On the Adversarial Robustness of Graph Contrastive Learning Methods

Filippo Guerranti, Zinuo Yi*, Anna Starovoit*, Rafiq Kamel*,
Simon Geisler, Stephan Günnemann
{f.guerranti, z.yi, r.kamel, a.starovoit, s.geisler, s.guennemann}@tum.de
Department of Computer Science & Munich Data Science Institute
Technical University of Munich

Abstract

Contrastive learning (CL) has emerged as a powerful framework for learning representations of images and text in a self-supervised manner while enhancing model robustness against adversarial attacks. More recently, researchers have extended the principles of contrastive learning to graph-structured data, giving birth to the field of graph contrastive learning (GCL). However, whether GCL methods can deliver the same advantages in adversarial robustness as their counterparts in the image and text domains remains an open question. In this paper, we introduce a comprehensive *robustness evaluation protocol* tailored to assess the robustness of GCL models. We subject these models to *adaptive* adversarial attacks targeting the graph structure, specifically in the evasion scenario. We evaluate node and graph classification tasks using diverse real-world datasets and attack strategies. With our work, we aim to offer insights into the robustness of GCL methods and hope to open avenues for potential future research directions.

1 Introduction

Contrastive learning (CL) (Hadsell et al., 2006; Bachman et al., 2019; Chen et al., 2020; Jaiswal et al., 2021; Liu et al., 2021; Khan et al., 2022) has emerged as a powerful framework within the field of self-supervised representation learning. Contrastive learning is primarily applied to text and images, and aims at learning latent representations by leveraging information inherent to the data. In contrastive learning, a common strategy involves generating multiple *augmentations* or views of an input entity and training the network to maximize the mutual information across these views (Bachman et al., 2019). This approach pushes the network to learn latent representations that are invariant to *data perturbations* similar to the augmentations used during training (Bachman et al., 2019). Notably, the choice of augmentations plays a pivotal role in influencing the performance of the model on downstream predictive tasks (Chen et al., 2020).

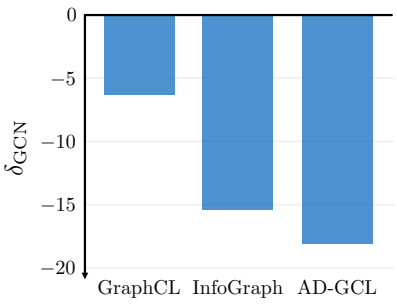

Figure 1: Average robustness improvement of graph contrastive learning methods w.r.t. the GCN baseline, across graph classification datasets. All assessed contrastive methods seem to *lower the robustness* in *evasion* scenarios, hence making the models more susceptible to adversarial attacks (see Table 1, and Equation (5)).

Despite not consistently matching the performance of fully supervised learning (SL) methods, CL exhibits remarkable potential in improving model robustness, as observed in previous works (Hendrycks et al., 2019; Shi et al., 2022). This advantage is especially pronounced when compared to the *target-driven* objective

*Equal contribution.

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

of SL methods. The benefits introduced by CL extend to various aspects of robustness, including robustness to adversarial examples (Kim et al., 2020), label corruptions (Xue et al., 2022), and distribution shifts (Shi et al., 2022).

While extensive research has investigated and demonstrated the advantages of CL in computer vision and natural language processing domains, its effectiveness in the realm of graph-structured data remain uncertain. Only recently, graph contrastive learning (GCL) methods have emerged to address this challenge (Veličković et al., 2018; Sun et al., 2019; You et al., 2020; Zhu et al., 2020, 2021; Xie et al., 2022). Analogous to CL, the choice of augmentations in GCL profoundly influences model accuracy, and researchers have introduced novel augmentation strategies to enhance the performance of GCL models on downstream tasks (Li et al., 2022; Suresh et al., 2021; Wang et al., 2022).

However, despite similarities with CL and the apparent impact of augmentations, the extent to which augmentations contribute to enhancing the robustness of learned representations in GCL remains an open question. Furthermore, works aiming to improve robustness by employing adversarially crafted augmentations often lack robustness evaluations (Suresh et al., 2021). Those that do assess robustness (Jovanović et al., 2021; Li et al., 2023) typically rely on simplistic proxies, such as random noise or transfer attacks (Zügner et al., 2018; Zügner and Günnemann, 2019), which can potentially overestimate actual robustness (Mujkanovic et al., 2022). Therefore, there exists a research gap necessitating a consistent evaluation of the robustness of GCL methods to adversarial attacks.

In this paper, we introduce the *robustness evaluation protocol* (Figure 2) designed to empirically assess the robustness of various GCL methods. These methods are evaluated against *adaptive* adversarial attacks that target the structure of graphs. Our focus include both node and graph classification tasks, particularly in the *evasion* (test time) scenario. We present our findings across multiple real-world datasets and diverse attack strategies. It is important to note that comparing the robustness of two models using an attack is a challenging task, as it essentially provides upper bounds on the worst-case perturbed robustness. However, we posit that these upper bounds provide valuable insights into the relative robustness of GCL models, even if they do not capture the entire landscape of robustness scenarios. Through extensive empirical analysis, our goal is to asses the efficacy of GCL methods in adversarial scenarios, thereby contributing to a deeper understanding of their practical utility and limitations in real-world applications. Our investigations shows the inadequacy of naive proxies often used to assess the robustness of GCL methods, highlighting how they tend to overestimate actual robustness under more realistic and adaptive attack conditions.

2 Background

Notation. We define $G = (V, E)$ as an undirected graph, where V represents the set of nodes with $n = |V|$ nodes, and E represents the set of edges with $m = |E|$ edges. In this representation, we allow for a node features matrix $\boldsymbol{X} \in \mathbb{R}^{n \times f}$ and denote the graph structure through a symmetric adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{n \times n}$. Here, $\boldsymbol{A}_{ij} = 1$ if there exists an edge between nodes i and j, and $\boldsymbol{A}_{ij} = 0$ otherwise. For node-level tasks within a given graph G, our objective is to learn the latent representation $\boldsymbol{h}_v$ for each node $v \in V$. Instead, for graph-level tasks involving a collection of graphs $\mathcal{G} = \{G_1, G_2, \dots\}$, our goal is to learn the latent representation $\boldsymbol{h}_G$ for each graph $G \in \mathcal{G}$. We let $\mathcal{P}$ denote the distribution of unlabeled graphs over the input space $\mathcal{G}$. To maintain simplicity and clarity throughout this paper, we consistently refer to both node and graph representations as $\boldsymbol{h}$.

Graph contrastive learning. Graph contrastive learning (GCL) has emerged as a framework specifically designed to train graph neural networks (GNNs) (Scarselli et al., 2009; Kipf and Welling, 2016; Gilmer et al., 2017; Bronstein et al., 2017) in a self-supervised manner. The fundamental concept behind GCL involves generating augmented views of an input graph and maximizing the *agreement* between these views pertaining to the same node or graph. This agreement is commonly quantified by the mutual information $\mathcal{I}(\boldsymbol{h}_i, \boldsymbol{h}_j) = D_{KL}(p(\boldsymbol{h}_i, \boldsymbol{h}_j) \| p(\boldsymbol{h}_i)p(\boldsymbol{h}_j))$ between a pair of representations $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, where D_{KL} represents the Kullback-Leibler divergence (Kullback and Leibler, 1951). Contrastive learning aims to maximize the mutual information between two views treated as random variables. Specifically, it trains the encoders to be pull together representations of positive pairs drawn from the joint distribution $p(\boldsymbol{h}_i, \boldsymbol{h}_j)$ and push apart representations of negative pairs derived from the product of marginals $p(\boldsymbol{h}_i)p(\boldsymbol{h}_j)$. To computationally estimate and maximize mutual information in contrastive learning, three lower bounds are commonly used (Hjelm et al., 2019): Donsker-Varadhan (DV) (Donsker and Varadhan, 1975), Jensen-Shannon (JS) (Nowozin

et al., 2016), and noise-contrastive estimation (InfoNCE) (Gutmann and Hyvärinen, 2010). A mutual information estimation is usually computed based on a discriminator $\mathcal{D} : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ that maps the representations of two views to an agreement score between them. In particular, given two representations $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ computed from the graph $G = (\boldsymbol{A}, \boldsymbol{X})$ and a discriminator $\mathcal{D}$, the Jensen-Shannon estimator is defined as follows:

$$\hat{\mathcal{I}}^{(\text{JS})}(\boldsymbol{h}_i, \boldsymbol{h}_j) = \mathbb{E}_{G\sim\mathcal{P}}[\log(\mathcal{D}(\boldsymbol{h}_i, \boldsymbol{h}_j))] + \mathbb{E}_{[G,G']\sim\mathcal{P}\times\mathcal{P}}[\log(1 - \mathcal{D}(\boldsymbol{h}_i, \boldsymbol{h}'_j))],$$

while the InfoNCE estimator can be written as:

$$\hat{\mathcal{I}}^{(\text{NCE})}(\boldsymbol{h}_i, \boldsymbol{h}_j) = \mathbb{E}_{[G,\boldsymbol{K}]\sim\mathcal{P}\times\mathcal{P}^N}\left[\log \frac{e^{\mathcal{D}(\boldsymbol{h}_i,\boldsymbol{h}_j)}}{\sum_{G'\in\boldsymbol{K}} e^{\mathcal{D}(\boldsymbol{h}_i,\boldsymbol{h}'_j)}}\right] + \log N,$$

where N is the number of negative samples, and $\boldsymbol{K}$ consists of N i.i.d. graphs sampled from $\mathcal{P}$. In practice we compute the InfoNCE over mini-batches of size $N + 1$ and minimize the following loss:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N+1}\sum_{G\in\mathcal{B}}\left[\log \frac{e^{\mathcal{D}(\boldsymbol{h}_i,\boldsymbol{h}_j)}}{\sum_{G'\in\mathcal{B}/\{G\}} e^{\mathcal{D}(\boldsymbol{h}_i,\boldsymbol{h}'_j)}}\right], \tag{1}$$

where $G \in \mathcal{B}$ is a graph in the mini-batch $\mathcal{B}$, and $\boldsymbol{h}_i, \boldsymbol{h}_j$ are representations computed from the graph. Intuitively, the optimization of Equation (1) aims at making the agreement between the representations $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ of views from the same graph G higher, while decreasing the agreement between the representation $\boldsymbol{h}_i$ of a view of graph G and the representation $\boldsymbol{h}'_j$ of a view from the N negative samples $\mathcal{B}/\{G\}$. A common contrastive loss that follows this principle is the NT-Xent loss (Sohn, 2016; Wu et al., 2018; Oord et al., 2018), which uses $\mathcal{D} = \langle g(\boldsymbol{h}_i), g(\boldsymbol{h}_j)\rangle/\tau$, where $g : \mathbb{R}^q \to \mathbb{R}^s$ projects the representations to a lower dimensional space, τ is a temperature parameter, and $\langle\cdot,\cdot\rangle$ represents the dot product.

For a description of the models considered in our evaluation, please refer to Appendix A. We also refer the reader to Xie et al. (2022, and references therein) for a comprehensive introduction to GCL.

Adversarial robustness. In the field of machine learning (ML), adversarial examples have become a significant concern (Goodfellow et al., 2014). Adversarial examples are perturbed inputs that, albeit being indistinguishable from the original input, lead to a change in the prediction of the model. Adversarial robustness (Carlini and Wagner, 2017; Madry et al., 2018), therefore, refers to the capability of an ML model to withstand such attacks and maintain accuracy, especially in important safety-critical applications. Various techniques have been developed to enhance adversarial robustness, including adversarial training, feature engineering, gradient masking, and ensemble methods (Chakraborty et al., 2018). Notably, contrastive learning has emerged as a promising approach to improve model robustness (Kim et al., 2020).

While much of the research has been focused on traditional data types like images and text, there is a growing interest in studying and enhancing the robustness of ML models applied to graph data (Dai et al., 2018; Günnemann, 2022). Graphs are used to represent complex relationships, and ensuring their robustness is critical in domains such as social network analysis, recommendation systems, and fraud detection. Addressing adversarial attacks in graph-based ML models is an evolving area, with methods tailored to graph structures, including node and edge perturbations (Zügner et al., 2018; Bojchevski and Günnemann, 2019), being actively explored to secure these systems.

Refer to Appendix B for a comprehensive description of the attacks considered in our evaluation. We also refer the reader to Günnemann (2022) for an introduction to adversarial robustness on graphs.

3 Evaluating adversarial robustness of graph contrastive learning methods

Problem setup. Our focus is on evaluating the adversarial robustness w.r.t. both structural and feature-based attacks on graph neural networks (GNNs) trained using graph contrastive learning methods for node and graph classification tasks. Specifically, we assess adversarial robustness during test time, i.e., *evasion* attacks.

3.1 Robustness evaluation protocol

Inspired by the linear evaluation protocol proposed by Veličković et al. (2019), we introduce our *robustness evaluation protocol* as illustrated in Figure 2. This protocol establishes a general yet

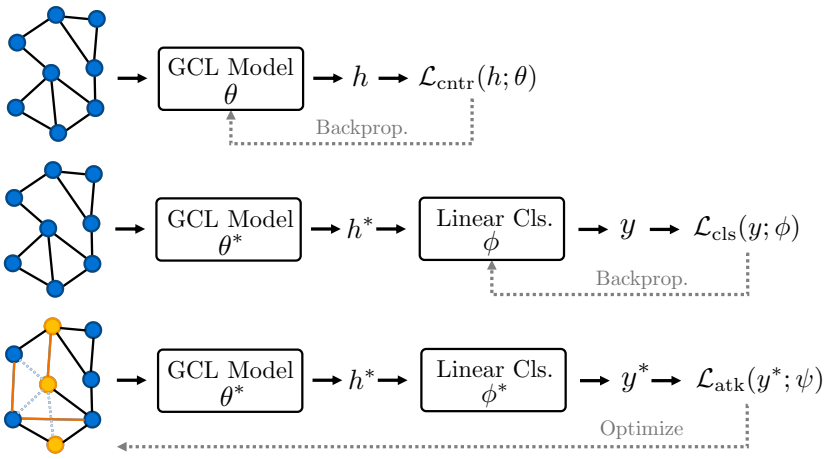

Figure 2: The three steps of our proposed *robustness evaluation protocol* for graph contrastive learning methods: (1) encoder training, (2) linear classifier training, (3) evasion attack.

consistent framework for assessing the robustness of self-supervised (or unsupervised) models against adversarial attacks across various attack schemes. The proposed robustness evaluation protocol can be applied to both node and graph classification tasks and is adaptable to different loss functions employed in the self-supervised encoder, linear classifier, and adversarial attack, respectively.

This evaluation protocol serves as a foundational structure that enables standardized and comprehensive evaluations of model robustness. By encompassing various loss functions and attack types, it ensures that the assessment is applicable to a wide range of scenarios. The evaluation consists of three steps, as described in the following paragraphs.

(Step 1) Encoder training. The initial step of the evaluation protocol is the training of the GCL encoder. The encoder, $f_\theta : \mathbb{R}^{n\times f} \times \{0,1\}^{n\times n} \to \mathcal{H}$, takes an input graph characterized by its node attributes and adjacency matrix, and outputs latent representations $\boldsymbol{h} \in \mathcal{H}$ for either the entire graph or its individual nodes, depending on the specific task it is trained on. Training the encoder is a critical phase where the model learns to capture meaningful graph structure and representation. This phase is guided by minimizing a contrastive loss function $\mathcal{L}_{\text{cntr}}(\boldsymbol{h};\theta)$, as the one defined in Equation (1). The objective is to ensure that similar instances in the graph result in representations that are close in the latent space, while dissimilar instances are pushed apart.

(Step 2) Linear classifier training. In this phase, we train a linear classifier, $g_\phi : \mathcal{H} \to \mathcal{Y}$, that takes as input the latent representations $\boldsymbol{h}^*$ generated by the encoder f_{θ^*} with learned *fixed* parameters θ^*, and outputs the downstream predictions $\boldsymbol{y} \in \mathcal{Y}$. In the classification setting, the classifier is trained by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{cls}} = -\sum_{i=1}^{n}\sum_{j=1}^{|\mathcal{Y}|} \bar{y}_{ij}\log(y_{ij}), \tag{2}$$

where $\bar{y}_{ij}$ represents the ground truth label for node i with respect to class j, and y_{ij} is the predicted probability assigned by the classifier to node i belonging to class j. It is important to highlight that the parameters θ^* of the encoder are kept fixed during this phase, and only the parameters of the linear classifier ϕ are updated, hence preserving the self-supervised nature of the encoder.

(Step 3) Adversarial attack. The last step of the evaluation protocol consists of attacking the encoder and the linear classifier. We perturb the input graph $G = (\boldsymbol{X}, \boldsymbol{A})$ by approximating:

$$\tilde{G} = (\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{A}}) = \arg\min_{\tilde{G}\in\varphi(G)} \mathcal{L}_{\text{atk}}(g_{\phi^*}(f_{\theta^*}(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{A}}))), \tag{3}$$

where $\varphi(G)$ defines the admissible perturbations that the attack optimizes over. Following the established literature on adversarial robustness of GNNs, we focus on perturbations related to the

structure of the graphs: $\varphi(G) = \{\tilde{G} \,|\, \|\tilde{\boldsymbol{A}} - \boldsymbol{A}\|_0 \leq 2\Delta \wedge \tilde{\boldsymbol{A}}^\top = \tilde{\boldsymbol{A}}\}$. This choice of $\varphi(G)$ permits up to Δ edge additions/removals while ensuring symmetry in $\tilde{\boldsymbol{A}}$, hence, that the perturbed graph is still undirected. In our study, we keep the parameters θ^* of the f_{θ^*} and the parameters ϕ^* of the g_{ϕ^*} *fixed* (evasion scenario). However, our framework naturally extends to training-time attacks (poisoning). We employ various attack methods, including random edge flips as a baseline, Projected Gradient Descent (PGD) (Xu et al., 2019), and two variants of Randomized Block Coordinate Descent (R-BCD) (Geisler et al., 2021). Furthermore, we use these attacks as *global* attacks, i.e., they attack the graph-level prediction or target the prediction of all test nodes jointly. This is in contrast to Nettack (Zügner et al., 2018), which is a *local* attack targeting individual nodes.

An important distinction in our evaluation, as opposed to prior GCL work, is that the PGD and R-BCD attacks are *adaptive* (Mujkanovic et al., 2022). That is, the attacks are fully white-box and capable of adapting to the different representations learned by GCL models, as opposed to supervised graph learning. As we will see in our empirical evaluation, employing adaptive and global attacks for assessment raises questions about the robustness benefits claimed by many GCL methods.

3.2 Measuring Adversarial Robustness

In machine learning, particularly in the context of evaluating adversarial robustness across diverse datasets and models, it is crucial to establish standardized metrics for comparative analysis.

Relative Adversarial Accuracy Drop. We introduce a new metric that provides a standardized approach for assessing adversarial robustness, enabling meaningful comparisons across different datasets and machine learning models, the *Relative Adversarial Accuracy Drop*, defined as follows:

$$\mathcal{R}_{\text{adv}}^{\text{clean}} = \mathcal{R}(\text{acc}_{\text{clean}}, \text{acc}_{\text{adv}}) = \frac{\text{acc}_{\text{clean}} - \text{acc}_{\text{adv}}}{\text{acc}_{\text{clean}}}. \tag{4}$$

$\mathcal{R}_{\text{adv}}^{\text{clean}}$ quantifies the relative reduction in accuracy between clean ($\text{acc}_{\text{clean}}$) and adversarial ($\text{acc}_{\text{adv}}$) predictions. The normalization by $\text{acc}_{\text{clean}}$ confines $\mathcal{R}$ within the interval $[0, 1]$, ensuring fair comparisons, even when models exhibit varying levels of performance on clean data. Our primary objective is to evaluate the robustness of the models rather than their absolute performance on a specific dataset. A lower value of $\mathcal{R}$ indicates higher robustness against adversarial attacks, signifying a smaller decrease in accuracy when exposed to adversarial inputs compared to clean data.

Model comparison. When comparing a model m to a *reference* model r across various datasets $\mathcal{D} = \{D_i \,|\, i = 1, \dots\}$[0], we employ the following metric:

$$\delta_r^m = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} (\mathcal{R}_m^D - \mathcal{R}_r^D). \tag{5}$$

Here, $\mathcal{R}_m^D$ and $\mathcal{R}_r^D$ represent the $\mathcal{R}_{\text{adv}}^{\text{clean}}$ of model m and the reference model r on dataset D, respectively. A positive δ_r^m indicates that model m consistently outperforms the reference model across the datasets, while a negative δ_r^m suggests the opposite. Therefore, the δ_r^m serves as a comprehensive measure of the overall comparative performance of the model of interest with respect to the reference model across diverse datasets.

4 Empirical Evaluation

Our empirical evaluation encompasses a comprehensive analysis involving multiple node and graph classification datasets, as well as a variety of graph contrastive learning (GCL) models. We measure both clean accuracy and perturbed accuracy under various *static* and *adaptive* attack scenarios and report the relative drop in accuracy compared to the clean accuracy.

4.1 Setup

In our evaluation, we use the model architectures as reported in the reference implementations. This includes the type and sequence of layers, choice of activation functions, and the application

[0]For instance, $\mathcal{D} = \{\text{PROTEINS}, \text{NCI1}, \text{DD}\}$.

Table 1: Graph classification performance: clean and perturbed accuracy mean ± std., and $\mathcal{R}_{\text{adv}}^{\text{clean}}$ (percentage), averaged over 15 runs with different weight initializations. The results include the accuracy for different contrastive and non-contrastive models, and different datasets under the attack scheme that most influences accuracy. The attacks are allowed (Δ) to change 5% of edges. Models achieving the three highest relative drop are highlighted as **first**, second, and *third*. The higher the worse.

Model		PROTEINS		NCI1		DD	
GCN	CLEAN	73.98 ± 1.00	$\downarrow$ 8.04	74.67 ± 0.80	$\downarrow$ 54.59	70.02 ± 0.54	$\downarrow$ **87.57**
	ATTACK	68.04 ± 0.90		33.91 ± 1.36		8.58 ± 7.35	
GIN	CLEAN	66.02 ± 4.60	$\downarrow$ 30.24	76.04 ± 0.91	$\downarrow$ 49.17	63.44 ± 3.19	$\downarrow$ *74.51*
	ATTACK	46.05 ± 6.86		38.65 ± 2.48		16.17 ± 6.16	
InfoGraph	CLEAN	63.93 ± 6.43	$\downarrow$ 50.83	66.87 ± 3.72	$\downarrow$ 78.10	64.77 ± 2.41	$\downarrow$ 68.92
	ATTACK	31.53 ± 7.67		14.71 ± 4.13		20.11 ± 4.24	
GraphCL	CLEAN	65.93 ± 4.02	$\downarrow$ *38.89*	73.09 ± 3.50	$\downarrow$ *55.94*	68.85 ± 4.83	$\downarrow$ 75.34
	ATTACK	40.29 ± 9.61		32.20 ± 3.48		15.74 ± 7.66	
AD-GCL	CLEAN	73.66 ± 1.36	$\downarrow$ **64.28**	71.79 ± 0.94	$\downarrow$ **80.01**	76.71 ± 2.07	$\downarrow$ 61.71
	ATTACK	26.31 ± 4.33		14.35 ± 1.34		29.17 ± 5.27	

of dropout. Given that our primary objective is to assess the extent to which GCL methods are affected by adversarial attacks, we do *not* perform hyperparameter tuning. Instead, we employ the best hyperparameters for each model as documented in their original implementations. We utilize the Adam optimizer (Kingma and Ba, 2014) across all models, and set the learning rate as defined in the reference papers (see Appendix E). Our results are averaged over 15 different initializations of the GCL encoder and linear classifier. We follow the robustness evaluation protocol, introduced in Section 3.1, for all contrastive-based models. For non-contrastive models, such as GCN (Kipf and Welling, 2016) and GIN (Xu et al., 2018), we train the network in a supervised manner, and then apply Step 3 of the protocol. Additionally, we compute the *Relative Adversarial Accuracy Drop* $\mathcal{R}_{\text{adv}}^{\text{clean}}$ (Section 3.2) and identify the minimum value across different attack schemes.

Graph classification. In the context of graph classification, we are given a dataset $\mathcal{D} = \{(G_i, y_i) \,|\, i = 1, \ldots, g\}$ that comprises a collection of g graphs G_i along with their respective labels y_i. The central objective of this task is to assign *entire* graphs to specific classes. To evaluate the performance of GCL models in this task, we employ three benchmark datasets: PROTEINS, NCI1, and DD (Morris et al., 2020). Table 3 in Appendix D reports detailed description of the dataset statistics. To ensure a consistent evaluation process across all datasets, we employ a random data split strategy, allocating 80% of the nodes for training and reserving the remaining 20% for testing. We perform mini-batch training with batch size of 64.

Node classification. In semi-supervised node classification, we are given a graph $G = (V, E)$, with $V = \{v_i \,|\, i = 1, \ldots, n\}$ and we assume only a subset of nodes $\tilde{V} \subset V$ are labeled, with node labels y_j for $j \in \tilde{V}$. The aim is to assign unlabeled nodes *within* a graph to specific classes. We conduct experiments on well-established node classification datasets, namely Cora, Citeseer Sen et al. (2008), Pubmed (Namata et al., 2012). We also include in our analysis the *large-scale* graph OGB-arXiv (Hu et al., 2020) (see Table 4 for a description of the dataset statistics). We adhere to the data splits proposed in (Yang et al., 2016) for Cora, Citeseer, and Pubmed, while opting for a random split in the case of OGB-arXiv. The random split allocates 80% of nodes for training and reserves the remaining 20% for testing. We perform full-batch training, using all nodes in the training set at every epoch.

4.2 Discussion

The results for graph and node classification tasks are presented in Tables 1 and 2. These tables report the model accuracies under two conditions: (*i*) clean, unperturbed data and (*ii*) data subjected to various adversarial attack schemes. Specifically, we report the minimum accuracy achieved across different attack schemes for each dataset and model. A more detailed report of the results across various attack schemes can be found in Tables 5 and 6. We emphasize the models that exhibit the worst three $\mathcal{R}_{\text{adv}}^{\text{clean}}$ values.

Table 2: Node classification performance: clean and perturbed accuracy mean ± std., and $\mathcal{R}_{\text{adv}}^{\text{clean}}$ (percentage), averaged over 15 runs with different weight initializations. The results include the accuracy for different contrastive and non-contrastive models, and different datasets under the attack scheme that most influences accuracy. The attacks are allowed (Δ) to change 5% of edges. Models achieving the three highest relative drop are highlighted as **first**, second, and *third*. The higher the worse.

Model		CORA		CITESEER		PUBMED		OGB-ARXIV	
GCN	CLEAN	77.57 ± 1.11	$\downarrow$ 23.50	63.99 ± 1.27	$\downarrow$ 25.46	75.31 ± 0.80	$\downarrow$ *24.31*	52.39 ± 0.79	$\downarrow$ **82.67**
	ATTACK	59.35 ± 1.70		47.69 ± 2.06		57.01 ± 1.28		9.08 ± 2.10	
DGI	CLEAN	83.24 ± 1.37	$\downarrow$ 9.08	72.91 ± 1.99	$\downarrow$ 7.47	81.46 ± 0.79	$\downarrow$ 5.23	60.18 ± 0.72	$\downarrow$ *11.45*
	ATTACK	75.69 ± 1.86		67.46 ± 2.13		77.19 ± 0.92		53.28 ± 0.69	
GraphCL	CLEAN	71.99 ± 1.35	$\downarrow$ **24.00**	59.57 ± 1.45	$\downarrow$ *21.89*	74.29 ± 1.75	$\downarrow$ 27.33	52.32 ± 0.22	$\downarrow$ 72.65
	ATTACK	54.71 ± 1.46		46.53 ± 1.87		53.99 ± 1.77		14.31 ± 0.19	
GCA	CLEAN	79.07 ± 1.36	$\downarrow$ *22.08*	60.14 ± 2.24	$\downarrow$ **27.51**	78.62 ± 0.98	$\downarrow$ **27.80**	OOM	
	ATTACK	61.63 ± 3.18		43.61 ± 2.13		56.77 ± 1.69			

In our analysis of graph classification, we observe that the GCL methods used in our evaluation (InfoGraph (Sun et al., 2019), GraphCL (You et al., 2020), AD-GCL (Suresh et al., 2021)) do not succeed in enhancing adversarial robustness. Notably, on PROTEINS and NCI1 datasets, these three GCL models consistently achieve the first, second, and third worst results, indicating a significant drop in accuracy under adversarially perturbed data compared to non-contrastive models like GCN and GIN. The DD dataset, however, shows slightly different results, with GCN being the weakest model in terms of robustness, while GraphCL, a GCL model, takes the second position.

Moving to node classification experiments, the results are mixed. GCL models (DGI (Veličković et al., 2018), GraphCL (You et al., 2020), GCA (Zhu et al., 2021)) generally perform as the worst or second-worst models across Cora, Citeseer, and Pubmed datasets. However, they show improved robustness when compared to GCN on the large-scale OGB-arXiv dataset. An interesting observation is that GCA goes out of memory (OOM) on the OGB-arXiv dataset due to the high complexity of loss computation. It is worth noting that DGI consistently performs the best across all datasets. This behavior might be attributed to DGI's particular training strategy, which involves creating corruptions of the original graph and forcing the network to recognize them as different, as opposed to many contrastive methods that are based on identifying different views as similar. A more in-depth analysis of this behavior remains a direction for future research.

When analyzing the impact of *adaptive* versus *static* attacks, and *global* versus *local* attacks, it becomes evident from both Table 5 and Table 6 that adaptive attacks generate perturbed graph structures that are more detrimental to every model. This reinforces our argument that static attacks, such as random flipping, or local attacks like Nettack (Zügner et al., 2018), are insufficient for evaluating the robustness of a method.

In summary, our findings indicate that in both graph and node classification tasks, GCL methods do not show a clear advantage in terms of improving the adversarial robustness of graph neural networks. In some instances, contrastive training can even lead to a further deterioration in performance.

5 Conclusion

We thoroughly evaluated the robustness of graph contrastive learning (GCL) methods to adaptive adversarial attacks on graph structures. Our investigation included node and graph classification tasks across multiple real-world datasets and various attack strategies.

Our results show that GCL methods do *not* consistently exhibit improved adversarial robustness compared to non-contrastive methods. In specific datasets and attack scenarios, GCL models perform even worse in robustness. This finding challenges the common belief that CL methods, successful in other domains, automatically translate to enhanced robustness in graph-structured data.

One notable discovery is the varying performance of GCL models, with DGI consistently outperforming others in node classification tasks. This behavior may be attributed to DGI's unique training strategy, differentiating between corruptions and the original graph instead of maximizing mutual

information between augmentations. This aspect deserves further investigation to comprehend its implications for adversarial robustness.

In conclusion, our findings suggest that while GCL methods hold promise for representation learning on graph-structured data, they do not inherently guarantee improved adversarial robustness. The effectiveness of GCL methods in adversarial scenarios is nuanced and context-dependent, necessitating further research to uncover the precise conditions under which they excel or falter. Exploring additional models, datasets, and a broader range of adversarial attacks should be considered in future research to comprehensively understand GCL robustness in practice.

Acknowledgments

This work was supported by the German Federal Ministry of Education and Research (BMBF) (HOPARL, 031L0289C). The authors of this work take full responsibility for its content.

References

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning Representations by Maximizing Mutual Information Across Views. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Aleksandar Bojchevski and Stephan Günnemann. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 695–704. PMLR, May 2019.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, July 2017.

Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE Computer Society, May 2017.

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial Attacks and Defences: A Survey. *arXiv preprint arXiv:1810.00069*, September 2018.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607. PMLR, November 2020.

Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial Attack on Graph Structured Data. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1115–1124. PMLR, July 2018.

Monroe D. Donsker and S. R. Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of Graph Neural Networks at Scale. In *Advances in Neural Information Processing Systems*, volume 34, pages 7637–7649. Curran Associates, Inc., 2021.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272. PMLR, July 2017.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, December 2014.

Lukas Gosch, Simon Geisler, Daniel Sturm, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Adversarial Training for Graph Neural Networks: Pitfalls, Solutions, and New Directions. In *Advances in Neural Information Processing Systems*, volume 36, pages 8954–8968. Curran Associates, Inc., 2023a.

Lukas Gosch, Daniel Sturm, Simon Geisler, and Stephan Günnemann. Revisiting Robustness in Graph Machine Learning. In *International Conference on Learning Representations*, 2023b.

Stephan Günnemann. Graph Neural Networks: Adversarial Robustness. In Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao, editors, *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 149–176. Springer Nature, Singapore, 2022.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304. PMLR, May 2010.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, June 2006.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, volume 32, pages 15663–15674. Curran Associates Inc., December 2019.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems*, volume 33, pages 22118–22133. Curran Associates, Inc., 2020.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A Survey on Contrastive Self-Supervised Learning. *Technologies*, 9(1):2, March 2021.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

Nikola Jovanović, Zhao Meng, Lukas Faber, and Roger Wattenhofer. Towards Robust Graph Contrastive Learning. *arXiv preprint arXiv:2102.13085*, February 2021.

Adnan Khan, Sarah AlBarri, and Muhammad Arslan Manzoor. Contrastive Self-Supervised Learning: A Survey on Different Architectures. In *2022 2nd International Conference on Artificial Intelligence (ICAI)*, pages 1–6, March 2022.

Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial Self-Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 2983–2994. Curran Associates, Inc., 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, December 2014.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, November 2016.

Solomon Kullback and Richard A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Disentangled Graph Contrastive Learning With Independence Promotion. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2022.

Wen-Zhi Li, Chang-Dong Wang, Jian-Huang Lai, and Philip S. Yu. Towards Effective and Robust Graph Contrastive Learning with Graph Autoencoding. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2023.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. Self-Supervised Learning: Generative or Contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876, June 2021.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*, February 2018.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, July 2020.

Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are Defenses for Graph Neural Networks Robust? In *Advances in Neural Information Processing Systems*, volume 35, pages 8954–8968. Curran Associates, Inc., December 2022.

Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven Active Surveying for Collective Classification. In *Workshop on Mining and Learning with Graphs, International Conference on Machine Learning*, volume 8, 2012.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. F-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Aaron Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*, July 2018.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, January 2009.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI Magazine*, 29(3):93–93, September 2008.

Yuge Shi, Imant Daunhawer, Julia E. Vogt, Philip H. S. Torr, and Amartya Sanyal. How Robust is Unsupervised Representation Learning to Distribution Shift? *arXiv preprint arXiv:2206.08871*, December 2022.

Kihyuk Sohn. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *International Conference on Learning Representations*, September 2019.

Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial Graph Augmentation to Improve Graph Contrastive Learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 15920–15933. Curran Associates, Inc., 2021.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5, April 2015.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint arXiv:physics/0004057*, April 2000.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, February 2018.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep Graph Infomax. In *International Conference on Learning Representations*, 2019.

Haonan Wang, Jieyu Zhang, Qi Zhu, and Wei Huang. Augmentation-Free Graph Contrastive Learning with Performance Guarantee. *arXiv preprint arXiv:2204.04874*, June 2022.

Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised Feature Learning via Non-parametric Instance Discrimination. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, June 2018.

Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-Supervised Learning of Graph Neural Networks: A Unified Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2412–2429, April 2022.

Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3961–3967. International Joint Conferences on Artificial Intelligence Organization, August 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, September 2018.

Yihao Xue, Kyle Whitecross, and Baharan Mirzasoleiman. Investigating Why Contrastive Learning Benefits Robustness against Label Noise. In *Proceedings of the 39th International Conference on Machine Learning*, pages 24851–24871. PMLR, June 2022.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 40–48. PMLR, June 2016.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep Graph Contrastive Representation Learning. *arXiv preprint arXiv:2006.04131*, July 2020.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph Contrastive Learning with Adaptive Augmentation. In *Proceedings of the Web Conference 2021*, WWW '21, pages 2069–2080, New York, NY, USA, April 2021. Association for Computing Machinery.

Daniel Zügner and Stephan Günnemann. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations*, May 2019.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 2847–2856, New York, NY, USA, July 2018. Association for Computing Machinery.

Appendix

A Graph contrastive learning models

Deep graph infomax (DGI) (Veličković et al., 2018) is a self-supervised approach for learning node representations in graph-structured data. It aims to maximize mutual information between patch representations and high-level graph summaries. DGI does not rely on random walk objectives, making it suitable for both transductive and inductive learning.

In the single-graph setup, the authors propose to corrupt the input graph $(\boldsymbol{X}, \boldsymbol{A})$ with a *corruption function* $\mathcal{C}$, and obtain a negative sample $(\tilde{\boldsymbol{X}}, \boldsymbol{A}) \sim \mathcal{C}(\boldsymbol{X}, \boldsymbol{A})$. They then use a trainable encoder $\mathcal{E}$ for both clean and corrupted input to obtain patch representations $\boldsymbol{h}_i$ and $\tilde{\boldsymbol{h}}_i$ for each node i in the clean and corrupted graph, respectively: $\boldsymbol{H} = \mathcal{E}(\boldsymbol{X}, \boldsymbol{A}) = \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\}$, and $\tilde{\boldsymbol{H}} = \mathcal{E}(\boldsymbol{X}, \boldsymbol{A}) = \{\tilde{\boldsymbol{h}}_1, \ldots, \tilde{\boldsymbol{h}}_M\}$. The encoder is a one-layer graph convolutional network (GCN) (Kipf and Welling, 2016), but the framework is general enough to use different models.The patch representations of the input graph are then used to create a summary of the graph itself $\boldsymbol{s} = \mathcal{R}(\boldsymbol{H}) = \sigma((\sum_{i=0}^{N} \boldsymbol{h}_i)/N)$, where $\mathcal{R}$ is a non-linear readout function. Patch representations coming from the input graph and those coming from the corrupted graph are then passed through a discriminator function $\mathcal{D}$ that scores summary-patch representation pairs by applying a simple bilinear scoring function. Scores are then converted to probabilities of $(\boldsymbol{h}_i, \boldsymbol{s})$ being a positive example, and $(\tilde{\boldsymbol{h}}_i, \boldsymbol{s})$ being a negative example. The whole network is trained end-to-end via cross-entropy, with label $+1$ assigned to positive examples, and -1 assigned to negative ones.

InfoGraph (Sun et al., 2019) learns graph-level representations by maximizing the mutual information between the representation of the whole graph and the representations of substructures of different scales (e.g., nodes, edges, triangles).

Given a set of N training graphs $\{G_j \in \mathcal{G}\}_{j=1}^{N}$ with empirical probability distribution $\mathbb{P}$ on the input space, InfoGraph behaves similarly to DGI, as it learns patch representations $\boldsymbol{h}_j^i$ for each node i in graph G_j and uses them to learn a representation for the whole graph $\boldsymbol{s}_j = \mathcal{R}(\boldsymbol{H}_j)$ through a readout function $\mathcal{R}$. The network is trained by maximization of the mutual information (MI) as:

$$\theta^* = \arg\max_{\theta} \sum_{G_j \in \mathcal{G}} \frac{1}{|G_j|} \sum_{i \in G_j} \text{MI}(\boldsymbol{h}_j^j, \boldsymbol{s}_j)$$

The authors use the Jensen-Shannon MI estimator (Nowozin et al., 2016):

$$\text{MI}(\boldsymbol{h}_j^j, \boldsymbol{s}_j) := \mathbb{E}_{\mathbb{P}}\left[-\sigma\left(-T(\boldsymbol{h}_j^i, \boldsymbol{s}_j)\right)\right] - \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\sigma\left(T(\tilde{\boldsymbol{h}}_j^i, \boldsymbol{s}_j)\right)\right],$$

where σ is the softplus function, $\boldsymbol{h}_j^i \sim \mathbb{P}$ is an input sample, $\tilde{\boldsymbol{h}}_j^i \sim \tilde{\mathbb{P}} = \mathbb{P}$ is a "negative" sample generated using all possible combinations of global and local patch representations across all graph instances in a batch, and T is a discriminator. Since $\boldsymbol{s}_j$ is encouraged to have high MI with patches that contain information at all scales, this favors encoding aspects of the data that are shared across patches and aspects that are shared across scales. Differently from DGI (Veličković et al., 2018), InfoGraph uses GIN (Xu et al., 2018) as graph convolutional encoders, and use the sum as the aggregator in $\mathcal{R}$ instead of the mean.

GraphCL (You et al., 2020) learns representation of graph data by proposing four different types of graph-level data augmentations techniques, namely node dropping, edge perturbation, attribute masking and subgraph sampling.

Given a set of N graphs $\{G_j \in \mathcal{G}\}_{j=1}^{N}$, the authors formulate the augmented graph $\hat{G} \sim q(\hat{G}|G)$, where $q(\cdot|G)$ is the augmentation distribution conditioned on the original graph. As common in contrastive learning, the authors propose to generate two augmented views $\hat{G}_i \sim q_i(\cdot|G)$ and $\hat{G}_j \sim q_j(\cdot|G)$. A GNN-based encoder f extracts graph-level representations vectors $\boldsymbol{h}_i, \boldsymbol{h}_j$ for the augmented graphs $\hat{G}_i, \hat{G}_j$, and a non-linear transformation g projects the representations to another latent space where the contrastive loss is computed. The parameters of the network are optimized via minimization of the NT-Xent loss Equation (1). The authors claim that GraphCL boost adversarial robustness, but their evaluation is only based on synthetic dataset, hence not capturing the complexity of real-world scenarios.

In **GCA**, Zhu et al. (2021) revisit the concept of augmentations in graph contrastive learning by arguing that data augmentation schemes should preserve intrinsic structures and attributes of graphs,

which will force the model to learn representations that are insensitive to perturbation on unimportant nodes and edges. They propose a novel graph contrastive representation learning method with adaptive augmentation that incorporates various priors for topological and semantic aspects of the graph.

For *topology-level augmentations*, the authors propose to corrupt the input graph by randomly removing edges. They sample a modified edge set $\tilde{E}$ from the original set of edges E as $P\{(u,v)\tilde{E}\} = 1 - \rho_{uv}$, where $(u,v) \in E$ and ρ_{uv} is the probability of removing (u,v). ρ_{uv} should reflect the importance of edge (u,v) in the graph, hence augmentations are more likely to corrupt unimportant edges while keeping important ones. The authors propose to compute ρ_{uv} based on (i) *degree centrality*, (ii) *eigenvector centrality*, and (iii) *PageRank centrality*.

Regarding *node-attribute-level* augmentations, the authors perform random node attributes masking. Each node feature vector is perturbed as $\tilde{\boldsymbol{x}} = \boldsymbol{x} \circ \boldsymbol{m}$, where $\boldsymbol{m} \in \{0,1\}^d$, $m_i \sim \text{Bernoulli}(1-\rho_i)$. Similarly to ρ_{uv}, ρ_i should capture the importance of the $i-th$ feature dimension. To do so, they define specific weights for each feature. The training procedure follows the same pipeline as the one introduced by (You et al., 2020), where the types of augmentations are the only difference.

In **AD-GCL**, Suresh et al. (2021) argue that related works in GCL are based on pre-determined data augmentation strategies that may capture redundand information about the graph. The author propose to optimize adversarial graph augmentation strategies used in GCL by minimization of the information bottleneck (Tishby et al., 2000; Tishby and Zaslavsky, 2015).

Given a graph $G \in \mathcal{G}$, $T(G)$ denotes a graph data augmentation of G, which is a distribution defined over $\mathcal{G}$ and conditioned on G. $t(G) \in \mathcal{G}$ is a sample of $T(G)$. Let $\mathcal{T}$ denote a family of different graph data augmentations $T_\phi(\cdot)$, with $T_\phi(\cdot)$ being a specific augmentation scheme with parameters ϕ. AD-GCL optimizes the following objective over a graph data augmentation family $\mathcal{T}$:

$$\min_{T \in \mathcal{T}} \max_{f} \text{MI}\left(f(G); f(t(G))\right), \quad \text{where } G \sim \mathbb{P}_{\mathcal{G}}, t(G) \sim T(G),$$

where f is a graph neural network encoder. Compared to the two graph augmentations usually adopted in GCL, AD-GCL views the original graph G as the anchor while pushing its perturbation $T(G)$ as far from the anchor as it can. The automatic search over $T \in \mathcal{T}$ saves a great deal of effort evaluating different combinations of graph augmentations.

For each graph $G = (V, E)$, the sample $t(G) \sim T_\phi(\cdot)$ is a graph that shares the same node set with G while the edge set of $t(G)$ is only a subset of E. Each edge $e \in E$ is associated with a random variable $p_e \sim \text{Bernoulli}(\omega_e)$, where $e \in t(G)$ if $p_e = 1$, and is dropped otherwise. To train $T(G)$ in an end-to-end fashion, the Bernoulli weights ω_e are parameterized through a GNN *augmenter*. and the discrete p_e are relaxed to be continuous variables in $[0,1]$. Gumbel-Max reparametrization trick (Maddison et al., 2017; Jang et al., 2017) is use to perform sampling.

B Adversarial attacks

In our evaluation, we consider various *adaptive* gradient-based adversarial attack schemes, such as PGD (Xu et al., 2019), PR-BCD and GR-BCD (Geisler et al., 2021), and *static* adversarial attacks as simple baselines. While we only consider a single global budget Δ, it is straightforward to include more sophisticated constraints when beneficial for the application at hand (Gosch et al., 2023b,a).

Random Edge Flipping is a straightforward attack scheme in which edges are randomly flipped with a given perturbation probability ρ. Formally, this can be expressed as $\tilde{A}_{ij} = \text{flip}(A_{ij}, \rho)$, where A_{ij} represents the edge (i,j) in the original adjacency matrix of the graph, and $\tilde{A}_{ij}$ is the perturbed edge.

Projected Gradient Descent (PGD) (Xu et al., 2019) uses continuous relaxation to approximate Equation (3). Specifically, the adjacency is relaxed from $\{0,1\}^{n \times n} \to [0,1]^{n \times n}$. Due to this relaxation, first-order methods can be applied as long as the targeted model can handle weighted edges (i.e., $g_{\phi^*}(f_{\theta^*}(\boldsymbol{X}, \tilde{\boldsymbol{A}}))$). Following, gradient descent is used with an additional projection to ensure that the budget is not exceeded and that the value range $[0,1]$ is not violated. The last step of the attack, then, discretizes the perturbed adjacency $[0,1]^{n \times n} \to \{0,1\}^{n \times n}$.

Projected Randomized Block Coordinate Descent (PR-BCD) (Geisler et al., 2021) works similar to the PGD attack of Xu et al. (2019), while avoiding its quadratic space complexity. The quadratic memory requirement arises from the fact that $\boldsymbol{A}$ has up to n^2 non-zero entries and PGD (typically)

Table 3: Graph classification datasets.

Dataset	# Graphs	# Nodes (min, max, median)	# Edges (min, max, median)
DD	1178	(30, 5748, 241)	(126, 28534, 1220)
NCI1	3327	(3, 111, 27)	(4, 238, 58)
PROTEINS	19717	(4, 620, 26)	(10, 2098, 98)

Table 4: Node classification datasets.

Dataset	# Nodes	# Edges	# Features	# Classes
Planetoid-Cora	2708	10556	1433	7
Planetoid-Citeseer	3327	9104	3703	6
Planetoid-Pubmed	19717	88648	500	3
OGB-arXiv	169343	1166243	128	40

optimizes over all of them. To circumvent the quadratic cost, PR-BCD performs the gradient update only for a random and non-contiguous block of entries in $\boldsymbol{A}$ at a time. Thereafter, in a survival-of-the-fittest manner, the random block is resampled. Specifically, relevant entries are kept in the next block, and the remainder is discarded as well as resampled. This way, the additional space complexity is linear in the random block size. In other words, PR-BCD uses randomization in the gradient update to obtain scalability. Geisler et al. (2021) apply the PR-BCD attack to graphs of up to 100 million nodes.

Greedy Randomized Block Coordinate Descent (GR-BCD) (Geisler et al., 2021) works similar to PR-BCD, with the notable exception of pursuing a greedy objective. First, the budget Δ is distributed over the desired number of greedy updates. Then, in each greedy update, the desired amount of edges in the randomly drawn block is flipped. The edges to be flipped are chosen based on the gradient $\nabla \mathcal{L}_{\text{atk}}(g_{\phi^*}(f_{\theta^*}(\boldsymbol{X}, \tilde{\boldsymbol{A}})))$. Due to the greediness, GR-BCD does not require the random block size to be larger than Δ, and, thus, GR-BCD is even more scalable than PR-BCD.

C Datasets

For graph classification, we utilize three datasets of graphs: PROTEINS, NCI1, and DD (Morris et al., 2020). For node classification, we conduct experiments on well-established node classification datasets, namely Cora, Citeseer Sen et al. (2008), Pubmed (Namata et al., 2012). We also include in our analysis the *large-scale* graph OGB-arXiv (Hu et al., 2020).

Tables 3 and 4 report the statistics for the datasets used in our empirical evaluation, for graph and node classification, respectively.

D Complete results

Here we report the extensions of Tables 1 and 2 to different attack schemes. As a general observation, we can notice that *adaptive* adversarial attacks (PGD, R-BCD), increase the drop in accuracies of the models in every dataset, compared to static attacks (random edge flipping).

Table 5: Graph classification performance: clean and perturbed accuracy mean $\pm$ std., and $\mathcal{R}_{\text{adv}}^{\text{clean}}$ (percentage), averaged over 15 runs with different weight initializations. Results include accuracy for different contrastive and non-contrastive models under various attack schemes and datasets. The attacks are allowed (Δ) to change 5% of edges. Models achieving the three highest relative drop are highlighted as **first**, **second**, and **third**.

Model	Attack	PROTEINS		NCI1		DD	
GCN	CLEAN	73.98 ± 1.00		74.67 ± 0.80		70.02 ± 0.54	
	RAND. EDGE FLIP.	73.41 ± 0.97	($\downarrow$ 0.78)	68.12 ± 1.29	($\downarrow$ 8.77)	60.29 ± 2.23	($\downarrow$ 13.98)
	PGD	68.68 ± 0.86	($\downarrow$ 7.16)	52.69 ± 1.05	($\downarrow$ 29.43)	51.21 ± 2.75	($\downarrow$ 26.85)
	PR-BCD	68.04 ± 0.90	($\downarrow$ 8.04)	33.91 ± 1.36	($\downarrow$ 54.59)	8.58 ± 7.35	($\downarrow$ 87.57)
	GR-BCD	72.50 ± 1.47	($\downarrow$ 2.00)	49.47 ± 4.21	($\downarrow$ 33.75)	57.33 ± 2.56	($\downarrow$ 18.24)
	MIN	68.04 ± 0.90	($\downarrow$ 8.04)	33.91 ± 1.36	($\downarrow$ 54.59)	8.58 ± 7.35	($\downarrow$ **87.57**)
GIN	CLEAN	66.02 ± 4.60		76.04 ± 0.91		63.44 ± 3.19	
	RAND. EDGE FLIP.	59.71 ± 4.63	($\downarrow$ 9.55)	54.47 ± 1.56	($\downarrow$ 28.37)	56.73 ± 3.61	($\downarrow$ 10.57)
	PGD	61.12 ± 4.76	($\downarrow$ 7.42)	71.59 ± 0.83	($\downarrow$ 5.86)	58.23 ± 3.23	($\downarrow$ 8.21)
	PR-BCD	48.34 ± 6.74	($\downarrow$ 26.77)	38.65 ± 2.48	($\downarrow$ 49.17)	16.17 ± 6.16	($\downarrow$ 74.51)
	GR-BCD	46.05 ± 6.86	($\downarrow$ 30.24)	43.99 ± 1.77	($\downarrow$ 42.15)	34.83 ± 8.31	($\downarrow$ 45.10)
	MIN	46.05 ± 6.86	($\downarrow$ 30.24)	38.65 ± 2.48	($\downarrow$ 49.17)	16.17 ± 6.16	($\downarrow$ **74.51**)
InfoGraph	CLEAN	63.93 ± 6.43		66.87 ± 3.72		64.77 ± 2.41	
	RAND. EDGE FLIP.	56.85 ± 6.22	($\downarrow$ 11.10)	50.88 ± 3.37	($\downarrow$ 23.83)	58.84 ± 2.66	($\downarrow$ 9.15)
	PGD	51.65 ± 7.50	($\downarrow$ 19.42)	30.68 ± 4.89	($\downarrow$ 54.18)	55.12 ± 3.81	($\downarrow$ 14.87)
	PR-BCD	31.53 ± 7.67	($\downarrow$ 50.83)	14.71 ± 4.13	($\downarrow$ 78.10)	20.11 ± 4.24	($\downarrow$ 68.92)
	GR-BCD	47.23 ± 7.55	($\downarrow$ 26.19)	41.13 ± 4.40	($\downarrow$ 38.45)	24.99 ± 5.33	($\downarrow$ 61.44)
	MIN	31.53 ± 7.67	($\downarrow$ **50.83**)	14.71 ± 4.13	($\downarrow$ **78.10**)	20.11 ± 4.24	($\downarrow$ 68.92)
GraphCL	CLEAN	65.93 ± 4.02		73.09 ± 3.50		68.85 ± 4.83	
	RAND. EDGE FLIP.	61.62 ± 4.94	($\downarrow$ 6.53)	59.43 ± 2.36	($\downarrow$ 18.69)	58.43 ± 5.11	($\downarrow$ 8.48)
	PGD	60.86 ± 4.02	($\downarrow$ 7.68)	68.32 ± 2.54	($\downarrow$ 10.63)	58.35 ± 6.21	($\downarrow$ 8.61)
	PR-BCD	49.48 ± 6.39	($\downarrow$ 24.95)	32.20 ± 3.48	($\downarrow$ 55.94)	15.74 ± 7.66	($\downarrow$ 75.34)
	GR-BCD	40.29 ± 9.61	($\downarrow$ 38.89)	37.09 ± 3.37	($\downarrow$ 49.26)	32.82 ± 6.93	($\downarrow$ 48.60)
	MIN	40.29 ± 9.61	($\downarrow$ **38.89**)	32.20 ± 3.48	($\downarrow$ **55.94**)	15.74 ± 7.66	($\downarrow$ **75.34**)
AD-GCL	CLEAN	73.66 ± 1.36		71.79 ± 0.94		76.71 ± 2.07	
	RAND. EDGE FLIP.	65.37 ± 2.24	($\downarrow$ 11.25)	58.12 ± 1.49	($\downarrow$ 19.04)	75.18 ± 2.03	($\downarrow$ 1.95)
	PGD	61.44 ± 2.57	($\downarrow$ 16.59)	50.32 ± 2.32	($\downarrow$ 29.91)	44.36 ± 5.41	($\downarrow$ 42.18)
	PR-BCD	26.31 ± 4.33	($\downarrow$ 64.28)	14.35 ± 1.34	($\downarrow$ 80.01)	29.17 ± 5.27	($\downarrow$ 61.71)
	GR-BCD	63.96 ± 2.90	($\downarrow$ 13.17)	54.52 ± 1.44	($\downarrow$ 24.06)	37.48 ± 7.03	($\downarrow$ 50.98)
	MIN	26.31 ± 4.33	($\downarrow$ **64.28**)	14.35 ± 1.34	($\downarrow$ **80.01**)	29.17 ± 5.27	($\downarrow$ 61.71)

Table 6: Node classification performance: clean and perturbed accuracy mean ± std., and $\mathcal{R}_{\text{adv}}^{\text{clean}}$ (percentage), averaged over 15 runs with different weight initializations. Results include accuracy for different contrastive and non-contrastive models under various attack schemes and datasets. The attacks are allowed (Δ) to change 5% of edges. Models achieving the three highest relative drop are highlighted as **first**, **second**, and **third**.

Model	Attack	Cora		Citeseer		Pubmed		OGB-arXiv	
GCN	Clean	77.57 ± 1.11		63.99 ± 1.27		75.31 ± 0.80		68.22 ± 1.15	
	Rand. Edge Flip.	76.55 ± 1.19	(↓ 1.32)	62.65 ± 1.58	(↓ 2.09)	74.34 ± 0.88	(↓ 1.31)	66.07 ± 1.09	(↓ 3.15)
	PR-BCD	59.35 ± 1.70	(↓ 23.50)	47.69 ± 2.06	(↓ 25.46)	57.01 ± 1.28	(↓ 24.31)	52.54 ± 1.08	(↓ 22.99)
	GR-BCD	70.37 ± 1.61	(↓ 9.29)	55.31 ± 2.21	(↓ 13.57)	64.33 ± 1.97	(↓ 14.59)	49.58 ± 1.37	(↓ 27.33)
	Min	59.35 ± 1.70	(↓ **23.50**)	47.69 ± 2.06	(↓ **25.46**)	57.01 ± 1.28	(↓ **24.31**)	49.58 ± 1.37	(↓ **27.33**)
DGI	Clean	83.24 ± 1.37		72.91 ± 1.99		81.46 ± 0.79		60.18 ± 0.72	
	Rand. Edge Flip.	82.65 ± 1.25	(↓ 0.71)	72.64 ± 1.96	(↓ 0.39)	80.72 ± 0.844	(↓ 0.91)	59.14 ± 0.76	(↓ 1.72)
	PR-BCD	75.69 ± 1.86	(↓ 9.08)	67.47 ± 2.13	(↓ 7.47)	77.19 ± 0.92	(↓ 5.23)	53.28 ± 0.69	(↓ 11.45)
	GR-BCD	80.13 ± 1.55	(↓ 3.73)	70.36 ± 2.44	(↓ 3.52)	77.12 ± 0.89	(↓ 5.33)	54.19 ± 0.87	(↓ 9.96)
	Min	75.69 ± 1.86	(↓ 9.08)	67.47 ± 2.13	(↓ 7.47)	77.12 ± 0.89	(↓ 5.33)	53.28 ± 0.69	(↓ **11.45**)
GraphCL	Clean	71.99 ± 1.35		59.57 ± 1.45		74.29 ± 1.75		62.78 ± 0.44	
	Rand. Edge Flip.	70.86 ± 1.27	(↓ 1.57)	58.61 ± 1.76	(↓ 1.62)	72.86 ± 1.52	(↓ 1.92)	60.37 ± 0.43	(↓ 3.83)
	PR-BCD	54.71 ± 1.46	(↓ 24.00)	46.53 ± 1.87	(↓ 21.89)	58.71 ± 1.45	(↓ 20.96)	49.07 ± 0.37	(↓ 21.84)
	GR-BCD	60.15 ± 1.26	(↓ 16.44)	48.99 ± 1.75	(↓ 17.77)	53.99 ± 1.77	(↓ 27.33)	36.57 ± 0.55	(↓ 41.75)
	Min	54.71 ± 1.46	(↓ **24.00**)	46.53 ± 1.87	(↓ **21.89**)	53.99 ± 1.77	(↓ **27.33**)	36.57 ± 0.55	(↓ **41.75**)
GCA	Clean	79.07 ± 1.36		60.14 ± 2.24		78.62 ± 0.98			
	Rand. Edge Flip.	78.30 ± 1.63	(↓ 0.98)	59.21 ± 2.15	(↓ 1.54)	76.09 ± 1.07	(↓ 3.22)		
	PR-BCD	61.63 ± 3.18	(↓ 22.08)	43.61 ± 2.13	(↓ 27.51)	56.77 ± 1.69	(↓ 27.80)	OOM	
	GR-BCD	72.59 ± 2.75	(↓ 8.22)	50.45 ± 2.65	(↓ 16.14)	57.67 ± 1.87	(↓ 26.67)		
	Min	61.63 ± 3.18	(↓ **22.08**)	43.61 ± 2.13	(↓ **27.51**)	56.77 ± 1.69	(↓ **27.80**)		

E Hyperparameters

We report the hyperparameters we used for our evaluation in Table 7.

Table 7: Models hyperparameters.

Model	**Dataset**	**Hyperparameters**					
		lr	epochs	patience	dropout	layers	hid. dim.
GCN	CORA	1e-2	200	10	0.5	2	16
	CITESEER	1e-2	200	10	0.5	2	16
	PUBMED	1e-2	200	10	0.5	2	16
	OGB-ARXIV	1e-2	500	10	0.5	3	256
	PROTEINS	5e-3	50			4	128
	NCI1	5e-3	50			4	128
	DD	5e-3	50			4	32
GIN	PROTEINS	1e-3	10			8	512
	NCI1	1e-4	10			12	512
	DD	1e-4	20			4	32
DGI	CORA	1e-3	1000	20		1	512
	CITESEER	1e-3	1000	20		1	512
	PUBMED	1e-3	1000	20		1	256
	OGB-ARXIV	1e-3	1000	20		2	512
GraphCL	CORA	1e-3	1000	20		1	512
	CITESEER	1e-3	1000	20		1	512
	PUBMED	1e-3	1000	20		1	512
	OGB-ARXIV	1e-3	1000	20		1	512
	PROTEINS	1e-3	10			8	512
	NCI1	1e-4	10			12	512
	DD	1e-4	20			4	32
GCA	CORA	1e-3	500	20			256
	CITESEER	1e-3	500	20			256
	PUBMED	1e-3	500	20			256
	OGB-ARXIV	1e-3	500	20			256 (OOM)
InfoGraph	PROTEINS	1e-3	100			8	256
	NCI1	1e-3	100			8	256
	DD	1e-3	100			8	256
AD-GCL	PROTEINS	1e-2	150	20	0.5	5	32
	NCI1	1e-2	150	20	0.5	5	32
	DD	1e-2	150	20	0.5	5	32