Robustness of Graph Neural Networks at Scale

Anonymous Author(s)
Affiliation
Address
email

Abstract

Graph Neural Networks (GNNs) are increasingly important given their popularity and the diversity of applications. Yet, existing studies of their vulnerability to adversarial attacks rely on relatively small graphs. We address this gap and study how to attack and defend GNNs at scale. We propose two sparsity-aware first-order optimization attacks that maintain an efficient representation despite optimizing over a number of parameters which is quadratic in the number of nodes. We show that common surrogate losses are not well-suited for global attacks on GNNs. Our alternatives can double the attack strength. Moreover, to improve GNNs' reliability we design a robust aggregation function, Soft Median, resulting in an effective defense at all scales. We evaluate our attacks and defense with standard GNNs on graphs more than 100 times larger compared to previous work. We even scale one order of magnitude further by extending our techniques to a scalable GNN.

1 Introduction

The evidence that Graph Neural Networks (GNNs) are not robust to adversarial perturbations is compelling [11, 18, 42]. However, the graphs in previous robustness studies are tiny. This is worrying, given that GNNs are already deployed in many real-world Internet-scale applications [3, 37]. For example, PubMed [28] (19,717 nodes) is often considered to be a large-scale graph and around 20 GB of memory is required for an attack based on its dense adjacency matrix. Such memory requirements are impractical and limit advancements of the field. In this work, we set the foundation for the holistic study of adversarial robustness of GNNs at scale. We study graphs with up to 111 million nodes for local attacks (i.e. attacking a single node) and 2.5 million nodes for global attacks (i.e. attacking all nodes at once). As it turns out, GNNs at scale are also highly vulnerable to adversarial attacks. In Fig. 1, we show the substantial improvement of memory efficiency of our attack over a popular prior work for attacking a GNN globally. Analogous to our attacks, our effective defense scales to graphs with 111 million nodes and beyond.

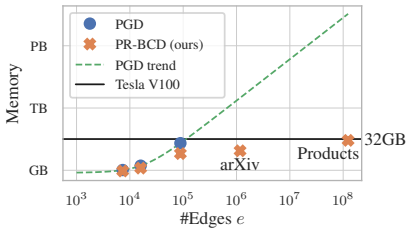


Figure 1: GPU memory consumption for a global attack with Projected Gradient Descent (PGD) [36], its quadratic extrapolation, and our *Projected Randomized Block Coordinate Descent (PR-BCD)* (§ 3). Both yield similar perturbed accuracy. Beyond attacks, our defense (§ 4) can also be applied on such scales.

Scope. We focus on adversarial robustness w.r.t. structure attacks on GNNs for node classification $\max_{\tilde{\boldsymbol{A}} \text{ s.t. } \|\tilde{\boldsymbol{A}} - \boldsymbol{A}\|_0 < \Delta} \mathcal{L}(f_\theta(\tilde{\boldsymbol{A}}, \boldsymbol{X}))$ with loss function $\mathcal{L}$ (or its surrogate $\mathcal{L}'$), budget Δ, and *fixed* model parameters θ. The GNN $f_\theta(\boldsymbol{A}, \boldsymbol{X})$ is applied to a graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ with node attributes $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, the adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{n \times n}$, and m edges. We solely consider *evasion* (test time) attacks, but our methods can be used in *poisoning* (train time) attacks [41]. We distinguish between *local* attacks on a single node and *global* attacks that target a large fraction of nodes with a shared budget Δ.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Broader impact. Since we enable the study of robustness at scale, which previously was practically infeasible, an adversary can potentially abuse our attacks. The risk is minimized given that we assume perfect knowledge about the graph, model, and labels. Nonetheless, our findings suggest that one should be careful when deploying GNNs, especially in a safety-critical application, and highlight that further research is needed. To mitigate this risk, we must be able to evaluate it. We also propose a scalable defense that shows strong performance empirically but we urge practitioners to consider potential trade-offs, e.g. improving robustness at the expense of accuracy for different groups of users.

Contributions. We address three major challenges hindering the study of GNNs' adversarial robustness at scale and propose viable solutions including an extensive empirical evaluation: (1) Previous losses are not well-suited for global attacks on GNNs at scale; (2) Attacks on GNNs scale quadratically in the number of nodes or worse; (3) Similarly, previous defenses are barely scalable.

(1) Surrogate loss. We study the limitation of state of the art surrogate losses for attacking the accuracy of a GNN over all nodes, following [6, 7, 22, 33, 35, 41] in § 2. Especially in combination with small/realistic budgets Δ and on large graphs, previous surrogate losses lead to weak attacks. In particular, Cross Entropy (CE) or the widely used Carlini-Wagner loss [4, 35] are weak surrogates for such global attacks. Our novel losses that overcome these limitations easily improve the strength of the attack by 100%. For larger datasets, this gap even becomes more significant.

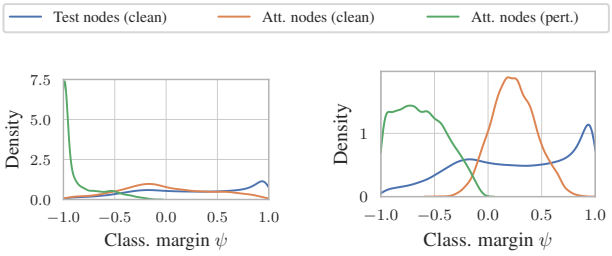
(2) Attacks. Attacks solving a discrete optimization problem easily become computationally infeasible because of the vast amount of potential adjacency matrices ($\mathcal{O}(2^{n^2})$). An approximate solution can be found with first-order optimization but we then still optimize over a quadratic number of parameters (n^2). There is no trivial way to sparsify existing attacks as we need to represent each edge explicitly to obtain its gradient (i.e. space complexity $\Theta(n^2)$). Nevertheless, we overcome this limitation and propose two strategies to apply first-order optimization without the burden of a dense adjacency matrix. In § 3, we describe how to add/remove edges between existing nodes based on Randomized Block Coordinate Descent (R-BCD) at an additional memory requirement of $\mathcal{O}(\Delta)$. Due to the limited scalability of traditional GNNs, we also consider the case where we attack PPRGo [3], a scalable GNN. Here, we even obtain an algorithm with constant complexity in the nodes n.

(3) Defense. We propose *Soft Median* – a computationally efficient, robust, differentiable aggregation function inspired by Geisler et al. [14], by taking advantage of recent advancements in differentiable sorting [26] (see § 4). Using our Soft Median we observe similar robustness to [14], but with a significantly lower memory footprint, which enables us to defend GNNs at scale.

2 Surrogate Loss

During training and in first-order attacks, we ideally wish to optimize a target metric which is often discontinuous (e.g. accuracy and 0/1 loss $\mathcal{L}_{0/1}$). However, for gradient-based optimization we commonly substitute the actual target loss by a *surrogate* $\mathcal{L}' \approx \mathcal{L}$ (e.g. cross entropy for $\mathcal{L}_{0/1}$). In the context of i.i.d. samples (e.g. images), a single example is attacked in isolation with its own budget, which is similar to a *local* attack for GNNs. In such a scenario it is often sufficient to *maximize* the cross entropy for the attacked node/image (untargeted attack): $\mathrm{CE}(y, \boldsymbol{p}) = -\log(p_{c^*})$. where y is the label and $\boldsymbol{p}$ is the vector of confidence scores.

Many *global* attacks for GNNs [6, 33, 35, 41] maximize the average cross entropy to attack all nodes with a combined budget Δ. Especially on large graphs, we often observed that the CE loss increases even though the accuracy does not decline (see § A.2). As we can see in Fig. 2, this is due to CE's bias towards nodes which have a low confidence score.



(a) Loss: cross entropy (b) Loss: tanh Margin

Figure 2: Margin ψ of test nodes vs. attacked nodes, before and after perturbation. We attack the arXiv graph (Table 1) and a single-layer GCN with one percent of edges ($\epsilon = 0.01$) as a budget. In stark contrast to the tanh Margin, the CE loss spends a lot of its budget on misclassified nodes (i.e. $\psi < 0$). See § A.2 for more variants.

With CE we primarily attack nodes that are already misclassified, which means that the classification margin $\psi = \min_{c \neq c^*} p_{c^*} - p_c$ is already negative[1]. For a longer discussion about CE see § A.2.

Global attack. In contrast to attacking a single image/node, a global attack on a GNN has to (1) keep house with the budget Δ and (2) find edges that degrade the overall accuracy maximally (i.e. target "fragile" nodes). Without additional information, intuitively, one would first attack low-confidence nodes close to the decision boundary. Hence, the surrogate loss should have a minimal (maximally negative) gradient at $\psi \to 0^+$ (i.e. approaching $\psi \to 0$ from $\psi \geq 0$). Moreover, if we solely want to lower the accuracy, then we can stop attacking a node once it is misclassified:

Definition 1 *A* budget-aware surrogate loss $\mathcal{L}'$ **(I)** *only incentivizes perturbing nodes that are correctly classified:* $\partial\mathcal{L}'/\partial z_{c^*}|_{\psi<0} = 0$ *and* **(II)** *favours nodes close to the decision boundary:* $\partial\mathcal{L}'/\partial z_{c^*}|_{\psi_0} < \partial\mathcal{L}'/\partial z_{c^*}|_{\psi_1}$ *for any* $0 < \psi_0 < \psi_1$.

Under certain assumptions such a *budget-aware* loss can be used to achieve the optimal solution in Proposition 1's greedy scheme (for proof see § A.2.1). *Assumption 1*: The set of attacked nodes is independent (their receptive fields do not overlap). Particularly on large graphs with small budgets, $\Delta/n \to 0$, deciding which node to attack becomes an increasingly local decision since the receptive field becomes insignificant in comparison to the rest of the graph. *Assumption 2*: The budget required to change the prediction of node i depends (only) on the margin: $\Delta_i = g(\psi_i)$ for some strictly increasing function $g(\cdot)$. That is, the larger the margin ψ_i, the harder it is to attack node i.

Proposition 1 *Let $\mathcal{L}'$ be the surrogate for the 0/1 loss $\mathcal{L}_{0/1}$ used to attack a node classification algorithm $f_\theta(\boldsymbol{A}, \boldsymbol{X})$ with a global budget Δ. Let $\Delta_i = g(|\psi_i|)$ be the budget required to move the prediction of node i over the decision boundary, with strictly increasing $g(|\psi_i|)$. Additionally, attacking node i does not influence other nodes. Suppose we greedily attack nodes in order of $\partial\mathcal{L}'/\partial z_{c^*}(\psi_0) \leq \partial\mathcal{L}'/\partial z_{c^*}(\psi_1) \leq \cdots \leq \partial\mathcal{L}'/\partial z_{c^*}(\psi_l)$ until the budget is exhausted $\Delta < \sum_{i=0}^{l+1} \Delta_i$. In this case, we obtain the global optimum of $\max_{\tilde{\boldsymbol{A}} \text{ s.t. } \|\tilde{\boldsymbol{A}} - \boldsymbol{A}\|_0 < \Delta} \mathcal{L}_{0/1}(f_\theta(\tilde{\boldsymbol{A}}, \boldsymbol{X}))$ if $\mathcal{L}'$ has the properties* **(I)** $\partial\mathcal{L}'/\partial z_{c^*}|_{\psi<0} = 0$ *and* **(II)** $\partial\mathcal{L}'/\partial z_{c^*}|_{\psi_0} < \partial\mathcal{L}'/\partial z_{c^*}|_{\psi_1}$ *for any* $0 < \psi_0 < \psi_1$.

Even under the simplifying Assumptions 1 & 2, the the Cross Entropy (CE) is not guaranteed to obtain the global optimum. The (CE) violates property (I) and in the worst case only perturbs nodes that are already misclassified (see Fig. 2). The Carlini Wagner (CW) [4, 36] loss CW $= \min(\max_{c \neq c^*} z_c - z_{c^*}, 0)$ violates property (II). It is also not guaranteed to obtain the global optimum, i.e. CW loss lacks focus on nodes close to the decision boundary. In the worst case, an attack with CW spends all budget on confident nodes—without flipping a single one.

We propose the Masked Cross Entropy MCE $= 1/|\mathbb{V}^+| \sum_{n \in \mathbb{V}^+} -\log(p^{(i)}_{c^*})$ which fulfills both properties by only considering correctly classified nodes $\mathbb{V}^+$ and, hence, reaches the global optimum under the stated assumptions. Empirically, for a greedy gradient-based attack the MCE comes with gains of more than 200% in strength (see Fig. 6). Surprisingly, if we apply MCE to a Projected Gradient Descent (PGD) attack, we observe hardly any improvement over CE. We identify two potential reasons for that. The first is due to the learning dynamics of PGD. Suppose a misclassified node does not receive any weight in the gradient update, now if the budget is exceeded after the update it is likely to be down-weighted. This can lead to nodes that oscillate around the decision boundary (similarly to the Carlini Wagner loss in § A.2). Second, the assumption about independence between the attacked nodes may be too strong in practice. In Definition 2, we relax properties (I)/(II) and propose to overcome these limitations via enforcing confidently misclassified nodes, i.e. we want the attacked nodes to be at a "safe" distance from the decision boundary. We propose the tanh of the margin in logit space, i.e. tanh Margin $= \tanh(\max_{c \neq c^*} z_c - z_{c^*})$. It obeys Definition 2 and its effectiveness is apparent from Fig. 2. For the empirical evaluation see § 5 and for more results as well as details on all selected losses see § A.2. In the appendix we also study further losses to deepen the understanding about the required properties.

Definition 2 *A* confidently misclassifying, budget-aware surrogate loss $\mathcal{L}'$ **(A)** *should saturate for confidently misclassified nodes:* $\lim_{\psi \to -1^+} \mathcal{L}' = k < \infty$ *and* **(B)** *favours points close to the decision boundary:* $\partial\mathcal{L}'/\partial z_{c^*}|_{\psi_0} < \partial\mathcal{L}'/\partial z_{c^*}|_{\psi_1} < 0$ *for any* $0 < \psi_0 < \psi_1 < 1$ *or* $-1 < \psi_1 < \psi_0 < 0$.

[1] We simply write $\boldsymbol{p} = f_\theta(\boldsymbol{A}, \boldsymbol{X})$ omitting that $\boldsymbol{p}$ belongs to specific node, i.e. $\boldsymbol{p}_i$ of node i. We also overload this with the logits / pre-softmax activation as $\boldsymbol{z} = f_\theta(\boldsymbol{A}, \boldsymbol{X})$. See § A.1 for an overview of the notation.

3 Scalable Attacks

Beginning with [11, 42], many adversarial attacks on the graph structure have been proposed. As discussed, gradient-based attacks such as [7, 33, 41, 42] aim for lower computational cost by approximating the corresponding discrete optimization problem. However, they optimize all possible entries in the *dense* adjacency matrix $\boldsymbol{A}$ which comes with quadratic space complexity $\Theta(n^2)$. Since previous attacks come with limited scalability (e.g. see Fig. 1), GNNs robustness on larger graphs is largely unexplored. First, we propose a family of attacks that does not require a dense adjacency matrix and comes with linear complexity w.r.t. the number of nodes. Then, we further improve the complexity of our attack for a scalable GNN called PPRGo [3].

Related work. Li et al. [21] analyze their *local* adversarial attack on the mini-batch network Cluster-GCN applied to a graph with around 200k nodes. We consider arbitrary Graph Neural Networks and we even scale our *global* attack to a graph ten times larger. With PPRGo we even outscale them by factor 500. Dai et al. [11] scale their reinforcement learning approach to a graph for financial transactions with 2.5 million nodes. In contrast to our work, they scale their *local* attack only using a tiny budget Δ of *a single edge deletion* and only need to consider the receptive field of a single node. We scale our local attack to 111M nodes and allow large budgets Δ.

Large scale optimization. In some big data use cases, the cost to calculate the gradient towards all variables can be prohibitively high. For this reason, coordinate descent has gained importance in machine learning and large-scale optimization [32]. Nesterov [24] proposed (and analyzed the convergence of) Randomized Block Coordinate Descent (R-BCD). In R-BCD only a subset (called a block) of variables is optimized at a time and, hence, only the gradients towards those variables are required. In many cases, this allows for a lower memory footprint and in some settings even converges faster than standard methods [25].

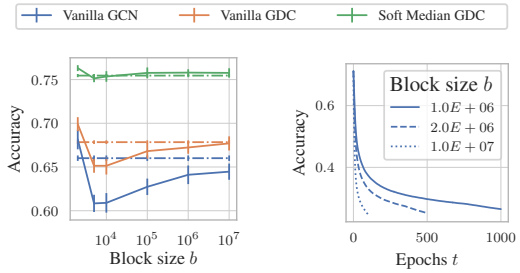
For clarity, we now model the perturbations $\boldsymbol{P} \in \{0,1\}^{n \times n}$ explicitly ($\boldsymbol{P}_{ij} = 1$ denotes an edge flip):

$$\max_{\boldsymbol{P} \text{ s.t. } \boldsymbol{P} \in \{0,1\}^{n \times n},\ \sum \boldsymbol{P} \leq \Delta} \mathcal{L}(f_\theta(\boldsymbol{A} \oplus \boldsymbol{P}, \boldsymbol{X}))\,. \tag{1}$$

Here, $\oplus$ stands for an element-wise exclusive or and Δ denotes the edge budget (i.e. the number of altered entries in the perturbed adjacency matrix). Naively, applying R-BCD to optimize towards the dense adjacency matrix would only save some computation on obtaining the respective gradient. It still has a space complexity of $\mathcal{O}(n^2)$ on top of the complexity of the attacked model because we still have to store up to n^2 parameters. Note that the L_0 perturbation constraint with limited budget Δ implies that the solution will be sparse. We build upon this fact and in each epoch, in a survival-of-the-fittest manner, we keep that part of the search space which is "promising" and resample the rest. Despite the differences, we simply call our approach **Projected Randomized Block Coordinate Descent (PR-BCD)** and provide the pseudo code in Algo. 1. On top of the GNN, PR-BCD comes with space complexity of $\Theta(b)$ where b is the block size (number of coordinates) since *everything can be implemented efficiently with sparse operations*. We typically choose Δ to be a fraction of m and $b > \Delta$, thus, in practice, we have a linear overhead.

PR-BCD. For L_0-norm PGD we relax the discrete edge perturbations $\boldsymbol{P}$ from $\{0,1\}^{(n \times n)}$ to $[0,1]^{(n \times n)}$ as proposed by Xu et al. [36]. Each entry of $\boldsymbol{P}$ denotes the probability for flipping it. In each epoch we only look at a randomly sampled, non-contiguous block of $\boldsymbol{P}$ of size b (line 3, line 10-13) and additionally ignore the diagonal elements (i.e. self-loops). If using an undirected graph, the potential edges are restricted to the upper/lower triangular $n \times n$ matrix. In each epoch $t \in \{1, 2, \dots\}$, $\boldsymbol{p}$ is added to / subtracted from the discrete edge weight (line 6). Note, we overload $\oplus$ s.t. $\boldsymbol{A}_{ij} \oplus p_{ij} = \boldsymbol{A}_{ij} + p_{ij}$ if $\boldsymbol{A}_{ij} = 0$ and $\boldsymbol{A}_{ij} - p_{ij}$ otherwise. We use $\boldsymbol{p}$ and $\boldsymbol{P}$ interchangeably while $\boldsymbol{p}$ only corresponds to the current subset/block of $\boldsymbol{P}_{\boldsymbol{i}_t}$. After each gradient update (line 7), the projection $\Pi_{\mathbb{E}[\text{Bernoulli}(\boldsymbol{p})]=\Delta}(\boldsymbol{p})$ adjusts the probability mass such that $\mathbb{E}[\text{Bernoulli}(\boldsymbol{p})] = \sum_{i \in b} p_i \approx \Delta$ and that $\boldsymbol{p} \in [0,1]$ (line 8). In the end we draw b sample s.t. $\boldsymbol{P} \in \{0,1\}^{(n \times n)}$ via $\boldsymbol{P} \sim \text{Bernoulli}(\boldsymbol{p})$ (line 16).

The projection $\Pi_{\mathbb{E}[\text{Bernoulli}(\boldsymbol{p})]=\Delta}(\boldsymbol{p})$ likely results in many zero elements, but is not guaranteed to be sparse (for details see § A.3.1). If $\boldsymbol{p}$ has more than 50% non-zero entries, we remove the entries with the lowest probability mass such that 50% of the search space is resampled. Otherwise, we resample all zero entries in $\boldsymbol{p}$. However, one also might apply a more sophisticated heuristic $h(\boldsymbol{p})$ which we leave for future work (see line 11). After $E_{\text{res.}}$ epochs we fine-tune $\boldsymbol{p}$, i.e. we stop resampling and decay the learning rate as in [36]. We also employ early stopping for both stages ($t \leq E_{\text{res.}}$ and $t > E_{\text{res.}}$ with the epoch t) such that we take the result of the epoch with highest loss $\mathcal{L}$.



(a) Cora ML ($n = 2.8k$) (b) arXiv ($n = 170k$)

Figure 3: Influence of block size b on PR-BCD (dashed its dense equivalent [36]) with tanh Margin loss and $\epsilon = 0.1$. (a) shows pert. accuracy with its three-sigma error over five seeds. We resample $E_{\text{res.}} = 50$ epochs and then fine-tune for 250. (b) shows pert. accuracy over epochs t. We choose E such that $E \cdot b = \text{const}$.

Algorithm 1 Projected and Randomized Block Coordinate Descent (PR-BCD)

Input: Gr. $(\boldsymbol{A}, \boldsymbol{X})$, lab. $\boldsymbol{y}$, GNN $f_\theta(\cdot)$, loss $\mathcal{L}$
Parameter: budget Δ, block size b, epochs E & $E_{\text{res.}}$, heuristic $h(\dots)$, learning rate α_t
Draw w/o replacement $\boldsymbol{i}_0 \in \{0, 1, \dots, n^2 - 1\}^b$
Initialize zeros for $\boldsymbol{p}_0 \in \mathbb{R}^b$
for $t \in \{1, 2, \dots, E\}$ **do**
  $\hat{\boldsymbol{y}} \leftarrow f_\theta(\boldsymbol{A} \oplus \boldsymbol{p}_{t-1}, \boldsymbol{X})$
  $\boldsymbol{p}_t \leftarrow \boldsymbol{p}_{t-1} + \alpha_{t-1} \nabla_{\boldsymbol{i}_{t-1}} \mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$
  Projection $\boldsymbol{p}_t \leftarrow \Pi_{\mathbb{E}[\text{Bernoulli}(\boldsymbol{p}_t)] = \Delta}(\boldsymbol{p}_t)$
  $\boldsymbol{i}_t \leftarrow \boldsymbol{i}_{t-1}$
  if $t \leq E_{\text{res.}}$ **then**
    $\text{mask}_{\text{res.}} \leftarrow h(\boldsymbol{p}_t)$
    $\boldsymbol{p}_t[\text{mask}_{\text{res.}}] \leftarrow \boldsymbol{0}$
    Resample $\boldsymbol{i}_t[\text{mask}_{\text{res.}}]$
  end if
end for
$\boldsymbol{P} \sim \text{Bernoulli}(\boldsymbol{p}_t)$ s.t. $\sum \boldsymbol{P} \leq \Delta$
Return $\boldsymbol{A} \oplus \boldsymbol{P}$

Block size b. With growing n it is unrealistic that each possible entry of the adjacency matrix was part of at least one random search space of (P)R-BCD. As is apparent, with a constant search space size, the number of mutually exclusive chunks of the perturbation matrix grows with $\Theta(n^2)$ and this would imply a quadratic runtime. However, as evident in randomized black-box attacks [31], it is not necessary to test every possible edge to obtain an effective attack. In Fig. 3 (a), we analyze the influence of the block size b on the perturbed accuracy. On small datasets and over a wide range of block sizes b, our method performs comparably (or sometimes even better) to its dense equivalent. For larger graphs, we observe that the block size b has a stronger influence on the perturbed accuracy. However, as shown in Fig. 3 (b), one might increase the number of epochs for an improved attack strength. This indicates that PR-BCD successfully identifies the harmful edges to keep.

GR-BCD. As an alternative to PR-BCD, we propose Greedy R-BCD (GR-BCD) which greedily flips the entries with the largest gradient in the block so that after E iterations the budget is met. It is even a little bit more scalable as it does *not* require $b > \Delta$ (see § A.3.2 for details).

Limitations. We solely propose approximate attacks that do not provide any guarantee on how well they approximate the actual optimization problem and, hence, only provide an *upper* bound on e.g. the perturbed accuracy. We also recommend to monitor the relaxation error. One could use certificates to get the respective lower bound, provided they were scalable enough. Even sparse smoothing [2] might be too slow since we need many forward passes. As our attacks rely on the gradient they also require that the victim model is (approximately) differentiable. Otherwise, the approximation can become inappropriate. Moreover, we are limited by the scalability of the attacked GNN as we discuss next.

Scalable GNNs. Up to now, we implicitly assumed that we have enough memory to obtain the predictions and gradient towards the edges. GNNs that typically process the whole graph "at once", are inherently limited in their scalability. Our PR-BCD attack is even applicable when operating at those limits (see experiments on Products in § 5). To push the limits further, we now consider more scalable GNNs. Some notable scalable GNNs either sample subgraphs [5, 10] or, such as PPRGo [3], simplify the message passing operation. Next, we extend our PR-BCD to a local attack on PPRGo with *constant complexity including the (Soft Median) PPRGo* (for the Soft Median PPRGo see § 4).

Attacking PPRGo. To scale to massive graphs effectively, we need to obtain sublinear/constant complexity w.r.t. the number of nodes. This severely restricts the possibilities of how one might approach a global attack and is the reason why we now focus on local attacks. For an L-layer message passing GNN we need to recursively compute the L-hop neighborhood to obtain the prediction of a single node. This makes it difficult to obtain a sublinear space complexity (here including the GNN)—especially if one considers arbitrary edge insertions. In contrast, PPRGo [3] leverages the Personalized Page Rank (PPR) matrix $\Pi = \alpha(\boldsymbol{I} - (1-\alpha)\boldsymbol{D}^{-1}\boldsymbol{A})^{-1}$ (row normalization) to reduce the number of explicit message passing steps to one (see Eq. 6). We use the Sherman-Morrison

formula for a closed-form, rank-one update of the PPR matrix:

$$\tilde{\Pi}_i = \alpha \left(\Pi'_i - \frac{\Pi'_{ii} \boldsymbol{v} \Pi'}{1 + \boldsymbol{v} \Pi'_{:i}} \right) \tag{2}$$

where $\Pi' = \alpha^{-1}\Pi$ and, with degree matrix $\boldsymbol{D}$, $\boldsymbol{v} = (\boldsymbol{D}_{ii} + \sum \boldsymbol{p})^{-1}(\boldsymbol{A}_i + \boldsymbol{p}) - \boldsymbol{D}_{ii}^{-1}\boldsymbol{A}_i$. This suffices to attack incoming edges of a node and since everything is differentiable ($\partial\mathcal{L}'/\partial\boldsymbol{p}$) we do not need a surrogate model (common practice [30, 42]). In § A.3.3, we give details on the derivation and show how we can leverage the fact that PPRGo uses a *top-k-sparsified* PPR matrix to obtain constant complexity $\mathcal{O}(b^2k)$ (assuming $b \ll n$ and $k \ll n$). With $\Delta < b$, our approach comes with no restriction on how we can insert or remove incoming edges of a specific node. Other approaches such as [11] gain scalability via restricting the set of admissible nodes for edge perturbations.

4 Scalable Defense

To complete the robustness picture we now shift focus to defenses. Unfortunately, we are not aware of any defense that scales to graphs significantly larger than PubMed. Thus, we propose a novel, scalable defense based on a robust message-passing aggregation, relying on recent advancements in differentiable sorting [26]. Our method *Soft Median* performs similarly to Geisler et al. [14]'s Soft Medoid, but comes with better complexity w.r.t. the neighborhood size, lower memory footprint, and enables us to scale to bigger graphs. We can also use this aggregation neatly in the PPRGo architecture resulting in the *first defense that scales to massive graphs with over 111M nodes*.

Background and related work. We typically have the message passing framework of a GNN:

$$\mathbf{h}_v^{(l)} = \sigma^{(l)} \left[\mathrm{AGG}^{(l)} \left\{ \left(\boldsymbol{A}_{vu}, \mathbf{h}_u^{(l-1)} \boldsymbol{W}^{(l)} \right), \forall\, u \in \mathbb{N}'(v) \right\} \right] \tag{3}$$

with the neighborhood $\mathbb{N}'(v) = \mathbb{N}(v) \cup v$ including the node itself, some message passing aggregation $\mathrm{AGG}^{(l)}$ of the l-th layer, the embedding $\mathbf{h}_v^{(l)}$, the normalized message passing matrix $\boldsymbol{A}$, the weights $\boldsymbol{W}^{(l)}$, and activations $\sigma^{(l)}$. We classify defenses into three categories: (1) preprocessing [13, 16, 33], (2) robust training [8, 36, 41], and (3) modifications of the architecture [14, 17, 29, 34, 38–40]. Our attacks lend themselves well for adversarial training but we leave it for future work due to the overhead during training. Instead, we build the observation of Geisler et al. [14] that common aggregations (e.g. sum or mean) in Eq. 3 are known to be non-robust. They propose a differentiable robust aggregation for $\mathrm{AGG}^{(l)}$ and call it Soft Medoid. It is a continuous relaxation of the Medoid and requires the row/column sum over the distance matrix of the embedding of the nodes in the neighborhood. Hence this operation has a quadratic complexity w.r.t. the neighborhood size and comes with a sizable memory overhead during training and inference.

Soft Median. We leverage two key facts to improve Soft Medoid. First, the Medoid is a multivariate generalization of the Median. Here we look into an alternative that is the dimension-wise Median. Second, we do not need to sort all inputs to obtain the median. This principle can be generalized to soft sorting which is a differentiable relaxation of the sort operation. In Eq. 4, we propose a differentiable relaxation of the Median that softly selects the embedding which is closest to the dimension-wise Median $\bar{\boldsymbol{x}}$, normalized by the dimensions d. We define the distances $\boldsymbol{d}$ as $d_v = \|\bar{\boldsymbol{x}} - \boldsymbol{X}_{v,:}\|$ and use $\boldsymbol{X}$ as well as $\mathbb{X}$ interchangably.

$$\mu_{\text{SoftMedian}}(\boldsymbol{X}) = \mathrm{softmax}\left(-\boldsymbol{d}/T\sqrt{d}\right)^\top \boldsymbol{X} = \boldsymbol{s}^\top \boldsymbol{X} \approx \arg\min_{\boldsymbol{x}' \in \mathbb{X}} \|\bar{\boldsymbol{x}} - \boldsymbol{x}'\|, \tag{4}$$

The temperature. The temperature parameter T controls the steepness of the weight distribution $\boldsymbol{s}$ between the neighbors. In the extreme case as $T \to 0$ we recover the instance which is closest to the dimension-wise Median (i.e. $\arg\min_{\boldsymbol{x}' \in \mathbb{X}} \|\bar{\boldsymbol{x}} - \boldsymbol{x}'\|$). In the other extreme case $T \to \infty$, the Soft Median is equivalent to the sample mean. We observe a similar empirical behavior as Geisler et al. [14] and we decide on a temperature value in our experiments by grid search.

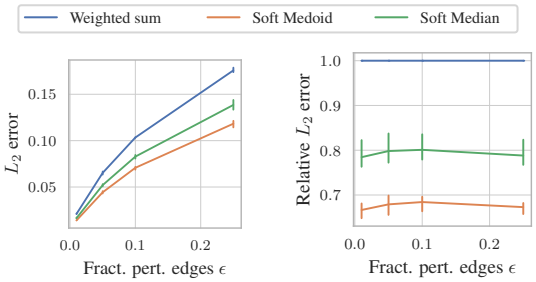
Breakdown point. For any finite T, our proposed Soft Median has the best possible breakdown point of 0.5 as we state formally in Theorem 1 (for proof see § A.4.1). Note that despite the lower complexity compared to Soft Medoid, we maintain the same breakdown point:

Theorem 1 *Let $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a collection of points in $\mathbb{R}^d$ with finite coordinates and temperature $T \in [0, \infty)$. Then the Soft Median location estimator (Eq. 4) has the finite sample breakdown point of $\epsilon^*(\mu_{\text{Soft Median}}, \boldsymbol{X}) = 1/n \lfloor (n+1)/2 \rfloor$ (asympt. $\lim_{n\to\infty} \epsilon^*(\mu_{\text{SoftMedian}}, \boldsymbol{X}) = 0.5$).*

We define the *weighted* Soft Median as:

$$\mu_{\text{WSM}}(\boldsymbol{X}, \mathbf{a}) = c\,(\boldsymbol{s} \circ \mathbf{a})^\top \boldsymbol{X} \tag{5}$$

where $\boldsymbol{s}$ is the softmax weight of Eq. 4 obtained using the weighted dimension-wise Median and c is a normalization s.t. $\sum \boldsymbol{s} \circ \mathbf{a} = \sum \mathbf{a}$. $\mathbf{a}$ stands for the edges weights in the neighborhood. This is similar to [14] and we recover the message passing operation of a GCN [19] for $T \to \infty$.



(a) absolute bias (b) relative bias

Figure 4: Empirical bias $B(\epsilon)$ for the second layer of a GCN with GDC preoroc. [20] network under PGD attack with L_2 distance. We use $T = 0.2$ and a relative budget of $\epsilon = 0.25$.

Empirical robustness. The optimal breakdown point does not necessarily imply that the proposed aggregation is more robust for finite perturbations. In Fig. 4, we analyze the L_2 distance in the latent space after the first message passing operation for a clean vs. perturbed graph. Empirically the Soft Median has a 20% lower error than the weighted sum (we call it sum since the weights do not sum up to 1). While here the Soft Medoid seems to be more robust, this is not consistent with the perturbed accuracy values in § 5.

Our Soft Median aggregation is highly scalable in combination with PPRGo (with feat. encoder f_{enc}):

$$\boldsymbol{p} = \text{softmax}\left[\text{AGG}\left\{\left(\pi(v)_u, f_{\text{enc}}(\boldsymbol{x}_u)\right), \forall\, u \in \mathbb{N}'(v)\right\}\right] \tag{6}$$

Limitations. Our Soft Median has the best possible breakdown point which is a (well-established) indicator for robustness but does not prove adversarial robustness. As for most defenses, ours can provide a false sense of robustness. If possible, use attacks and certification techniques to verify the application specific efficacy. Similarly to the Soft Medoid we expect that our Soft Median can also improve the certified robustness [14, 27]. Naturally, our Soft Median also comes with higher cost than e.g. a naïve summation despite having the same asymptotic complexity. Nevertheless, the overhead seems to be reasonable as we show in our experiments and in combination with PPRGo one can mitigate the slightly higher memory requirements with a smaller batch size.

5 Empirical Evaluation

In the following, we present our experiments (consisting of approx. 2,500 runs) to show the efficacy and scalability of our methods on the six graphs detailed in Table 1. We open source the code with configurations that reproduce the results and provide it here anonymously. **Attacks:** We benchmark **our** GR-BCD and PR-BCD against *PGD* [36], *greedy FGSM* (similar to Dai et al. [11]) as well as *DICE* [31]. **Defenses:** Besides regular/vanilla GNNs we compare **our** Soft Median GDC/PPRGo with *Soft Medoid GDC* [14], *SVD GCN* [13], *RGCN* [40], and *Jaccard GCN* [33]. We follow Soft Medoid GDC [14] and diffuse the adjacency matrix with PPR/GDC [20]. For the OGB datasets we use the public splits and otherwise sample 20 nodes per class for training/validation. The full setup and details about baselines are given in § A.5.1. We typically report the average over three random seeds/splits and the 3-sigma error of the mean.

Table 1: Dataset summary. For the dense adjacency matrix we assume 4 bytes per entry. We represent the sparse (COO) matrix via two 8 byte integer pointers and a 4 bytes float value per edge. We highlight configurations above 30 GB.

Dataset	**#Nodes** n	**Size (dense)**	**Size (sparse)**
Cora ML [1]	2.8 k	35.88 MB	168.32 kB
Citeseer [23]	3.3 k	43.88 MB	94.30 kB
PubMed [28]	19.7 k	1.56 GB	1.77 MB
arXiv [15]	169.3 k	**114.71 GB**	23.32 MB
Products [15]	2.4 M	**23.99 TB**	2.47 GB
Papers 100M [15]	111.1 M	**49.34 PB**	**32.31 GB**

Time and memory cost. We want to stress again that most of the baselines barely scale to PubMed using a common 11GB GeForce GTX 1080 Ti (as we do). We only use a 32GB Tesla V100 for the experiments on Products with a full-batch GNN, since a three-layer GCN requires roughly 30 GB already during training. Extrapolating the overhead on PubMed to the largest dataset, Papers 100M, *traditional attacks and defenses would require roughly 1 exabyte* (10^{18} *bytes) while for ours 11 GB suffice*. Our attacks and defenses are also reasonably fast. On arXiv (170 k nodes), we train

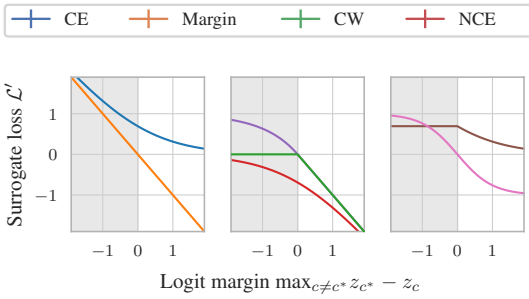
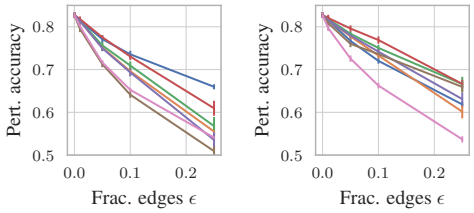


Figure 5: Losses for the binary case. The losses are grouped via their basic properties (see text).



(a) FGSM (b) PGD

Figure 6: Attacking GCN on Cora ML. The lower p. acc. the better the loss.

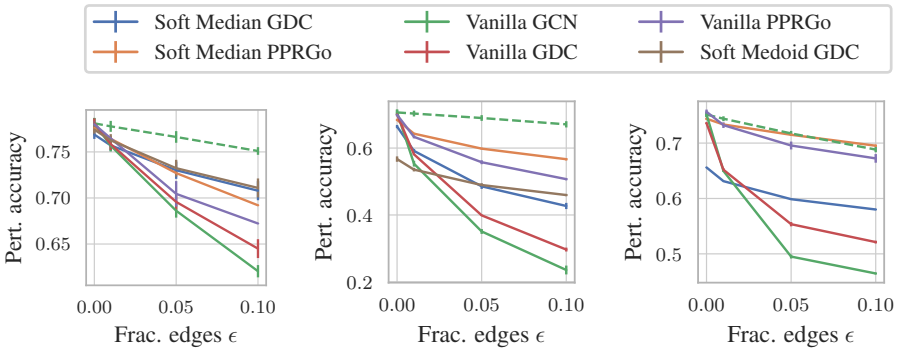
for 500 epochs and run the global PR-BCD attack for 500 epochs. The whole training and attacking procedure requires less than 2 minutes. Moreover, one epoch on Papers 100M with the local PR-BCD attack takes less than 10 seconds. For further details see § A.5.2.

Surrogate Loss. We illustrate the losses in Fig. 5, where we clustered the losses in three groups. (1) incentivizing very low margins: Cross Entropy CE and Margin. (2) focusing on high-confidence nodes: Carlini-Wagner CW, the (neg.) CE of the most-likely, non-target class NCE, and ELU Margin. (3) focusing on nodes close to the decision boundary: MCE **(ours)** and tanh Margin **(ours)**. In Fig. 6, we see that the losses of category (3), or equivalently obeying the properties of Definition 1 or Definition 2, are superior to the other losses. For example, with MCE and FGSM the accuracy drops twice as much as with CE. A detailed discussion and mathematical formulation of all losses can be found in § A.2. Additionally, we report further experiments backing our claims and discuss the losses' properties in more detail. Subsequently, we use MCE for greedy attacks and tanh Margin otherwise.

Robustness w.r.t. global attacks. In Table 2, we present the experimental results for our proposed global attacks on the small dataset Cora ML since most baselines do not scale much further. Our attacks are as strong as their dense equivalents despite being much more scalable. In Fig. 7, we compare our PR-BCD attack on all baselines that fit into memory or can be trained within 24 hours on the bigger datasets. Our defense Soft Median GDC and Soft Median PPRGo are consistently among the best models tested over all scales. For example with $\epsilon = 0.1$, the accuracy of a Vanilla GCN drops by an absolute 20% while for the Soft Median PPRGo we only lose 5%. To fit our Soft Median GDC on Products into memory, we had to reduce the number of hidden dimensions in comparison to its baselines. However, note that even a Vanilla GCN requires almost the entire memory of the 32 GB GPU. Despite the small sacrifice in clean accuracy, we already outperform most baselines for a budget of $\epsilon > 0.01$. We also faced similar scaling limitations for the Soft Medoid GDC baseline on arXiv. This highlights the lower memory requirements for our Soft Median. In § A.5.3, we give experiments on more budgets, further baselines, and also present adaptive/direct attacks for the models that allow it.

Table 2: Comparing attacks (transfer from Vanilla GCN) and defenses. We show the perturbed accuracy for $\epsilon = 0.1$ on Cora ML, and the clean test accuracy (last column). We only highlight the **strongest defense** as the attacks perform similarly. Our approaches are underlined. See § A.5.3 for more datasets, budgets, and adaptive/direct attacks.

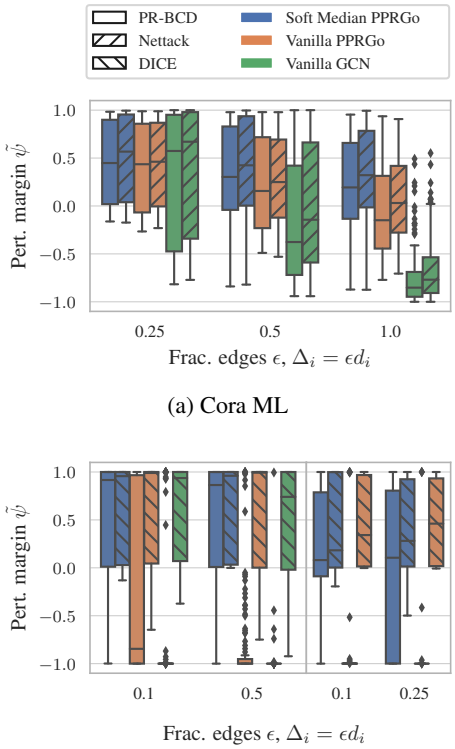
Attack	FGSM	GR-BCD	PGD	PR-BCD	Acc.
Soft Median GDC	0.769 ± 0.002	0.765 ± 0.001	0.758 ± 0.002	0.752 ± 0.002	0.824 ± 0.002
Soft Median PPRGo	$\mathbf{0.778 \pm 0.001}$	$\mathbf{0.781 \pm 0.002}$	$\mathbf{0.769 \pm 0.001}$	$\mathbf{0.770 \pm 0.001}$	0.821 ± 0.001
Vanilla GCN	0.641 ± 0.003	0.622 ± 0.003	0.662 ± 0.003	0.645 ± 0.002	0.827 ± 0.003
Vanilla GDC	0.672 ± 0.005	0.677 ± 0.005	0.679 ± 0.002	0.674 ± 0.004	$\mathbf{0.842 \pm 0.003}$
Vanilla PPRGo	0.724 ± 0.003	0.726 ± 0.002	0.704 ± 0.001	0.700 ± 0.002	0.826 ± 0.002
Soft Medoid GDC	0.773 ± 0.005	0.775 ± 0.003	0.759 ± 0.003	0.761 ± 0.003	0.819 ± 0.002
SVD GCN	0.751 ± 0.007	0.755 ± 0.006	0.719 ± 0.005	0.724 ± 0.006	0.781 ± 0.005
Jaccard GCN	0.661 ± 0.002	0.664 ± 0.001	0.673 ± 0.002	0.667 ± 0.003	0.818 ± 0.003
RGCN	0.654 ± 0.007	0.665 ± 0.005	0.671 ± 0.007	0.664 ± 0.004	0.819 ± 0.002



(a) Pubmed (b) arXiv (c) Products

Figure 7: PR-BCD (DICE dashed) on the large datasets (transfer).

Robustness w.r.t. local attacks. In Fig. 8, we compare the results of our local PR-BCD with Nettack on Cora ML (undirected). We define the budget $\Delta_i = \epsilon d_i$ and select the nodes for each budget s.t. $\Delta_i \geq 1$. Similarly to Zügner et al. [42], we apply the attack only to the 10 nodes with highest confidence, 10 with lowest, and 20 random nodes (all correctly classified). For more datasets and budgets see § A.5.4. Our attack seems to be slightly stronger than Nettack on all architectures and budgets. Nettack and PR-BCD use a different strategy to make the original combinatorial optimization problem feasible (see § 1). Nettack uses a linearlized surrogate model to select the adversarial edges. Evidently, this leads to a weaker attack compared to relaxing the optimization problem as we proposed with PR-BCD. On the large datasets Products and Papers 100M (directed), we outperform the simple DICE baseline substantially. We compare to DICE since Nettack is not scalable enough. In comparison to the small datasets, the Vanilla GCN/PPRGo are extremely fragile and much lower budgets Δ_i suffice to flip almost every node's prediction. Our proposed defense Soft Median PPRGo on the other hand remains similarly robust as on the small datasets. On Papers 100M with $\Delta_i = 0.25$, the Soft Median PPRGo reduces the attacker's success rate from around 90% to just 30% (90% vs. 1% on Products with $\Delta_i = 0.5$).



(a) Cora ML

(b) Products (left) & Papers100M (right)

Figure 8: Perturbed classification margins $\tilde{\psi}_i$ of the attacked nodes. In (a)-(d), we compare our local PR-BCD attack with Nettack [42] on (undirected) Cora ML and Citeseer. In (e) and (f), we show the results on the (directed) large-scale datasets Products (2.5 million nodes) and Papers 100M (111 million nodes), respectively. Our Soft Medoid PPRGo resists the attacks much better than the baselines.

6 Conclusion

We study the adversarial robustness of GNNs at scale. We tackle all three of the identified challenges: (1) we introduce *budget-aware* surrogate losses that can double the attack strength, (2) we principally scale first-order attacks that optimize over the quadratic number of possible edges, and (3) we propose a scalable defense using our novel Soft Median which is differentiable as well as provably robust. We show that our attacks and defenses are practical by scaling to graphs of up to 111 million nodes. In some settings our defense reduces the attack's success rate from around 90 % to 1 %. Most importantly, our work enables the assessment of adversarial robustness for real-world massive-scale applications with GNNs.

References

[1] A. Bojchevski and S. Günnemann. Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking. *6th International Conference on Learning Representations, ICLR*, 2018.

[2] A. Bojchevski, J. Klicpera, and S. Günnemann. Efficient Robustness Certificates for Graph Neural Networks via Sparsity-Aware Randomized Smoothing. *37th International Conference on Machine Learning, ICML*, 2020.

[3] A. Bojchevski, J. Klicpera, B. Perozzi, A. Kapoor, M. Blais, B. Rózemberczki, M. Lukasik, and S. Günnemann. Scaling Graph Neural Networks with Approximate PageRank. *ACM SIGKDD 2020*, 2020.

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

[5] J. Chen, T. Ma, and C. Xiao. FASTGCN: Fast learning with graph convolutional networks via importance sampling. *6th International Conference on Learning Representations, ICLR*, 2018.

[6] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan. Fast gradient attack on network embedding. *arXiv*, 2018.

[7] J. Chen, Y. Chen, H. Zheng, S. Shen, S. Yu, D. Zhang, and Q. Xuan. MGA: Momentum Gradient Attack on Network. *IEEE Transactions on Computational Social Systems*, 2020.

[8] 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.

[9] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training Deep Nets with Sublinear Memory Cost. *arXiv preprint arXiv:1604.06174*, 2016.

[10] W. L. Chiang, Y. Li, X. Liu, S. Bengio, S. Si, and C. J. Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. *International Conference on Knowledge Discovery and Data Mining, KDD*, 2019.

[11] H. Dai, H. Li, T. Tian, H. Xin, L. Wang, Z. Jun, and S. Le. Adversarial attack on graph structured data. *35th International Conference on Machine Learning, ICML*, 3, 2018.

[12] D. Donoho and P. J. Huber. The notion of breakdown point. In *A Festschrift For Erich L. Lehmann*. Wadsworth Statist./Probab. Ser., Wadsworth, Belmont, CA, 1983, 1983.

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

[14] S. Geisler, D. Zügner, and S. Günnemann. Reliable Graph Neural Networks via Robust Aggregation. *Neural Information Processing Systems, NeurIPS*, (NeurIPS), 2020.

[15] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. 2020.

[16] M. Jin, H. Chang, W. Zhu, and S. Sojoudi. Power up! Robust Graph Convolutional Network against Evasion Attacks based on Graph Powering. *AAAI Conference on Artificial Intelligence*, 2019.

[17] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang. Node Similarity Preserving Graph Convolutional Networks. *WSDM 2021 - Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021.

[18] W. Jin, Y. Li, H. Xu, Y. Wang, and J. Tang. Adversarial Attacks and Defenses on Graphs. *ACM SIGKDD Explorations Newsletter*, 2021.

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

[20] J. Klicpera, S. Weißenberger, and S. Günnemann. Diffusion Improves Graph Learning. *Neural Information Processing Systems, NeurIPS*, 2019.

[21] J. Li, T. Xie, L. Chen, F. Xie, X. He, and Z. Zheng. Adversarial attack on large scale graph. *arXiv*, (i), 2020.

[22] J. Ma, S. Ding, and Q. Mei. Towards More Practical Adversarial Attacks on Graph Neural Networks. *Neural Information Processing Systems, NeurIPS*, (NeurIPS), 2020.

[23] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 2000.

[24] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 22, 2012.

[25] Y. Nesterov and S. Stich. Efficiency of accelerated coordinate descent method on structured optimization problems. *Siam J. Optim.*, 27, 2017.

[26] S. Prillo and J. Martin Eisenschlos. SoftSort: A Continuous Relaxation for the argsort Operator. *37th International Conference on Machine Learning, ICML*, 119, 2020.

[27] J. Schuchardt, A. Bojchevski, J. Klicpera, and S. Günnemann. Collective robustness certificates. *9th International Conference on Learning Representations, ICLR*, 2021.

[28] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 2008.

[29] X. Tang, Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang. Transferring robustness for graph neural network against poisoning attacks. *Conference on Web Search and Data Mining, WSDM*, 2020.

[30] J. Wang, M. Luo, F. Suya, J. Li, Z. Yang, and Q. Zheng. Scalable attack on graph data by injecting vicious nodes. *Data Mining and Knowledge Discovery*, 34(5), 2020.

[31] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2), 2018.

[32] S. J. Wright. Coordinate Descent Algorithms. *Mathematical Programming*, 151, 2015.

[33] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu. Adversarial examples for graph data: Deep insights into attack and defense. *IJCAI International Joint Conference on Artificial Intelligence*, 2019-Augus, 2019.

[34] T. Wu, H. Ren, P. Li, and J. Leskovec. Graph Information Bottleneck. *Neural Information Processing Systems, NeurIPS*, (NeurIPS), 2020.

[35] K. Xu, C. Li, Y. Tian, T. Sonobe, K. I. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. *35th International Conference on Machine Learning, ICML*, 12, 2018.

[36] K. Xu, H. Chen, S. Liu, P. Y. Chen, T. W. Weng, M. Hong, and X. Lin. Topology attack and defense for graph neural networks: An optimization perspective. *IJCAI International Joint Conference on Artificial Intelligence*, 2019-Augus, 2019.

[37] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[38] X. Zhang and M. Zitnik. GNNGuard: Defending Graph Neural Networks against Adversarial Attacks. *Neural Information Processing Systems, NeurIPS*, (NeurIPS), 2020.

[39] Y. Zhang, S. Pal, M. Coates, and D. Ustebay. Bayesian Graph Convolutional Neural Networks for Semi-Supervised Classification. *AAAI Conference on Artificial Intelligence*, 33, 2019.

[40] D. Zhu, P. Cui, Z. Zhang, and W. Zhu. Robust graph convolutional networks against adversarial attacks. *International Conference on Knowledge Discovery and Data Mining, KDD*, 2019.

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

[42] D. Zügner, A. Akbarnejad, and S. 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 § 3 and § 4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See § 1.
   (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? [Yes] For losses see § 2 and for the breakdown point of the Soft Median see § A.4.1.
   (b) Did you include complete proofs of all theoretical results? [Yes] For proofs see § A.2.1 and § A.4.1.
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] We give brief summary in the main paper and full details in § A.5.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] They are either presented in the main text or in the respective section in the appendix.
   (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 § A.5.1.
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] E.g. see Table 1.
   (b) Did you mention the license of the assets? [Yes] We give available information in § A.5.1.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code will be open sourced presumably MIT-licensed.
   (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]