Graph Contrastive Backdoor Attacks

Hangfan Zhang [1] **Jinghui Chen** [1] **Lu Lin** [1] **Jinyuan Jia** [1] **Dinghao Wu** [1]

Abstract

Graph Contrastive Learning (GCL) has attracted considerable interest due to its impressive node representation learning capability. Despite the wide application of GCL techniques, little attention has been paid to the security of GCL. In this paper, we systematically study the vulnerability of GCL in the presence of malicious backdoor adversaries. In particular, we propose *GCBA*, the first backdoor attack for graph contrastive learning. GCBA incorporates three attacks: poisoning, crafting, and natural backdoor, each targeting one stage of the GCL pipeline. We formulate our attacks as optimization problems and solve them with a novel discrete optimization technique to overcome the discrete nature of graph-structured data. By extensively evaluating GCBA on multiple datasets and GCL methods, we show that our attack can achieve high attack success rates while preserving stealthiness. We further consider potential countermeasures to our attack and conclude that existing defenses are insufficient to mitigate GCBA. We show that as a complex paradigm involving data and model republishing, GCL is vulnerable to backdoor attacks, and specifically designed defenses are needed to mitigate the backdoor attacks on GCL.

1. Introduction

Graph-structured data is commonly seen nowadays, such as social media networks, mobile payment networks, and credit networks (Shchur et al., 2018; Hamilton et al., 2017; Zeng et al., 2019). In such networks, identities are connected with each other, and each of them contains rich information about the identity's properties. Extracting useful information from these graph structure data can provide better services to many downstream tasks on graphs such as recommendations (Wu et al., 2020; Monti et al., 2017; Ying et al., 2018), link prediction (Zhang & Chen, 2018; Zhu et al., 2023; Qu et al., 2020), and node classification (Kipf & Welling, 2016; Veličković et al., 2017; Xu et al., 2018). Recently, Graph Neural Networks (GNNs) (Kipf & Welling, 2016; Veličković et al., 2017; Xu et al., 2018) are widely adopted as the feature extractor for graph structure data. However, well-trained GNN models usually require large amounts of labeled data, which are expensive and time-consuming to collect, especially when the graph size is large.

Very recently, graph contrastive learning (GCL) (Velickovic et al., 2019; Hassani & Khasahmadi, 2020; You et al., 2020; Zhang et al., 2021a; Zhu et al., 2020; 2021; Lin et al., 2022b) were proposed to address this challenge. GCL has achieved impressive effectiveness in dealing with a large quantity of graph-structured data in the absence of supervisory signals. Its unsupervised property is especially favorable since collecting and labeling data in a traditional supervised manner is extremely costly. GCL aims to pretrain a GNN encoder mapping each node in graph-structured data to a low-dimensional representation called node embedding. Fine-tuning a downstream classifier based on a GNN encoder significantly reduces the cost of collecting and labeling data.

Prior works revealed that GNNs are vulnerable to so-called backdoor attacks (Xi et al., 2021; Zhang et al., 2021b; Yang et al., 2022). A graph backdoor adversary aims to inject a hidden backdoor into the victim GNN model. The interfered GNN will behave like a genuine model on clean inputs. However, once the input is stamped with an adversary-defined "trigger", the victim GNN will behave maliciously, such as misclassifying the poisoned input into a specific class. This is commonly achieved by poisoning the training set involving modifying the labels. However, existing graph backdoor attacks do not fit GCL settings since they need to access labels to inject the backdoor. Nevertheless, human-fed supervisory labels are not required and thus unavailable in GCL.

Despite the excessiveness of research on backdoor attacks (Gu et al., 2017; Chen et al., 2017; 2021; Li et al., 2022), the vulnerability of GCL against backdoor attacks has yet to be explored, which is highly concerning given the broad application of GCL in security-sensitive domains

[1]Pennsylvania State University. Correspondence to: Dinghao Wu <dinghao@psu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

(Qian et al., 2022; Jia et al., 2021). For instance, graph encoders can help classify credit network users as either high credit or low credit. A user marked as low credit will have restricted privileges compared to high credit ones. Such a distinction protects the credit system and encourages appropriate and legitimate user behavior. However, if a backdoor adversary injects a backdoor into the encoder, the encoder will be misled to produce erroneous user embeddings, causing misclassification of the user's credit status from low to high credit. If the backdoor adversary can arbitrarily flip the credit status, the system will become insecure, which may lead to serious consequences. However, previous backdoor attacks in the graph domain assumed a supervised training scenario, thus cannot directly serve for evaluating the vulnerability of GCL to backdoor attacks.

Our work. To bridge this gap, we present GCBA, the first backdoor attack for graph contrastive learning. GCBA aims to manipulate the behavior of the downstream classifier built on the pre-trained GNN encoder. To achieve this goal, GCBA adversary injects a hidden backdoor into the encoder. The embedded backdoor can be activated by a pre-defined "trigger". Once triggered, the backdoor will force the victim GNN encoder to produce a specified result, such as mapping the input node to a specific embedding. In particular, GCBA adversary guides the victim GNN encoder to learn a connection between the pre-defined trigger and an adversary-chosen embedding pattern referred to as target embedding. In this paper, we adopt a single injected node attached to the victim node as the trigger. We expect the victim GNN encoder to map any trigger-attached node to the target embedding.

To achieve the attackers' goals, we formulate GCBA as a bilevel optimization problem to search for the optimal model parameter and trigger node attributes. We define our attacking loss as the difference between the embedding of trigger-attached nodes and the target embedding. By minimizing the attacking loss, we can force the victim GNN encoder to map trigger-attached nodes to the target embedding and the corresponding trigger attributes. Due to the discrete nature of node attributes in graph-structured data, conventional continuous optimization does not directly apply to our attack. To address this challenge, we design a novel discrete optimization method based on convex relaxation. By adopting a latent variable approximating the discrete solution space, we can apply gradient-based methods to solve the optimization problem.

We evaluate GCBA on different datasets collected from the real world. Empirical experiments show that GNN encoders trained by GCL are highly vulnerable to GCBA attacks under different scenarios. Regardless of the limitations on the adversary's capabilities, GCBA can achieve considerable effectiveness. Moreover, our attack can inject a backdoor into any pre-trained GNN encoders while keeping the backdoor stealthy to downstream users. Our experiments also demonstrate that existing defenses in the graph domain may not be able to mitigate GCBA sufficiently.

Contributions. To the best of our knowledge, this paper proposes the first backdoor attacks on pre-trained GNN encoders under graph contrastive learning frameworks. Our contributions can be summarized as follows.

- We propose the detailed design of GCBA, the first backdoor attacks against graph contrastive learning. The proposed attack aims to compromise GCL under various circumstances.

- We systematically evaluate the backdoor robustness of current GCL frameworks under the attack by GCBA in different scenarios and show the effectiveness of the proposed attack.

- We demonstrate that existing defenses in the graph domain may not be able to mitigate GCBA, entailing an urgent demand for specifically designed defenses.

2. Preliminary

2.1. Graph Neural Networks

A Graph Neural Network (GNN) (Kipf & Welling, 2016; Xu et al., 2018; Veličković et al., 2017; Qu et al., 2021; Zhu & Wang, 2022) f receives a graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$ as the input and outputs $\boldsymbol{\mathcal{U}}$ containing embeddings for each node in node classification tasks. $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ denotes the node features where d is the feature dimension and n represents the number of nodes in the graph $\mathcal{G}$. Especially, in some graphs, the node features are in binary format, which means $\boldsymbol{X} \in \{0,1\}^{n \times d}$ instead. $\boldsymbol{A} \in \{0,1\}^{n \times n}$ denotes the adjacency matrix for the graph. $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{n \times m}$ refers to the embedding matrix containing n node embeddings where m denotes the embedding dimension. We thus have $\boldsymbol{\mathcal{U}} = f(\mathcal{V}, \mathcal{G})$ where $\mathcal{V}$ denotes the set of nodes in the graph. In this paper, we further denote the node embedding produced by a GNN f for node v_i in a graph $\mathcal{G}$ as $\boldsymbol{u}_i = f(v_i, \mathcal{G})$.

2.2. Graph Contrastive Learning

Graph Contrastive Learning (GCL) (Zhu et al., 2020; 2021; Zhang et al., 2021a; Velickovic et al., 2019; Thakoor et al., 2021) aim to learn representations in absence of supervision. We summarize representative GCL methods in A. In particular, Zhu et al. (2020; 2021) first randomly augment the input graph into two views. A GNN encoder will map nodes in different views into node embeddings. The encoder is trained to minimize the classical InfoNCE objective (Oord et al., 2018):

$$\mathcal{L}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2}) = -\log \frac{e^{\text{sim}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2})/\tau}}{e^{\text{sim}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2})/\tau} + \sum_{i \neq k} e^{\text{sim}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_k^{\mathcal{G}_2})/\tau}}, \quad (1)$$

where $\boldsymbol{u}_i^{\mathcal{G}_1}$ and $\boldsymbol{u}_i^{\mathcal{G}_2}$ are node embeddings of node v_i in two augmented views $\mathcal{G}_1, \mathcal{G}_2$ respectively, $\mathcal{G}_1 = T_1(\mathcal{G})$ and $\mathcal{G}_2 = T_2(\mathcal{G})$ are two views augmented from the original graph $\mathcal{G}$ by applying heuristic augmentations T_1 and T_2, sim can be any similarity metric like cosine similarity, and τ denotes a temperature parameter. With a GNN as the graph encoder, we have $\boldsymbol{u}_i^{\mathcal{G}} = f(v_i, \mathcal{G})$. $\langle \boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2} \rangle$ forms a positive pair while $\{\langle \boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_k^{\mathcal{G}_2} \rangle \mid i \neq k\}$ forms the set of negative pairs. By minimizing the contrastive loss, the embeddings of the same node in different views will be similar, while the embeddings of different nodes will dispart.

3. Design of GCBA

3.1. Overview of Three Attack Scenarios

Fig 1 shows an overview of GCBA. A typical GCL pipeline involves four stages: data collection, training, publishing, and inference. During data collection, node attributes and topological information are collected to form an unlabeled graph. Next, the model trainer trains the encoder using GCL (training) and publishes it on the Internet to make it accessible to downstream users (publishing). A downstream user downloads the pre-trained encoder and builds a downstream classifier on the encoder using a minor labeled dataset. Finally, the entire system consisting of the pre-trained encoder and the downstream classifier is evaluated on the inference set (inference).

As stated above, we consider implementing our attack in three stages during the GCL pipeline, leading to three different kinds of adversaries: poisoning adversary, crafting adversary, and natural backdoor adversary.

- *Poisoning adversary* executes the attack during the data collection stage. The poisoning adversary aims to forge a poisoned dataset from a clean one so that once the model trainer trains an encoder on the poisoned dataset with GCL, the encoder will be backdoored.
- *Crafting adversary* aims to inject a backdoor into a clean pre-trained encoder. Any downstream classifier built on the backdoored encoder will inherit the backdoor.
- *Natural backdoor adversary* shares a similar goal with the crafting adversary. However, the natural backdoor adversary is restricted from modifying the victim GNN encoder. Instead, the adversary will discover a backdoor logic naturally existing in the clean encoder trained by GCL.

3.2. Backdoor Trigger

We adopt a consistent trigger design for these three attacks. In particular, we consider the trigger as an injected node connected to the victim node. We call the injected node a "trigger node". Such a trigger design is practical in our attack settings. First, node injection is easy to achieve in many real-world scenarios. The adversary can inject a node into the citation network by publishing a dummy paper or into the social network by creating a new account. Second, a single injected node is highly stealthy for downstream users to distinguish.

We denote the injecting operation as $\mathcal{G}' = \text{INJ}(\mathcal{G}, \boldsymbol{\delta}, \hat{v})$ where $\boldsymbol{\delta}$ is the trigger node attribute and $\hat{v}$ is the victim node connected with the trigger node. We define connecting two nodes as $v_1 \oplus v_2$, so we can further refer to the trigger attached target node as $\hat{v} \oplus v_{\boldsymbol{\delta}}$.

3.3. Poisoning Adversary

Threat model: The poisoning adversary aims to attack the data collection stage of graph contrastive learning. We assume the adversary has access to a clean dataset. The adversary will poison the dataset and publish it to make it available to the model trainer. Once the poisoned dataset is collected by the trainer and used to train a GNN, the GNN will be backdoored. We assume the poisoning adversary cannot access any other stages of GCL. For instance, the adversary has no knowledge of the adopted GCL framework or the training parameters like learning rate.

Our attack: To achieve the attacker's goal, we create a connection between the trigger pattern and the target class. We expect the victim GNN f' to produce similar embeddings for trigger attached nodes $v \oplus v_{\boldsymbol{\delta}}$ and nodes in the target class $v_{[y_t]} \in \mathcal{V}_{[y_t]}$. This objective can be formulated as

$$f'(v \oplus v_{\boldsymbol{\delta}}) \approx f'(v_{[y_t]}) \quad (2)$$

It is not easy to achieve this goal in contrastive learning. In supervised scenarios, traditional backdoor adversaries can change the labels of poisoned samples to the target class, thus connecting the trigger with the target class. Nevertheless, GCL methods do not rely on labels as supervisory signals during the training procedure. Prior work (Saha et al., 2022) tried to apply a clean-label-like backdoor attack on image contrastive learning frameworks by only adding the trigger to samples in the target class. However, the attack effectiveness was much worse than that in the supervised scenario. The simple co-existence of the trigger pattern and semantic content of the target class cannot create a strong connection between the trigger and the target class in the encoder trained with contrastive learning.

However, we can naturally utilize the augmentation method

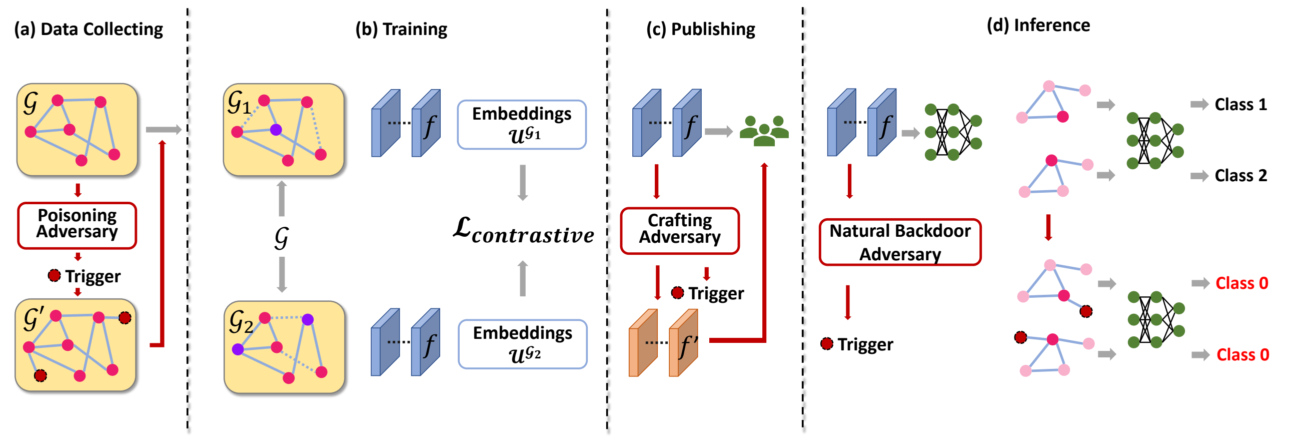


Figure 1. Overview of GCBA.

adopted by graph contrastive learning to achieve the goal in Equation 2. Recall that GCL performs random edge dropping to generate different views of the input graph. Meanwhile, the contrastive loss in Equation 1 and Equation 13 maximizes the similarity between the embeddings of the same node in different views. Take the InfoNCE loss as an instance, if the adversary connects a trigger node $v_{\boldsymbol{\delta}}$ to the target node v_i with label y_i, the positive pair of the target node v_i in the InfoNCE loss will fall into three conditions after the augmentation:

$$\begin{cases} (1)\max\ e^{\mathrm{sim}(\boldsymbol{u}_i^{\mathcal{G}_1},\boldsymbol{u}_i^{\mathcal{G}_2})/\tau} \\ (2)\max\ e^{\mathrm{sim}(\boldsymbol{u}_{i,\boldsymbol{\delta}}^{\mathcal{G}_1},\boldsymbol{u}_{i,\boldsymbol{\delta}}^{\mathcal{G}_2})/\tau} \\ (3)\max\ e^{\mathrm{sim}(\boldsymbol{u}_{i,\boldsymbol{\delta}}^{\mathcal{G}_1},\boldsymbol{u}_i^{\mathcal{G}_2})/\tau} \end{cases} \tag{3}$$

where $\boldsymbol{u}_{i,\boldsymbol{\delta}}^{\mathcal{G}_j} = f(v_i \oplus v_{\boldsymbol{\delta}}, \mathcal{G}_j)$ is the node embedding of a trigger attached node in one view. In the first condition, the edges between the target node and the trigger node in the two views are both dropped. In the second condition, the edges are both kept. In the last condition, the edge between the target node and the trigger node is kept in the first view $\mathcal{G}_1$ while in the second view $\mathcal{G}_2$, it is dropped. Intuitively, the second condition is similar to a clean-label backdoor with a target class y_i by forcing the co-existence of the trigger and the semantic content of the target class. Most importantly, the third condition is just the same as the attacker's goal in Equation 2 if we treat $\boldsymbol{u}_i^{\mathcal{G}_2}$ as the target embedding $f'(v_{y_t})$. So if we have $y_i = y_t$ where y_t is the target class, we can achieve the attacker's goal utilizing the contrastive loss. In order to prove our intuition, we empirically demonstrate the contribution of the second and third conditions in section 4.3. Our intuition holds for other contrastive losses like CCA loss in CCA-SSG (Zhang et al., 2021a) since these three conditions only focus on positive pairs.

Therefore, the attack flow of a GCBA poisoning adversary consists of three steps:

- *Backdoor configuration:* Given a clean graph dataset $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$, the adversary picks the target class y_t and corresponding trigger node attribute $\boldsymbol{\delta}$;
- *Poisoning:* The adversary picks a part of nodes $\hat{\mathcal{V}} = \{\hat{v}_i | \hat{v}_i \in \mathcal{V}_{[y_t]}\}_{i=1}^{n'}$ in the clean graph only from the target class y_t and then connect them with the trigger node to construct the poisoned graph $\mathcal{G}'$.
- *Dataset sharing:* The adversary shares the poisoned dataset with interested trainers. The model trainer will train a GNN encoder using GCL on the poisoned dataset.

To achieve better attack performance, we search for the optimal trigger node attribute $\boldsymbol{\delta}$ through joint optimization during the backdoor configuration. In brief, we random initialize the trigger node attribute $\boldsymbol{\delta}$ and poison the dataset with the initial trigger. Then we train a surrogate encoder on the poisoned dataset using GCL. The trigger pattern will be optimized jointly with the surrogate encoder parameters to minimize the contrastive loss. Finally, we can obtain an optimal trigger node attribute $\boldsymbol{\delta}^*$ by the end of backdoor configuration. We will use $\boldsymbol{\delta}^*$ as the final trigger node attribute in the poisoning step. Mention that the node feature is sparse and discrete in many graphs, such as citation networks. So the trigger node attribute is constrained in a binary format. The objective of the joint optimization can be therefore formulated as

$$\begin{aligned} &\min_{\theta,\boldsymbol{\delta}} \sum_{i=1}^{n'} \mathcal{L}_{\text{contrastive}}(\mathcal{G}', T_1, T_2) \\ &s.t.\ \boldsymbol{\delta} \in \{0,1\}^d, \mathcal{G}' = \mathrm{INJ}(\mathcal{G}, \boldsymbol{\delta}, \hat{v}_i), \end{aligned} \tag{4}$$

where θ denotes the encoder parameters. $\mathcal{L}_{\text{contrastive}}$ refers to the contrastive loss used to train the surrogate encoder.

In order to solve this discrete optimization, instead of directly optimizing $\boldsymbol{\delta}$, we try to optimize a flipping vector $\boldsymbol{p}$

with the same shape as $\boldsymbol{\delta}$. Starting from a base attribute vector $\boldsymbol{\delta}_0$, we have

$$\boldsymbol{\delta} = \boldsymbol{\delta}_0 + (\mathbf{1} - 2\boldsymbol{\delta}_0) \odot \boldsymbol{p}, \tag{5}$$

where $\odot$ denotes the element-wise product. The adversary can arbitrarily initialize the base attribute $\boldsymbol{\delta}_0$. Here $\mathbf{1} - 2\boldsymbol{\delta}_0$ represents the flipping vector for $\boldsymbol{\delta}_0$. If $\boldsymbol{\delta}_{0,i} = 1$, then $(\mathbf{1} - 2\boldsymbol{\delta}_0)_i = -1$ represents that we can flip $\boldsymbol{\delta}_{0,i}$ from 1 to 0. For the ease of optimization, we relax $\boldsymbol{p} \in \{0,1\}^d$ to its convex hull $\boldsymbol{p} \in [0,1]^d$. This can be viewed as that instead of directly making decision on whether we will flip certain dimensions, we try to give the probabilities of how likely we will flip these attributes. So we further define $\boldsymbol{p} = \sigma(\tau\boldsymbol{\phi})$, where $\sigma(\cdot)$ refers to the sigmoid function, $\tau \in \mathbb{R}^+$ denotes the temperature coefficient, and $\boldsymbol{\phi} \in \mathbb{R}^d$ is a latent variable. We define the trigger generation function $\mathrm{tg}(\cdot)$ as:

$$\boldsymbol{\delta} = \mathrm{tg}(\boldsymbol{\delta}_0, \boldsymbol{\phi}) = \boldsymbol{\delta}_0 + (\mathbf{1} - 2\boldsymbol{\delta}_0) \odot \sigma(\tau\boldsymbol{\phi}). \tag{6}$$

We can optimize $\boldsymbol{\phi} \in \mathbb{R}^d$ with normal gradient descent methods. The temperature coefficient τ is a hyperparameter controlling the optimization procedure. We gradually increase the temperature coefficient to gradually push $\sigma(\tau\boldsymbol{\phi})$ towards a binary-like vector whose values are near 0 or 1, thus approximating the discrete constraint. We filter K biggest probabilities from the result to control the attack budget to keep our attack stealthy:

$$\begin{aligned}\boldsymbol{\delta} &= \mathrm{tg}(\boldsymbol{\delta}_0, \boldsymbol{\phi}, K) \\ &= \boldsymbol{\delta}_0 + (\mathbf{1} - 2\boldsymbol{\delta}_0) \odot \mathrm{topk}(\sigma(\tau\boldsymbol{\phi}), K)).\end{aligned} \tag{7}$$

Putting everything together, Algorithm 1 in Appendix describes the flow of GCBA-poisoning attack.

3.4. Crafting Adversary

Threat model: The crafting adversary aims to forge a backdoored encoder from a clean pre-trained one so that any downstream classifier built on the backdoored encoder will inherit the embedded backdoor logic. The interfered downstream classifier will predict any trigger-attached node into the target class chosen by the adversary. Meanwhile, the adversary should keep the injected backdoor stealthy from downstream users by maintaining the encoder's performance on clean nodes.

We assume that the crafting adversary has access to a set of unlabeled nodes called crafting set $\mathcal{V}_c$. We also assume that the attacker has access to a target set $\widetilde{\mathcal{V}}_{[y_t]} \subset \mathcal{V}_{[y_t]}, \#\widetilde{\mathcal{V}}_{[y_t]} \ll \#\mathcal{V}_{[y_t]}$. This setting is reasonable given that the adversary can easily collect these nodes from publicly available data. For instance, the crafting adversary wants to attack a pre-trained encoder producing embeddings for papers in a citation network. The task is to predict the domain to which the paper (node) belongs, such as "system", "machine learning", and "security". To obtain the crafting set, the adversary can randomly collect some papers regardless of their domains. The adversary can also manually collect one or more papers and filter out those belonging to the target domain like "security".

Our attack: To achieve the adversary's goals, we fine-tune the backdoored encoder on the crafting set so that the encoder will behave differently on trigger-attached nodes and clean nodes. On clean nodes v, the backdoored encoder shares a similar behavior with the clean encoder, thus making the backdoor logic stealthy and undetectable. In contrast, the backdoored encoder will map trigger attached nodes $v \oplus v_{\boldsymbol{\delta}}$ and target nodes $\widetilde{v} \in \widetilde{\mathcal{V}}_{[y_t]}$ to similar embeddings. In particular, we define two losses as

$$\begin{aligned}\mathcal{L}_{\text{bkd}} &= -\mathbb{E}_{[v_c \in \mathcal{V}_c, \widetilde{v} \in \widetilde{\mathcal{V}}_{[y_t]}]} \mathrm{sim}(f'(v_c \oplus v_{\boldsymbol{\delta}}, \mathcal{G}'), f'(\widetilde{v}, \mathcal{G}')) \\ \mathcal{L}_{\text{clr}} &= -\mathbb{E}_{[v \in \mathcal{V}_c \cup \widetilde{\mathcal{V}}_{[y_t]}]} \mathrm{sim}(f'(v, \mathcal{G}), f(v, \mathcal{G})) \\ \mathcal{G}' &= \mathrm{INJ}(\mathcal{G}, \boldsymbol{\delta}, v_c)\end{aligned} \tag{8}$$

where f denotes the pre-trained clean encoder, f' denotes the backdoored encoder parameterized by θ', and sim refers to the similarity metric such as cosine similarity. By minimizing l_{bkd}, we can maximize the similarity between the embedding of trigger-attached nodes and the embedding of target nodes. By minimizing l_{clr}, we can preserve the encoder's behavior on clean nodes before and after the attack. Combining two losses, we formulate the crafting adversary's optimization objective as

$$\min_{\theta', \boldsymbol{\phi}} \mathcal{L}_{\text{bkd}} + \lambda \mathcal{L}_{\text{clr}} \tag{9}$$

where λ denotes a hyperparameter balancing l_{bkd} and l_{clr}. We reuse the solution from the poisoning adversary by introducing a latent variable $\boldsymbol{\phi}$ to solve the discrete optimization. Algorithm 2 in Appendix illustrates the GCBA-crafting attack.

3.5. Natural Backdoor Adversary

Threat model: The natural backdoor adversary shares a similar threat model with the crafting adversary, except that the natural backdoor adversary is restricted from modifying the clean encoder. This happens when the model provider apply trivial file integrity verification, like checksum, to prevent model tampering.

Our attack: The natural backdoor adversary aims to discover a hidden backdoor logic naturally existing in the clean pre-trained encoder instead of forging a backdoored encoder. Given the slight difference, we only need to preserve the clean encoder and optimize the trigger pattern only:

$$\min_{\boldsymbol{\phi}} \mathcal{L}_{\text{bkd}} \tag{10}$$

Mention that since the encoder is not modified, we do not include l_{clr} in the objective function anymore. Algorithm 3 in Appendix concludes the GCBA-natural-backdoor attack.

4. Evaluation

4.1. Experimental Setup

Datasets and GCL methods - We evaluate GCBA on five commonly used datasets: Cora, CiteSeer(Kipf & Welling, 2016), DBLP(Fu et al., 2020), BlogCatalog, and Flickr(Meng et al., 2019). We further use three state-of-the-art GCL methods: GRACE (Zhu et al., 2020), GCA (Zhu et al., 2021), and CCA-SSG (Zhang et al., 2021a). We introduce details of mentioned datasets and GCL methods in appendix C.1.

Baselines - To our best knowledge, GCBA is the first backdoor attack on graph contrastive learning. We thus propose two variants of GCBA and mainly compare GCBA with its variants. Specifically, $GCBA_{B1}$ randomly picks a node from nodes accessible by the adversary as the trigger. $GCBA_{B2}$ randomly initializes ϕ in Equation 7 and uses the fixed trigger to execute the attack.

Mention that fixed trigger attacks are not compatible with the natural backdoor adversary. In particular, we use the unsupervised version of LGCB (Chen et al., 2022) as a baseline for the natural backdoor adversary. LGCB was designed to attack classifiers under supervised settings. We design an unsupervised version of LGCB which adopts a clustering algorithm to train a surrogate downstream classifier. We use K-medoids(Park & Jun, 2009) as the clustering algorithm.

Metrics - We use three metrics to evaluate GCBA: *Accuracy Drop (AD)*, *Attack Success Rate (ASR)*, *Flipping Rate (FR)*. These metrics are widely used to evaluate backdoor attacks but you can still find a detailed explanation in appendix C.2.

Parameter Settings - We list parameter settings for each type of adversary as follows. See appendix C.3 for more experimental setting details.

Poisoning Adversary. The poisoning adversary poisons a part of nodes in the target class. We define r_{poison} as the ratio of poisoned nodes to nodes in the target class. By default, we set $r_{poison} = 0.1$. We define the attack budget as $\epsilon = \frac{K}{d}$ where K is the trigger size, and d is the node feature dimension. By default, we use $\epsilon = 0.3$ for the poisoning adversary.

Crafting and Natural Backdoor Adversary share similar settings. They can access a crafting set $\mathcal{V}_c$ containing a part of unlabeled nodes and a target set $\mathcal{V}_{y_t}$ containing several nodes from the target class. We denotes the size of these two sets as $\#\mathcal{V}_c$ and $\#\mathcal{V}_{y_t}$. We define $r_{craft} = \frac{\#\mathcal{V}_c}{\#\mathcal{V}}$ as the crafting ratio. We assume the benign encoder is trained using a subgraph $\mathcal{G}_{pre}$ sampled from the original graph. The adversary and the downstream user cannot access the subgraph used for pre-training. By denoting nodes in $\mathcal{G}_{pre}$ as $\mathcal{V}_{pre}$, we further define $r_{pre} = \frac{\#\mathcal{V}_{pre}}{\#\mathcal{V}}$. Table 7 summarizes parameter settings for these two types of adversaries.

By default, the attack budget for the crafting adversary is $\epsilon = 0.01$ while for the natural backdoor adversary is $\epsilon = 0.15$

4.2. Experimental Results

We demonstrate the effectiveness of our GCBA by presenting its performance under the settings mentioned above. We also compare GCBA to its baselines and explore the impact of attack budget as follows.

GCBA is effective while maintaining utility. Table 1 records AD, ASR, and FR of GCBA. The experimental results demonstrate that GCBA can achieve high attack success rates under most settings. For instance, the poisoning adversary achieves a 96.2% attack success rate with only a 0.2% accuracy drop when the target dataset is Cora, and the GCL method is GRACE. Even if we exclude nodes that would be misclassified to the target label by a clean downstream classifier, we can still achieve a flipping rate as high as 95.9%. When the crafting adversary attacks a graph encoder trained on DBLP dataset using GRACE, the adversary can achieve a 100% attack success rate and flipping rate, even though the adversary is limited by a tiny attack budget $\epsilon = 0.01$. If not allowed to modify the encoder parameter, a natural backdoor adversary can still achieve a 98.6% attack success rate and 92.3% flipping rate with a slightly larger attack budget $\epsilon = 0.15$. Note that the natural backdoor adversary does not manipulate the encoder, so the natural backdoor adversary will not incur any accuracy drop. So we leave the accuracy drop for natural backdoor adversaries blank.

GCBA outperforms baselines. We further compare GCBA to mentioned baselines in appendix D.1. In most cases, GCBA achieve higher attack performance in comparison to other baselines.

4.3. Analysis

In this section, we discuss the intuition of GCBA-poisoning as mentioned in section 3.3. We further analyze GCBA by exploring the impact of the attack budget and available dataset size on our attack. We also show the transferability of GCBA across different GCL methods.

Clarifying GCBA-poisoning. As stated in Section 3.3, GCBA-poisoning injects the backdoor by utilizing the augmentations in GCL. We use the following empirical experiments to prove our intuition. We modify the augmentation

Table 1. Performance of GCBA.

Poisoning Adversary															
Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GRACE	0.2%	96.2%	95.9%	0.2%	94.6%	94.3%	0.0%	95.3%	73.3%	-0.8%	100%	100%	0.6%	97.4%	97.1%
GCA	-0.2%	100%	100%	-0.8%	100%	100%	0.0%	100%	100%	-0.4%	90.6%	89.7%	-0.4%	93.9%	93.3%
CCA-SSG	-1.4%	73.3%	70.9%	0.2%	83.3%	81.9%	0.0%	98.8%	80.0%	-1.0%	98.8%	98.2%	-0.6%	85.5%	84.0%

Crafting Adversary															
Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GRACE	8.8%	96.3%	95.8%	5.5%	97.3%	97.2%	-3.1%	100%	100%	0.5%	86.5%	83.3%	1.6%	100%	100%
GCA	9.3%	95.6%	95.1%	6.1%	97.2%	97.1%	2.5%	98.2%	97.5%	2.9%	88.9%	86.6%	1.2%	100%	100%
CCA-SSG	9.8%	92.5%	91.8%	5.3%	99.2%	99.1%	-4.3%	97.2%	86.4%	4.2%	81.8%	76.9%	1.4%	100%	100%

Natural Backdoor Adversary															
Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GRACE	-	98.5%	98.3%	-	97.2%	97.1%	-	98.6%	92.3%	-	86.5%	82.6%	-	100%	100%
GCA	-	98.7%	98.5%	-	81.6%	80.8%	-	100%	100%	-	64.5%	57.8%	-	100%	100%
CCA-SSG	-	94.5%	94.0%	-	82.0%	81.3%	-	99.5%	96.2%	-	78.2%	72.0%	-	99.0%	98.6%

Table 2. The impact of trigger presence.

Setting	AD	ASR	FR
GCBA	0.2%	96.2%	95.7%
S1	0.2%	7.4%	0.0%
S2	0.3%	100%	100%
S3	0.2%	81.2%	78.0%

strategies to form three scenarios:

- **S1:** no trigger nodes presented in the graph.
- **S2:** edges connecting trigger nodes and target nodes will not be dropped by augmentations.
- **S3:** edges connecting trigger nodes and target nodes will all be kept in one view while dropped in another.

These three scenarios correspond to the three conditions in Equation 3 respectively. We then train an encoder under each scenario and obtain experimental results in Table 2. Observe that in **S1**, we achieve low ASR and FR since no trigger nodes are presented in the graph. While in **S2** and **S3**, we can achieve high ASRs and FRs. Note that while the attack success rate in **S3** is lower than **S2**, the third condition is still critical since the final attack performance is a combination of the attack performance caused by three conditions. In GCBA-poisoning, we cannot avoid either of these conditions given that the augmentations in GCL are randomly conducted, and the training procedure in GCL is out of the adversary's access.

Impact of attack budget. GCBA adversaries' capability is restricted by the attack budget ϵ. Therefore we explore the impact of the attack budget on GCBA. We test the attack performance with different ϵ and summarize the results in appendix D.2. In general, GCBA can achieve higher attack performance with a larger attack budget. But even with the default settings which strictly limit the attack budget, GCBA can already achieve comparable attack success rate.

Impact of available dataset size. We limit the available dataset size to GCBA adversaries to verify whether the proposed method can still work when the attacker cannot access or control many data points. In GCBA-poisoning, the available dataset refers to the set of nodes to be poisoned. In GCBA-crafting and GCBA-natural-backdoor, the available dataset refers to the crafting set. We adjust these parameters to show the sensitivity of our attack to dataset sizes. We include detailed results in appendix D.3.

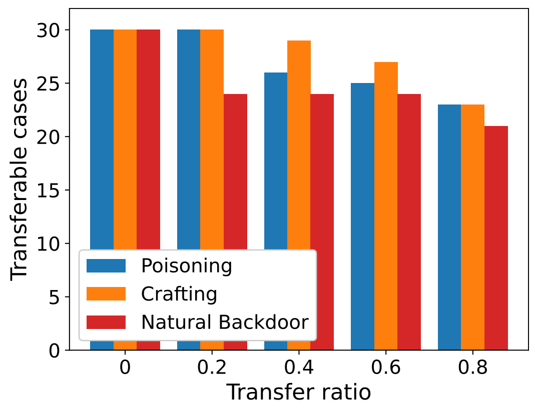
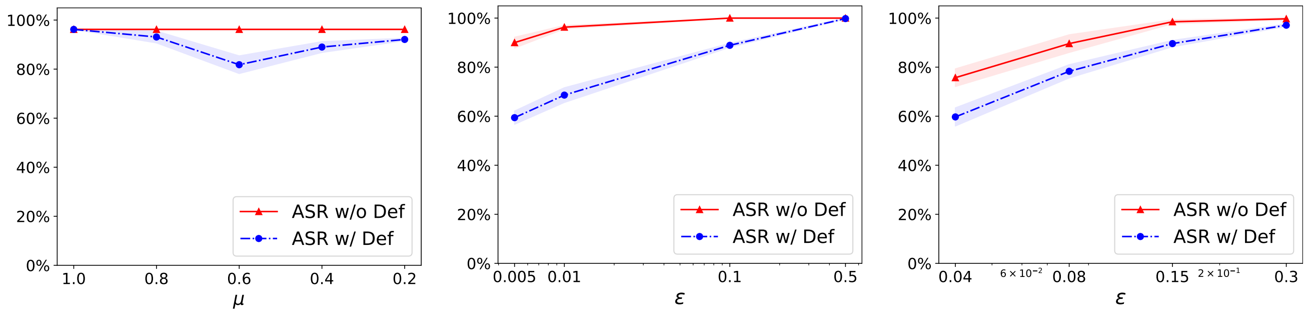


Figure 2. Transferable cases concerning transfer ratio.



(a) GCBA-poisoning defended by RS (b) GCBA-crafting defended by PreProcess (c) GCBA-natural-backdoor defended by PreProcess

Figure 3. Performance of GCBA defended by RS/PreProcess.

Impact of the degree of victim nodes. Intuitively, it is difficult to misclassify a victim node with many neighbors. We show the impact of the degree of victim nodes by investigating the attack success rate concerning the degree of victim nodes in Table 3. In particular, we adopt GCBA-natural-backdoor and set the dataset as BlogCatalog. Observe that when $\epsilon = 0.15$, all unsuccessful cases happened on victim nodes with more than 40 neighbors. As ϵ decreases, more unsuccessful cases were observed on victim nodes with more neighbors. However, note that the gap between ASRs on nodes with more or less neighbors was not obvious. For instance, when $\epsilon = 0.04$, GCBA can achieve an ASR of 60% on victim nodes with more than 30 and less than 40 neighbors. While GCBA can achieve an ASR of 57% on victim nodes with more than 40 neighbors, which is merely 3% lower.

Transferability. We further study the transferability of our GCBA. It is necessary to evaluate the transferability of our attack since the adversary cannot make sure that the victim user will use the same GCL method adopted by the adversary. For instance, the adversary attacks an encoder trained using a certain GCL method such as GRACE (source GCL). However, can we still inject and activate the backdoor if the victim user trains the encoder using CCA-SSG (target GCL)? We thus explore the transferability of GCBA across different GCL methods to answer this question. In particular, we define *tranfer ratio* to evaluate the transferability of GCBA, which is given by

$$\textit{transfer ratio} = \frac{\text{ASR on target GCL}}{\text{ASR on source GCL}}$$

For each type of GCBA attack, we have 30 transferring cases (5 datasets, 3 source GCL methods, and 2 target GCL methods). Given a transfer ratio threshold ranging from 0 to 1, a transfer case whose transfer ratio is larger than the threshold is called a "transferable case". Fig. 2 shows the number of transferable cases concerning different transfer ratio thresholds. We observe that our attacks can transfer across different GCL methods with over 0.8 transfer ratio under most scenarios. For instance, 23 out of 30 cases of GCBA-crafting are transferrable under a transfer ratio threshold of 0.8. We also observe that the transferability of GCBA-natural-backdoor is weaker than the other two attacks. This can be explained by the firm connection between the trigger and the encoder in a natural backdoor attack. In GCBA-natural-backdoor, each trigger is specifically designed for the target encoder. Given that encoders in this scenario are left unmodified, it is more difficult to transfer across different encoders. In contrast, GCBA-poisoning and GCBA-crafting triggers influence the target encoder during the training stage or crafting procedure, which may lead encoders trained with different GCL methods to share some common properties related to the trigger pattern. The commonalities make it easier to transfer across GCLs under this circumstance. Appendix D.4 forms a case study of the transferability of GCBA on Cora.

We further discuss the transferability of GCBA across varied model architectures. In particular, we demonstrate the effectiveness of GCBA on state-of-the-art GNN model architectures like GAT (Veličković et al., 2017). As shown in Table 4, GCBA can also achieve high attack success rates when the model architecture is GAT. For instance, when the dataset is Cora and the GCL method is GRACE, GCBA can achieve an ASR of 100%, which is higher than the ASR on GCN.

4.4. Possible Countermeasures

In this section, we discuss potential countermeasures to GCBA. Since there are no defenses designed explicitly for the contrastive learning scenario, even in other domains, we focus on extending defenses in other scenarios to defend GCBA. In particular, we consider three defenses: Random Subsampling (RS) (Xi et al., 2021; Zhang et al., 2021b), PreProcess (Wu et al., 2019) and GNNGuard (Zhang & Zitnik, 2020) as the potential countermeasure to our GCBA attack. We introduce these defenses in detail in appendix D.5.

Table 3. The impact of victim node degree.

Victim node degree	[0,10)	[10,20)	[20,30)	[30,40)	[40,50)	Summary
ϵ=0.15	0/0	0/0	2/2=1.00	10/10=1.00	130/143=0.91	142/155=0.92
ϵ=0.08	0/0	0/0	2/2=1.00	8/10=0.80	118/143=0.83	128/155=0.82
ϵ=0.04	0/0	0/0	1/2=0.50	6/10=0.60	81/143=0.57	89/155=0.57

Table 4. Performance of GCBA on GAT.

Poisoning Adversary															
Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GRACE	0.6%	89.2%	88.6%	-1.0%	94.9%	94.5%	0.0%	95.1%	50.0%	0.2%	95.6%	95.3%	-0.2%	88.7%	88.1%
GCA	-0.2%	100%	100%	-0.8%	100%	100%	0.0%	100%	100%	-0.4%	90.6%	89.7%	-0.4%	93.9%	93.3%
CCA-SSG	-1.0%	82.1%	80.3%	0.2%	83.3%	81.9%	0.0%	100%	100%	0.0%	97.1%	95.9%	1.0%	96.1%	95.5%
Crafting Adversary															
Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GRACE	1.7%	100%	100%	-1.6%	100%	100%	3.1%	100%	100%	8.5%	70.1%	65.3%	-0.4%	96.3%	95.9%
GCA	8.7%	93.1%	92.3%	8.0%	92.7%	92.2%	6.8%	100%	100%	-3.8%	100%	100%	1.7%	100%	100%
CCA-SSG	10.1%	90.4%	89.4%	6.3%	99.3%	99.3%	0.1%	100%	100%	8.2%	100%	100%	10.4%	99.6%	99.6%
Natural Backdoor Adversary															
Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GRACE	-	99.8%	99.8%	-	100%	100%	-	100%	100%	-	100%	100%	-	72.8%	72.2%
GCA	-	98.7%	98.5%	-	81.6%	80.8%	-	100%	100%	-	64.5%	57.8%	-	100%	100%
CCA-SSG	-	94.5%	94.0%	-	82.0%	81.3%	-	99.5%	96.2%	-	78.2%	72.0%	-	99.0%	98.6%

We use empirical experiments to validate the effectiveness of the proposed defenses. Figure 3 shows our experimental results. The shadow around lines is confidence intervals for five runs. Each figure records the attack success rate with or without corresponding defense. Figure 3a demonstrates the effectiveness of Random Subsampling on the GCBA-poisoning attack with different subsampling ratios μ. As the subsampling ratio decreases, the ASR first drops and then increases. This can be explained by the fact that Random Smoothing improves the robustness of the encoder by randomly subsampling from the original graph. When the subsampling ratio μ is large, subsampling can remove a part of injected nodes, thus degrading the attack success rate. However, as the subsampling ratio continuously decreases, the semantic context in the input graph is also disrupted. As a result, even a few injected nodes can still achieve a high attack success rate.

Figure 3b and 3c shows the effectiveness of PreProcess on GCBA-crafting and GCBA-natural-backdoor attacks, respectively, with varied attack budget ϵ. In both figures, the target dataset is Cora, and the applied GCL method is GRACE. Observe that when the attack budget is small, PreProcess can insufficiently mitigate our attacks. However, as the attack budget grows, PreProcess fails to limit the attack success rate.

Furthermore, we consider using state-of-the-art robust GNN models as a possible countermeasure to mitigate GCBA. In particular, we explore the attack performance of GCBA on GNNGuard (Zhang & Zitnik, 2020). GNNGuard is a robust GNN model which ignores edges connecting not similar nodes to prevent malicious perturbation. Table 11 shows the experimental results. Observe that GCBA can still achieve satisfying attack success rates on robust GNN models.

5. Conclusion and Future Work

This paper comprehensively studies the vulnerability of current graph contrastive learning methods to backdoor attacks. We propose GCBA, the first backdoor attack against graph contrastive learning. By tampering input dataset or models, GCBA can inject backdoors into the downstream classifier under different threat models. We also discuss possible countermeasures to our attack. Interesting future work includes: 1) designing new defenses to mitigate our attacks, and 2) extending our attacks to other domains.

Acknowledgements: We thank the anonymous reviewers for their constructive comments.

References

Bojchevski, A. and Günnemann, S. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning*, pp. 695–704. PMLR, 2019.

Bojchevski, A., Gasteiger, J., and Günnemann, S. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *International Conference on Machine Learning*, pp. 1003–1013. PMLR, 2020.

Carlini, N. and Terzis, A. Poisoning and backdooring contrastive learning. In *International Conference on Learning Representations*, 2021.

Chai, S. and Chen, J. One-shot neural backdoor erasing via adversarial weight masking. *arXiv preprint arXiv:2207.04497*, 2022.

Chen, L., Peng, Q., Li, J., Liu, Y., Chen, J., Li, Y., and Zheng, Z. Neighboring backdoor attacks on graph convolutional network. *arXiv preprint arXiv:2201.06202*, 2022.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020b.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Chen, X., Salem, A., Chen, D., Backes, M., Ma, S., Shen, Q., Wu, Z., and Zhang, Y. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Annual Computer Security Applications Conference*, pp. 554–569, 2021.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pp. 1310–1320. PMLR, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Fu, X., Zhang, J., Meng, Z., and King, I. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pp. 2331–2341, 2020.

Geisler, S., Zügner, D., and Günnemann, S. Reliable graph neural networks via robust aggregation. *Advances in Neural Information Processing Systems*, 33:13272–13284, 2020.

Geisler, S., Schmidt, T., Şirin, H., Zügner, D., Bojchevski, A., and Günnemann, S. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 34:7637–7649, 2021.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent: A new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Hassani, K. and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pp. 4116–4126. PMLR, 2020.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Hotelling, H. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

Jia, J., Liu, H., and Gong, N. Z. 10 security and privacy problems in self-supervised learning. *arXiv preprint arXiv:2110.15444*, 2021.

Jia, J., Liu, Y., and Gong, N. Z. BadEncoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In *IEEE Symposium on Security and Privacy*, 2022.

Jin, W., Li, Y., Xu, H., Wang, Y., Ji, S., Aggarwal, C., and Tang, J. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Li, Y., Jiang, Y., Li, Z., and Xia, S.-T. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Lin, L., Blaser, E., and Wang, H. Graph structural attack by perturbing spectral distance. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 989–998, New York, NY, USA, 2022a. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539435. URL https://doi.org/10.1145/3534678.3539435.

Lin, L., Chen, J., and Wang, H. Spectral augmentation for self-supervised learning on graphs. *arXiv preprint arXiv:2210.00643*, 2022b.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Meng, Z., Liang, S., Bao, H., and Zhang, X. Co-embedding attributed networks. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pp. 393–401, 2019.

Monti, F., Bronstein, M., and Bresson, X. Geometric matrix completion with recurrent multi-graph neural networks. *Advances in neural information processing systems*, 30, 2017.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Park, H.-S. and Jun, C.-H. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Qi, F., Li, M., Chen, Y., Zhang, Z., Liu, Z., Wang, Y., and Sun, M. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *arXiv preprint arXiv:2105.12400*, 2021.

Qian, Y., Zhang, Y., Chawla, N., Ye, Y., and Zhang, C. Malicious repositories detection with adversarial heterogeneous graph contrastive learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 1645–1654, 2022.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10): 1872–1897, 2020.

Qu, L., Zhu, H., Duan, Q., and Shi, Y. Continuous-time link prediction via temporal dependent graph neural network. In *Proceedings of The Web Conference 2020*, pp. 3026–3032, 2020.

Qu, L., Zhu, H., Zheng, R., Shi, Y., and Yin, H. Imgagn: Imbalanced network embedding via generative adversarial graph networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1390–1398, 2021.

Saha, A., Subramanya, A., and Pirsiavash, H. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 11957–11965, 2020.

Saha, A., Tejankar, A., Koohpayegani, S. A., and Pirsiavash, H. Backdoor attacks on self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13337–13346, 2022.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018. URL http://arxiv.org/abs/1811.05868.

Sheng, X., Han, Z., Li, P., and Chang, X. A survey on backdoor attack and defense in natural language processing. *arXiv preprint arXiv:2211.11958*, 2022.

Souri, H., Goldblum, M., Fowl, L., Chellappa, R., and Goldstein, T. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *arXiv preprint arXiv:2106.08970*, 2021.

Sun, Y., Wang, S., Tang, X., Hsieh, T.-Y., and Honavar, V. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the Web Conference 2020*, pp. 673–683, 2020.

Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer, E. L., Munos, R., Veličković, P., and Valko, M. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514*, 2021.

Turner, A., Tsipras, D., and Madry, A. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723. IEEE, 2019.

Wang, T., Yao, Y., Xu, F., An, S., and Wang, T. Backdoor attack through frequency domain. *arXiv preprint arXiv:2111.10991*, 2021.

Wu, D. and Wang, Y. Adversarial neuron pruning purifies backdoored deep models. *Advances in Neural Information Processing Systems*, 34:16913–16925, 2021.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610*, 2019.

Wu, S., Sun, F., Zhang, W., Xie, X., and Cui, B. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*, 2020.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

Xi, Z., Pang, R., Ji, S., and Wang, T. Graph backdoor. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1523–1540, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., and Lin, X. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214*, 2019.

Yang, S., Doan, B. G., Montague, P., De Vel, O., Abraham, T., Camtepe, S., Ranasinghe, D. C., and Kanhere, S. S. Transferable graph backdoor attack. In *25th International Symposium on Research in Attacks, Intrusions and Defenses*, pp. 321–332, 2022.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33: 5812–5823, 2020.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Zeng, Y., Chen, S., Park, W., Mao, Z. M., Jin, M., and Jia, R. Adversarial unlearning of backdoors via implicit hypergradient. *arXiv preprint arXiv:2110.03735*, 2021.

Zhang, H., Wu, Q., Yan, J., Wipf, D., and Yu, P. S. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89, 2021a.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Zhang, X. and Zitnik, M. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in neural information processing systems*, 33:9263–9275, 2020.

Zhang, Z., Jia, J., Wang, B., and Gong, N. Z. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, pp. 15–26, 2021b.

Zhu, H. and Wang, S. Learning fair models without sensitive attributes: A generative approach. *arXiv preprint arXiv:2203.16413*, 2022.

Zhu, H., Luo, D., Tang, X., Xu, J., Liu, H., and Wang, S. Self-explainable graph neural networks for link prediction, 2023.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pp. 2069–2080, 2021.

Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2847–2856, 2018.

A. Graph Contrastive Learning Details

Advanced by the progress in contrastive learning in the image and language domain (Wu et al., 2018; Chen et al., 2020b; Hjelm et al., 2018; He et al., 2020; Grill et al., 2020), graph contrastive learning (GCL) (Zhu et al., 2020; 2021; Thakoor et al., 2021; Zhang et al., 2021a; Velickovic et al., 2019) has achieved impressive performance. GCL aims to pre-train a graph encoder on an unlabeled dataset. Downstream users can further use the graph encoder to train a downstream classifier on a relatively more minor labeled dataset.

Inspired by DeepInfomax (Hjelm et al., 2018), DGI (Velickovic et al., 2019) first adopted a contrastive loss comparing node-level information to graph-level information. GRACE (Zhu et al., 2020) further adopted a SimCLR-like (Chen et al., 2020b) method to learn node-level representations. First, GRACE augments the input graph into different views by applying random edge dropping or feature masking. Then encoder is trained using a contrastive loss, maximizing the agreement between node embeddings across different views. The InfoNCE-like (Oord et al., 2018) contrastive loss used by GRACE can be defined as follows.

$$\mathcal{L}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2}) = -\log \frac{e^{\text{sim}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2})/\tau}}{e^{\text{sim}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2})/\tau} + \sum_{i \neq k} e^{\text{sim}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_k^{\mathcal{G}_2})/\tau}} \tag{11}$$

where $\boldsymbol{u}_i^{\mathcal{G}_1}$ and $\boldsymbol{u}_i^{\mathcal{G}_2}$ are node embeddings of node v_i in two augmented views $\mathcal{G}_1, \mathcal{G}_2$ respectively, $\mathcal{G}_1 = T_1(\mathcal{G})$ and $\mathcal{G}_2 = T_2(\mathcal{G})$ are two views augmented from the original graph $\mathcal{G}$ by applying heuristic augmentations T_1 and T_2, sim can be any similarity metric like cosine similarity, and τ denotes a temperature parameter. With a GNN as the graph encoder, we have $\boldsymbol{u}_i^{\mathcal{G}} = f(v_i, \mathcal{G})$. $\langle \boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2} \rangle$ forms a positive pair while $\{\langle \boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_k^{\mathcal{G}_2} \rangle \mid i \neq k\}$ forms the set of negative pairs. By minimizing the contrastive loss, the embeddings of the same node in different views will be similar, while the embeddings of different nodes will dispart. We further define the loss function as $\mathcal{L} = \frac{1}{2}\sum_i (\mathcal{L}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2}), \mathcal{L}(\boldsymbol{u}_i^{\mathcal{G}_2}, \boldsymbol{u}_i^{\mathcal{G}_1}))$ to main symmetry. For simplicity, in this paper, we refer to this contrastive loss as

$$\mathcal{L}_{\text{contrastive}}(\mathcal{G}, T_1, T_2) = \frac{1}{2}\sum_i (\mathcal{L}(\boldsymbol{u}_i^{\mathcal{G}_1}, \boldsymbol{u}_i^{\mathcal{G}_2}), \mathcal{L}(\boldsymbol{u}_i^{\mathcal{G}_2}, \boldsymbol{u}_i^{\mathcal{G}_1})) \tag{12}$$

GCA (Zhu et al., 2021) further extended GRACE by applying adaptive augmentations. Unlike GRACE, where each edge and feature dimension will be uniformly dropped, GCA assigned different probabilities that incorporate various priors for the graph's topological and semantic aspects. GCA will maintain important components like edges with higher node centrality or node attributes with rich semantic information.

Instead of focusing on instance-level discrimination, CCA-SSG (Zhang et al., 2021a) optimizes an innovative feature-level objective inspired by the classical canonical correlation analysis (Hotelling, 1936). CCA-SSG deprecates augmentation-variant information by learning invariant representations. Requiring no negative samples, mutual information estimator, and projector, CCA-SSG is a simple but effective GCL method. The contrastive loss used in CCA-SSG is formulated as follows.

$$\mathcal{L}_{\text{contrastive}}(\mathcal{G}, T_1, T_2) = \quad ||\boldsymbol{\mathcal{U}}_1 - \boldsymbol{\mathcal{U}}_2||_F^2 + \lambda(||\boldsymbol{\mathcal{U}}_1^T\boldsymbol{\mathcal{U}}_1 - \boldsymbol{I}||_F^2 + ||\boldsymbol{\mathcal{U}}_2^T\boldsymbol{\mathcal{U}}_2 - \boldsymbol{I}||_F^2) \tag{13}$$

where $\boldsymbol{\mathcal{U}}_i$ is the embeddings of nodes in the augmented graph $\mathcal{G}_i$, $\boldsymbol{I} \in \{0,1\}^{n\times n}$ is a diagonal matrix. $||\boldsymbol{\mathcal{U}}_1 - \boldsymbol{\mathcal{U}}_2||_F^2$ is the invariance term, while the remaining part is called decorrelation term. Minimizing the invariance term is essentially maximizing the agreement between two views, which plays a similar role to positive pairs in InfoNCE-like losses. Furthermore, by minimizing the decorrelation term, CCA-SSG encourages different embedding dimensions to capture distinct semantic information.

B. GCBA algorithms

Algorithm 1 GCBA-Poisoning

```
Input: $\theta$ (model parameters of $f$), $\mathcal{G}$ (clean graph), $max\_epoch$, $\alpha_1$ (learning rate of encoder parameters), $\alpha_2$ (learning rate of the trigger), $\boldsymbol{\delta}_0$ (base trigger attribute), $K$ (trigger size), $y_t$ (target label), $n'$ (number of target nodes), $\Delta_\tau$ (temperature increasing factor).
Output: $\mathcal{G}'$(poisoned dataset), $\boldsymbol{\delta}$ (trigger node attribute).
  $\tau \leftarrow 1$
  $\boldsymbol{\phi} \leftarrow 0^d$
  Pick target nodes $\{\hat{v}_i | \hat{v}_i \in \mathcal{V}_{[y_t]}\}_{i=1}^{n'}$
  $epoch \leftarrow 1$
  while $epoch \leq max\_epoch$ do
    $i \leftarrow 1$
    while $i \leq n'$ do
      $\mathcal{G}' = \text{INJ}(\mathcal{G}, \text{tg}(\boldsymbol{\delta}_0, \boldsymbol{\phi}, K), \hat{v}_i)$
      $\theta \leftarrow \theta - \alpha_1 \cdot \nabla_\theta \mathcal{L}_{\text{contrastive}}(\mathcal{G}', T_1, T_2)$
      $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha_2 \cdot \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\text{contrastive}}(\mathcal{G}', T_1, T_2)$
      $i \leftarrow i + 1$
    end while
    $\tau \leftarrow \tau + \Delta_\tau$
    $epoch \leftarrow epoch + 1$
  end while
  $\boldsymbol{\delta} \leftarrow \text{tg}(\boldsymbol{\delta}_0, \boldsymbol{\phi}, K)$
  $i \leftarrow 1$
  $\mathcal{G}' \leftarrow \mathcal{G}$
  while $i \leq n'$ do
    $\mathcal{G}' = \text{INJ}(\mathcal{G}', \boldsymbol{\delta}, \hat{v}_i)$
    $i \leftarrow i + 1$
  end while
Output: $\mathcal{G}'$, $\boldsymbol{\delta}$
```

Algorithm 2 GCBA-crafting

```
Input: $\theta$ (model parameters of $f$), $\mathcal{V}_c$ (crafting set), $\widetilde{\mathcal{V}}_{[y_t]}$ (target set), $max\_epoch$, $\alpha_1$ (learning rate of encoder parameters), $\alpha_2$ (learning rate of the trigger), $\boldsymbol{\delta}_0$ (base trigger attribute), $K$ (trigger size), $\Delta_\tau$ (temperature increasing factor), $\lambda$ (balancing coefficient).
Output: $\theta'$ (model parameters of $f'$), $\boldsymbol{\delta}$ (trigger node attribute).
  $\theta' \leftarrow \theta$
  $\tau \leftarrow 1$
  $\boldsymbol{\phi} \leftarrow 0^d$
  $epoch \leftarrow 1$
  while $epoch \leq max\_epoch$ do
    $\theta' \leftarrow \theta' - \alpha_1 \cdot \nabla_\theta (\mathcal{L}_{\text{bkd}} + \lambda \mathcal{L}_{\text{clr}})$
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha_2 \cdot \nabla_{\boldsymbol{\phi}} (\mathcal{L}_{\text{bkd}} + \lambda \mathcal{L}_{\text{clr}})$
    $\tau \leftarrow \tau + \Delta_\tau$
    $epoch \leftarrow epoch + 1$
  end while
  $\boldsymbol{\delta} \leftarrow \text{tg}(\boldsymbol{\delta}_0, \boldsymbol{\phi}, K)$
Output: $\theta'$,$\boldsymbol{\delta}$
```

Algorithm 3 GCBA-natural-backdoor

```
Input: $f$ (clean encoder), $\mathcal{V}_c$ (crafting set), $\widetilde{\mathcal{V}}_{[y_t]}$ (target set), $max\_epoch$, $\alpha$ (learning rate), $\boldsymbol{\delta}_0$ (base trigger attribute), $K$ (trigger size), $\Delta_\tau$ (temperature increasing factor).
Output: $\boldsymbol{\delta}$ (trigger node attribute).
  $\tau \leftarrow 1$
  $\boldsymbol{\phi} \leftarrow 0^d$
  $epoch \leftarrow 1$
  while $epoch \leq max\_epoch$ do
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha \cdot \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\text{nbkd}}$
    $\tau \leftarrow \tau + \Delta_\tau$
    $epoch \leftarrow epoch + 1$
  end while
  $\boldsymbol{\delta} \leftarrow \text{tg}(\boldsymbol{\delta}_0, \boldsymbol{\phi}, K)$
Output: $\boldsymbol{\delta}$
```

C. Experimental Details

C.1. Details of used Datasets and GCL Methods

We evaluate GCBA on five commonly used datasets: Cora, CiteSeer (Kipf & Welling, 2016), DBLP (Fu et al., 2020), BlogCatalog, and Flickr (Meng et al., 2019). The first three datasets are citation networks in which each node represents a paper in the network, edges denote the citation relationship between papers, and node features refer to the presence of words. The BlogCatalog dataset is a social network containing blog users as nodes. Edges in BlogCatalog denote users' social relationships, and node features are generated from keywords in user profiles. Flickr is also a social network with users as nodes and social connections as edges. Moreover, for each dataset, we pick one consistent target label y_t across different attacker settings. The details of used datasets are summarized in Table 5 (in Appendix).

In this paper, we use three state-of-the-art GCL methods: GRACE (Zhu et al., 2020), GCA (Zhu et al., 2021), and CCA-SSG (Zhang et al., 2021a). GRACE uses a SimCLR-like (Chen et al., 2020b) way to learn node-level representations. GCA further adopts adaptive augmentation to highlight important information. CCA-SSG requires neither negative pairs nor mutual information estimators. Using GCL methods with different architectures and implementations, we factor out the impact of varied learning strategies. Table 6 summarizes their performance in terms of accuracy on node classifications tasks. For all these GCL methods, we obey their authors' default settings presented in papers.

C.2. Metrics

We use three metrics to evaluate GCBA: *Accuracy Drop (AD)*, *Attack Success Rate (ASR)*, *Flipping Rate (FR)*.

- **AD** is the difference between the clean and backdoor downstream classifier accuracy on the clean downstream dataset. A lower AD demonstrates that the evaluated attack method does not change the encoder behavior significantly. Since we expect our attack to preserve the backdoor downstream classifier's classification accuracy compared to the clean one, a lower AD is preferred.

- **ASR** of a backdoor downstream classifier is defined as the ratio of predicting a trigger-attached node into the target class. We only attack and test one victim node at each iteration to avoid impacting neighbors of the victim node. We then sum them up to calculate the ASR following

$$\text{ASR} = \frac{\text{\# successfully attacked nodes}}{\text{\# target nodes}} \tag{14}$$

- **FR** of a backdoor downstream classifier derived from ASR. When calculating ASR, we also count in nodes that will be misclassified into the target class even if we do not execute our attack. We exclude these false positive data points to get the FR. FR can accurately reflect the effectiveness of the evaluated attack.

Table 5. The statistics of datasets.

Dataset ↓ Property →	Nodes	Edges	Classes	Features	y_t
Cora	2708	10556	7	1433	3
CiteSeer	3327	9228	6	3703	3
DBLP	17716	105734	4	1639	0
BlogCatalog	5196	343486	6	8189	4
Flickr	7575	479476	9	12047	0

Table 6. Accuracy (%±σ) of graph contrastive learning methods on node classification.

Dataset ↓ GCL →	GRACE	GCA	CCA-SSG
Cora	81.6 ± 0.5	81.9 ± 0.4	81.81 ± 0.8
CiteSeer	65.8 ± 0.7	66.5 ± 0.4	64.4 ± 0.1
DBLP	81.8 ± 0.3	82.4 ± 0.5	81.9 ± 0.3
BlogCatalog	76.1 ± 0.2	77.2 ± 0.6	82.3 ± 0.4
Flickr	47.6 ± 0.1	48.0 ± 0.3	50.1 ± 0.1

C.3. Implementation Details

We use PyTorch (Paszke et al., 2019) as the deep learning framework for implementations. In our implementation, we follow the default settings as stated in (Zhu et al., 2020; 2021; Zhang et al., 2021a). We adopt AdamW (Loshchilov & Hutter, 2017) to optimize the trigger node attribute $\boldsymbol{\delta}$ with a learning rate of 0.0015. For GCBA-poisoning and GCBA-crafting, we update the trigger and encoder for 600 epochs; for GCBA-natural-backdoor, we update the trigger for 1000 epochs. We adopt AdamW to optimize the downstream classifiers with a learning rate of 0.001 and a weight decay of 0.1. The downstream classifiers are updated for 1000 epochs. We set the base trigger pattern $\boldsymbol{\delta}_0$ in Equation 5 as the attributes of a node randomly selected from the attacker-accessible nodes. In $\boldsymbol{p} = \sigma(\tau\boldsymbol{\phi})$, we initialize $\boldsymbol{\phi}$ as 0^d. Therefore at the beginning, we have $\boldsymbol{p} = 0.5^d$ which indicates that we are uncertain about which dimension to flip. As the training proceeds, $\boldsymbol{\phi}$ is optimized to assign different flip probabilities for $\boldsymbol{p}$. Recall that we filter K biggest probabilities, so when $K = 0$, we simply inject a node already existing in the graph. Furthermore, we set the poisoning factor as 25%, which means we poison at most 25% of the trigger nodes for each compromised victim node. To train the downstream classifier, we used half of the nodes from the downstream dataset as the downstream training set and used the remaining nodes as the testing set.

Table 7. Parameter settings for the crafting and natural-backdoor adversary

Dataset ↓ Parameter →	$\#\mathcal{V}_{pre}(r_{pre})$	$\#\mathcal{V}_c$ (r_{craft})
Cora	1899 (0.7)	216 (0.08)
CiteSeer	2329 (0.7)	266 (0.08)
DBLP	8858 (0.5)	213 (0.012)
BlogCatalog	2598 (0.5)	125 (0.024)
Flickr	3788 (0.5)	182 (0.024)

D. More Experimental Results

D.1. Compare GCBA to baselines

Table 8 compares the performance of GCBA to corresponding baselines on GRACE. We can observe that for the poisoning adversary, GCBA can significantly improve the attack success rate and flipping rate. In comparison to GCBA$_{B_1}$, GCBA can improve the ASR by at least 7.1% and at most 100%. Mention that the slightest improvement in ASR is observed when the target dataset is DBLP. However, the FR of GCBA$_{B_1}$ on DBLP is only 0%, which means that GCBA$_{B_1}$ fails to flip any node to the target label. The high misclassification rate of the clean downstream classifier causes the high ASR achieved by GCBA$_{B_1}$ on DBLP. In this setting, GCBA can successfully flip nodes to the target label and improve the FR from 0% to 73.3%. In comparison to GCBA$_{B_2}$, GCBA also improves both ASR and FR. These attacks are generally ranked as GCBA $>$ GCBA$_{B_2}$ $>$ GCBA$_{B_1}$. This observation holds for the crafting adversary. Since the crafting adversary can manipulate the encoder parameters, GCBA$_{B_1}$ and GCBA$_{B_2}$ can also achieve comparable ASRs and FRs, but GCBA still outperforms baselines. For the natural backdoor adversary, we compare GCBA to unsupervised LGCB. On most datasets, GCBA significantly improves both ASRs and FRs. On CiteSeer, GCBA even improve the ASR and FR by 79.8% and 80.8%, respectively. LGCB only achieves higher ASR and FR on DBLP, but GCBA still obtains close results. Experimental results on GCA and CCA-SSG are shown in Table 9 and Table 10, respectively. Our observation is consistent with these two GCL methods. We can thus conclude that GCBA outperforms baselines under different settings.

Table 8. Comparison of baselines on GRACE.

Poisoning Adversary

Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GCBA$_{B1}$	**0.0%**	27.4%	18.2%	0.6%	22.3%	17.2%	0.0%	88.2%	0.0%	0.2%	0.0%	0.0%	0.8%	10.5%	3.7%
GCBA$_{B2}$	0.4%	90.3%	89.5%	0.2%	87.2%	86.6%	**-0.2%**	94.1%	53.3%	-0.2%	49.4%	44.8%	**-0.8%**	91.8%	91.1%
GCBA	0.2%	**96.2%**	**95.9%**	**0.2%**	**94.6%**	**94.3%**	0.0%	**95.3%**	**73.3%**	**-0.8%**	**100%**	**100%**	0.6%	**97.4%**	**97.1%**

Crafting Adversary

Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GCBA$_{B1}$	10.3%	79.9%	77.7%	5.4%	66.2%	64.8%	-3.9%	94.4%	75.3%	1.6%	75.7%	69.6%	1.6%	100%	100%
GCBA$_{B2}$	9.5%	87.0%	85.5%	**4.8%**	87.2%	86.7%	**-3.0%**	100%	100%	**-0.3%**	75.9%	74.9%	**1.1%**	100%	100%
GCBA	**8.8%**	**96.3%**	**95.8%**	5.5%	**97.3%**	**97.2%**	-3.1%	**100%**	**100%**	0.5%	**86.5%**	**83.3%**	1.6%	**100%**	**100%**

Natural Backdoor Adversary

Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
LGCB	-	85.3%	83.3%	-	17.4%	16.3%	-	**100%**	**100%**	-	60.0%	60.0%	-	100%	100%
GCBA	-	**98.5%**	**98.3%**	-	**97.2%**	**97.1%**	-	98.6%	92.3%	-	**86.5%**	**82.6%**	-	**100%**	**100%**

Table 9. Comparison of baselines on GCA.

Poisoning Adversary

Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GCBA$_{B1}$	-0.4%	14.4%	4.6%	**-1.0%**	35.4%	27.8%	0.0%	100%	100%	0.2%	7.6%	0.3%	1.0%	0.0%	0.0%
GCBA$_{B2}$	**-1.2%**	98.7%	98.6%	-0.6%	90.8%	90.1%	0.0%	100%	100%	-0.0%	65.3%	62.2%	0.2%	68.4%	65.9%
GCBA	-0.2%	**100%**	**100%**	-0.8%	**100%**	**100%**	**0.0%**	**100%**	**100%**	**-0.4%**	**90.6%**	**89.7%**	**-0.4%**	**93.9%**	**93.3%**

Crafting Adversary

Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GCBA$_{B1}$	**8.5%**	79.6%	77.4%	6.1%	81.8%	80.7%	**1.7%**	66.0%	53.2%	2.9%	62.6%	55.9%	3.2%	85.0%	84.4%
GCBA$_{B2}$	8.5%	89.1%	88.0%	6.7%	90.0%	89.5%	1.9%	87.2%	81.9%	**2.3%**	71.29%	65.7%	**0.8%**	100%	100%
GCBA	9.3%	**95.6%**	**95.1%**	**6.1%**	**97.2%**	**97.1%**	2.5%	**98.2%**	**97.5%**	2.9%	**88.9%**	**88.6%**	1.2%	**100%**	**100%**

Natural Backdoor Adversary

Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
LGCB	-	81.2%	81.0%	-	53.1%	51.0%	-	60%	60.0%	-	57.3%	47.2%	-	80.0%	80.0%
GCBA	-	**98.7%**	**98.5 %**	-	**81.6%**	**80.8%**	-	**100%**	**100%**	-	**64.5%**	**57.8%**	-	**100%**	**100%**

D.2. Impact of attack budget

Fig. 4, Fig. 5 , and Fig. 6 show our experimental results of attacks with different attack budget ϵ. Each figure contains three rows corresponding to poisoning, crafting, and natural backdoor adversary. The attack budget ϵ is varied among GCLs and datasets. In each subfigure, the hatched bars refer to attack performances under default settings.

Table 10. Comparison of baselines on CCA-SSG.

Poisoning Adversary															
Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GCBA$_{B1}$	0.2%	20.0%	9.8%	**-1.4%**	10.3%	3.8%	0.0%	94.1%	0.0%	1.0%	15.3%	1.7%	-0.2%	13.9%	0.9%
GCBA$_{B2}$	-0.4%	44.9%	40.1%	-0.2%	67.4%	65.7%	**-0.2%**	94.1%	0.0%	0.0%	82.4%	76.6%	**-0.8%**	15.5%	5.3%
GCBA	**-1.4%**	**73.3%**	**70.9%**	0.2%	**83.3%**	**81.9%**	0.0%	**98.8%**	**80.0%**	**-1.0%**	**98.8%**	**98.2%**	0.6%	**85.5%**	**84.0%**

Crafting Adversary															
Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GCBA$_{B1}$	11.1%	80.1%	78.1%	6.3%	89.7%	89.3%	-4.3%	89.5%	62.5%	6.6%	59.5%	51.2%	1.4%	98.7%	98.4%
GCBA$_{B2}$	10.1%	86.7%	85.4%	**5.0%**	95.7%	95.6%	-3.9%	83.8%	68.1%	6.6%	65.0%	56.1%	**1.2%**	100%	100%
GCBA	**9.8%**	**92.5%**	**91.8%**	5.3%	**99.2%**	**99.1%**	**-4.3%**	**97.2%**	**86.4%**	**4.2%**	**81.8%**	**76.9%**	1.4%	**100%**	**100%**

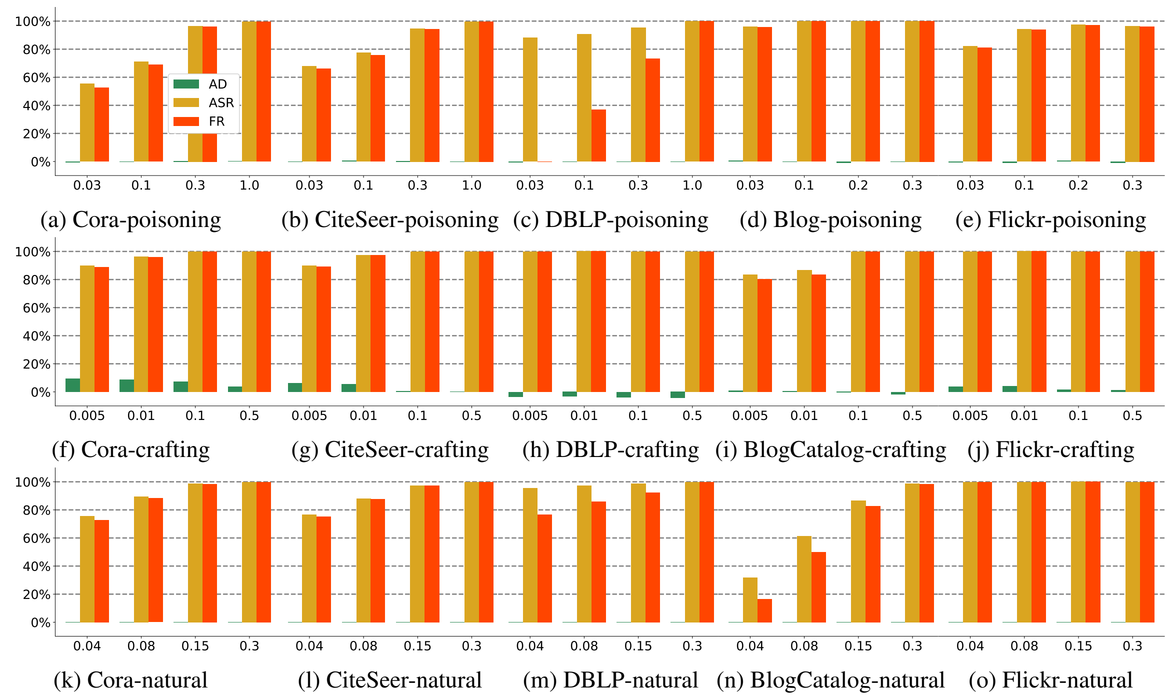
Natural Backdoor Adversary															
Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
LGCB	-	19.7%	16.9%	-	1.0%	0.0%	-	89.0%	65.2%	-	23.0%	11.4%	-	20.5%	15.3%
GCBA	-	**94.5%**	**94.0%**	-	**82.0%**	**81.3%**	-	**99.5%**	**96.2%**	-	**78.2%**	**72.0%**	-	**99.0%**	**98.6%**

We have the following observations about the impact of attack budget on our attacks. First, all three GCBA attacks achieve higher ASRs and FRs as the attack budget grows. Second, for those attacks achieving relatively low ASRs or FRs under default settings, the attack performance quickly rises as the attack budget increases. For instance, in Fig 6a, GCBA-poisoning achieves 73.3% ASR on Cora-CCA. When we enlarge ϵ to 1, we can achieve 97.7% ASR. Third, the crafting adversary can achieve a higher ASR even if the attack budget is small. This phenomenon can be explained by the crafting adversary being allowed to modify the encoder parameters. So even if the attack budget is strictly limited, the crafting adversary is less impacted compared to the other two adversaries. Last, we can observe that the accuracy drop of crafting adversary also becomes smaller as the attack budgets increase. This observation conflicts with other backdoor attacks, in which a larger attack budget (trigger size) commonly incurs a larger utility corruption. This can be attributed to the difference between our trigger design and traditional backdoor trigger design. In prior works, the backdoor adversary "covers" the target samples with the trigger pattern. For instance, in the image domain, a trigger can be a white square located at the corner of the input image; in GTA(Xi et al., 2021) a trigger is a subgraph substituting nodes in the input graph. These triggers will "delete" a part of the input, thus corrupting the semantic content of clean inputs. However, in GCBA, the trigger is designed as an attached node, which will not significantly interfere with the clean input. Moreover, with a larger attack budget, the graph encoder can easily distinguish the trigger-attached nodes from other clean nodes and map them to the target embedding. This can explain why we can obtain a smaller accuracy drop as the attack budget increases.

D.3. Impact of available dataset size

Fig. 7 shows the impact of available dataset size on GCBA. The lines represent the mean of the results and the shadow around lines are confidence intervals for five runs. We adjust the ratio of available dataset size from 10% to 150%. With a 50% ratio of available dataset size, we use half of the available dataset for attacking. In GCBA-poisoning, the available dataset refers to the set of nodes to be poisoned. The size of poisoned nodes can be given by the multiplication of r_{poison} and $\#\mathcal{V}_{[y_t]}$. Given that by default we set $r_{poison} = 0.1$, Fig 7a shows the attack performance when the r_{poison} varies from 0.01 to 0.15. Observe that a smaller or larger available dataset size does not significantly impact the attack performance.

In GCBA-crafting and GCBA-natural-backdoor, the available dataset refers to the crafting set. We, therefore, adjust r_{craft} to study the impact of crafting set size on GCBA. We have the following interesting observations. First, a smaller crafting set size will not degrade the attack success rate and flipping rate much. For instance, in Fig. 7b, even if we only use a crafting set sized 10% of the default size, GCBA-crafting can still achieve over 80% attack success rate. While in Fig. 7c, GCBA-natural-backdoor achieves nearly the same ASR with different crafting set sizes. Second, GCBA-crafting can achieve a lower accuracy drop with a smaller crafting set size. This can be explained by that GCBA-crafting has a smaller impact on the encoder parameters if the crafting set size is limited.

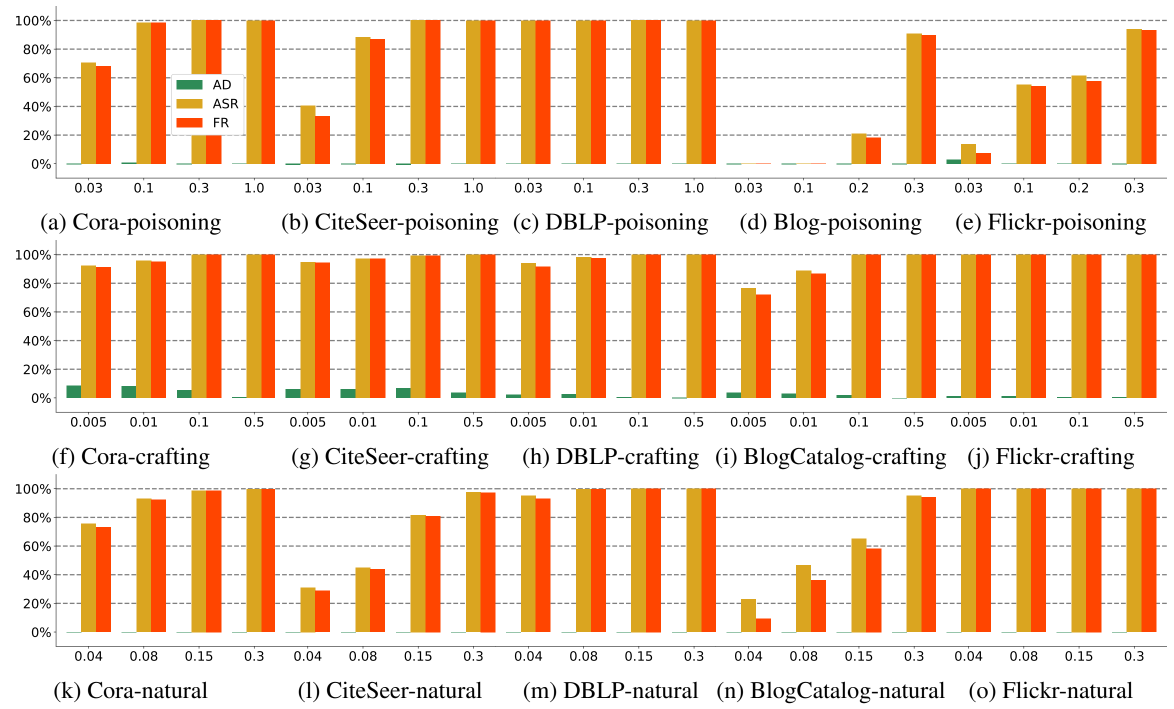


(a) Cora-poisoning (b) CiteSeer-poisoning (c) DBLP-poisoning (d) Blog-poisoning (e) Flickr-poisoning

(f) Cora-crafting (g) CiteSeer-crafting (h) DBLP-crafting (i) BlogCatalog-crafting (j) Flickr-crafting

(k) Cora-natural (l) CiteSeer-natural (m) DBLP-natural (n) BlogCatalog-natural (o) Flickr-natural

Figure 4. The impact of attack budget on GCBA, GRACE. Hatched bars refer to performances under default settings



(a) Cora-poisoning (b) CiteSeer-poisoning (c) DBLP-poisoning (d) Blog-poisoning (e) Flickr-poisoning

(f) Cora-crafting (g) CiteSeer-crafting (h) DBLP-crafting (i) BlogCatalog-crafting (j) Flickr-crafting

(k) Cora-natural (l) CiteSeer-natural (m) DBLP-natural (n) BlogCatalog-natural (o) Flickr-natural

Figure 5. The impact of attack budget on GCBA, GCA. Hatched bars refer to performances under default settings

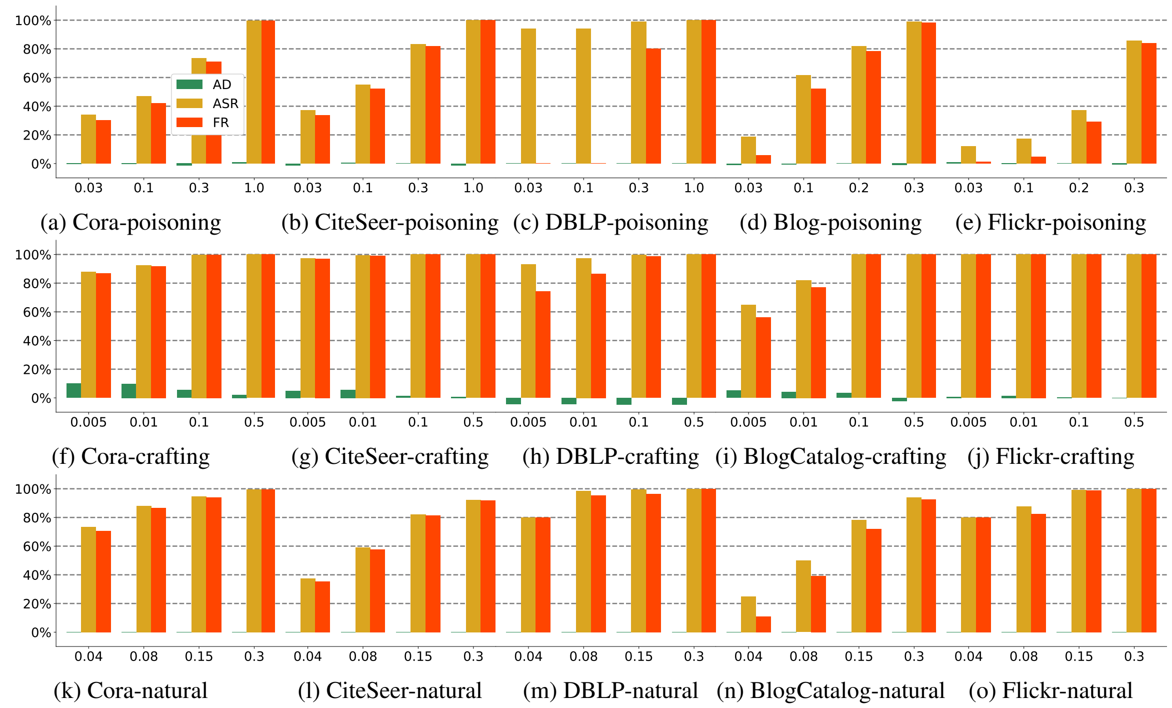


Figure 6. The impact of attack budget on GCBA, CCA-SSG. Hatched bars refer to performances under default settings

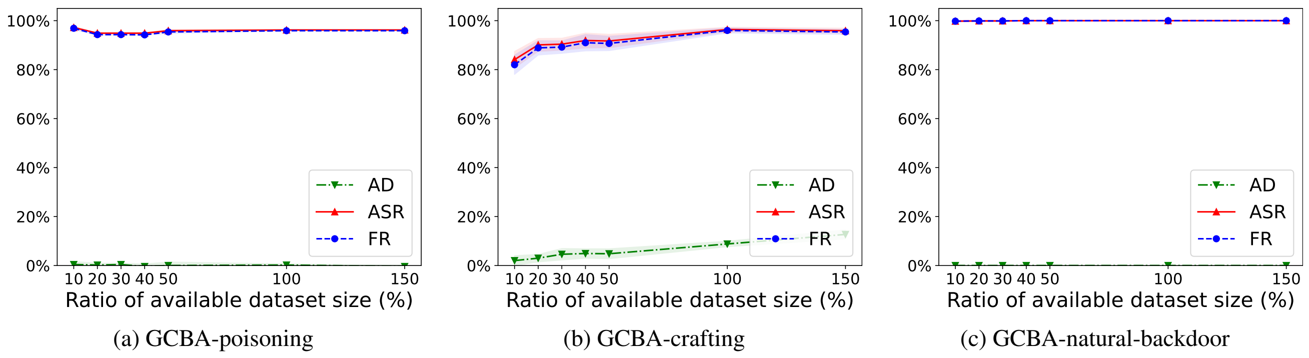


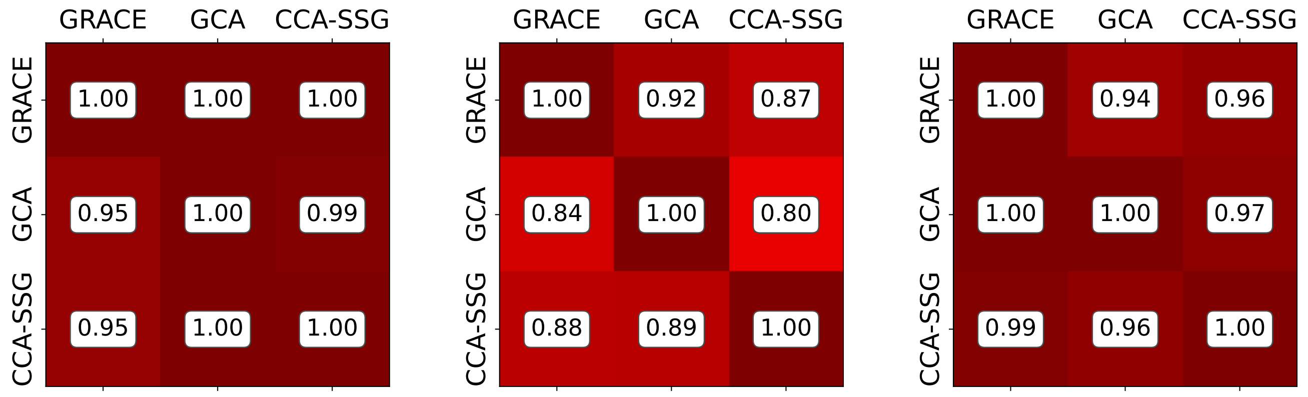
Figure 7. The impact of available dataset size on GCBA.

D.4. Case study on the transferability of GCBA

Fig. 8 forms a case study of the transferability of GCBA on Cora. We visualize our experimental results using heatmaps. Fig. 8a, 8b, 8c shows the transferability of GCBA-poisoning, GCAB-crafting, and GCAB-natural-backdoor attack respectively. The rows of heatmaps represent the source GCL method, while the columns represent the target GCL method. The values in each unit are *transfer ratio*. As shown in the figure, all three attacks are transferable across different GCL methods with high transfer ratios. Observe that GCBA-crafting achieves slightly lower ASRs after transferring among these three attacks. Two reasons may cause this. First, the crafting adversary modifies the encoder parameters, and the trigger is paired with the backdoor encoder. The trigger cannot perfectly match up with the backdoor encoder trained using the target GCL method. Second, the crafting adversary has a significantly smaller attack budget than the other two attacks.

D.5. Countermeasures

In this paper, we mainly consider three countermeasures: **Random Smoothing (RS).** For this defense, we consider the model trainer as the defender. The defender collects a graph $\mathcal{G}$ as the training dataset. During the training process, the defender randomly subsamples a set of subgraphs $\{\mathcal{G}_i\}_{i=1}^{n}$ from $\mathcal{G}$. The set of subgraphs is further used to train the graph



(a) Poisoning (b) Crafting (c) Natural Backdoor

Figure 8. The transferability of GCBA on Cora.

Table 11. Performance of GCBA on GNNGuard.

Setting	Cora			CiteSeer			DBLP			BlogCatalog			Flickr		
	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR	AD	ASR	FR
GRACE	8.6%	68.0%	66.7%	6.4%	57.5%	55.3%	1.8%	95.9%	95.5%	0.2%	88.4%	80.2%	9.1%	54.8%	53.3%
GCA	5.6%	52.8%	50.7%	8.0%	57.9%	55.7%	1.0%	100%	100%	-2.6%	61.9%	56.1%	8.6%	100%	100%
CCA-SSG	7.3%	78.8%	77.5%	3.9%	58.8%	53.7%	2.5%	98.2%	96.6%	-0.3%	87.1%	84.0%	8.9%	100%	100%

encoder. The subsampling procedure is controlled by a hyperparameter μ (subsampling ratio). In particular, if the defender sets $\mu = 0.6$, the defender will randomly remove 40% $(1 - \mu)$ nodes from the graph and mask 40% of the remaining nodes' features. The defender repeats this subsampling procedure for n times to obtain n subgraphs. The defender trains the graph encoder using these subgraphs iteratively to get a smooth graph encoder. We only apply Random Smoothing in the poisoning scenario since we consider the model trainer a Random Smoothing defender. However, in the crafting and natural backdoor scenario, the model trainer trains the encoder on a clean graph.

PreProecss. PreProcess was first proposed to mitigate adversarial attacks in the graph domain. This method can improve the robustness of GNN by removing edges connecting nodes with low feature similarity. We extend PreProcess to fit our threat model. We consider the downstream user as the defender. The PreProcess defender aims to build a downstream classifier given a pre-trained graph encoder and a downstream dataset. The defender first removes edges connecting dissimilar nodes from the downstream training dataset. Then, the defender trains the classifier on the pre-processed dataset. Finally, during the inference stage, the defender drops edges connecting nodes with low similarity in the testing set. PreProcess purifies the training set before the training stage and filters incoming malicious samples in the inference stage. Same as (Wu et al., 2019), we adopt Jaccard similarity as a measurement of the edge similarity score. Edges with a 0 similarity score will be removed from the graph.

GNNGuard. GNNGuard was proposed as a robust GNN architecture to mitigate the threats of adversarial attacks. Particularly, GNNGuard adopted neighbor importance estimation which quantifies how relevant node u is to another linked node v so that it preserved the successful routing of messages through edges. By default, GNNGuard used the cosine similarity of node attributes to measure the relevance between linked nodes. Edges with a low node similarity were assigned lower weights and contributed less to the message aggregation. GNNGuard further pruned edges with a node similarity lower than a given threshold. To smooth the evolution of edge pruning, GNNGuard also applied layer-wide graph memory, in which pruned edges were partially kept across layers. In particular, we initialize $\boldsymbol{\delta}_0$ as the victim node attributes to bypass GNNGuard. The experimental results are shown in Table 11.

E. Related Works

E.1. Backdoor Attacks

With the blossom of pre-trained models (Devlin et al., 2018; Chen et al., 2020a; Vaswani et al., 2017; Qiu et al., 2020; Khan et al., 2022; Dosovitskiy et al., 2020), backdoor attacks (Gu et al., 2017; Chen et al., 2017) have become an immense threat to the security of Deep Neural Networks (DNNs). Since being proposed, backdoor attacks and corresponding defenses have been extensively explored in the image and text domain under supervised scenarios (Wang et al., 2021; Turner et al., 2019; Saha et al., 2020; Souri et al., 2021; Li et al., 2022; Sheng et al., 2022; Qi et al., 2021; Chen et al., 2021; Wang et al., 2019; Chai & Chen, 2022; Wu & Wang, 2021; Zeng et al., 2021). Supervised backdoor adversaries commonly poison the dataset to create a connection between the trigger and the target class.

Most recently, there have been several backdoor attacks against unsupervised training (Carlini & Terzis, 2021; Jia et al., 2022; Saha et al., 2022). Carlini & Terzis (2021) proposed poisoning and backdoor attacks for multi-modal contrastive learning on (image, text) pairs. Instead of modifying the labels, they inject keywords related to the target class into image captions. BadEncoder (Jia et al., 2022) aims to forge a backdoored image encoder from a clean one. In BadEncoder, embeddings of samples in the target class are used as the supervisory information. BadEncoder forces the victim encoder to produce similar embeddings for trigger-stamped images and images in the target class. Saha et al. (2022) proposed a clean-label-like backdoor attack. The attack is achieved by guaranteeing that the trigger pattern is only added to images in the target class. However, this attack is merely effective under limited scenarios and can be easily mitigated by transferring or distillation.

E.2. Security Threats to GNNs

With the application of GNNs on security-sensitive domains, attacks and defenses on GNNs have been extensively studied in previous works (Bojchevski & Günnemann, 2019; Bojchevski et al., 2020; Geisler et al., 2020; Jin et al., 2021; Xu et al., 2019; Zhang & Zitnik, 2020; Sun et al., 2020; Zügner et al., 2018; Lin et al., 2022a). Particularly, Sun et al. (2020); Xu et al. (2019); Zügner et al. (2018) are adversarial attacks to graph-structured data. Bojchevski & Günnemann (2019) proposed the adversarial attack to early-stage unsupervised node embedding learning algorithms such as Random Walk (Perozzi et al., 2014). From the defender's side, Zhang & Zitnik (2020) and Geisler et al. (2020; 2021) proposed robust GNN backbones to defend GNN models from potential attackers by filtering out abnormal nodes from the input graph. Bojchevski et al. (2020) firstly transferred Randomized Smoothing (Cohen et al., 2019) to the graph domain. However, Bojchevski et al. (2020) cannot be directly applied to GCL since it needs to access the labels of inputs during the training stage.

In this work, we mainly focus on backdoor attacks on state-of-the-art GCL algorithms, which have not been studied in existing works. In recent years, backdoor attacks in the graph domain also attract attention from researchers. Zhang et al. (2021b) proposed to use a fixed subgraph as tFhe trigger pattern and implemented the first graph backdoor attack focusing on the graph classification task. GTA (Xi et al., 2021) introduced a target-specific trigger generator to obtain an adaptive trigger for each target sample. There were also attempts to adopt different trigger designs. Yang et al. (2022) used a structural trigger derived from the gradient score matrix for graph structure. In LGCB (Chen et al., 2022), the trigger derives from a linear approximation of target GNNs by maximizing the classification error.

However, prior works on graph backdoors only focused on attacking GNNs trained in supervised manners. They all require control over ground truth labels leading to incompatibility with graph contrastive learning paradigm.