.

By stacking multiple graph convolutional layers, a deep neural network model is constructed. This model is used to obtain the final hidden representation of each node, which encapsulates information from its multi-hop neighbors as well. The depth of the network allows the learned representation to integrate features from increasingly distant nodes, thereby capturing more complex structural and feature information.

The final step in the node classification process involves using the learned node representations to predict the class labels. This is typically achieved by applying a softmax function to the output of the last convolutional layer, producing a probability distribution over the possible classes for each node. The node with the highest probability is then assigned as the predicted class label. Figure 2.1 shows the node classification process using GCN. Numerous research studies have focused on GNN-based node classification tasks and have enhanced the performance of GNNs through various methods. For instance, Huang et al. (Hang et al. 2021) address the challenge of improving the expressiveness of Graph Neural Networks (GNNs) for node classification tasks by incorporating Collective Inference (CI). Despite GNNs being strong classifiers, they are often not most-expressive. The authors propose a novel framework, CL+GNN, which combines collective learning with GNNs to enhance their representation power. This framework utilizes self-supervised learning and Monte Carlo sampling to incorporate label dependencies among neighboring nodes. Theoretical analysis and extensive experiments demonstrate that CL+GNN consistently improves node classification accuracy across various state-of-the-art WL-GNNs on partially-labeled graphs.

Chowdhury et al. (Chowdhury et al. 2023) present novel input and output intervention techniques to enhance the accuracy of semi-supervised node classifica-

tion using GNNs. Their input intervention technique increases the number of non-contiguous training nodes by selecting nodes from the same class that are spread out across different subgraphs. This technique leverages variations of random walks and node embedding methods combined with clustering techniques like K-means and KNN. The output intervention technique identifies misclassified nodes by analyzing the confidence vector and relabeling low-confidence nodes with the labels of their nearest high-confidence neighbors. Both techniques are modular and can be applied as pre- and post-processing steps to existing GNN methods without requiring additional knowledge about node classes.

Duan et al. (L. Duan et al. 2024) propose a novel Graph Structure Learning (GSL) framework for node classification that leverages structural information theory to optimize graph structures. This framework addresses the limitations of existing GSL methods by incorporating hierarchical community information to reduce noise in complex real-world graphs. The core contributions include proving that an encoding tree with minimal structural entropy can effectively classify nodes and eliminate redundant noise, designing an efficient algorithm to construct such encoding trees, and developing a fusion mechanism to generate the optimal graph structure by combining community influence and prediction confidence.

These studies enhance the ability of GNNs to handle node classification tasks on complex datasets, demonstrating the powerful representation learning capabilities of GNNs.

2.2.2 Link Prediction

The task of link prediction is to determine the existence of an edge between two unconnected nodes in a graph. Existing link prediction algorithms estimate the proximity of different node pairs, where pairs with higher proximity are more likely to interact.

A fundamental approach to link prediction involves the use of Graph Autoen-

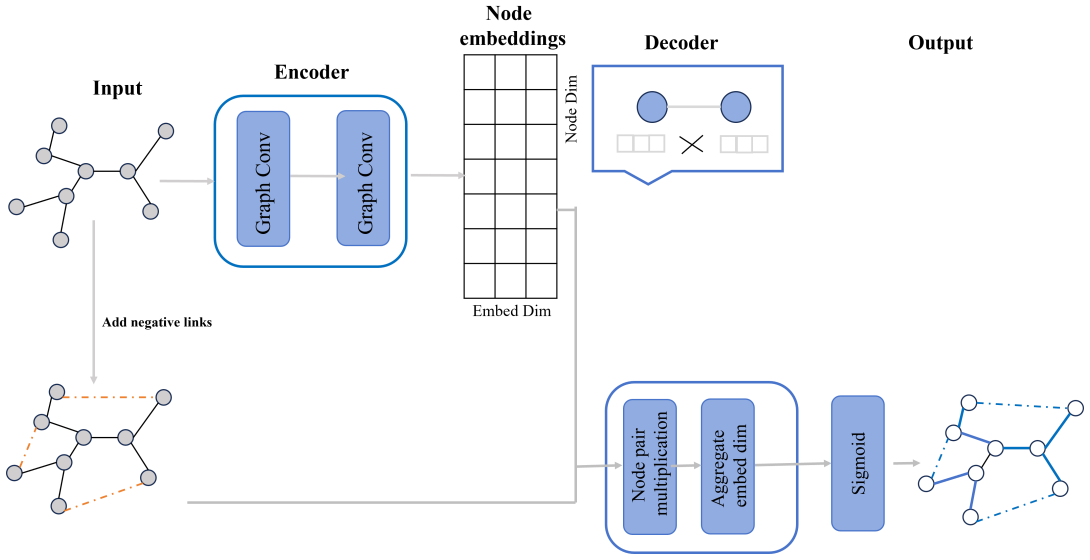


Figure 2.2: Link prediction using GAE

coders (GAEs). GAEs leverage the power of neural networks to learn node representations that can be used for predicting links. The process of using GAEs for link prediction can be summarized as follows:

- Graph Encoding: The first step involves encoding the graph structure and node features into a latent space. This is typically done using a graph convolutional network (GCN) encoder, which aggregates feature information from a node's local neighborhood. The encoder learns a compressed representation (embedding) for each node.

- Latent Space Representation: The GCN encoder produces a low-dimensional representation for each node. This latent representation captures the structural and feature information of the graph, facilitating the estimation of proximity between node pairs.

- Graph Decoding: The decoder component of the GAE reconstructs the graph from the latent space representations. This step often involves predicting the likelihood of an edge between pairs of nodes. A common approach is to use a simple inner product decoder, which computes the dot product of the latent vectors of two nodes to estimate the probability of an edge between them.

- Loss Function: The model is trained using a reconstruction loss, typically the binary cross-entropy loss, which measures the difference between the actual graph adjacency matrix and the predicted adjacency matrix. The objective is to minimize this loss, thereby improving the accuracy of link predictions.

- Training: During training, the GAE optimizes the node embeddings to accurately reconstruct the original graph's adjacency matrix. This involves iteratively updating the model parameters through backpropagation.

- Link Prediction: Once the GAE is trained, it can be used to predict links by evaluating the reconstructed adjacency matrix. Node pairs with higher predicted scores are more likely to have an edge between them, indicating a potential link. Figure 2.2 presents the GAE method for link prediction.

Various Graph Neural Networks (GNNs) techniques for link prediction have been proposed. Key methods include GCN (Yao et al. 2019), GAT (Velikovi et al. 2018), SAGE (Yang et al. 2016), and GAE (T. Kipf et al. 2016), which leverage the message-passing paradigm to assimilate multi-hop graph structures. These approaches integrate connectivity patterns and node features to enhance the predictive accuracy of potential links within graphs, showcasing the diversity and depth of GNN methodologies in capturing complex relational data for link prediction tasks.

2.2.3 Graph-based Anomaly Detection

Anomaly detection involves identifying patterns that significantly deviate from typical observations, a task with increasing importance and application in various domains. Research in anomaly detection could date back to Grubbs et al. (Grubbs 1969), who firstly proposed the notion of anomaly. With the advancements in graph mining, graph anomaly detection has gained considerable attention. Figure 2.3 shows some common anomalies on graphs, including structural anomalies and con-

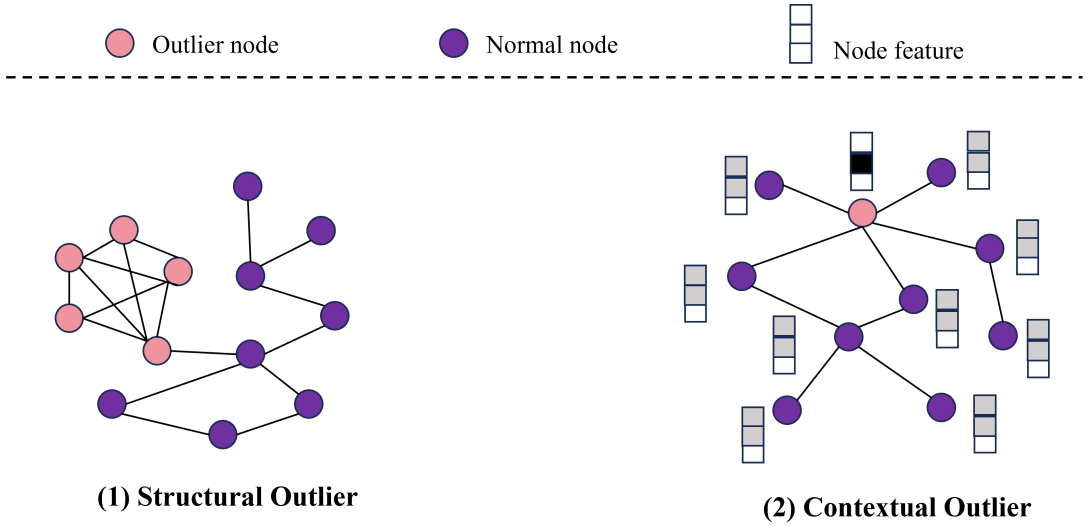


Figure 2.3: Anomaly Detection

textual/feature anomalies. Recently, Graph Neural Networks (GNNs) have been adopted to detect anomalies in graphs efficiently, leveraging their expressive capability via the message-passing mechanism for learning graph representations. GNNs facilitate the extraction of anomalous patterns from complex graph structures or attributes, handling graphs with attributes as input data.

This review mainly focuses on GNN-based static graph anomaly detection, specifically:

- Anomalous Node Detection: Detecting anomalous nodes in attributed graphs by extracting node attribute and structural information using GNNs. Anomalous nodes can be categorized as global anomalies, structural anomalies, and community anomalies (X. Ma et al. 2023). Various GNN-based methods, often built upon the graph autoencoder (GAE) framework, use different decoders and anomaly scoring functions to identify abnormal nodes.

- GCN-Based GAE Framework: GAE is widely used for detecting graph anomalies. Methods like DOMINANT (K. Ding et al. 2019) use GCN as the encoder to detect global and structural anomalies. Dual-SVDAE (F. Zhang et al. 2022) and GUIDE (H. Chen et al. 2019) enhance anomaly detection by addressing

complex interactions and high-dimensional graphs.

- GCN Framework: Techniques like semi-GCN (Kumagai et al. 2021) and HCM (T. Huang et al. 2023) leverage both supervised and unsupervised learning for detecting global anomalies. Methods like ResGCN (Pei et al. 2021) and CoLA (Y. Liu, Z. Li, et al. 2021) address issues of sparsity and over-smoothing.

- GAT-Based GAE Framework: GAT-based approaches, such as AnomalyDAE (H. Fan et al. 2020) and GATAE (You et al. 2020), overcome the limitations of GCN by using attention mechanisms to better capture complex interactions.

- Anomalous Edge Detection: This task focuses on identifying atypical interactions between nodes. Methods like AANE (D. Duan et al. 2020) and eFraudCom (G. Zhang et al. 2022) use GCN-based GAE frameworks to detect anomalous edges by adjusting fitting and anomaly-aware losses.

In summary, GNN-based approaches for anomaly detection in static graphs utilize various architectures and learning strategies to identify anomalies at the node, edge, and subgraph levels, leveraging the strengths of GNNs in handling complex graph data.

2.3 Poisoning Attacks against Graph Neural Networks

Recent studies have highlighted that adversarial attacks on training data, known as data poisoning attacks, can significantly compromise the training phase of graph neural networks (Jin, Y. Li, et al. 2020). These attacks involve subtle, deliberate modifications to nodes and edges within the graph data, aiming to undermine the training process (B. Zhang et al. 2024). Data poisoning attacks pose a considerable threat to the reliability of GNNs by deteriorating their performance. For example, spammers on social networks can act as adversarial samples, disseminating fake and sensitive content, hijacking trending topics, and misusing mention functions. These

actions can disrupt the utility of GNNs applied to normal users due to extensive connections within the graph, leading to less accurate predictions and threatening model reliability.

While defensive strategies for data types like images have been extensively explored, the complexity of graph structures complicates the defense against attacks. In graphs, perturbations targeting even a single node can impact many connected nodes due to their interconnected nature (Zihan Liu, Luo, Lirong Wu, S. Li, et al. 2022).

Data poisoning attacks on graphs can be formulated as a bilevel optimization problem:

Given a graph $G = (A, X)$, let $f : G \rightarrow Z$ be a GNN model, where Z denotes the node embeddings. The data poisoning attacks on graphs can be formulated as

$$\max_{G' \in F} L_{atk}(G', f^*), s.t. f^* = argmin_f L(G', f), \tag{2.2}$$

where L_{atk} denotes the attacker's objective function and F denotes the feasible space of poisoned graphs. A and X represent the adjacency matrix and attributes of G, respectively

Poisoning attacks can be categorized based on the attacker's knowledge into three types: white-box, gray-box, and black-box attacks. In white-box attacks, the attacker has complete access to the victim model and data, including model parameters, training data, and ground truth labels, which allows them to precisely tailor their attack strategy and generate fine-tuned perturbations that degrade the model's performance. The attacker's ability to manage and modify a significant portion of the dataset enables the introduction of highly effective poisoned data points, making white-box attacks the most effective among the three types. Gray-box attacks, on the other hand, provide limited knowledge, excluding access to exact model parameters. The attacker may have some information about the model

architecture or hyperparameters and often uses a surrogate GNN model to predict the victim models behavior. While gray-box attacks can still be potent, they are generally less effective than white-box attacks because the surrogate model may not perfectly replicate the victim model's behavior, leading to less precise perturbations and a reduced ability to craft highly effective poisoned data. In black-box attacks, the attacker's knowledge is further reduced, granting them access only to input graph data and corresponding model output. This minimal knowledge forces the attacker to rely on trial and error or heuristic-based methods, resulting in less targeted and more generalized perturbations. Additionally, the attacker's limited control over the dataset in black-box scenarios makes it more challenging to significantly degrade the model's performance. The effectiveness of a poisoning attack is directly influenced by the level of knowledge the attacker has and their ability to manage and manipulate the dataset. With more control, as seen in white-box scenarios, the attacker can achieve a greater impact, while reduced knowledge and control in gray-box and black-box scenarios lead to less effective attacks. We summarize the poisoning attacks against GNNs for different applications below:

Node Classification. For node classification tasks, white-box poisoning attacks can be framed as bilevel optimization problems, using methods like Projected Gradient Descent (PGD) (Jorge et al. 2006). Nettack (Zügner et al. 2018), a seminal gray-box method, uses a simpler GCN surrogate to optimize by greedily flipping edges. Metattack (Daniel et al. 2019) employs meta-learning to approximate the meta gradient of the loss function, while AtkSE (Zihan Liu, Luo, Lirong Wu, S. Li, et al. 2022) incorporates strategies for edge discrete sampling and momentum gradient ensemble to mitigate optimization instability. Wang et al. (B. Wang and Gong 2019) reformulated poisoning attacks against collective classification as a constrained optimization problem, using Lagrange multipliers with PGD. Beyond edge flipping, attackers may introduce adversarial nodes into the graph, as seen in NIPA (Y. Sun

et al. 2020). G-FairAttack (B. Zhang et al. 2024) presents a gray-box approach that subtly undermines group fairness without noticeably affecting model utility.

Link Prediction. Opt-Attack (M. Sun et al. 2018) exemplifies how the quality of unsupervised graph embeddings, such as DeepWalk (Perozzi et al. 2014), can be compromised in link prediction tasks. When targeting knowledge graphs, attackers may alter links or entities to disrupt embeddings, reducing link prediction accuracy. Methods like those proposed by Bhardwaj et al. (Bhardwaj et al. 2021) use influence functions to identify and replace the knowledge triples that most significantly impact predictions. Some methods, like Milani Fard and Wang's (Milani Fard et al. 2013), use neighborhood randomization to locally disrupt link predictions, while others, such as Yu et al.'s evolutionary graph community attack (Yu et al. 2018), focus on defending link privacy but struggle with computational demands. Sun et al.'s technique (M. Sun et al. 2018) leverages gradient descent to attack node embeddings, impacting link prediction. Notably, Chen et al.'s IGA (J. Chen et al. 2020) and Ding et al.'s VertexSerum (R. Ding et al. 2023) stand out by directly manipulating graph structures and node attributes, respectively, showcasing diverse approaches to undermining GNN-based link prediction systems.

Anomaly Detection. Data poisoning attacks also impact community detection and network embedding. CD-Attack (J. Li et al. 2020) generates adversarial graphs to obscure targeted entities, while methods like those by Bojchevski et al. (Perozzi et al. 2014) maximize the unsupervised loss function for DeepWalk. These attacks can extend to other graph embedding methods such as node2vec (Grover et al. 2016) and GCN. Zhou et al. (X. Zhou et al. 2022) introduce a hierarchical poisoning attack (HAA) strategy aimed at GNN-based intrusion detection in IoT systems, designed to operate within a limited budget. This black-box approach alters node features through a hierarchical node selection process, which uses a random walk with restart (RWR) to identify and prioritize the most vulnerable nodes for attack.

Similarly, Venturi et al. (Venturi et al. 2024) describe an evasion attack targeting intrusion detection systems by either perturbing the feature values of malicious netflows (feature attacks) or modifying the structure of the test graph (structural attacks).

Table 2.1: Categorization of representative attack methods

Attack Methods	Attack Knowledge	Targeted or Non-targeted	Evasion or Poisoning	Perturbation Type	Application	Victim Model
(Jorge et al. 2006)	White-box	Untargeted	Evasion	Add/Delete edges	Node Classification	GNN
(B. Wang and Gong 2019)	White-box, Gray-box	Targeted	Poisoning	Add/Delete edges	Node Classification	GNN
(B. Wang and Gong 2019)	White-box, Gray-box	Targeted	Poisoning	Add/Delete edges	Node Classification	GNN
(B. Wang, Pang, et al. 2023)	White-box, Gray-box	Untargeted	Poison-ing/Evasion	Add/Delete edges	Node Classification	GNN
(Zügner et al. 2018)	Gray-box	Targeted	Both	Add/Delete edges, Modify features	Node Classification	GNN
(Daniel et al. 2019)	Gray-box	Untargeted	Poisoning	Add/Delete edges	Node Classification	GNN
(Y. Sun et al. 2020)	Gray-box	Untargeted	Poisoning	Inject nodes	Node Classification	GNN
(R. Ding et al. 2023)	Black-box	Untargeted	Poisoning	Add/Delete edges/Features	Link prediction	GNN
(J. Chen et al. 2020)	Gray-box	Untargeted	Poisoning	Add/Delete edges	Link prediction	GNN
(M. Sun et al. 2018)	White-box	Untargeted	Poisoning	Add/Delete edges	Link prediction	Network Embedding
(Venturi et al. 2024)	Gray-box	Both	Evasion	Add/Delete edges/Features	Intrusion Detection	GNN
(Perozzi et al. 2014)	Black-box	Both	Poisoning	Add/Delete edges	Node Classification, Community Detection	Network Embedding
(X. Zhou et al. 2022)	Black-box	Targeted	Poisoning	Add/Delete features	Intrusion detection	GNN
(J. Li et al. 2020)	Black-box	Targeted	Poisoning	Add/Delete edges	Community Detection	Community Detection Algorithm

In summary, data poisoning attacks on GNN models involve sophisticated techniques targeting various graph-related tasks, posing significant threats to model reliability.

Chapter 3

Cost Aware Untargeted Poisoning Attack against Node Classification

Graph Neural Networks (GNNs) have become widely used in the field of graph mining. However, these networks are vulnerable to structural perturbations. While many research efforts have focused on analyzing vulnerability through poi- soning attacks, we have identified an inefficiency in current attack losses. These losses steer the attack strategy towards modifying edges targeting misclassified nodes or resilient nodes, resulting in a waste of structural adversarial perturbation. To address this issue, we propose a novel attack loss framework called the Cost Aware Poisoning Attack (CA-attack) to improve the allocation of the attack budget by dynamically considering the classification margins of nodes. Specifically, it prioritizes nodes with smaller positive margins while postponing nodes with negative margins. Our experiments demonstrate that the proposed CA-attack significantly enhances existing attack strategies.

3.1 Preliminaries

3.1.1 GNNs

We define an undirected graph as $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_N\}$ represents the node set, and $e_{ij} \in E$ is edge connecting the nodes v_i and v_j. Node attributes are represented by $X \in \mathbb{R}^{N \times d}$, where d is the dimension of the attribute. Additionally, the graph structure can be represented by an adjacent matrix $A \in \{0, 1\}^{N \times N}$, with $A_{ij} = 1$ if an edge exists between two nodes, otherwise 0. Given a subset of labeled nodes $V_L \subset V$ where each node $v \in V_L$ has a label $y_v \in \mathcal{C} = \{c_1, c_2, \cdots, c_k\}$, GNNs aim to learn a function f_θ to predict the remaining unlabeled nodes $V_U = V \setminus V_L$ into classes of $\mathcal{C}$. We use $f_\theta(G)_v$ to denote the prediction of the model f_θ for node v. Its parameters θ are optimized by minimizing a loss $\mathcal{L}_{train}$ over the labeled nodes, typically using losses like negative log-likelihood loss or C&W loss (Carlini et al. 2017).

3.1.2 Poisoning Attacks against GNNs

Based on the classification task, the general form of node-level graph poisoning attacks can be defined as a bi-level optimization process where the attacker optimizes the selection of the edges to be attacked to maliciously modify the graph structure, while the parameters of a surrogate model are optimized using this poisoned graph:

$$\begin{aligned} A' &= arg\,\min \mathcal{L}_{atk}\left(f_{\theta^*}\left(\widehat{G}\right)\right) \\ s.t., \theta^* &= arg\,\min_{\theta} \mathcal{L}_{train}\left(f_\theta\left(\widehat{G}\right)\right), \end{aligned} \tag{3.1}$$

where $\mathcal{L}_{atk}$ is the attack loss and usually chosen as $\mathcal{L}_{atk} = -\mathcal{L}_{train}$, $\widehat{G}$ is the graph modified from original G by the adversarial attack, y_v denotes the labels of node v.

Given a budget Δ, the attacker aims to ensure that perturbations are unnoticeable by maintaining the l_0 norm difference between the original and perturbed graph: $\|A - A'\|_0 \leq \Delta$. Δ is generally no more than 10% of the number of edges in the original graph. To avoid detection, the attacker also refrains from making significant changes to the graph's degree distribution or introducing isolated nodes, as suggested in (Zügner et al. 2018).

A conventional poisoning attack process can be divided into three steps. The first step is to retrain a surrogate model $f_{\theta^*}(G)$ using a linearized graph convolutional network (GCN) which is expressed as:

$$f_\theta(G) = soft\max\left(\hat{A}^2 XW\right), \tag{3.2}$$

where $\hat{A} = D^{-1/2}(A + I)D^{-1/2}$ is the normalized adjacent matrix, X are the node features, D is the diagonal matrix of the node degrees, and $\theta = \{W\}$ are the set of learnable parameters. In the second step, the attacker uses pseudo-labels y_v generated by the surrogate model $f_{\theta^*}(G)$ to construct the attack loss of the node v as $\ell\left(f_{\theta^*}\left(\widehat{G}\right)_v, y_v\right)$ under the gray-box attack scenario. In the third step, the attack loss of each node is backpropagated to produce a partial gradient matrix:

$$g_v = \nabla_A \ell\left(f_{\theta^*}\left(\widehat{G}\right)_v, y_v\right), \tag{3.3}$$

The overall gradient information passed to the attack strategy is the average of all the partial gradient matrices. Thereafter, the attacker selects the edges to be perturbed (adding/deleting) based on the saliency of their gradients.

3.1.3 Attack Budget

For structural poisoning attacks, the attack budget refers to the number of edges that can be added or removed by the attacker. Usually, attacks can only have a

Δ budget, meaning that the original adjacency matrix and the modified adjacency matrix should be limited as follows:

$$\sum |A - \hat{A}| \leq \Delta, \tag{3.4}$$

3.1.4 Classification Margin

The classification margin of a node v is commonly defined as:

$$\varphi(v) = z_{c^*} - \max_{c \neq c^*} z_c, \tag{3.5}$$

where z represents the vector of logits produced by the model towards node v, and c^* refers to the true label of the node v. If a node has a negative margin, it indicates misclassification.

3.2 Limitations of Previous Attacks

Graph Neural Networks (GNNs) (Jin, Y. Ma, et al. 2020; Velikovi et al. 2018) have emerged as an effective machine learning approach for structured data, generalizing neural networks to manage graph-structured information. They have shown potential in various applications such as social network analysis (W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, et al. 2019), anomaly detection (Chaudhary et al. 2019; Jianheng Tang et al. 2022; J. Duan et al. 2023), and natural language processing (Lingfei Wu et al. 2023), excelling in tasks like node classification and link prediction. However, current research (Dai et al. 2018; Lai et al. 2023; Zhu, Lai, et al. 2022; Zhu, Michalak, et al. 2022) indicated that they are susceptible to poisoning attacks, where the attack easily manipulates the graph structure (i.e., adding/deleting edges) before the training of the GNN models, and leads to incorrect predictions. Such vulnerabilities not only compromise the model's reliability but also present opportunities for malicious attacks.

To analyze the vulnerabilities of GNNs, several studies have investigated potential poisoning attacks on these models, such as Minmax (H. Chen et al. 2019), Metattack (Daniel et al. 2019), and the Certify Robustness Inspired Attack Framework (B. Wang, Pang, et al. 2023). These attacks leverage gradient-based techniques to approximate the complex bi-level optimization problem associated with poisoning attacks. However, there is still room for improvement in current methodologies, particularly regarding the allocation of the limited perturbation budget.

Poisoning attacks (Dai et al. 2018) occur during the training phase of machine learning models and involve altering the training data by introducing perturbations. These attacks can lead the model to produce incorrect predictions during the testing phase. Various methodologies can facilitate poisoning attacks, including node injection (Y. Sun et al. 2020), feature modification (Q. Li et al. 2023), and edge perturbation (Daniel et al. 2019). This discussion focuses on untargeted attack strategies that employ edge perturbation.

A notable contribution in this area was made by Zügner et al. (Daniel et al. 2019), who introduced the Meta-Self attack model. This model uses the meta-gradient of a surrogate model to modify the graph's structure or node features within a gray-box context, marking the first instance of a gradient-based attack strategy. Following this, subsequent research (H. Wu et al. 2019) has explored new tactics and improvements, focusing on utilizing gradient data from the surrogate model to manipulate node features and the graph's structure. These studies typically employ an attack objective function, often a negative cross-entropy loss (H. Wu et al. 2019; Daniel et al. 2019), to guide the gradient's backpropagation through the adjacent matrix.

However, current approaches face limitations in budget allocation, largely due to the design of the attack objective function. For example, Metattack, which uses loss functions such as cross entropy loss and C&W loss (Carlini et al. 2017), can suffer from budget waste. In such cases, some budget allocations do not contribute

to decreased classification accuracy. Specifically, when employing cross entropy as the attack objective, a node that has already been successfully misclassified tends to attract higher meta-gradients, resulting in further budget allocation to that specific node.

To address these issues, Wang et al. (B. Wang, Pang, et al. 2023) proposed a novel attack loss framework called the Certify Robustness Inspired Framework (CR-framework) to enhance attack performance. This method assigns larger meta-gradients to nodes with smaller certify robustness sizes, as these nodes are more vulnerable to attacks. It accomplishes this by reweighting the loss of each node according to its certify robustness size. While this approach optimizes budget allocation for existing attack losses and further decreases the accuracy of victim models, it faces computational inefficiencies during the assessment of certified robustness sizes. We show this budget waste problem by the gradient of every node's loss on adjacent matrix. The attack strategy involves a basic edge perturbation selection method based on the saliency of gradients, represented as follows:

$$\tilde{A}^{\text{grad}} = \begin{cases} A_{i,j}^{\text{grad}} & \text{if } A_{i,j}^{\text{grad}} > 0 \text{ and } A_{i,j} = 0 \\ -A_{i,j}^{\text{grad}} & \text{if } A_{i,j}^{\text{grad}} < 0 \text{ and } A_{i,j} = 1 \\ 0 & \text{otherwise,} \end{cases} \tag{3.6}$$

where $\tilde{A}^{grad}$ filters out candidates that the gradient signs and edge states are inconsistent, A^{grad} is the average gradient information of all the partial gradient matrices g_v. $A_{i,j}$ represents the edge between node i and node j. $A_{i,j}^{grad}$ is the gradient of edge between node i and node j. Since poisoning attack aims to minimize the attack loss (i.e., maximize training loss) by modifying an edge with the largest gradient at each iteration, the node connected to this edge will be attacked with higher priority according to Equation (3.3). Figure 3.1 illustrates the priority of a node in a clean graph to be attacked based on the l_2 norm of partial gradient matric for each node

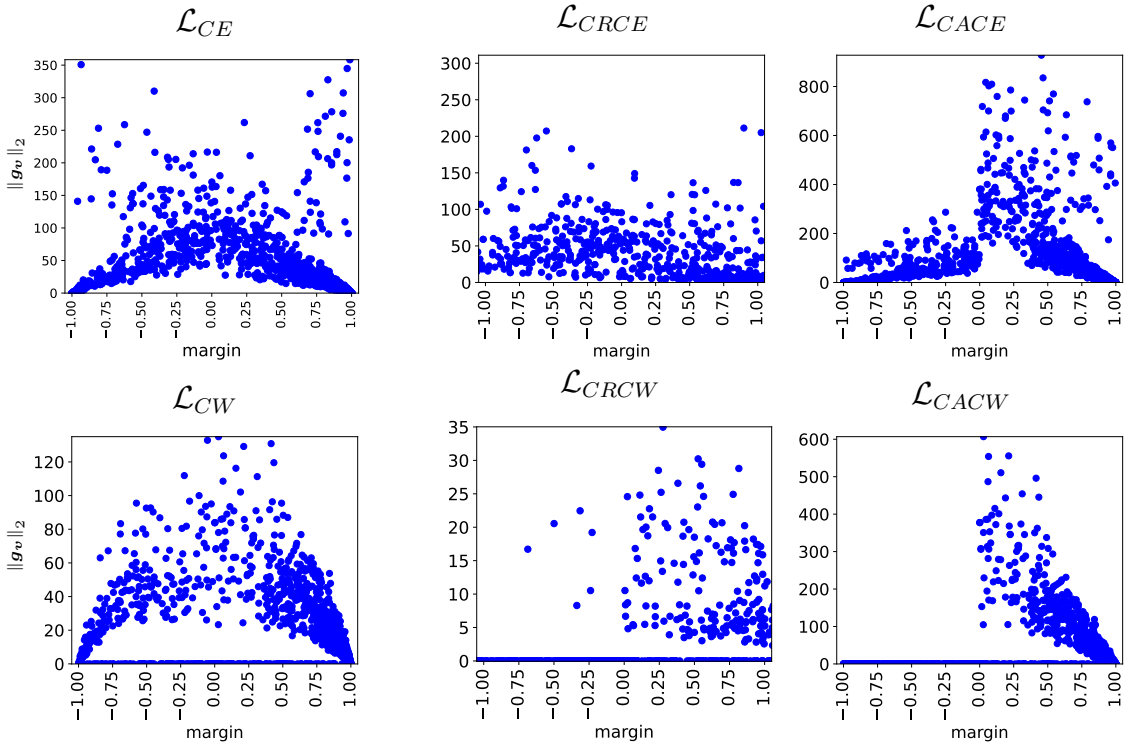


Figure 3.1: Scatterplot of the nodes margins with l_2 norm of their partial gradient matrices of cora dataset. The terms $\mathcal{L}_{CE}$, $\mathcal{L}_{CRCE}$, and $\mathcal{L}_{CACE}$ respectively represent the cross entropy loss, the CR framework, and the CA framework and terms $\mathcal{L}_{CW}$, $\mathcal{L}_{CRCW}$, and $\mathcal{L}_{CACW}$ respectively represent the C&W loss, the CR framework, and the CA framework

in Metattack using different attack loss functions. We can observe that the cross entropy loss and the CR framework result in significant gradients for nodes with negative margins. Additionally, the cross entropy loss produces significant gradients for nodes with large margins. In contrast, our CA framework generates significant gradients on nodes with small positive margins. We can also observe that the C&W loss imparts significant gradients to nodes possessing negative margins and the CR framework exhibits difficulty concentrating on nodes with minute margins when subjected to C&W loss. When applying the CA framework to the C&W loss, we only assign weights to nodes with positive margins, while nodes with negative margins are given a weight of zero. In Section 5.1, we will demonstrate that the CA attack loss outperforms previous attack losses.

To address the limitations above, we introduce a Cost Aware Poisoning Attack

Loss Framework (CA-attack) to improve the allocation of attack budget to maximize the impact of the attack. Specifically, we dynamically reweight the nodes according to their classification margins in the attack loss. This means that the weights of the nodes are adjusted during the optimization process of the attack objective. Through extensive testing on benchmark datasets, the CA-attack consistently surpasses previous attack methods in terms of effectiveness. Our research makes the following contributions:

- To address the inefficiencies of budget allocation, we propose the CA-attack, a budget-efficient attack loss framework.
- Through rigorous empirical assessments on three datasets, we demonstrate that CA-attack improves existing methods, highlighting its potential as a plug-and-play solution for various graph poisoning attacks.

3.3 Cost-Aware Attack

In this section, we demonstrate our proposed CA-attack attack loss framework that aims to improve the approximation of this challenging bi-level optimization problem (3.6) by redesigning the $\mathcal{L}_{atk}$. Specifically, we incorporate the classification margin of nodes as dynamic weights for the attack loss.

3.3.1 Attack Model

We present an overview of our attack model in Figure 3.2, illustrating the flow of perturbing an edge in a graph. The first step involves retraining a surrogate model $f_{\theta^*}(G_t)$ with the perturbed graph G_t. This surrogate model is a linear 2-layer graph convolutional network (GCN): $f_\theta(G) = \text{softmax}(\hat{A}^2XW)$, where $\hat{A} = D^{-1/2}(A+I)D^{-1/2}$ is the normalized adjacency matrix, and $W$is a learnable weight matrix.

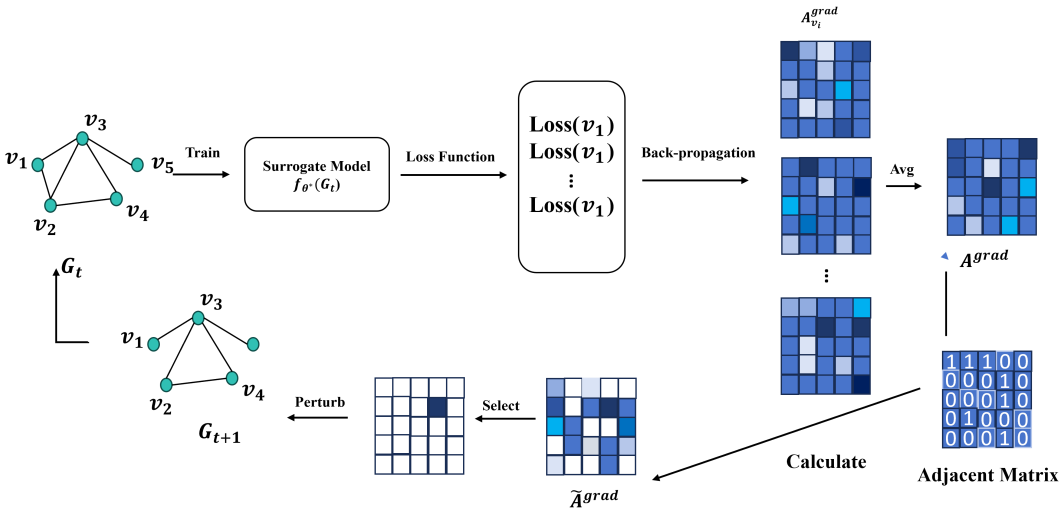


Figure 3.2: Poisoning attack model generating one perturbation

In the second step, we construct the attack loss for each node using pseudo-labels. The pseudo-labels are derived from the surrogate model $f_{\theta^*}(G)$ trained on the original graph G. We ensure the surrogate model is consistent with the model generating pseudo-labels to avoid inconsistencies. The attack loss L_{atk} for node v_i relates to the margin $\varphi(v)$ and pseudo-labels detailed in Section 3.3.2.

In the third step, the attack loss is backpropagated to produce partial gradient matrices $A^{v_i}_{\text{grad}}$, representing the structural gradient from each node v_i. For a GCN aggregating h times, a node generates gradients on edges with its h-hop neighbors. The overall gradient A_{grad} is the average of all partial gradients. The attack strategy is flipping the edge with the largest gradient. The modified graph then enters the next loop, and this process continues until the loop ends.

3.3.2 Cost Aware Loss

Intuitively, a larger margin suggests that a node is more resilient to attacks, whereas a smaller but positive margin suggests a higher potential for a successful attack.

With this intuition in mind, we propose the introduction of a cost-aware loss (CA-loss) that takes into account the margins of nodes. By doing so, our aim is to assign higher priority to nodes with smaller but positive margins. While a simplistic

strategy is to rank the nodes based on margins and perturb them sequentially, this approach is computationally demanding and potentially suboptimal due to the complex interplay of nodes and edges in predictions. Instead, we refine the attack loss $\mathcal{L}_{atk}$ by incorporating node margins to weight the nodes dynamically. The resulting CA-loss is defined as follows:

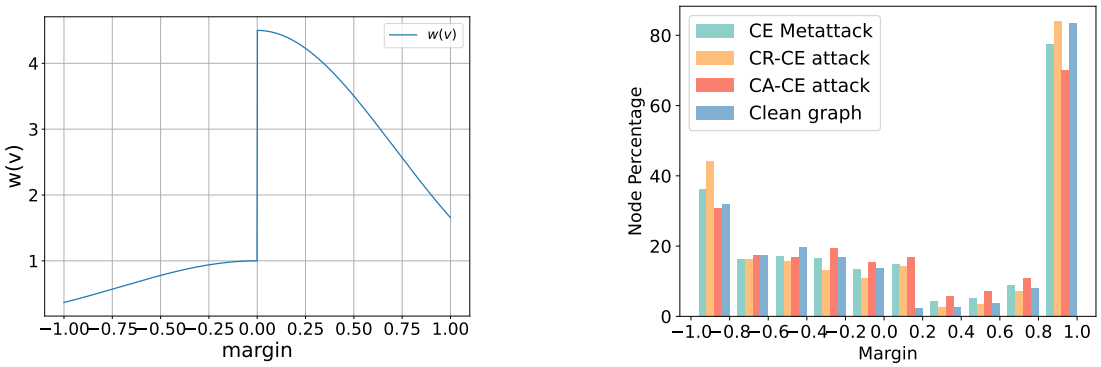
$$\mathcal{L}_{CA}\left(f_{\theta^*}, \widehat{G}\right) = \sum_{v \in V_U} w\left(v\right) \cdot \ell\left(f_{\theta^*}\left(\widehat{G}\right)_v, y_v\right), \tag{3.7}$$

where $w\left(v\right)$ is the weight of the node v. When all nodes are assigned equal weight, our CA-loss reduces to the conventional loss. To introduce the weight $w(v)$, which captures the inverse relationship between the node's margin $\varphi(v)$ and the weight, we utilize the exponential function. The reasoning behind using the exponential function is that it provides a smooth and continuous way to assign higher weights to nodes with smaller margins, thereby prioritizing them in the loss function. The weight $w(v)$ is defined as follows:

$$w(v) = \alpha \times e^{-\beta \times \varphi(v)^2}, \tag{3.8}$$

where α and β are tunable hyper-parameters. The weight $w(v)$ exponentially decreases as the node's margin $\varphi(v)$ increases. This design ensures that nodes with smaller margins (i.e., those more vulnerable to attacks) receive significantly higher weights, thereby focusing the attack efforts on these nodes. The exponential function is particularly suitable for this purpose because it rapidly decreases, allowing for a clear distinction in priority between nodes with small and large margins.

By dynamically weighting the nodes based on their margins, our CA-loss function effectively balances the computational complexity and the attack efficacy, ensuring that the most vulnerable nodes are targeted more aggressively without the need for exhaustive ranking and sequential perturbation. Figure 3.3 (a) is a visualization of $w(v)$.



(a) (b)

Figure 3.3: (a) Comparative illustration of node weights in Cora dataset. (b) Node distribution of different margin ranges with a 5% perturbation ratio in Cora dataset.

3.3.3 Theoretical Analysis of Budget Allocation

In this section, we analyze the effectiveness of CA-loss from a theoretical perspective.

Margin-based prioritization is a key advantage of the CA-loss (Cost-Aware Loss) approach over traditional loss functions such as the Cross Entropy Loss(CE loss) and C&W loss. Traditional loss functions do not consider the specific vulnerability of each node, treating all misclassifications or margin violations equally. In contrast, CA-loss introduces a weighting mechanism based on node margins, assigning higher weights to nodes with smaller margins. This ensures that the most vulnerable nodes are targeted more aggressively. The theoretical advantage of this approach can be understood through the concept of gradient magnitude. For a node v, the margin φ indicates its vulnerability. CA-loss assigns higher weights $w(v)$ to nodes with smaller margins, effectively increasing their contribution to the overall loss gradient. The weighted gradient for node v is given by

$$\nabla_A L_{CA}(v) = w(v) \cdot \nabla_A L(f_{\theta^*}(v), y_v), \tag{3.9}$$

where L represents the underlying loss function. For CE loss,

$$L_{CE} = -\sum y_v \log P(y_v | f_{\theta^*}(v)), \tag{3.10}$$

and for C&W,

$$L_{C\&W} = \max(0, f_{\theta^*}(v)_{c^*} - f_{\theta^*}(v)_c + \kappa), \tag{3.11}$$

CE loss measures the difference between the predicted probability distribution and the true distribution. For a given input v, the model $f_{\theta^*}(v)$ predicts a probability distribution $\log P(y_v|f_{\theta^*}(v))$ over possible classes. The CE loss calculates the negative log-likelihood of the true class under this predicted distribution. If the confidence of a node's true class is small (less than 0.5), meaning the margin of this node is negative, the loss value of CE is large on this node, resulting in a large gradient impact on the adjacency matrix. That means CE loss leads to more budget allocation to nodes with negative margins, resulting in budget wastage. The C&W loss compares the model's output for the true class c and a target class c^*. The goal is to make the output for the target class c^* larger than the output for the true class c by at least a margin κ. The difference of a node's loss impact on budget allocation is due to the confidence difference between the true label and the target label, thus it will not allocate much budget to nodes with negative margins because the loss is 0 for these nodes. According to the loss formulation, C&W loss theoretically has less wastage than CE loss.

Integrated with the CA loss framework, nodes with smaller positive margins have larger weights $w(v)$, thus their gradients are magnified, guiding the attack optimization process more strongly towards these vulnerable nodes. By focusing on nodes with smaller positive margins in a dynamic poisoning attack setting, meaning that the weight for every node will be adjusted according to their changing margins during the attack process, CA-loss reduces the attack budget spent on nodes that are already misclassified and extremely resilient to attacks. This ensures a more effective attack by concentrating efforts on the most vulnerable nodes.

Furthermore, the exponential weighting mechanism in CA-loss provides another layer of advantage. Traditional loss functions do not differentiate between nodes based on their margin sizes, potentially leading to suboptimal attack strategies. CA-loss uses the exponential weighting mechanism stated in function 4.12, ensuring that even small differences in margins are magnified, providing clear prioritization of nodes. This smooth and continuous priority avoids abrupt changes that might destabilize the optimization process. For small positive values of φ, the weight $w(v)$ is large, ensuring high priority. The rapid decrease in weight as φ increases ensures that nodes with larger margins do not dominate the loss function. This is expressed as

$$w(v) = \alpha \cdot e^{-\beta \times \varphi^2} \quad \text{where} \quad \frac{\partial w(v)}{\partial \varphi} = -2\beta \cdot \varphi \cdot w(v), \tag{3.12}$$

This rapid decrease ensures the focus remains on the most vulnerable nodes.

By incorporating the margin-based prioritization and exponential weighting mechanism, CA-loss provides a theoretically superior approach to traditional loss functions.

3.3.4 Cost Aware Poisoning Attack Algorithm

In our approach to conducting a poisoning attack, we operate under the assumption that the attacker lacks access to the target classifier's internal parameters, its output, or any explicit knowledge of its architectural design. To navigate this limitation, the attacker employs a surrogate model as a stand-in to facilitate the poisoning attacks. The manipulated data resulting from this attack is then utilized to train deep learning models, such as a Graph Convolutional Network (GCN), with the aim of assessing the extent of performance degradation inflicted by the attack.

By integrating the cost-aware attack loss function, which we previously discussed in Section 3.3.2, with the optimization goal delineated in (3.6), we present Algorithm

1. This algorithm outlines the procedural steps required to tackle this optimization challenge effectively.

Algorithm 1 Cost-aware Poisoning Attack on Graph Neural Networks

Input: Graph $G = (A, X)$, modification budget Δ, number of training iterations T, training class labels CL, node classification margin $\varphi(v)$
Output: Modified graph $\hat{G} = (\hat{A}, X)$
$\theta^* \leftarrow$ train surrogate model on G using CL
$\hat{C}_U \leftarrow$ predict labels of unlabeled nodes using θ^*
$\hat{A} \leftarrow A$
while $\|\hat{A} - A\|_0 < 2\Delta$ do
    Randomly initialize θ_0 of surrogate model
    for $t = 0$ to $T - 1$ do
        $\theta_{t+1} \leftarrow \text{step}(\theta_t, \nabla_{\theta_t} L_{\text{train}}(f_{\theta_t}(\hat{A}, X), CL))$
    end for
    Calculate $\varphi(v)$ for each node v to form the margin matrix $\Phi(V)$
    $\nabla \hat{A} \leftarrow \nabla_{\hat{A}} \mathcal{L}_{CA}(f\theta_T(\hat{A}, X), \hat{C}_U, \Phi(V))$
    $\hat{A} \leftarrow$ modify by adding or deleting the edge with the most significant gradient influence
end while
$\hat{G} \leftarrow (\hat{A}, X)$
return $\hat{G}$

Chapter 4

Applicability to Other Graph Learning Tasks

4.1 Cost Aware Poisoning Attack against Graph Neural Networks for Link Prediction

Graph Neural Networks (GNNs) have revolutionized the processing of graph structured data, finding widespread applications from social network analysis to fraud detection. However, the use of sensitive graph connections such as social ties and transaction records poses significant security risks. Current link inference attacks, while effective, often suffer from inefficient budget allocation, leading to diminished attack impact due to wasted resources.

In the realm of link prediction attacks, various strategies target different aspects of GNNs to compromise their predictive capabilities. Some methods, like Milani Fard and Wang's (Milani Fard et al. 2013), employ neighborhood randomization to disrupt link predictions locally, while others, such as Yu et al.'s evolutionary graph community attack (Yu et al. 2018), aim to defend link privacy but face practicality issues due to computational demands. Techniques like Sun et al.'s leverage gradient descent for attacking node embeddings with downstream effects on link

prediction (M. Sun et al. 2018). Notably, Chen et al.'s IGA (J. Chen et al. 2020) and Ding et al.'s VertexSerum (R. Ding et al. 2023) stand out by directly manipulating graph structures and node attributes, respectively, demonstrating the diverse approaches to undermining GNN-based link prediction systems. Nonetheless, the current approaches face limitations in budget allocation, largely attributable to the design of the attack objective function. In our work, we propose a refined attack objective function that prioritizes edges based on their vulnerability to attacks. Recognizing that not all edges possess the same level of robustness, we advocate for a strategic allocation of the attack budget, targeting edges that are less resilient to ensure a more effective utilization of resources. To counter these challenges, we present MetaLinkAttack, a novel graph poisoning approach crafted to maximize the strategic use of the graph structure budget. Distinctively, MetaLinkAttack assigns varying weights to edges based on their prediction probability margins, aiming to degrade the inferential accuracy of GNNs more effectively. Leveraging meta-gradient techniques, MetaLinkAttack intricately solves the bilevel optimization problem by considering the graph structure as an optimizable hyperparameter. Our comprehensive evaluations across several benchmark datasets indicate that MetaLinkAttack outperforms existing state-of-the-art link inference attacks by judiciously allocating the attack budget to significantly lower the Area Under the Curve (AUC) scores, thereby demonstrating its superior efficiency and effectiveness in disrupting GNN performance.

4.1.1 Limitations of Previous Attacks

Graph Neural Networks (GNNs) have become integral to analyzing complex data structures across multiple fields, including finance (D. Wang et al. 2019), social sciences (J. Sun et al. 2022), and healthcare (Choi et al. 2017), due to their ability to capture high-dimensional attributes and intricate network relationships (J. Zhou et al. 2020). The growing application of GNNs in these sensitive areas raises significant

security concerns, especially where graph data encapsulate confidential relationships.

Recent research has unveiled vulnerabilities in Graph Neural Networks (GNNs), especially through poisoning attacks that degrade link prediction performance. Chen et al.'s Iterative Gradient Attack (IGA) (J. Chen et al. 2020) employs gradient information from a graph autoencoder model's loss function, targeting graph edges to maximize training loss, effectively compromising the structure. Meanwhile, Ding et al.'s VertexSerum (R. Ding et al. 2023) introduces a novel approach, aiming to intensify private link information leakage by altering node attributes. This diversity in methods enriches the spectrum of strategies against GNNs, each exploiting different aspects of graph data for potential breaches.

A notable limitation in existing link inference attacks by adding or deleting edges is their inefficiency selecting edges to attack. We have found that current attack framework with negative log likelihood loss has budget waste problem that some budgets do not contribute to decrease the link prediction accuracy. For example, when employing the commonly used negative log likelihood loss as the attack objective, a link that has already been successfully misclassified tends to attract higher meta-gradients, resulting in a further allocation of the attack budget to that specific edge.

This paper presents MetaLinkAttack, which introduces an innovative data poisoning strategy for GNNs, optimizing attack budget utilization for heightened efficacy. The attack dynamically adjusts edge weights based on prediction probability margins, refining the attack's focus during optimization. Comprehensive evaluations on standard datasets demonstrate MetaLinkAttack's superior performance over existing methods. This research contributes a novel edge reweighting mechanism in attack strategies, enhancing the understanding and development of robust GNNs.Our research makes the following contributions:

- To address the inefficiencies in budget allocation, we propose MetaLinkAttack, a budget-efficient attack framework specifically applied to the class imbalance

task of link prediction, where negative links far outnumber positive links.

- Through rigorous empirical assessments on three datasets, we demonstrate that MetaLinkAttack improves existing methods, highlighting its potential as a plug-and-play solution for various graph poisoning attacks.

4.1.2 Method

In this section, we define and formulate the link prediction attack problem and then introduce how a poisoned graph is generated by MetaLinkAttack.

4.1.2.1 Problem Definition

Definition 1 (Link Prediction): For a graph $G = (V, E)$, with V as nodes and E as links, E is divided into observable links E_o and predicted links E_u, with $E_o \cap E_u = \varnothing$ and $E_o \cup E_u = E$. Link prediction aims to predict E_u using V and E_o.

Definition 2 (Poisoned Graph): For a graph $G = (V, E)$, a poisoned graph $\hat{G} = (V, \hat{E})$ introduces perturbations E_α to E, forming $\hat{E} = E + E_\alpha$.

Definition 3 (Link Prediction Poisoning Attack): Given G and target link E_t, the goal is to generate an poisoned graph $\hat{G}$ to impede accurate prediction of E_t. The attack can be formulated to maximize the inconsistency between the link prediction method's outputs on the original graph G and the poisoned graph $\hat{G}$ for a target link E_t:

$$\max_{\hat{G}} \; I(f_\theta(G, E_t) \neq f_\theta(\hat{G}, E_t)) \quad \text{s.t. } |E_\alpha| \leq \Delta, \tag{4.1}$$

where I represents the indicator function, f_θ is the graph neural network (GNN) model parameterized by θ, and Δ imposes a limit on the magnitude of perturbations E_α. This setup aims to maximize the difference in the GNN model's predictions between the original and perturbed graphs, within the bounds of allowed perturbations.

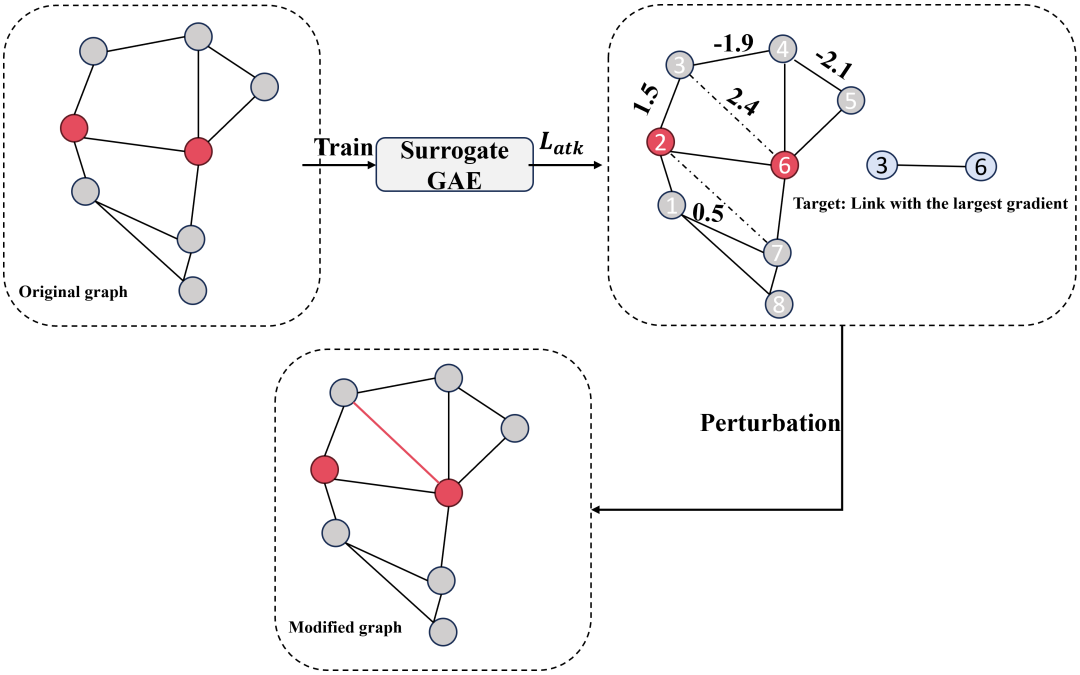


Figure 4.1: Poisoning attack model against link prediction generating one perturbation

Definition 4 (Classification Margin): The classification margin of an edge E_t is commonly defined as:

$$\varphi(E_t) = z_{c^*} - \max_{c \neq c^*} z_c, \tag{4.2}$$

where z represents the vector of logits produced by the model towards edge E_t, and c^* refers to the true label of the edge E_t. If an edge has a negative margin, it indicates misclassification.

4.1.2.2 Attack Model of MetaLinkAttack

Building upon IGA's framework for link prediction poisoning in graphs, the crux of poisoning attacks lies in crafting high-quality poisoned graphs, typically by targeting a well-established GNN model. Figure 4.1 presents the process of generating a poisoned graph in link prediction tasks.

Surrogate Model for Link Prediction. Graph Auto-Encoders (GAEs), leveraging Graph Convolutional Networks (GCNs), effectively encode node structural and feature information. Their encode-decode architecture, akin to CNNs in computer vision, allows for efficient extraction of hidden relational data between nodes, positioning them as nearly state-of-the-art for link prediction tasks. Therefore, we choose GAE as our surrogate model to perform MetaLinkAttack.

Given a graph with adjacency matrix A, the GAE model computes the node embedding matrix $Z \in \mathbb{R}^{N \times F}$ as:

$$Z(A) = \bar{A}\sigma\left(\bar{A} I_N W^{(0)}\right) W^{(1)}, \tag{4.3}$$

where $\bar{A} = \tilde{D}^{-\frac{1}{2}}(A + I_N)\tilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix, $\tilde{D}_{ii} = \sum_j (A + I_N)_{ij}$, and $W^{(0)} \in \mathbb{R}^{N \times H}, W^{(1)} \in \mathbb{R}^{H \times F}$ are weight matrices for the GCN layers.

Node pair similarity is calculated as:

$$\tilde{A} = s(ZZ^T), \tag{4.4}$$

with s being the sigmoid function. Links with scores above a threshold (set to 0.5) are predicted to exist.

Gradient Extraction and Poisoned Graph Generation. The poisoned graph is crafted by perturbing the original graph based on gradients from the GAE model. For a target link E_t, the loss function is defined as:

$$\mathcal{L}_{atk}\left(f_{\theta^*}, \widehat{G}\right) = \sum_{E_t \in E} w(E_t) \cdot \ell\left(f_{\theta^*}\left(\widehat{G}\right)_{E_t}, Y_t\right), \tag{4.5}$$

$$w(E_t) = \alpha \times e^{-\beta \times \varphi(E_t)^2}, \tag{4.6}$$

where Y_t denotes the true label for the link, and θ^* represents the parameters trained on a graph that has been altered for poisoning purposes. The term α and β are

adjustable hyperparameters influencing the weight $w(E_t)$, which decreases exponentially as the edge's margin $\varphi(E_t)$ increases. This weighting approach, alongside the gradient of the loss with respect to the adjacency matrix, is utilized to guide modifications in the graph's structure, aiming to increase the loss and thereby reduce the predictive accuracy of the model on targeted links.

The gradient with respect to the adjacent matrix of a target link E_t connecting node i and node j is given by:

$$g_{ij} = \frac{\partial \mathcal{L}_{atk}}{\partial A}, \tag{4.7}$$

To generate the poisoned graph, the approach accounts for the discrete nature of graph data, allowing only for the addition or removal of links based on the gradient's magnitude, considering the constraints of undirected graphs.

4.1.2.3 MetaLinkAttack Algorithm

In this section, we present the MetaLinkAttack algorithm to demonstrate the attack process on link prediction tasks. The goal of the MetaLinkAttack algorithm is to generate a poisoned graph that impedes the accurate prediction of a target link. This is achieved by strategically perturbing the original graph structure to maximize the difference in the graph neural network's (GNN) predictions between the original and poisoned graphs. The algorithm proceeds as follows: Start by initializing the poisoned graph $\hat{G}$ as the original graph G. Compute the initial node embeddings Z using the Graph Auto-Encoder (GAE) model on the original graph G. For each target link E_t, compute its classification margin $\varphi(E_t)$ and the corresponding weight $w(E_t)$, which captures the inverse relationship between the node's margin and the perturbation impact. While the perturbation budget $|E_\alpha|$ is within the allowed limit Δ, compute the gradients for each link in the graph, select the link with the maximum gradient magnitude to perturb the graph, update the poisoned graph $\hat{G}$ by adding or removing the selected link, and recompute the node embeddings using the GAE model on the updated graph. Finally, return the final poisoned graph $\hat{G}$.

Algorithm 2 Link Prediction Poisoning Attack (MetaLinkAttack)

Require: Graph $G = (V, E)$, Target link E_t, Perturbation budget Δ
Ensure: Poisoned graph $\hat{G} = (V, \hat{E})$

```
Initialize Ĝ = G
Compute initial node embeddings Z using the GAE model on G with loss function ℓ(Z, A)
while |E_α| ≤ Δ do
    for each target link E_t in E do
        Compute classification margin φ(E_t) using Eq. (4.2)
        Calculate weight w(E_t) using Eq. (4.6)
    end for
    for each link (i, j) in G do
        Compute gradient g_ij = ∂L_atk/∂A' using Eq. (4.7)
    end for
    Select link (i, j) with maximum gradient magnitude
    Perturb Ĝ by adding/removing selected link (i, j)
    Recompute node embeddings Z using the GAE model on Ĝ with loss function
      ℓ(Z, A')
end while
return poisoned graph Ĝ
```

In the MetaLinkAttack algorithm, the key component is the strategic perturbation of the graph based on the gradients derived from the GAE model. By focusing on the links with the highest gradient magnitudes, the algorithm ensures that the perturbations are both effective and efficient, thereby degrading the performance of the link prediction model.

4.2 Cost Aware Poisoning Attack against Anomaly Detection

Graph neural networks (GNNs) play a fundamental role in anomaly detection, excelling at identifying node anomalies by aggregating information from neighboring nodes. Nonetheless, they are vulnerable to attacks, where even minor alterations in the graph structure or node attributes can significantly degrade performance. Although there are some studies investigating attacks for community detection or anomaly detection, there is a lack of focus on GNN-based anomaly detection methods.

GNNs are crucial for anomaly detection, but their vulnerabilities in this context need further exploration. In our study, we first adapt the Nettack framework for GNNs and then apply our cost-aware poisoning attack framework, MetaAD, to demonstrate its impact on anomaly detection.

Our empirical findings, derived from extensive experiments conducted on benchmark anomaly detection datasets, show that our attack can degrade the performance of GNNs and reveal their vulnerabilities.

4.2.1 Anomaly Detection on Graphs

The purpose of anomaly detection is to distinguish between abnormal and normal items in datasets, which can be framed as a binary classification problem. However, obtaining sufficient labeled data is arduously expensive and sometimes infeasible. Therefore, we formulate anomaly detection on a partially labeled graph G as follows: For a given graph $G = \{V, E, X, A, Y^L\}$, where Y^L is the set of partial labels on nodes and each $y_i \in Y^L$is a binary value taking the value 1 if the corresponding node v_i is abnormal and 0 otherwise. The objective of the anomaly detection model is to learn a predictive function defined as:

$$f_\theta := E_{x_i \sim G} logp(y_i | x_i, y_i \in Y^L), y_i \in \{0, 1\}, \tag{4.8}$$

We obtain the final node representation using a GNN encoder f_θ and employ a classifier (such as an MLP) to distinguish between normal and abnormal nodes, where θ represents the parameters to be learned.

4.2.2 Limitations of Previous Attacks

Anomaly detection, essential for identifying deviations from expected patterns within datasets (S. Zhou, X. Huang, N. Liu, H. Zhou, et al. 2023; S. Zhou, X. Huang, N. Liu, Tan, et al. 2022; S. Zhou, Tan, et al. 2021) plays a crucial role in areas such as

credit card fraud detection, spam filtering, and hacker intrusion detection (Akoglu et al. 2014; X. Ma et al. 2023). This task is challenging due to data sparsity and the implicit features that characterize anomalies (Zheng et al. 2023). For instance, on e-commerce platforms, while most users genuinely review products, a few malicious users may manipulate ratings for personal gain. These malicious users blend in with the majority, making it difficult for classifiers to distinguish them from ordinary users due to the subtlety of their features.

Recently, advancements in graph neural networks (GNNs) have led to the development of GNN-based anomaly detection methodologies. These techniques construct a graph connecting various objects and use GNNs' capabilities to differentiate anomalies from normal instances. GNN-based detectors benefit from end-to-end and semi-supervised training, reducing the need for extensive feature engineering and costly data annotation (Jianheng Tang et al. 2022). Unlike conventional models, GNN-based detectors generate high-quality node embeddings by iteratively aggregating information from neighboring nodes, capturing the essential characteristics of anomalous data.

Despite their success, existing GNN-based anomaly detectors are highly susceptible to disruptions from attacks on the graph structure and node features, which can significantly affect their performance. Figure 4.2 illustrates an attacked financial transaction network, highlighting this vulnerability in GNN-based fraud detection systems.

Currently, some research investigates adversarial attacks similar to anomaly detection. Zhou et al. (X. Zhou et al. 2022) propose a hierarchical poisoning attack (HAA) generation method against graph neural network (GNN)-based intrusion detection in IoT systems with a limited budget. This is a black-box attack that modifies node features using a hierarchical node selection algorithm based on random walk with restart (RWR) to select a set of more vulnerable nodes with high attack priority. Venturi et al. (Venturi et al. 2024) propose an evasion attack against intru-

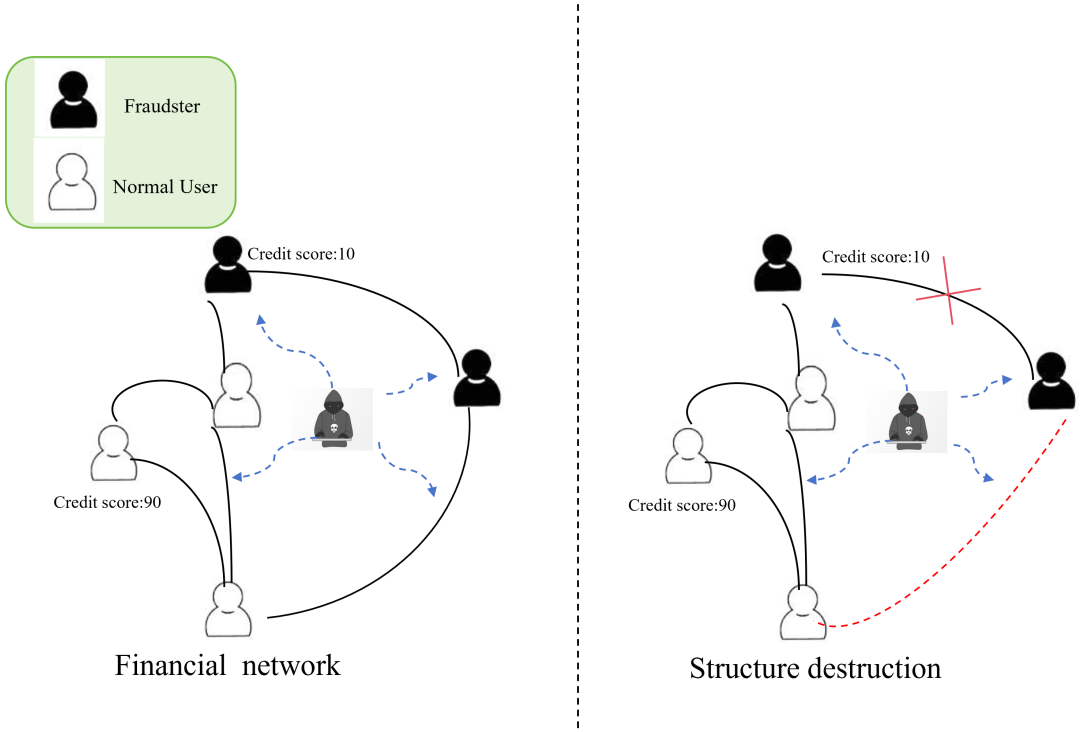


Figure 4.2: The illustration presents a financial transaction network under attack. In this network, normal users are represented by white nodes, and fraudsters are represented by black nodes. The attacker destroys the graph structure by deleting edge (cross) and adding edge (dotted line) to reduce the fraudsters detectability. For example, the fraudster camouflages himself by connecting to multiple normal users.

sion detection systems by perturbing the feature values of their malicious netflows (feature attacks) or by altering the test graph itself (structural attacks).

Although there are some studies about adversarial attacks for anomaly detection, there are not enough. According to our knowledge, there are no studies investigating the vulnerability of GNN-based anomaly detectors in a gray-box scenario and realizing poisoning attacks through structural attacks. To address this gap, in this section, we first show the application of Nettack (Zügner et al. 2018) for the anomaly detection task. Then we demonstrate the inefficient budget allocation in Nettack for anomaly detection. To improve the performance of Nettack, we propose MetaAD, which combines our cost-aware framework with the Nettack approach against GNN-based anomaly detection. Our research makes the following contributions:

- To address the current gap in poisoning attacks against anomaly detection, we

propose MetaAD, a budget-efficient targeted attack framework that leverages the CA framework to enhance the effectiveness of targeted attacks.

- Through rigorous empirical assessments on benchmark datasets, we demonstrate that MetaAD improves upon existing methods, highlighting its potential as a plug-and-play solution for various graph poisoning attacks against anomaly detection.

4.2.3 Cost Aware Framework for anomaly detection

Our methodology comprises two distinct components. The first component is an adaptation of Nettack for anomaly detection, specialized in structure attacks by modifying the edges according to the gradient saliency map to protect abnormal nodes from detection. The second component, our designed MetaAD, modifies the attack objective based on the Nettack algorithm. MetaAD assigns different weights to abnormal nodes according to their margins to prioritize nodes differently and incorporates the weight function into the objective function of Nettack, thereby allocating budgets more reasonably and enhancing the performance of Nettack.

4.2.3.1 Attack Model against Anomaly Detection

Under the Nettack setting, our goal is to attack abnormal nodes V_a i.e., we aim to change V_a's detection status, from abnormal to normal. To ensure that the attacker cannot modify the graph completely, we further limit the number of allowed changes by a budget Δ:

$$\sum |A - A'| \leq \Delta, \tag{4.9}$$

where A is the clean adjacent matrix and A' is the poisoned one.

Given this basic setup, our problem is defined as:

Given a graph $G^{(0)} = (A^{(0)}, X^{(0)})$ and a set of target nodes V_a. Let c_{old} denote the anomaly status label "1" for v_0 based on the graph $G^{(0)}$ (predicted or using some

ground truth). Determine:

$$\begin{aligned} &\text{argmax}_{(A',X)\in P_{G^{(0)},\Delta,A}} \max_{c\neq c_{\text{old}}} \left(\ln Z^*_{v_0,c} - \ln Z^*_{v_0,c_{\text{old}}}\right) \\ &\text{subject to} \quad Z^* = f_{\theta^*}(A',X) \quad \text{with} \quad \theta^* = \arg\min_{\theta} L(\theta; A', X), \end{aligned} \tag{4.10}$$

The objective is to find a perturbed graph G' that misclassifies the target node v_0 as a new class c_{new}. Specifically, the inner maximization aims to find the class c (different from c_{old} that maximizes the difference in the log-probabilities). Essentially, it seeks the class that v_0 is most likely to be misclassified as, compared to its original class c_{old}. The outer maximization seeks the optimal perturbations A' within the allowed perturbation budget $P_{G0,\Delta,A}$ to achieve the objective of the inner maximization. It finds the best possible graph and feature perturbations to maximize the misclassification likelihood. Since we assume that the attacker does not have access to the target classifiers parameters, outputs, or even knowledge about its architecture in the gray-box setting, the attacker uses a surrogate model to perform the poisoning attacks. We use Graph Isomorphism Network (GIN)(Xu et al. 2019) as the surrogate model. The objective remains to degrade the performance of the anomaly detection system by introducing strategic perturbations.

4.2.3.2 Our MetaAD

Based on the discussion in Section 4.2.3.1, MetaAD combines our cost-aware framework in the attack objective as follows:

$$\begin{aligned} &L_{\text{atk}} = \text{argmax}_{(A',X)\in P_{G^{(0)},\Delta,A}} \sum_{v\in V_a} w(v) \cdot \max_{c\neq c_{\text{old}}} \left(\ln Z^*_{v,c} - \ln Z^*_{v,c_{\text{old}}}\right) \\ &\text{subject to} \quad Z^* = f_{\theta^*}(A',X) \quad \text{with} \quad \theta^* = \arg\min_{\theta} L(\theta; A', X), \end{aligned} \tag{4.11}$$

where $w(v)$ is the weight of node v. To introduce the weight $w(v)$, which captures the inverse relationship between the node's margin $\varphi(v)$ and the weight, we utilize

the exponential function. The weight $w(v)$ is defined as follows:

$$w(v) = \alpha \times e^{-\beta \times \varphi(v)^2}, \tag{4.12}$$

where α and β are adjustable hyperparameters that control the influence of the margin on the weight. This framework ensures that the attack is cost-effective by focusing on perturbations that are likely to have the greatest impact on the anomaly detection system. By weighting the nodes based on their margins, MetaAD prioritizes changes that are more likely to cause misclassification, thereby maximizing the efficiency of the attack within the given budget constraints.

4.2.3.3 MetaAD Algorithms

In this section, we present the MetaAD algorithm to show the attack process on anomaly detection tasks. The goal of the MetaAD algorithm is to generate a poisoned graph that degrades the performance of the anomaly detection system by strategically perturbing the graph structure. The process begins by initializing the poisoned graph $\hat{G}$ as the original graph G. Next, the classification margin $\varphi(v)$ is computed along with the corresponding weight $w(v)$, which captures the inverse relationship between the node's margin and the perturbation impact. The algorithm then enters a perturbation loop where, as long as the perturbation budget $|\sum(A - A')|$ remains within the allowed limit Δ, it computes the gradients for each edge in the adjacency matrix A, selects the edge with the maximum gradient magnitude, perturbs the graph $\hat{G}$ by adding or removing the selected edge. Finally, the process concludes by returning the final poisoned graph $\hat{G}$. This comprehensive explanation and pseudocode in algorithm 3 provide a clear understanding of the MetaAD algorithm and its steps.

Algorithm 3 MetaAD: Cost-Aware Poisoning Attack for Anomaly Detection

Input: Graph $G = (A, X)$, modification budget Δ, Target nodes V_a, number of training iterations T, training data labels CL
Output: Modified graph $\hat{G} = (A', X)$
$A' \leftarrow A$
while $|\sum(A - A')| \leq \Delta$ do
    Randomly initialize θ_0 of surrogate model
    for $t = 0$ to $T - 1$ do
        $\theta_{t+1} \leftarrow \text{step}(\theta_t, \nabla_{\theta_t} L_{\text{train}}(f_{\theta_t}(A', X), CL))$
    end for
    for each node $v \in V_a$ do
        Calculate $\varphi(v)$ for each node v to form the margin matrix $\Phi(V)$ (Eq. 4.2)
    end for
    $\nabla A' \leftarrow \nabla_{A'} \mathcal{L}_{atk}(f_{\theta_T}(A', X), V_a, \Phi(V))$
    $A' \leftarrow$ modify by adding or deleting the edge with the most significant gradient influence
end while
$\hat{G} \leftarrow (A', X)$
return poisoned graph $\hat{G}$

Chapter 5

Evaluation

In this section, we provide the evaluation of our proposed cost-aware poisoning attack framework on node classification, link prediction, and anomaly detection tasks.

5.1 Evaluation of CA-Attack

In this section, We present the experimental results by answering the following questions:

- $RQ1$: Can our CA attack loss framework enhance the efficacy of current poisoning attacks?
- $RQ2$: Can our CA attack loss framework serve as a universal framework for conventional poisoning attack losses?
- $RQ3$: Is our framework more computationally efficient compared to baselines?
- $RQ4$: How does our framework perform in terms of transfer learning?

5.1.1 Set Up

Datasets and GNN models. We evaluate our attacks on three benchmark graph datasets. They are Cora (McCallum et al. 2000), Citeseer (Sen et al. 2008) and

Table 5.1: Dataset statistics. Following (Jin, Y. Ma, et al. 2020), we only consider the largest connected component (LCC).

Dataset	Nodes	Edges	Classes	Features
Cora	2,485	5,069	7	1,433
Citeseer	2,110	3,668	6	3,703
Polblogs	1,222	16,714	2	/

Table 5.2: Hyperparameters of CA combined with CE loss.

Dataset	α_1	α_2	β_1	β_2
Cora	4.5	1.0	1.0	1.0
Citeseer	4.5	1.0	1.0	1.0
Polblogs	1.0	1.0	0.5	0.1

Table 5.3: Hyperparameters of CA combined with CW loss.

Dataset	α_1	α_2	β_1	β_2
Cora	4.5	0.0	1.0	1.0
Citeseer	4.5	0.0	1.0	1.0
Polblogs	1.0	0.0	0.5	0.1

Polblogs (Adamic 2005). Table 5.1 shows the basic statistics of these datasets. The datasets are partitioned into 10% labeled nodes and 90% unlabeled nodes. The true labels of the unlabeled nodes are hidden from both the attacker and the surrogate, serving only as a benchmark for assessing the effectiveness of the adversarial attacks. We employ the Graph Convolutional Network (GCN) as our target GNN model.

Baseline Attacks. We choose DICE (Waniek et al. 2018)[1], MetaSelf (Daniel et al. 2019)[2] and Certify robustness(CR) inspired attack framework (B. Wang, Pang, et al. 2023) as the base attack methods. Specifically, DICE (Delete Internally, Connect Externally) is an innovative approach designed to manipulate social networks by selectively removing internal edges within a community while adding external connections. This method aims to disrupt the typical structure of a community, making it less detectable by common network analysis tools, thus preserving the privacy of its members without significantly impacting their influence or connectivity within the larger network. MetaSelf employs a sophisticated adversarial framework that strategically alters the topology of the graph through the insertion or deletion of

[1] https://github.com/DSE-MSU/DeepRobust
[2] https://github.com/DSE-MSU/DeepRobust

Table 5.4: Experimental results of attacks for GCNs.

Dataset	Cora		Citeseer		Polblogs	
Pert Rate(%)	5%	10%	5%	10%	5%	10%
Clean	84.61±0.28		73.95±0.28		95.34±0.19	
DICE	82.79±0.17	81.92±0.22	72.32±0.24	71.59±0.28	86.54±0.42	80.71±0.29
CE-MetaSelf	76.56±0.27	66.96±0.55	71.49±0.39	66.20±0.30	77.22±0.34	69.73±0.44
CR-CE-MetaSelf	75.79±0.34	68.50±0.29	65.28±0.43	56.93±0.33	78.68±0.32	70.43±0.85
CA-CE-MetaSelf	66.98±0.56	56.30±0.42	62.58±0.28	53.22±0.30	76.58 ±0.46	69.43±0.22
CW-MetaSelf	72.75±0.50	61.85±1.18	64.37±1.07	52.17±0.25	83.71±0.66	79.45±0.47
CR-CW-MetaSelf	71.40±0.61	60.84±0.45	64.76±0.88	54.98±1.29	83.09±0.92	79.87±1.02
CA-CW-MetaSelf	69.69±0.64	58.50±0.73	60.79±0.51	53.28±0.62	83.24±0.45	76.99±0.46

Table 5.5: Comparison of computational efficiency.

Methods	Cora	Citeseer	Polblogs
DICE	0.04s	0.01s	0.06s
CE-MetaSelf	107.46s	97.90s	114.07s
CW-MetaSelf	158.39s	129.12s	236.52s
CR-CE-MetaSelf	84533.67s	53089.36s	42780.71s
CR-CW-MetaSelf	87566.54s	56345.78s	53390.56s
CA-CW-MetaSelf	153.58s	122.74s	200.68s
CA-CE-MetaSelf	104.39s	107.88s	112.37s

edges. By leveraging meta-learning techniques, this method iteratively adjusts the graph structure to maximize the misclassification rate of the target graph neural network (GNN) model. The Certify Robustness (CR) inspired attack framework adopts certified robustness principles to craft more potent attacks against GNN vulnerabilities. It commences by pinpointing nodes' certified perturbation thresholds through randomized smoothing, earmarking those with minimal thresholds as particularly susceptible due to their diminished graph perturbation resilience. This pivotal understanding begets the creation of a loss function inspired by certified robustness which bolsters existing attack strategies' impact. For clarity in presentation, we denote the variant of MetaSelf utilizing the cross entropy loss as CE-MetaSelf. When employing the CW loss (Carlini et al. 2017), it is referred to as CW-MetaSelf. When integrating the CR-inspired attack framework with MetaSelf using the cross entropy loss, we term it CR-CE-MetaSelf. Similarly, its adaptation with the CW loss is named CR-CW-MetaSelf.

Table 5.6: Experimental results of transfer learning on GAT and GraphSage with 5% perturbation rate.

Dataset	Cora		Citeseer		Polblogs	
Victim Model	GAT	GraphSage	GAT	GraphSage	GAT	GraphSage
Clean	83.54±1.41	83.21±1.54	74.11±1.69	73.28±1.02	94.19±1.77	93.73±1.98
DICE	82.62±2.0	81.51±1.33	73.75±2.25	72.61±1.58	87.63±3.01	83.81±2.98
CEMetaSelf	80.24±3.58	77.93±1.53	72.80±2.57	71.31±1.41	87.26±2.26	83.26±3.36
CR-CEMetaSelf	79.62±4.50	76.59±1.33	69.43±3.57	66.41±1.39	88.34±2.24	84.72±2.55
CA-CEMetaSelf	76.65±2.66	72.55±2.88	68.75±3.24	65.23±2.95	86.48±1.77	82.11±3.31
CWMetaSelf	78.06±5.77	74.20±1.82	68.89±3.24	65.40±1.64	89.82±3.10	86.96±3.73
CR-CWMetaSelf	78.44±3.21	74.95±1.54	69.16±3.52	66.00±1.73	90.34±3.52	87.71±3.48
CA-CWMetaSelf	76.38±3.56	71.69±2.17	66.72±4.58	62.14±1.84	90.88±3.35	87.45±2.54

Implementation Details All attacks are implemented in PyTorch and run on a Linux server with 16 core 1.0 GHz CPU, 24 GB RAM, and 10 Nvidia-RTX 3090Ti GPUs. Each experiment is repeated for ten times under different initial random seeds, and the uncertainties are presented by 95% confidence intervals around the mean in our tables. We separately set the perturbation budget Δ as 5% and 10% of the total number of edges in a graph. The specific parameters associated with our cost aware loss were selected by grid search to achieve optimal attack performance and details are in Table 5.2 and Table 5.3. For $\varphi(v) > 0$, we use α_1 and β_1. Conversely, for $\varphi(v) < 0$, we employ α_2 and β_2. For the baseline CR inspired attack framework, we adopted the parameter settings as outlined in their original paper and computed the certify robustness size at every $20th$ iteration during the generation of the poisoned graph.

5.1.2 Results and Analysis

Performance Comparison ($RQ1$) Table 5.4 presents a comparison of our attack performance against other methods. We can observe that our CA attack loss framework combined with negative log-likelihood loss can enhance the CE-MetaSelf performance in all datasets. For instance, when attacking GCN with a 5% perturbation ratio, our CA-CE-MetaSelf demonstrates a gain of 9.58% and 9.74% over CE-MetaSelf on Cora and Citeseer, respectively. Figure 3.3 (b) shows the CA-CE-

MetaSelf method's effectiveness in minimizing budget wastage on attacked nodes, with fewer nodes in the -1 to -0.8 margin compared to the CE-MetaSelf method. These findings indicate that our framework enhances poisoning attack more effectively within the same budget constraints.

Universality Analysis ($RQ2$) Table 5.4 highlights the effectiveness of our weight assignment strategy when applied to the CW loss, underscoring the versatility of our approach. For example, when considering a 5% perturbation ratio, our proposed approach exhibits a relative gain of 3.58% and 3.06% over the CW-MetaSelf method for the Citeseer and Cora datasets, respectively. Additionaly, CW-CE-MetaSelf has a gain of 1.71% and 3.97% over the CR-CW-MetaSelf for the Cora and Citeseer datasets, respectively. Since we use the same parameters with negative log likelihood loss for CW loss to reduce the time for parameter adjustments, it may not be optimal for the CW loss which could result in suboptimal results compared to the baselines. The results of our attack can be improved with more detailed parameter tuning for the CW loss within the CA attack loss framework. Furthermore, the experimental results of CE-MetaSelf and CW-MetaSelf show that the C&W loss enhances the poisoning attack performance. Our CA framework is more suitable for CE loss since it enhances the performance of CE-MetaSelf more compared to CW-MetaSelf. According to the analysis in Section 3.3.3, our framework generates a larger impact on CE loss because the CA framework reduces the attack budget allocated to nodes with negative attack margins. Since the C&W loss already has a better attack budget allocation compared to CE loss, the performance improvement of our CA framework integrated with C&W loss is not as significant as with CE loss.

Computational Efficiency($RQ3$) We compared the computational efficiency of our method with the baselines, shown in Table 5.5. DICE has the highest computa-

Table 5.7: Dataset statistics.Following (Jin, Y. Ma, et al. 2020), we only consider the largest connected component (LCC).

Dataset	Nodes	Edges	Classes	Features
Cora	2,485	5,069	7	1,433
Citeseer	2,110	3,668	6	3,703
CoraML	2,810	7,981	7	2,879

tional efficiency since it depends on randomness without gradient derivation. The computational efficiency of CR-CE-MetaSelf and CR-CW-MetaSelf are the lowest because they need much time to compute certify robustness size for every node. CA-CE-MetaSelf and CA-CW-MetaSelf have similar computational efficiency with CE-MetaSelf and CW-MetaSelf. Therefore, the computational efficiency of CA attack loss framework is satisfactory while the performance is competitive.

Transfer Learning Performance Comparison($RQ4$) Table 5.6 shows the experimental results for the case where the structures of the victim model and the surrogate model are different. Both CA-CE-MetaSelf and CA-CW-MetaSelf have competitive transfer performance compared to other baselines. In evaluations using GATs on Cora, our CA-CE-MetaSelf outperformed CR-CE-MetaSelf by 2.97%. Similarly, with GraphSAGE on Citeseer, CA-CW-MetaSelf surpassed CR-CW-MetaSelf by 3.09%.

5.2 Evaluation of MetaLinkAttack

In this section, We present the experimental results

Table 5.8: Hyperparameters of MetaLinkAttack.

Dataset	α_1	α_2	β_1	β_2
Cora	3.6	1.0	1.0	1.1
Citeseer	3.6	1.0	1.0	1.1
CoraML	3.6	1.0	1.0	1.1

Table 5.9: ROC-AUC value of the attacks against GAE.

Dataset	Cora		Citeseer		CoraML	
Pert Rate(%)	5%	10%	5%	10%	5%	10%
Clean	91.20±0.23		88.32±0.45		89.44±0.25	
IGA	85.88±0.11	79.97±0.2	83.69±0.37	77.20±0.13	83.34±0.28	78.75±0.15
MetaLinkAttack	79.45±0.23	69.30±0.25	74.55±0.23	71.56±0.25	76.58 ±0.35	68.66±0.24

5.2.1 Set Up

Datasets and GNN models. We evaluate our attacks on three benchmark graph datasets.They are Cora (McCallum et al. 2000), Citeseer (Sen et al. 2008) and CoraML (McCallum et al. 2000). Table 5.7 shows the basic statistics of these datasets. The datasets are citation networks where nodes represent publications, and links indicate citations among them. We employ the Graph Autoencoder Model (GAE) (T. Kipf et al. 2016) as our target GNN model. We assume the model is trained on 70% of the nodes and evaluated on the remaining in the graph. To train the target model, we collect all linked node pairs and randomly sample the same number of unlinked node pairs. We split this dataset into 70% for training and 30% for validation.

Metric. ROC-AUC is a key metric for assessing binary classification tasks, including link prediction (X. He et al. 2021). It evaluates a model's skill in differentiating between connected and non-connected node pairs, with higher AUC values indicating better discrimination ability.

Baseline Attacks. We choose IGA as the base attack method. Specifically, IGA employs a sophisticated adversarial framework that strategically alters the topology

of the graph through the insertion or deletion of edges. This method iteratively adjusts the graph structure to maximize the misclassification rate of the target graph neural network (GNN) model.

Implementation Details. All attacks are implemented in PyTorch and run on a Linux server with 16 core 1.0 GHz CPU, 24 GB RAM, and 10 Nvidia-RTX 3090Ti GPUs. Each experiment is repeated for ten times under different initial random seeds, and the uncertainties are presented by 95% confidence intervals around the mean in our tables. We separately set the perturbation budget Δ as 5% and 10% of the total number of edges in a graph. The specific parameters associated with our cost aware loss were selected by grid search to achieve optimal attack performance and details are in Table 5.8. For $\varphi(v) > 0$, we use α_1 and β_1. Conversely, for $\varphi(v) < 0$, we employ α_2 and β_2.

5.2.2 Results and Analysis

Table 5.9 presents a comparison of our attack performance against other methods. We can observe that our MetaLinkAttack combined with negative log-likelihood loss can enhance the IGA performance in all datasets. For instance, when attacking GAE with a 5% perturbation ratio, our MetaLinkAttack demonstrates a gain of 6.43% and 9.14% over IGA on Cora and Citeseer, respectively. These findings indicate that our framework enhances poisoning attack more effectively within the same budget constraints.

5.3 Evaluation of MetaAD

In this section, We evaluate the performance of our proposed MetaAD.

Table 5.10: Dataset statistics.

Dataset	Nodes	Edges	Features	Anomaly%
Cora	2,708	5,429	1,433	5%
Citeseer	3,312	4,732	3,703	5%
DBLP	5,484	8,117	6,775	5%

Table 5.11: Hyperparameters of MetaAD.

Dataset	α_1	α_2	β_1	β_2
Cora	3.8	1.0	1.0	1.0
Citeseer	3.6	1.0	1.0	1.0
DBLP	2.7	1.0	1.0	1.0

5.3.1 Set Up

Datasets and GNN models. We conduct experiments on three datasets introduced in Table 5.10. Cora is a citation network dataset including $2,708$ nodes, with $5,429$ edges indicating citation relationships. Each node represents a scientific publication described by a binary feature vector. DBLP (Yuan et al. 2021) is a citation network dataset composed of $5,484$ scientific publications collected from the DBLP Computer Science Bibliography. The $8,117$ edges represent citation relationships among different papers, and node attributes are extracted from the article titles. Citeseer is a citation network dataset that includes $3,312$ scientific publications. It contains $4,732$ edges indicating citation relationships between publications. Each node in the Citeseer dataset is characterized by a sparse bag-of-words feature vector, representing the words present in the paper.

We use the GIN as our target GNN model. To generate structural outliers according to the method described by (K. Ding et al. 2019), we randomly select m nodes from the input graph and then make those nodes fully connected. All m nodes in the clique are regarded as outliers. This process is iteratively repeated until n cliques are generated, resulting in a total of $m \times n$ structural outliers. In our experiments, the size of m is set to 15. Since we control the structural anomaly

rate to be 5% in each dataset, The number n is fine-tuned according to different datasets.

We split the dataset into a training set and a test set. The training set contains 50% of the normal nodes and 50% of the anomalous nodes, while the test set includes the remaining normal nodes and anomalous nodes.

Evaluation Metric. We evaluate our attack performance by assessing the number of anomaly nodes that have evaded detection and by determining the amount of budget resources utilized for the attack. Therefore, we compute the recall score within the top-k predictions (Rec@K) at different attack budget levels for the node anomaly detection task. A lower Rec@K indicates better poisoning attack performance. Recall measures how many of the actual anomaly data points are correctly identified. The formula is as follows:

$$\text{Rec@K} = \frac{\text{Number of actual anomalies in top-k predictions}}{\text{Total number of actual anomalies in the test set}} \tag{5.1}$$

Baseline Attacks. We adopt Nettack (Daniel et al. 2019)[3] as our baseline attack method. Nettack, originally developed for node classification tasks via graph convolutional networks, is a targeted attack aiming to misclassify specific nodes by manipulating their features and the graph structure. By exploiting the relational nature of graph data, Nettack introduces direct and influencer attacks. These attacks are designed to be imperceptible by preserving the graphs degree distribution and feature co-occurrences, ensuring that changes remain unnoticed even in a discrete, relational domain. Our implementation of Nettack leverages this approach to efficiently perturb the graph, maintaining its overall structural properties while achieving effective adversarial manipulation.

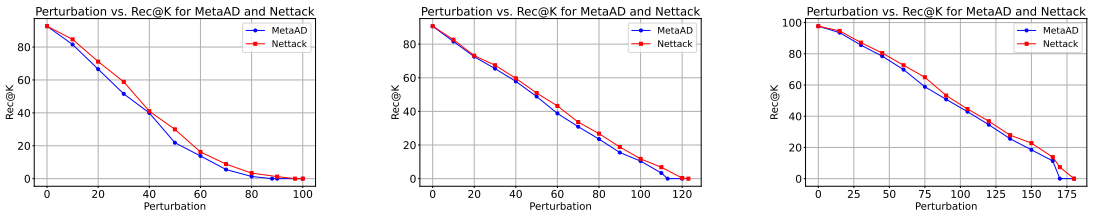
In this study, we employ the Nettack algorithm for anomaly detection tasks. Specifically, we target the structure of the graphs to prevent the detection of anoma-

[3] https://github.com/DSE-MSU/DeepRobust

lous nodes by detectors. This adaptation of Nettack to anomaly detection involves strategically altering the graph to hide anomalies from being correctly identified by the detection algorithms.

Implementation Details. All attacks are implemented in PyTorch and run on a Linux server with 16 core 1.0 GHz CPU, 24 GB RAM, and 10 Nvidia-RTX 3090Ti GPUs. Each experiment is repeated for ten times under different initial random seeds. The specific parameters associated with our cost aware framework were selected by grid search to achieve optimal attack performance and details are in Table 5.11. For $\varphi(v) > 0$, we use α_1 and β_1. Conversely, for $\varphi(v) < 0$, we employ α_2 and β_2.

5.3.2 Results and Analysis



(a) Cora (b) Citeseer (c) DBLP

Figure 5.1: Comparison of attack performance between Nettack and MetaAD on targeted GIN detector

Figure 5.1 presents a comparison of the performance of our MetaAD method and Nettack on the anomaly detection task. We observe that MetaAD enhances the performance of Nettack across all datasets. For instance, when attacking GIN under the same budget, the Rec@K scores of MetaAD on three datasets are lower than those of Nettack, indicating that MetaAD performs better. Additionally, to cause all anomaly nodes to evade detection (Rec@K score equals 0), the budgets used by MetaAD are lower than those required by Nettack. For example, on the Cora dataset, MetaAD uses 88 budgets to achieve an Rec@K score of 0, while Nettack

uses 97. On the Citeseer dataset, MetaAD uses 111 budgets compared to Nettack's 120. On the DBLP dataset, MetaAD uses 170 budgets, whereas Nettack requires 178. These results indicate that our cost-aware framework significantly reduces the budget compared to Nettack and effectively develops a poisoning attack on anomaly detection tasks.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we delve into the realm of graph poisoning attacks on graph neural networks (GNNs), based on the finding that current attack methods have a budget waste problem which impairs the attack performance for identifying vulnerabilities of graph neural networks. We study this problem across different graph neural network applications.

First, we address the budget waste issue in node classification tasks. We present an innovative Cost Aware Poisoning Attack Loss Framework (CA-attack) utilizing the concept of classification margin. Our research begins by examining the budget inefficiencies present in previous attack methods. Subsequently, we develop a cost-aware loss framework that assigns dynamic weights to victim nodes based on their margins. Through evaluations conducted on various datasets, we demonstrate that our attack loss can considerably decrease the budget waste and improve the performance of existing attacks.

Continuing our study, we investigate the budget waste problem in link prediction and anomaly detection tasks. First, we propose a cost-aware link prediction poisoning attack called MetaLinkAttack, which also utilizes the concept of classification

margin. Similarly, we identify the budget inefficiencies in previous attack methods and develop a cost-aware poisoning attack that assigns dynamic weights to edges based on their margins. Through evaluations conducted on various datasets, we demonstrate that our attack method can considerably decrease budget waste and enhance the performance of existing attacks.

Second, we introduce MetaAD, integrated with our cost-aware poisoning attack framework, which aims to attack graph neural network-based anomaly detection. Experiments show that our MetaAD uses fewer budgets to ensure all anomaly nodes evade anomaly detection compared to the baseline method. This indicates that MetaAD utilizes attack budgets more efficiently and exhibits superior attack performance.

Our research has identified the issue of budget waste and developed a new attack framework to optimize budget allocation. Evaluations demonstrate that our cost-aware framework can be effectively combined with various attack objectives, enhancing attack performance across key graph neural network applications, including node classification, link prediction, and anomaly detection. Our investigation offers a significant improvement for current poisoning attacks and can help uncover more vulnerabilities in graph neural networks across their applications.

6.2 Limitations

Despite the promising results, our study has certain limitations. Firstly, our cost-aware framework is specifically adapted to gradient-based poisoning attacks. If the attacks are not based on gradients, for example, some black-box poisoning attacks that rely on reinforcement learning or non-gradient strategies to modify the input data, our framework may not be effective.

Secondly, although our framework can enhance the performance of existing poisoning attacks by integrating with attack objectives and using budgets more effi-

ciently, the budget waste problem still persists. This is because our framework cannot address all the budget waste in a dynamic attack process, especially in graph datasets where nodes have connections and can be influenced by neighboring nodes,

6.3 Future Work

In the future, we plan to apply our cost-aware poisoning attack framework to poisoning node features and building feature attacks for different applications. Currently, we have only investigated structural attacks by adding or deleting edges in the graph. We will also explore attacking both structure and features simultaneously under a limited budget to allocate resources efficiently between feature and structure modifications.

Additionally, for anomaly detection tasks specifically, we aim to extend our framework to various anomaly score functions to attack different kinds of anomalies and uncover their vulnerabilities.

Furthermore, with the development of large language models (LLMs), we will investigate whether LLMs can help solve the current budget waste problem, ensuring that every budget allocation effectively contributes to the degradation of graph neural network performance.

References

Adamic, Lada (2005). "The Political Blogosphere and the 2004 U.S. Election: Divided They Blog". In: Proceedings of the 3rd International Workshop on Link Discovery.

Akoglu, Leman et al. (2014). "Graph based anomaly detection and description: a survey". In: Data Mining and Knowledge Discovery 29, pp. 626–688.

Bhardwaj, Peru et al. (2021). "Adversarial Attacks on Knowledge Graph Embeddings via Instance Attribution Methods". In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 8225–8239.

Bruna, Joan et al. (2014). "Spectral networks and locally connected networks on graphs". In: International Conference on Learning Representations (ICLR2014), CBLS, April 2014.

Carlini, Nicholas et al. (2017). "Towards evaluating the robustness of neural networks". In: ieee symposium on security and privacy (sp).

Chaudhary, Anshika et al. (2019). "Anomaly Detection using Graph Neural Networks". In: International Conference on Machine Learning, Big Data, Cloud and Parallel Computing.

Chen, Deli et al. (Apr. 2020). "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View". In: Proceedings of the AAAI Conference on Artificial Intelligence 34.04, pp. 3438–3445.

Chen, Hongge et al. (Aug. 2019). "Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective". In: pp. 3961–3967.

Chen, Jinyin et al. (2020). “Link Prediction Adversarial Attack Via Iterative Gradient Attack”. In: IEEE Transactions on Computational Social Systems 7.4, pp. 1081–1094.

Choi, Edward et al. (2017). “GRAM: Graph-based Attention Model for Healthcare Representation Learning”. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 787–795.

Chowdhury, Anjan et al. (2023). “Improving Node Classification Accuracy of GNN through Input and Output Intervention”. In: ACM Trans. Knowl. Discov. Data 18.1.

Dai, Hanjun et al. (2018). “Adversarial Attack on Graph Structured Data”. In: Proceedings of the 35th International Conference on Machine Learning.

Daniel, Zügner et al. (2019). “Adversarial Attacks on Graph Neural Networks via Meta Learning”. In: International Conference on Learning Representations.

Ding, Kaize et al. (2019). “Deep Anomaly Detection on Attributed Networks”. In: Proceedings of the 2019 SIAM International Conference on Data Mining (SDM), pp. 594–602.

Ding, Ruyi et al. (2023). “VertexSerum: Poisoning Graph Neural Networks for Link Inference”. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 4509–4518.

Duan, Dongsheng et al. (2020). “AANE: Anomaly Aware Network Embedding For Anomalous Link Detection”. In: 2020 IEEE International Conference on Data Mining (ICDM), pp. 1002–1007.

Duan, Jingcan et al. (2023). “Graph Anomaly Detection via Multi-Scale Contrastive Learning Networks with Augmented View”. In: Proceedings of the AAAI Conference on Artificial Intelligence.

Duan, Liang et al. (2024). “Structural Entropy Based Graph Structure Learning for Node Classification”. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 38. 8, pp. 8372–8379.

Duvenaud, David et al. (Sept. 2015). “Convolutional Networks on Graphs for Learning Molecular Fingerprints”. In: Advances in Neural Information Processing Systems (NIPS) 13.

Fan, Haoyi et al. (2020). “Anomalydae: Dual Autoencoder for Anomaly Detection on Attributed Networks”. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5685–5689.

Fan, Wenqi, Yao Ma, Qing Li, Yuan He, Eric Zhao, et al. (2019). “Graph Neural Networks for Social Recommendation”. In: The World Wide Web Conference.

Fan, Wenqi, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, et al. (2019). “Graph Neural Networks for Social Recommendation”. In: The World Wide Web Conference.

Gori, Marco et al. (2005). “A new model for learning in graph domains”. In: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005., 729–734 vol. 2.

Grover, Aditya et al. (2016). “node2vec: Scalable feature learning for networks”. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855–864.

Grubbs, Frank E. (1969). “Procedures for Detecting Outlying Observations in Samples”. In: Technometrics 11.1, pp. 1–21.

Hamilton, William L. et al. (2017). “Inductive representation learning on large graphs”. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1025–1035.

Han, Yuwei et al. (2024). “Cost Aware Untargeted Poisoning Attack Against Graph Neural Networks”. In: IEEE International Conference on Acoustics, Speech, and Signal Processing.

Hang, Mengyue et al. (18–24 Jul 2021). “A Collective Learning Framework to Boost GNN Expressiveness for Node Classification”. In: Proceedings of the 38th In-

ternational Conference on Machine Learning. Vol. 139. Proceedings of Machine Learning Research, pp. 4040–4050.

He, Xinlei et al. (2021). "Stealing Links from Graph Neural Networks". In: 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, pp. 2669–2686.

Huang, Tianjin et al. (2023). "Hop-Count Based Self-supervised Anomaly Detection onăAttributed Networks". In: Machine Learning and Knowledge Discovery in Databases. Springer International Publishing, pp. 225–241.

Jin, Wei, Yaxin Li, et al. (2020). "Adversarial Attacks and Defenses on Graphs". In: ACM SIGKDD Explorations Newsletter 22, pp. 19–34.

Jin, Wei, Yao Ma, et al. (2020). "Graph Structure Learning for Robust Graph Neural Networks". In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.

Jorge, Nocedal et al. (2006). In: Numerical Optimization.

Ju, Mingxuan et al. (2023). "Let graph be the go board: gradient-free node injection attack for graph neural networks via reinforcement learning". In: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence. AAAI Press.

Kipf, Thomas et al. (2016). "Variational Graph Auto-Encoders". In: ArXiv.

Kipf, Thomas N. et al. (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: International Conference on Learning Representations.

Kumagai, Atsutoshi et al. (2021). "Semi-supervised Anomaly Detection on Attributed Graphs". In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–8.

Lai, Yuni et al. (2023). "Dual-Space Attacks against Random-Walk-based Anomaly Detection". In: arXiv preprint arXiv:2307.14387.

Li, Guohao et al. (2021). “Training Graph Neural Networks with 1000 Layers”. In: Proceedings of Machine Learning Research 139, pp. 6437–6449.

Li, Jia et al. (2020). “Adversarial Attack on Community Detection by Hiding Individuals”. In: WWW ’20. Taipei, Taiwan, pp. 917–927.

Li, Qing et al. (2023). “PAGCL: An unsupervised graph poisoned attack for graph contrastive learning model”. In: Future Generation Computer Systems 149, pp. 240–249.

Liu, Yixin, Kaize Ding, et al. (2023). “Learning Strong Graph Neural Networks with Weak Information”. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. KDD ’23, pp. 1559–1571.

Liu, Yixin, Zhao Li, et al. (Apr. 2021). “Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning”. In: IEEE Transactions on Neural Networks and Learning Systems PP, pp. 1–15.

Liu, Zihan, Yun Luo, Lirong Wu, Siyuan Li, et al. (2022). “Are Gradients on Graph Structure Reliable in Gray-box Attacks?” In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. New York, NY, USA: Association for Computing Machinery, pp. 1360–1368.

Liu, Zihan, Yun Luo, Lirong Wu, Zicheng Liu, et al. (2022). “Towards Reasonable Budget Allocation in Untargeted Graph Structure Attacks via Gradient Debias”. In: Advances in Neural Information Processing Systems.

Ma, Xiaoxiao et al. (2023). “A Comprehensive Survey on Graph Anomaly Detection With Deep Learning”. In: IEEE Transactions on Knowledge and Data Engineering 35.12, pp. 12012–12038.

McCallum, Andrew et al. (2000). “Automating the Construction of Internet Portals with Machine Learning”. In: Information Retrieval.

Milani Fard, Amin et al. (2013). “Neighborhood randomization for link privacy in social network analysis”. In: World Wide Web.

Pei, Yulong et al. (2021). “ResGCN: Attention-based Deep Residual Modeling for Anomaly Detection on Attributed Networks”. In: 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–2.

Perozzi, Bryan et al. (2014). “DeepWalk: online learning of social representations”. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD ’14. New York, New York, USA, pp. 701–710.

Scarselli, Franco et al. (2009). “The Graph Neural Network Model”. In: IEEE Transactions on Neural Networks 20, pp. 61–80.

Sen, Prithviraj et al. (2008). “Collective Classification in Network Data”. In: AI Magazine 29.3.

Sperduti, A. et al. (1997). “Supervised neural networks for the classification of structures”. In: IEEE Transactions on Neural Networks 8.3, pp. 714–735.

Sun, Jianyong et al. (2022). “Graph Neural Network Encoding for Community Detection in Attribute Networks”. In: IEEE Transactions on Cybernetics 52.8, pp. 7791–7804.

Sun, Mingjie et al. (2018). “Data Poisoning Attack against Unsupervised Node Embedding Methods”. In: ArXiv.

Sun, Yiwei et al. (2020). “Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach”. In: Proceedings of The Web Conference 2020. WWW ’20, pp. 673–683.

Tang, Jiabin et al. (2023). “Explainable Spatio-Temporal Graph Neural Networks”. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. CIKM ’23, pp. 2432–2441.

Tang, Jianheng et al. (2022). “Rethinking Graph Neural Networks for Anomaly Detection”. In: Proceedings of the 39th International Conference on Machine Learning.

Tian, Zhiyi et al. (2022). “A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning”. In: ACM Comput. Surv. 55.8.

Vel-ickovic, Petar et al. (2017). “Graph Attention Networks”. In: Proceedings of the 5th International Conference on Learning Representations.

Velikovi, Petar et al. (2018). “Graph Attention Networks”. In: Proceedings of the 5th International Conference on Learning Representations.

Venturi, Andrea et al. (2024). “Problem space structural adversarial attacks for Network Intrusion Detection Systems based on Graph Neural Networks”. In: arXiv preprint arXiv:2403.11830.

Wang, Binghui and Neil Zhenqiang Gong (2019). “Attacking Graph-based Classification via Manipulating the Graph Structure”. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. CCS ’19. London, United Kingdom, pp. 2023–2040.

Wang, Binghui, Meng Pang, et al. (June 2023). “Turning Strengths Into Weaknesses: A Certified Robustness Inspired Attack Framework Against Graph Neural Networks”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 16394–16403.

Wang, Daixin et al. (2019). “A Semi-Supervised Graph Attentive Network for Financial Fraud Detection”. In: 2019 IEEE International Conference on Data Mining (ICDM), pp. 598–607.

Waniek, Marcin et al. (2018). “Hiding Individuals and Communities in a Social Network”. In: Nature Human Behaviour.

Wu, Huijun et al. (2019). “Adversarial Examples for Graph Data: Deep Insights into Attack and Defense”. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. Ed. by Sarit Kraus, pp. 4816–4823.

Wu, Lingfei et al. (2023). “Graph Neural Networks for Natural Language Processing: A Survey”. In: Foundations and Trends in Machine Learning 16.

Xu, Keyulu et al. (2019). “How Powerful are Graph Neural Networks?” In: International Conference on Learning Representations.

Yang, Zhilin et al. (20–22 Jun 2016). “Revisiting Semi-Supervised Learning with Graph Embeddings”. In: Proceedings of The 33rd International Conference on Machine Learning. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 40–48.

Yao, Liang et al. (2019). “Graph convolutional networks for text classification”. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence. ISBN: 978-1-57735-809-1.

You, Ziquan et al. (2020). “GATAE: Graph Attention-based Anomaly Detection on Attributed Networks”. In: 2020 IEEE/CIC International Conference on Communications in China (ICCC), pp. 389–394.

Yu, Shanqing et al. (Sept. 2018). Target Defense Against Link-Prediction-Based Attacks via Evolutionary Perturbations.

Yuan, Xu et al. (2021). “Higher-order Structure Based Anomaly Detection on Attributed Networks”. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 2691–2700.

Zhang, Binchi et al. (2024). “Adversarial Attacks on Fairness of Graph Neural Networks”. In: The Twelfth International Conference on Learning Representations.

Zhang, Fengbin et al. (2022). “Deep Dual Support Vector Data description for anomaly detection on attributed networks”. In: International Journal of Intelligent Systems 37.2, pp. 1509–1528.

Zhang, Ge et al. (Mar. 2022). “eFraudCom: An E-commerce Fraud Detection System via Competitive Graph Neural Networks”. In: ACM Transactions on Information Systems 40.3.

Zhao, Kaifa et al. (2021). “Structural Attack against Graph Based Android Malware Detection”. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 3218–3235.

Zheng, Xiangping et al. (2023). “Select The Best: Enhancing Graph Representation with Adaptive Negative Sample Selection”. In: ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5.

Zhou, Jie et al. (2020). “Graph neural networks: A review of methods and applications”. In: AI Open 1, pp. 57–81.

Zhou, Shuang, Xiao Huang, Ninghao Liu, Qiaoyu Tan, et al. (Apr. 2022). “Unseen Anomaly Detection on Networks via Multi-Hpersphere Learning”. In: SIAM International Conference on Data Mining (SDM22), pp. 262–270.

Zhou, Shuang, Xiao Huang, Ninghao Liu, Huachi Zhou, et al. (2023). “Improving Generalizability of Graph Anomaly Detection Models via Data Augmentation”. In: IEEE Transactions on Knowledge and Data Engineering 35.12, pp. 12721–12735.

Zhou, Shuang, Qiaoyu Tan, et al. (2021). “Subtractive Aggregation for Attributed Network Anomaly Detection”. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. CIKM ’21, pp. 3672–3676.

Zhou, Xiaokang et al. (2022). “Hierarchical Adversarial Attacks Against Graph-Neural-Network-Based IoT Network Intrusion Detection System”. In: IEEE Internet of Things Journal 9.12, pp. 9310–9319.

Zhu, Yulin, Yuni Lai, et al. (2022). “Binarizedattack: Structural poisoning attacks to graph-based anomaly detection”. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE).

Zhu, Yulin, Tomasz Michalak, et al. (2022). “Towards Secrecy-Aware Attacks Against Trust Prediction in Signed Graphs”. In: arXiv preprint arXiv:2206.13104.

Zügner, Daniel et al. (2018). “Adversarial Attacks on Neural Networks for Graph Data”. In: ACM Special Interest Group on Knowledge Discovery and Data Mining.