# Are Defenses for Graph Neural Networks Robust?

**Anonymous Author(s)**
Affiliation
Address
email

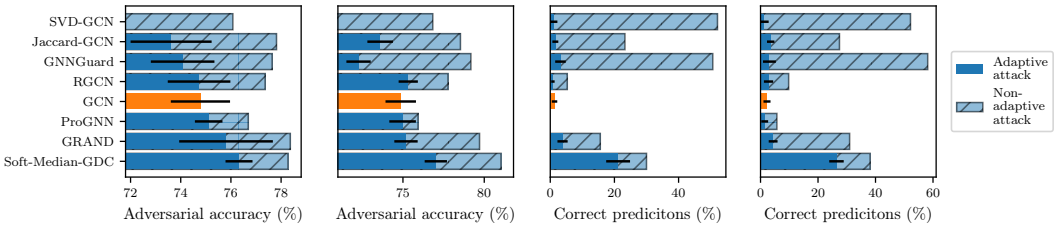
## Abstract

A cursory reading of the literature suggests that we have made a lot of progress in designing effective adversarial defenses for Graph Neural Networks (GNNs). Yet, the standard methodology has a serious flaw – virtually all of the defenses are evaluated against non-adaptive attacks leading to overly optimistic robustness estimates. We perform a thorough robustness analysis of 7 of the most popular defenses spanning the entire spectrum of strategies, i.e. aimed at improving the graph, the architecture, or the training. The results are sobering – most defenses show no or only marginal improvement compared to an undefended baseline. We advocate using custom adaptive attacks as a gold standard and we outline the lessons we learned from successfully designing such attacks. Moreover, our diverse collection of perturbed graphs forms a (black-box) unit test offering a first glance at a model's robustness.

## 1 Introduction

The vision community learned a bitter lesson – we need specific carefully crafted attacks to properly evaluate the adversarial robustness of a defense. Consequently, adaptive attacks are considered the gold standard [38]. This was not always the case, until recently most defenses were tested only against relatively weak static attacks. The turning point was Carlini & Wagner [3]'s work showing that 10 methods for detecting adversarial attacks can be easily circumvented. Shortly after Athalye et al. [1] showed that 7 out of the 9 defenses they studied can be broken since they (implicitly) rely on obfuscated gradients. So far, this bitter lesson is completely ignored in the graph domain.

Virtually no existing work that proposes an allegedly robust Graph Neural Network (GNN) evaluates against adaptive attacks, leading to overly optimistic robustness estimates.




(a) Global, Poisoning (b) Global, Evasion (c) Local, Pois. (d) Local, Evasion

Figure 1: Adaptive attacks draw a different picture of robustness. All defenses are less robust than reported, with an undefended GCN [30] outperforming some. We show results on Cora ML for both poisoning (attack before training) and evasion (attack after training), and both global (attack the test set jointly) and local (attack individual nodes) setting. The perturbation budget is relative w.r.t. the #edges for global attacks (5% evasion, 2.5% poisoning) and w.r.t the degree for local attacks (100%). In (a)/(b) SVD-GCN is catastrophically broken – our adaptive attacks reach 24%/9% (not visible).

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

To show the seriousness of this methodological flaw we categorize 45 works that propose a robust GNN and are published at major conferences/journals. We then choose one defense per category (usually the most highly cited). Not surprisingly, we show that none of the assessed models are as robust as originally advertised in their respective papers. In Fig. 1 we summarize the results for 7 of the most popular defenses, spanning the entire spectrum of strategies (i.e. aimed at improving the graph, the architecture, or the training, see Table 1).

We see that in both local and global settings, as well as for both evasion and poisoning, the adversarial accuracy under our adaptive attacks is significantly smaller compared to the routinely used non-adaptive attacks. Even more troubling is that many of the defenses perform worse than an undefended baseline (a vanilla GCN [30]). Importantly, the 7 defenses are not cherry-picked. We report the results for each defense we assessed and selected each defence before running any experiments.

Adversarial robustness measures the local generalization capabilities of a model, i.e. sensitivity to (bounded) worst-case perturbations. Certificates typically provide a lower bound on the actual robustness while attacks provide an upper bound. Since stronger attacks directly translate into tighter bounds our goal is to design the strongest attack possible. Our adaptive attacks have perfect knowledge of the model, the parameters, and the data, including all defensive measures. In contrast, non-adaptive attacks (e.g. transferred from an undefended proxy or an attack lacking knowledge about defense measures), only show how good is the defense at suppressing a specific subset of input perturbations.[1]

Tramer et al. [38] showed that even adaptive attacks can be tricky to design with many subtle challenges. The graph domain comes with additional challenges since graphs are typically sparse and discrete and the representation of any node depends on its neighborhood. For this reason, we describe the recurring themes, the lessons learned, and our systematic methodology for designing strong adaptive attacks for all examined models. Additionally, we find that defenses are *sometimes* sensitive to a common attack vector and transferring attacks can also be successful. Thus, the diverse collection of perturbed adjacency matrices resulting from our attacks forms a (black-box) unit test that any truly robust model should pass before we need an adaptive evaluation. In summary:

- We survey and categorize *45 defenses* published across prestigious machine learning venues.
- We design custom attacks for 7 defenses (16 %) , covering the spectrum of defense techniques. All examined models forfeit a large fraction of previously reported robustness gains.
- We provide a transparent methodology and guidelines for designing strong adaptive attacks.
- Our collection of perturbed graphs (to be published) can serve as a robustness unit test for GNNs.

## 2 Background and preliminaries

We follow the most common setup and assume GNN [18, 30] classifiers $f_\theta(\mathbf{A}, \mathbf{X})$ that operate on a symmetric binary adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$ with node features $\mathbf{X} \in \mathbb{R}^{n\times d}$ and node labels $\mathbf{y} \in \{1, 2, \dots, C\}^n$ where $C$ is the number of classes, $n$ is the number of nodes, and $m$ the number of edges. A poisoning attack perturbs the graph (flips edges) prior to training, optimizing

$$\max_{\tilde{\mathbf{A}}\in\Phi(\mathbf{A})} \ell_{\text{attack}}(f_{\theta^*}(\tilde{\mathbf{A}}, \mathbf{X}), \mathbf{y}) \quad \text{s.t.} \quad \theta^* = \arg\min_\theta \ell_{\text{train}}(f_\theta(\tilde{\mathbf{A}}, \mathbf{X}), \mathbf{y}) \tag{1}$$

where $\ell_{\text{attack}}$ is the attacker's loss, which is possibly different from $\ell_{\text{train}}$ (see § 4). In an evasion attack, $\theta^*$ is kept fixed and obtained by training on the clean graph $\min_\theta \ell_{\text{train}}(f_\theta(\mathbf{A}, \mathbf{X}), \mathbf{y})$. In both cases, the locality constraint $\Phi(\mathbf{A})$ limits the perturbation to an $L_0$-ball around the clean adjacency matrix s.t. $\|\tilde{\mathbf{A}} - \mathbf{A}\|_0 \leq 2\Delta$. Attacks on $\mathbf{X}$ also exist, however, they are not our focus since feature perturbations are more similar to other domains where adversarial robustness is studied.

**Threat model.** Our attacks aim to either cause misclassification of the entire test set (*global*) or a single node (*local*). To obtain the strongest attack possible (i.e. tightest robustness upper bound) we use white-box attacks. We do not constrain the attacker beyond a simple budget constraint that enforces a maximum number of perturbed edges. For our considerations on unnoticeability see § A.

**Greedy attacks.** Attacking a GNN typically corresponds to solving a constrained discrete non-convex optimization problem that – evident by this work – is hard to solve. Commonly, greedy

[1] From a security perspective non-adaptive attacks (typically transfer attacks) are also relevant since a real-world adversary is unlikely to know everything about the model and the data.

algorithms are used to approximate the solution of these optimization problems (e.g. see Nettack [61] or Metattack [60]). These greedy approaches have the nice side-effect that an attack for a high budget $\Delta$ directly gives all attacks for budgets lower than $\Delta$, but they tend to be relatively weaker.

**Projected Gradient Descent (PGD).** Alternatively, PGD [47] has been applied to GNNs where the discrete adjacency matrix is relaxed to $[0,1]^{n\times n}$ during the gradient-based optimization (e.g. $\nabla_{\mathbf{A}} f_{\theta^*}(\mathbf{A},\mathbf{X})$) and the resulting weighted change reflects the probability of flipping an edge. After each gradient update, the changes are projected back such that the budget holds in expectation $\|\mathbb{E}[\tilde{\mathbf{A}}] - \mathbf{A}\|_0 \leq 2\Delta$. Finally, multiple samples are obtained and the strongest perturbation $\tilde{\mathbf{A}}$ is chosen that obeys the budget $\Delta$. The biggest caveats while applying $L_0$-PGD are the relaxation gap and limited scalability (see Geisler et al. [16] for a detailed discussion and a scalable alternative).

**Evasion vs. poisoning.** Evasion can be considered the easier setting from an attack perspective since the model is fixed $f_{\theta^*}$. For poisoning, on the other hand, the adjacency matrix is perturbed before training (Eq. 1). Two general strategies exist for poisoning attacks: (1) transfer a perturbed adjacency matrix from an evasion attack [61]; or (2) attack directly by e.g. unrolling the training procedure to obtain gradients through training [60]. Xu et al. [47] propose to solve Eq. 1 with alternating optimization which was shown to be even weaker than the evasion transfer (1). Note that evasion is particularly of interest for inductive learning and poisoning for transductive learning.

# 3 Adversarial defenses

We select the defenses s.t. we capture the entire spectrum of methods improving robustness against structure perturbations. For the selection, we extend the taxonomy proposed in [19]. We selected the subset without cherry-picking based on the criteria elaborated below before experimentation.

**Taxonomy.** The taxonomy includes the top-level categories *Improving Graph* (preprocessing graph), *Improving Training* (e.g. adv. training or augmentations), and *Improving Architecture*. Many defenses for structure perturbations either fall into the category of improving the graph or adaptively (down-) weighting edges through an improved architecture. Thus, we introduce further subcategories. Similar to [19]'s discussion, unsupervised improvement of the graph finds clues in the structure and node features, while supervised improvement incorporates gradient information from the learning objective. Conversely, for adaptive edge weighting, we identify three prevalent approaches: rule-based (e.g. using a statistical test or a simple metric), probabilistic (e.g. modeling a latent distribution), and robust aggregations (e.g. a median aggregation). We assign each defense to the most appropriate category (more details in § B) but exclude [53, 25, 58] since these defenses are covered by their subsequent works.

**Selected defenses.** To evaluate a diverse set of defenses, we select one per leaf taxon, prioritizing highly cited defenses published at renowned venues with publicly available code. We implement all defenses in one unified pipeline. We present the categorization of defenses and our selection in Table 1. Similarly to Tramer et al. [38], we exclude defenses in the "robust training" category (see § C for a discussion). The three models in the "miscellaneous" category report some improvement in robustness, but they are not explicitly designed for defense purposes so we exclude them from our study. Some works evaluate only against evasion [42], others only poisoning [12, 15, 52], and the rest tackle both [57, 27, 16]. In some cases the evaluation setting is not explicitly stated. For completeness we consider each defense in all four settings (local/global and evasion/poisoning). Next, we provide a short summary of the key ideas behind each defense (details in § D).

Table 1: Categorization of selected defenses. Our taxonomy extends the one by Günnemann [19].

| Taxonomy | | | Selected Defense | Other Defenses |
|---|---|---|---|---|
| Improving Graph | Unsupervised | Feature-based | Jaccard GCN [42] | [23, 44] |
| | | Structure-based | SVD GCN [12] | [26, 54] |
| | Supervised | | ProGNN [27] | [10, 22, 21, 45, 50, 55] |
| Improving Training | Robust training | | N/A (see § C) | [6, 9, 14, 20, 24, 35, 46, 47, 48] |
| | Further Training Principles | | GRAND [15] | [5, 11, 32, 33, 36, 37, 49, 59] |
| Improving Architecture | Adaptively Weighting Edges | Rule-based | GNNGuard [52] | [28, 31] |
| | | Probabilistic | RGCN [57] | [8, 13] |
| | | Robust Agg. | Soft-Median-GDC [16] | [7, 16, 41, 51] |
| | Miscellaneous | | N/A (see above) | [34, 40, 43] |

**Improving the graph.** *Jaccard GCN* [42] uses a preprocessing step to remove all edges between nodes whose features exhibit a Jaccard similarity below a certain threshold. This was motivated by the homophily assumption which is violated by prior attacks that tend to insert edges between dissimilar nodes. *SVD GCN* [12] replaces the adjacency matrix with a low-rank approximation prior to plugging it into a regular GNN. This defense was motivated by the observation that the perturbations from Nettack [61] tend to disproportionately affect the high-frequency spectrum of the adjacency matrix. The key idea in *ProGNN* [27] is to learn the graph structure by alternatingly optimizing the parameters of the GNN and the adjacency matrix (the edge weights). The loss for the latter includes the standard cross-entropy loss, the distance to the original graph, and three other objectives designed to promote sparsity, low rank, and feature smoothness.

**Improving training.** *GRAND* [15] relies on random feature augmentations (setting some features to 0) coupled with neighbourhood augmentations $\bar{\mathbf{X}} = (\mathbf{AX} + \mathbf{AAX} + \cdots)$. All (randomly) augmented copies of $\bar{\mathbf{X}}$ are passed through the same MLP or GCN unifying the predictions using a custom loss.

**Improving the architecture.** *GNNGuard* [52] filters edges in each message passing aggregation via cosine-similarity (smoothed over training iterations). In the first layer of *RGCN* [57] we learn a Gaussian distribution over the feature matrix and the subsequent layers then manipulate this distribution (instead of using point estimates). For the loss we then sample from the resulting distribution. In addition, in each layer, RGCN assigns higher/lower weights to features with low/high variance. *Soft-Median-GDC* [16] replaces the message passing aggregation function in GNNs (typically a weighted mean) with a more robust alternative by relaxing the median using differentiable sorting.

**Common themes**. One theme shared by some defenses is to first discover some property that can discriminate clean from adversarial edges (e.g. high vs. low feature similarity), and then propose a strategy based on that property (e.g. filter low similarity edges). Often they analyze the edges from only a single attack such as Nettack [61]. The obvious pitfall of this strategy is that the attacker can easily adapt by restricting the adversarial search space to edges that will bypass the defense's (implicit) filter. Another theme is to add additional loss terms to promote some robustness objectives. Similarly, the attacker can incorporate the same terms in the attack loss to negate their influence.

# 4 Methodology: How to design strong adaptive attacks

In this section, we describe our general methodology and the lessons we learned while designing adaptive attacks. We hope these guidelines can serve as a reference for testing new defenses.

**Step 1 – Understand how the defense works** and categorize it. For example, some defenses rely on preprocessing by filtering out edges that meet certain criteria (e.g. Jaccard-GCN [42]). Others introduce additional losses during training (e.g. GRAND [15]), or change the architecture (e.g. RGCN [57]). Different defenses might require different attacks or impose extra requirements on the attacks.

**Step 2 – Probe for obvious weaknesses**. Some examples include: (a) transfer adversarial edges from another (closely related) model (see also § 6); (b) use a gradient-free (black-box) attack. For example, in our local experiments, we use a greedy brute force attack and in each step we iterate through all possible single edge flips and choose the one that contributes most to the attack objective.

**Step 3 – Launch a gradient-based adaptive attack** such as PGD. If the defense contains non-differentiable components (or components that obfuscate gradients [1]) then either: (a) replace them with a differentiable counterpart, e.g. drop "hard" filtering of edges as done in the preprocessing of Soft-Median-GDC [16]; or (b) attack a proxy model that removes them, e.g. for ProGNN [27] we ignore the nuclear norm loss term during training (while of course keeping it intact for the final evaluation of the defense).

**Step 4 – Adjust the attack loss**. In previous works the attack loss is often chosen to be the same as the training loss, i.e. maximize the cross-entropy (CE). This is suboptimal since CE is not *consistent* as defined by Tramer et al. [38] – higher loss values do not indicate a stronger attack. Thus, we use the consistent Carlini-Wagner (CW) loss [4] for local attacks, i.e. the logit difference between the target and most-likely non-target class. However, as discussed by Geisler et al. [16], for global attacks CW is also suboptimal since it can "waste" budget on already misclassified nodes. Their tanh margin (TM) loss resolves this issue. If not indicated otherwise, we either use TM or the probability margin (PM) loss [4] – a slight variant of TM that computes the margin after the softmax (rather than before).

**Step 5 – Tune the (attack) hyperparameters** such as the number of PGD steps, the attack learning rate, the optimizer, etc. For example, for meta attacks (poisoning) we observed that using the ADAM optimizer [29] can weaken the attack and replacing it with SGD can increase the effectiveness. Moreover, many defenses combine several distinct components – potentially introducing several loss terms. For example, for ProGNN [27] we find it important to tune the weights for these new terms.

**Remarks on poisoning attacks.** For poisoning we either: (a) transfer the edges obtained from an evasion attack; or (b) unroll the training procedure to obtain a gradient $\nabla_{\tilde{\mathbf{A}}}$ for determining good edges for a greedy attack. Not surprisingly, we find that applying PGD [47] on the (unrolled) meta gradients often yields even stronger attacks than the greedy Metattack [60] previously proposed. We refer to this attack as Meta-PGD. In some cases, it was helpful to alter the training to obtain better behaved gradients $\nabla_{\tilde{\mathbf{A}}}$, but naturally we always report the results on the original training procedure.

**Lessons learned.** We provide a detailed description of each adaptive attack and the necessary actions to make it as strong as possible in § D. Here, we highlight some important recurring challenges that should be kept in mind when designing adaptive attacks. (1) Numerical issues, e.g. due to division by tiny numbers can lead to weak attacks, and we typically resolve them via clamping. (2) In some cases we observed that for PGD attacks it is beneficial to clip the gradients to stabilize the adversarial optimization. (3) For a strong attack it is essential to tune its hyperparameters. (4) Relaxing non-differentiable components and deactivating operations that filter edges/embeddings based on a threshold in order to obtain gradients for every edge is an effective strategy. (5) If the success of evasion-poisoning transfer depends on a fixed random initialization (see § H), it helps to use multiple clean auxiliary models trained with different random seeds for the PGD attack – in each PGD step we choose one model randomly. (6) Components that make the optimization more difficult but barely help the defense can be safely deactivated. (7) It is sometimes beneficial to control the randomness in the training loop of Meta-PGD. (8) For Meta-PGD it can help to initialize the attack with non-zero perturbations and e.g. use the perturbed graph of a different attack.

**Example 1 – SVD-GCN.** To illustrate the attack process (especially step 3) we present a case study of how we construct an adaptive attack against SVD-GCN. Gradient-free attacks like Nettack do not work well here as they waste budget on adversarial edges which are filtered out by the low-rank approximation (LRA). Moreover, to the demise of gradient-based attacks, the gradients of the adjacency matrix are very unstable due to the SVD and thus less useful. Still, we start with a gradient-based attack as it is easier to adapt, specifically FGSM whose quick runtime enables rapid prototyping. To replace the LRA with a function whose gradients are better behaved, we first decompose the perturbed adjacency matrix $\tilde{\mathbf{A}} = \mathbf{A} + \delta\mathbf{A}$ and, thus, only need gradients for $\delta\mathbf{A}$. Next, we notice that the eigenvectors of $\mathbf{A}$ usually have few large components. Perturbations along those principal dimensions are representable by the eigenvectors, hence most likely are neither filtered out nor impact the eigenvectors. Knowing this, we approximate the LRA in a tractable manner by element-wise multiplication of $\delta\mathbf{A}$ with weights that quantify how well an edge aligns with the principal dimensions (details in § D). In short we replace $\mathrm{LRA}(\mathbf{A} + \delta\mathbf{A})$ with $\mathrm{LRA}(\mathbf{A}) + \delta\mathbf{A} \circ \mathrm{Weight}(\mathbf{A})$, which admits useful gradients. This approach carries over to other attacks such as Nettack – we can incorporate these weights into its score function to avoid selecting edges that will be filtered out.

**Example 2 – ProGNN.** While we approached SVD-GCN with a theoretical insight, breaking a composite defense like ProGNN requires engineering and tinkering. When attacking ProGNN with PGD and transferring the perturbations to poisoning we observe that the perturbations are only effective if the model is trained with the same random seed. This over-sensitivity can be avoided by employing lesson (5) in § 4. As ProGNN is very expensive to train due to its nuclear norm regularizer, we drop that term when training the set of auxiliary models without hurting attack strength. For unrolling the training we again drop the nuclear norm regularizer since it is non-differentiable. Sometimes PGD does not find a state with high attack loss, which can be alleviated by random restarts. As Meta-PGD optimization quickly stalls, we initialize it with a strong perturbation found by Meta-Attack on GCN. All of these tricks combined are necessary to successfully attack ProGNN.

**Effort.** Breaking Jaccard-GCN (and SVD-GCN) required around half an hour (resp. three days) of work for the initial proof of concept. Some other defenses require various adjustments that need to be developed over time, but reusing those can quickly break even challenging defenses. It is difficult to quantify this effort, but it can be greatly accelerated by adopting our lessons learned in § 4. In any case, we argue that authors proposing a new defense must put in reasonable effort to break it.

# 5 Evaluation of adaptive attacks

First, we provide details on the experimental setup and used metrics. We then report the main results and findings. We provide additional results in § G, we refer to § A for the list of attacks, and § D for their adaptive counter parts. We provide the code, configurations, and a collection of perturbed graphs anonymously: `https://figshare.com/s/0ecabc74ad7809a84c8d`.

**Setup.** We use the two most widely used datasets in the literature, namely Cora ML [2] and Citeseer [17]. Unfortunately, larger datasets are barely possible since most defenses are not very scalable. We repeat the experiments for five different data splits and we report the mean and the variance. We use an internal cluster with Nvidia GTX 1080Ti GPUs. Most experiments can be reproduced within a few hours. However, the experiments with ProGNN and GRAND will likely require several GPU days.

**Defense hyperparameters.** When first attacking the defenses we observed that many exhibit poor robustness using the hyperparameters provided by their authors. To not accidentally dismiss a defense as non-robust, we tune the hyperparameters such that the clean accuracy remains constant but the robustness w.r.t. adaptive attacks is improved. We run all experiments on the untuned defenses as well to confirm we achieve this goal. In the same way, we also tune the GCN model, which we use as a reference to asses whether a defense has merit. We report the configurations and comparisons in § F.

**Attacks and budget.** In the *global* setting, we run the experiments for budgets $\Delta$ of up to 15% of the total number of edges in the dataset. We apply the attacks FGSM and PGD [47] for evasion. For poisoning, we transfer the found perturbations and additionally run Metattack [60] and our Meta-PGD. In the *local* setting, we first draw sets of 20 target nodes per split with degrees 1, 2, 3, 5, 8-10, and 15-25 respectively (total of 120 nodes). This enables us to study how the attacks affect different types of nodes – lower degree nodes are often conjectured to be less robust. We then run the experiments for relative budgets $\Delta$ of up to 200% of the target node's degree. For example, if a node has 10 neighbors, and the budget $\Delta = 70\%$ then the attacker can change up to $10 \cdot 0.7 = 7$ edges. This commonly used setup ensures that we treat both low and high-degree nodes fairly. We use Nettack [61], FGSM, PGD, and our greedy brute force attack for evasion. For poisoning, we only transfer the found perturbations. Recall that where necessary, we adapt the attacks to the defense as discussed in § 4.

In alignment with our threat model, we evaluate each found perturbation by the test set accuracy it achieves (*global*) or the number of target nodes it correctly classifies (*local*). For each budget, we choose the strongest attack over different attack types also considering lower budgets (i.e. we enforce the curve to be monotonically decreasing). Choosing the strongest attack per budget among all attempts (e.g. among PGD, Metattack, Meta-PGD) gives rise to an envelope curve as seen in Fig. 2. For all reported numbers the higher the value the more robust is the model (conversely, the lower the value the stronger the attack).

Finally, we generate additional attacks by transferring the perturbations supporting the envelope curve to every other defense for both poisoning and evasion (see § 6 for a breakdown). They also contribute to the envelope curve of each defense, but in most cases their contribution is marginal. By non-adaptive attacks we refer to the best transfer attack from the untuned GCN (often Metattack) for the global setting and Nettack for the local setting, mirroring the attacks used by most defenses.

**Area Under the Curve (AUC).** An envelope curve gives us a detailed breakdown of the empirical robustness of a defense for different adversarial budgets. However, it is difficult to quantitatively compare different defenses without visually comparing their curves in a figure (e.g. see Fig. 4 or Fig. 5). Therefore, in addition to this breakdown per budget, we use the Area Under the Curve (AUC) to quantify the relations between different attacks and defenses using a single summary metric. We use the standard AUC for local attacks.

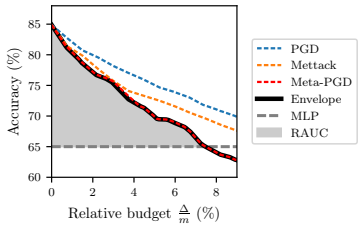
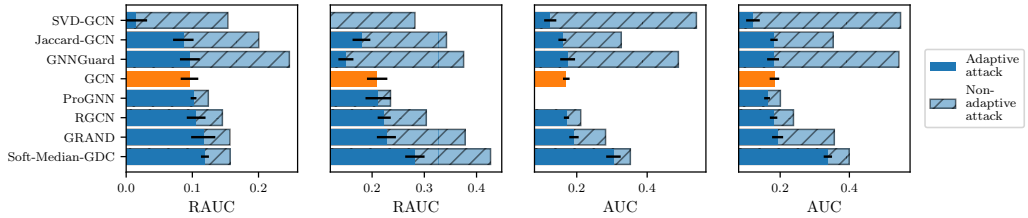



Figure 2: Strongest global poisoning attack per budget (i.e. envelope curve) made up of three attacks (dotted lines) for a GCN on Cora ML. The dashed line denotes the accuracy of an MLP. The shaded area is the RAUC.

For global attacks, we need to be more careful. We desire a metric that assigns a single number to the curve, and we require it to: (a) be independent of the specific choice of budget $\Delta$; (b) disregard high budgets where the model performs worse than an MLP; and (c) punish defenses that achieve robustness by trading off too much clean accuracy. All three properties are




(a) Global, Poisoning (b) Global, Evasion (c) Local, Pois. (d) Local, Evasion

Figure 3: Adaptive vs. non-adaptive attacks with budget-agnostic (R)AUC on Cora ML (compare to Fig. 1). SVD-GCN (b) is catastrophically broken – our adaptive attacks reach <0.02 (not visible).

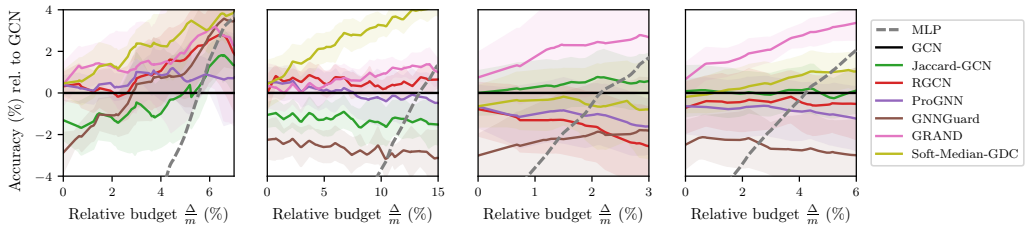
fulfilled by the area between the envelope curve and the line signifying the accuracy of an MLP. We call this metric Relative AUC (RAUC). More formally, $\mathrm{RAUC}(c) = \int_0^{b'} (c(b) - a_{\mathrm{MLP}})\mathrm{d}b$ s.t. $b \lesseqgtr b' \implies c(b) \gtreqless a_{\mathrm{MLP}}$ where $c(\cdot)$ is a piecewise linear robustness per budget curve, and $a_{\mathrm{MLP}}$ is the accuracy of the MLP baseline. We normalize RAUC s.t. 0% is the performance of an MLP and 100% is the optimal score (i.e. 100% accuracy).

On Fig. 2 we explicitly show the area that is computed with RAUC. It captures the robustness relative to an MLP – a model that is perfectly robust w.r.t. structure perturbations since it is oblivious to the graph structure, at the expense of a substantially lower clean accuracy than a GNN. We argue that robustness becomes less relevant when accuracy drops beyond the MLP threshold, as then, discarding the graph entirely is a better defense. Nonetheless, since some defenses seem to have the largest gains for perturbations beyond this threshold we show more severe budgets in § E for completeness.

**Finding 1 – Our adaptive attacks lower robustness by 40% on average.** In Fig. 3 we compare non-adaptive attacks, the current standard to evaluate defenses, with our adaptive attacks which we propose as a new standard. The achieved (R)AUC in each case drops on average by 40% (similarly for Citeseer, see § E). In other words, the reported robustness in the original works proposing a defense is roughly 40% too optimistic. We confirm a statistically significant drop ($p < 0.05$) with a one-sided t-test in 85% of all cases. Soft-Median-GDC is one partial exception showing small gains in 3 out of 4 cases. Regardless, the gains are noticeably smaller than reported.

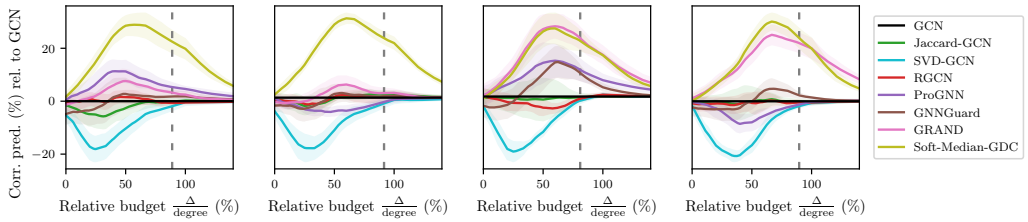
Considering adversarial accuracy for (small) fixed adversarial budget (Fig. 1) instead of the summary (R)AUC over all budgets tells the same story: non-adaptive attacks are too weak to be reliable indicators of robustness and adaptive attacks massively shrink the alleged robustness gains.

**Finding 2 – The robustness of GCN is not easily improved.** In Fig. 4 (global) and Fig. 5 (local) we provide a more detailed view for different adversarial budgets and different graphs. For easier comparison we show the accuracy relative to the undefended GCN baseline. We exclude SVD-GCN from Fig. 4 since it is so catastrophically broken that including it will make the other defenses illegible (it has accuracy of <24% already for a budget of 2% on Cora ML). Overall, the decline is substantial. Almost half of the examined defenses perform worse than GCN and most remaining defenses neither meaningfully improve nor lower the robustness (see also Fig. 1 and Fig. 2). GRAND and Soft-Medoid-GCN retain robustness in some settings, but the gains are smaller than reported.




(a) Cora ML, Pois. (b) Cora ML, Evas. (c) Citeseer, Pois. (d) Citeseer, Evas.

Figure 4: Difference (defense – undefended GCN) of adversarial accuracy for the strongest global attack per budget. Almost half of the examined defenses perform worse than the GCN.




(a) Cora ML, Pois. (b) Cora ML, Evas. (c) Citeseer, Pois. (d) Citeseer, Evas.

Figure 5: Difference (defense – undefended GCN) of fraction of correct predictions for the strongest local attack per budget. Most defenses show no or only marginal gain in robustness. The dashed vertical line shows where 95% of nodes for a GCN are misclassified on average.

**Finding 3 – Defense effectiveness depends on dataset.** As we can see in Fig. 4 and Fig. 5, our ability to circumvent specific defenses tends to depend on the dataset. It appears that some defenses are more suited for different datasets. For example, GRAND seems to be a good choice for Citeseer while it is not as strong on Cora ML. The results for local attacks (Fig. 5) paint a similar picture, here we see that Cora ML is more difficult to defend. This points to another potentially problematic pitfall – most defenses are developed only using these two datasets as benchmarks. Is the robustness even worse on other graphs? We leave this question for future work.

**Finding 4 – We cannot confirm a trade-off between accuracy and robustness.** Instead, Fig. 6 shows that defenses with high clean accuracy also exhibit high RAUC, i.e. are more robust against our attacks. This appears to be in contrast to the image domain [39]. However, we cannot exclude that future more powerful defenses might manifest this trade-off in the graph domain.

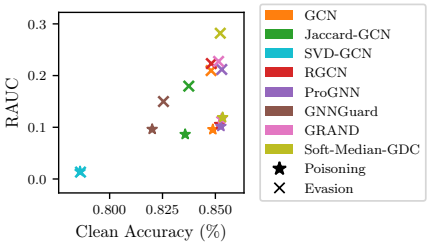



Figure 6: Each defense's clean accuracy vs. RAUC values of the strongest global attacks on Cora ML. We do not observe a robustness accuracy trade-off. Conversely, we even find that models with higher clean accuracy are more robust.

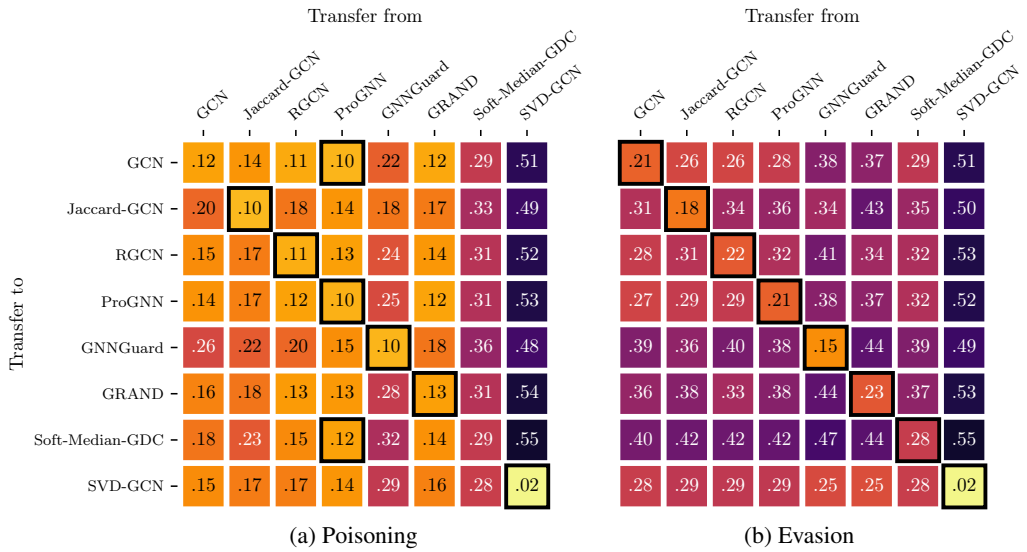
**Additional analysis.** During this project we generated a treasure trove of data. We perform a more in-depth analysis of our attacks in the appendix. First, we study how node degree affects attacks (see § I). For local attacks, the required budget to misclassify a node is usually proportional to the node's degree. Global attacks tend to be oblivious to degree and uniformly break nodes. Next, we perform a breakdown of each defense in terms of the sensitivity to different attacks (see § G). In short, global attacks are dominated by PGD for evasion and Metattack/Meta-PGD for poisoning with the PM or TM loss. For local attacks, PGD and especially our greedy brute-force attack are most effective. Finally, we analyze the properties of the adversarial edges in terms of different graph statistics such as edge centrality (see § J).

## 6 Robustness unit test

Next we systematically study how well the attacks transfer between the defenses. In Fig. 7 we see that in 14 out of 16 cases the adaptive attack is the most effective strategy (see main diagonal). However, often there is a model combination for which transfer attacks are very strong as well. Thus, we can probe a new defense's robustness simply by testing against all perturbed graphs. Recall, if there exists at least one perturbed graph that diminishes the robustness gains we can immediately conclude that a defense is not robust (in the worst-case) without the elaborate process of designing a new attack.

Motivated by the effectiveness of transfer attacks (e.g. from ProGNN [27] to Soft-Median-GDC [16] and GRAND [15]) we suggest this set of perturbed graphs to be used as a bare minimum robustness unit test. There is no need for adaptive attacks if a defense does not withstand this black box (ensemble) attack. We provide this collection in the supp. material, and we will open source it upon acceptance.

Nevertheless, we cannot stress enough that this collection does not replace a properly developed adaptive attack. For example, if one would come up with SVD GCN and would use our collection (excluding the perturbed graphs for SVD GCN) the unit test would partially pass. However, as we can see in e.g. Fig. 3, SVD GCN can be broken with an – admittedly very distinct – adaptive attack.




(a) Poisoning (b) Evasion

Figure 7: (R)AUC for transfer of global adaptive attacks between models/defenses. In each columns we have the model for which the adaptive attacks where created and the rows contain the performance after the transfer. With only two exceptions, adaptive attacks (diagonal) are most effective.

## 7 Related work

Excluding attacks on undefended GNNs, previous works studying adaptive attacks in the graph domain are scarce. The recently proposed graph robustness benchmark [56] also only studies transfer attacks. Such transfer attacks (typically from a GCN proxy, and typically Nettack or Metattack) are so common in the graph domain that their usage is often not even explicitly stated. Often the authors of a defense only state that they use PGD [47] (aka "topology attack") without further explanations. In this case, the authors most certainly refer to a PGD transfer attack on a GCN proxy. They almost never apply PGD to their actual defense which would yield a possibly weak but adaptive attack.

An exception where the defense authors study an adaptive attack is SVD-GCN [12]. Their attack collects the edges flipped by Nettack in a difference matrix $\delta\mathbf{A}$, replaces its most significant singular values and vectors with those from the clean adajcency matrix $\mathbf{A}$, and finally adds it to $\mathbf{A}$. Notably, this yields a dense continuous perturbed adjacency matrix. While their SVD GCN is susceptible to these perturbations, the results however do not appear as catastrophic as with our adaptive attacks, despite the severe violation of the threat model (see § 2). Geisler et al. [16] is another exception where gradient-based greedy and PGD attacks are directly applied to their Soft-Median defense, making them adaptive. Still, our attacks manage to further reduce its robustness estimate.

## 8 Discussion

We hope that the adversarial learning community for GNNs will reflect on the bitter lesson that evaluating adversarial robustness is not trivial. We show that on average adversarial robustness estimates are overstated by 40%. To ease the transition into a more reliable regime of robustness evaluation for GNNs we share our recipe for successfully designing strong adaptive attacks.

Using adaptive (white-box) attacks is also interesting from a security perspective. If a model successfully defends such strong attacks it is less likely to have remaining attack vectors for a real-world adversary. Practitioners can use our methodology to evaluate their models in hope to avoid an arms race right with attackers. Moreover, the white-box assumption lowers the chance that real-world adversaries can leverage our findings, as it is unlikely that the they have perfect knowledge.

We also urge for caution since the attacks only provide an upper bound (which with our attacks is now 40% tighter). Nevertheless, we argue that the burden of proof that a defense is truly effective should lie with the authors proposing it. Following our methodology the effort to design a strong adaptive attack is reduced, so we advocate for adaptive attacks as the gold-standard for future defenses.

# References


[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *35th International Conference on Machine Learning, ICML*, July 2018.

[2] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

[3] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *ACM Workshop on Artificial Intelligence and Security, AISec 2017*, 2017.

[4] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *IEEE Symposium on Security and Privacy*, 2017.

[5] Heng Chang, Yu Rong, Tingyang Xu, Yatao Bian, Shiji Zhou, Xin Wang, Junzhou Huang, and Wenwu Zhu. Not All Low-Pass Filters are Robust in Graph Convolutional Networks. In *Neural Information Processing Systems, NeurIPS*, May 2021.

[6] J. Chen, X. Lin, H. Xiong, Y. Wu, H. Zheng, and Q. Xuan. Smoothing Adversarial Training for GNN. *IEEE Transactions on Computational Social Systems*, 2020.

[7] Liang Chen, Jintang Li, Qibiao Peng, Yang Liu, Zibin Zheng, and Carl Yang. Understanding Structural Vulnerability in Graph Convolutional Networks. In *30th International Joint Conference on Artificial Intelligence*, Montreal, Canada, Aug. 2021. International Joint Conferences on Artificial Intelligence Organization.

[8] Lingwei Chen, Xiaoting Li, and Dinghao Wu. Enhancing Robustness of Graph Convolutional Networks via Dropping Graph Connections. In Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera, editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, Cham, 2021. Springer International Publishing.

[9] Zhijie Deng, Yinpeng Dong, and Jun Zhu. Batch Virtual Adversarial Training for Graph Convolutional Networks. *ICML 2019 Workshop: Learning and Reasoning with Graph-Structured Representations*, May 2019.

[10] Dongsheng Duan, Lingling Tong, Yangxi Li, Jie Lu, Lei Shi, and Cheng Zhang. AANE: Anomaly Aware Network Embedding for Anomalous Link Detection. In *IEEE International Conference on Data Mining (ICDM)*, Nov. 2020.

[11] Pantelis Elinas, Edwin V. Bonilla, and Louis Tiao. Variational Inference for Graph Convolutional Networks in the Absence of Graph Data and Adversarial Settings. *Neural Information Processing Systems, NeurIPS*, 33, 2020.

[12] Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. All you need is Low (rank): Defending against adversarial attacks on graphs. *International Conference on Web Search and Data Mining, WSDM*, 2020.

[13] Boyuan Feng, Yuke Wang, Z. Wang, and Yufei Ding. Uncertainty-aware Attention Graph Neural Network for Defending Adversarial Attacks. *AAAI Conference on Artificial Intelligence*, 2021.

[14] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. Graph Adversarial Training: Dynamically Regularizing Based on Graph Structure. *IEEE Transactions on Knowledge and Data Engineering*, 33(06), June 2021.

[15] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph Random Neural Network for Semi-Supervised Learning on Graphs. *35th International Conference on Machine Learning, ICML*, Sept. 2021.

[16] Simon Geisler, Tobias Schmidt, Hakan Sirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of Graph Neural Networks at Scale. *Neural Information Processing Systems, NeurIPS*, 2021.

[17] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, 1998.

[18] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *34th International Conference on Machine Learning, ICML*, 3, 2017.

[19] 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*, chapter 8, . Springer, Singapore, 2021.

[20] Weibo Hu, Chuan Chen, Yaomin Chang, Zibin Zheng, and Yunfei Du. Robust graph convolutional networks with directional graph adversarial training. *Applied Intelligence*, 51(11), Nov. 2021.

[21] Vassilis N. Ioannidis and Georgios B. Giannakis. Edge Dithering for Robust Adaptive Graph Convolutional Networks. Technical Report AAAI Conference on Artificial Intelligence, arXiv, 2020.

[22] Vassilis N. Ioannidis, Antonio G. Marques, and Georgios B. Giannakis. Tensor Graph Convolutional Networks for Multi-relational and Robust Learning. *IEEE Transactions on Signal Processing*, 68, 2020.

[23] Vassilis N. Ioannidis, Dimitris Berberidis, and Georgios B. Giannakis. Unveiling Anomalous Nodes Via Random Sampling and Consensus on Graphs. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2021.

[24] Hongwei Jin and Xinhua Zhang. Latent Adversarial Training of Graph Convolution Networks. *ICML 2019 Workshop: Learning and Reasoning with Graph-Structured Representations*, 2019.

[25] Hongwei Jin and Xinhua Zhang. Robust Training of Graph Convolutional Networks via Latent Perturbation. In Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera, editors, *{Machine Learning and Knowledge Discovery in Databases - European Conference, {ECML} {PKDD}*. Springer International Publishing, Cham, 2021.

[26] Ming Jin, Heng Chang, Wenwu Zhu, and Somayeh Sojoudi. Power up! Robust Graph Convolutional Network against Evasion Attacks based on Graph Powering. *AAAI Conference on Artificial Intelligence*, 2021.

[27] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph Structure Learning for Robust Graph Neural Networks. In *International Conference on Knowledge Discovery and Data Mining, KDD*, 2020.

[28] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. Node Similarity Preserving Graph Convolutional Networks. *International Conference on Web Search and Data Mining, WSDM*, 2021.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, {ICLR} 2015*, 2015.

[30] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations, ICLR*, 2017.

[31] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic Graph Neural Networks. In *38th International Conference on Machine Learning*. PMLR, July 2021.

[32] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to Drop: Robust Graph Neural Network via Topological Denoising. *International Conference on Web Search and Data Mining, WSDM*, 2021.

[33] Florence Regol, Soumyasundar Pal, Jianing Sun, Yingxue Zhang, Yanhui Geng, and Mark Coates. Node copying: A random graph model for effective graph sampling. *Signal Processing*, 192, Mar. 2022.

[34] Uday Shankar Shanthamallu, Jayaraman J. Thiagarajan, and Andreas Spanias. Uncertainty-Matching Graph Neural Networks to Defend Against Poisoning Attacks. *AAAI Conference on Artificial Intelligence*, 2021.

[35] Ke Sun, Zhouchen Lin, Hantao Guo, and Zhanxing Zhu. Virtual Adversarial Training on Graph Convolutional Networks in Node Classification. In Zhouchen Lin, Liang Wang, Jian Yang, Guangming Shi, Tieniu Tan, Nanning Zheng, Xilin Chen, and Yanning Zhang, editors, *Pattern Recognition and Computer Vision, PRCV*, Lecture Notes in Computer Science, Cham, 2019. Springer International Publishing.

[36] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. Transferring Robustness for Graph Neural Network Against Poisoning Attacks. *13th International Conference on Web Search and Data Mining*, Jan. 2020.

[37] Shuchang Tao, H. Shen, Q. Cao, L. Hou, and Xueqi Cheng. Adversarial Immunization for Certifiable Robustness on Graphs. *International Conference on Web Search and Data Mining, WSDM*, 2021.

[38] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On Adaptive Attacks to Adversarial Example Defenses. *Neural Information Processing Systems, NeurIPS*, 33, 2020.

[39] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. In *7th International Conference on Learning Representations, {ICLR}*, 2019.

[40] Haibo Wang, Chuan Zhou, Xin Chen, Jia Wu, Shirui Pan, and Jilong Wang. Graph Stochastic Neural Networks for Semi-supervised Learning. In *Neural Information Processing Systems, NeurIPS*. Curran Associates, Inc., 2020.

[41] Yiwei Wang, Shenghua Liu, Minji Yoon, Hemank Lamba, Wei Wang, Christos Faloutsos, and Bryan Hooi. Provably Robust Node Classification via Low-Pass Message Passing. In *2020 IEEE International Conference on Data Mining (ICDM)*, Sorrento, Italy, Nov. 2020. IEEE.

[42] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. *IJCAI International Joint Conference on Artificial Intelligence*, 2019-Augus, 2019.

[43] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph Information Bottleneck. *Neural Information Processing Systems, NeurIPS*, 2020.

[44] Yang Xiao, Jie Li, and Wengui Su. A Lightweight Metric Defence Strategy for Graph Neural Networks Against Poisoning Attacks. In Debin Gao, Qi Li, Xiaohong Guan, and Xiaofeng Liao, editors, *Information and Communications Security*, Lecture Notes in Computer Science, Cham, 2021. Springer International Publishing.

[45] Hui Xu, Liyao Xiang, Jiahao Yu, Anqi Cao, and Xinbing Wang. Speedup Robust Graph Structure Learning with Low-Rank Information. In *30th ACM International Conference on Information & Knowledge Management*, Virtual Event Queensland Australia, Oct. 2021. ACM.

[46] Jiarong Xu, Yang Yang, Junru Chen, Chunping Wang, Xin Jiang, Jiangang Lu, and Yizhou Sun. Unsupervised Adversarially-Robust Representation Learning on Graphs. *AAAI Conference on Artificial Intelligence*, 2022.

[47] 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. *IJCAI International Joint Conference on Artificial Intelligence*, 2019-Augus, 2019.

[48] Kaidi Xu, Sijia Liu, Pin-Yu Chen, Mengshu Sun, Caiwen Ding, Bhavya Kailkhura, and Xue Lin. Towards an Efficient and General Framework of Robust Training for Graph Neural Networks. *{IEEE} International Conference on Acoustics, Speech and Signal Processing, {ICASSP}*, Feb. 2020.

[49] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph Contrastive Learning with Augmentations. In *Neural Information Processing Systems, NeruIPS*. Curran Associates, Inc., 2020.

[50] Baoliang Zhang, Xiaoxin Guo, Zhenchuan Tu, and Jia Zhang. Graph alternate learning for robust graph neural networks in node classification. *Neural Computing and Applications*, 34 (11), June 2022.

[51] Li Zhang and Haiping Lu. A Feature-Importance-Aware and Robust Aggregator for GCN. In *29th ACM International Conference on Information & Knowledge Management*, CIKM '20, New York, NY, USA, Oct. 2020. Association for Computing Machinery.

[52] Xiang Zhang and Marinka Zitnik. GNNGUARD: Defending Graph Neural Networks against Adversarial Attacks. *Neural Information Processing Systems, NeurIPS*, 2020.

[53] Yingxue Zhang, Sakif Hossain Khan, and Mark Coates. Comparing and Detecting Adversarial Attacks for Graph Deep Learning. *Representation Learning on Graphs and Manifolds Workshop, ICLR, New Orleans, LA, USA*, 2019.

[54] Yingxue Zhang, Florence Regol, Soumyasundar Pal, Sakif Khan, Liheng Ma, and Mark Coates. Detection and Defense of Topological Adversarial Attacks on Graphs. In *24th International Conference on Artificial Intelligence and Statistics, AISTATS*. PMLR, Mar. 2021.

[55] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust Graph Representation Learning via Neural Sparsification. In *International Conference on Machine Learning, ICML*. PMLR, Nov. 2020.

[56] Qinkai Zheng, Xu Zou, Yuxiao Dong, Yukuo Cen, Da Yin, Jiarong Xu, Yang Yang, and Jie Tang. Graph Robustness Benchmark: Benchmarking the Adversarial Robustness of Graph Machine Learning. *Neural Information Processing Systems, NeurIPS*, Aug. 2021.

[57] Dingyuan Zhu, Peng Cui, Ziwei Zhang, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. *International Conference on Knowledge Discovery and Data Mining, KDD*, 2019.

[58] Jun Zhuang and Mohammad Al Hasan. Defending Graph Convolutional Networks against Dynamic Graph Perturbations via Bayesian Self-supervision. In *AAAI Conference on Artificial Intelligence*, Mar. 2022.

[59] Jun Zhuang and Mohammad Al Hasan. How Does Bayesian Noisy Self-Supervision Defend Graph Convolutional Networks? *Neural Processing Letters*, Feb. 2022.

[60] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. *7th International Conference on Learning Representations, ICLR*, 2019.

[61] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. *International Conference on Knowledge Discovery and Data Mining, KDD*, 2018.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See § 8
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See § 8
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See § 5
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See § 5, § F and provided code
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All experiments are repeated for five random data splits
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See beginning of § 5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See beginning of § 5
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]