Robust Graph Neural Networks at Scale

ABSTRACT

Adversarial robustness of Graph Neural Networks (GNNs) has become exceedingly important due to the popularity and the diverse applications of GNNs. Even though it is challenging to design attacks because of the discrete optimization domain, structure perturbations can affect a GNN severely. All existing adversarial attacks for structure perturbations that rely on first-order optimization require a dense adjacency matrix and, therefore, can only be applied to small graphs (space complexity $\Theta(n^2)$ in the number of nodes n). In this work, we scale adversarial attacks to evaluate GNNs in settings closer to practice, i.e. on massive graphs. First, we show that the widely used surrogate losses are not well-suited for global attacks on GNNs which becomes particularly apparent on large graphs. We provide two alternatives that overcome these limitations. Second, we propose two attacks based on first-order optimization that do not require a dense adjacency matrix and come with linear space complexity. We use our methods for global attacks on graphs more than 100 times larger than previously evaluated. Moreover, we adapt our attack to a scalable GNN, namely PPRGo, which allows us to scale to even larger graphs. We propose a differentiable robust aggregation function, Soft Median, which we use to improve one of the most effective defense strategies for GNNs. Also PPRGo can be equipped with our aggregation and allows us to scale almost indefinitely.

ACM Reference Format:

. 2021. Robust Graph Neural Networks at Scale. In *Proceedings of 27th ACM SIGKDD Conference On Knowledge Discovery and Data Mining (KDD '21).* ACM, New York, NY, USA, 9 pages. https://doi.org/TBD

1 INTRODUCTION

The adversarial robustness of GNNs has been widely studied in recent research beginning with [10, 34]. However, most previous work largely focused on graphs with less than 20,000 nodes. In particular, from the perspective of real-world internet-scale applications, those graphs are tiny. To attack a small dataset such as PubMed (19,717 nodes) [22], typically around 20 GB is required if using a dense adjacency matrix. We argue that such memory requirements are impractical, hinder practitioners to assess adversarial robustness, and limit advancements of the field. In this work, we set the foundation for the holistic study of adversarial robustness of GNNs on real-world social/citation networks. We study graphs with up to 111 million nodes and therefore assess the adversarial robustness of GNNs on graphs 500 times larger than before. In Fig. 1, we compare the memory requirements of a previous attack with ours for attacking a GNN globally (i.e. aiming for a reduced accuracy). Analogously to our attacks, we can scale our defense also to graphs with 111 million nodes and beyond.

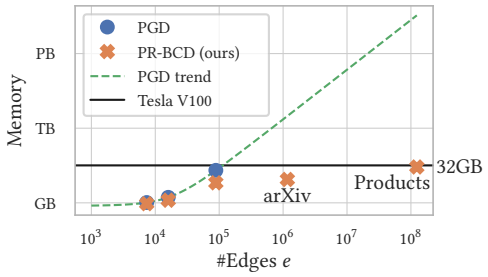


Figure 1: GPU memory consumption for a global attack with Projected Gradient Descent (PGD) [30], its quadratic extrapolation, and our Projected Randomized Block Coordinate Descent (PR-BCD) (Sec. 3.1). Both yield similar perturbed accuracy. Beyond attacks, our defense (Sec. 4) can also be applied on such scales. Using a scalable GNN and a local attack we can scale attacks and defenses to even larger graphs.

Scope. In this work, we focus on adversarial robustness w.r.t. structure perturbations of GNNs for node classification. The GNN $f(\boldsymbol{A}, \boldsymbol{X})$ is applied to a Graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ with adjacency matrix $\boldsymbol{A}$ and node attributes $\boldsymbol{X}$. We solely consider *evasion* (test time) attacks, but our approach could also serve as a building block in scaling *poisoning* (train time) attacks [33]. We distinguish between *local* attacks on a single or small group of nodes and *global* attacks that consider all or a large fraction of nodes with a shared global budget Δ denoting the maximum number of adversarial perturbations. It is apparent that local attacks are much easier to scale than global attacks since for a local attack we only need to obtain the prediction for a few (or a single) nodes. We study an adversary with perfect knowledge about the graph, GNN, and test labels. Even this white-box scenario has not been studied for large graphs and our primary goal is to empirically assess worst-case adversarial robustness.

Challenges. We identify three major challenges hindering the study of GNNs' adversarial robustness at scale: (1) Previous losses are not well-suited for global attacks on GNNs at scale. (2) Attacks on GNNs typically come with quadratic space complexity. (3) Virtually no adversarial defense exists for large graphs. We tackle all three in this paper.

Surrogate losses (1). We study the limitation of state of the art surrogate losses for attacking the accuracy of a GNN over all nodes, following [7, 28, 29, 33] (Sec. 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 [5] are bad surrogates for such global attacks. Our novel losses that overcome these limitations

KDD '21, August 14–18, 2021, Online

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/TBD

easily improve the strength of the attack by 100%. For larger datasets, we observe gains of more than 200%. Are GNNs perhaps even more fragile than previously believed?

Attacks (2). Scaling attacks is non-trivial. Attacks based on combinatorial approaches easily become computationally infeasible because of the vast amount of potential adjacency matrices ($O(2^{n^2})$). We can approximate the resulting discrete combinatorial optimization problem using first-order optimization attacks. Such attacks typically require the gradient towards all entries of the adjacency matrix, reducing the complexity to $\Theta(n^2)$ which is still not feasible for massive graphs. We propose two strategies to apply first-order optimization without the burden of a dense adjacency matrix. In Sec. 3.1, we describe how to add/remove edges between existing nodes based on Randomized Block Coordinate Descent (R-BCD) at an additional memory requirement of $O(\Delta)$[1]. Due to the limited scalability of traditional GNNs, we also consider the case where we attack PPRGo [4], a scalable GNN. We even obtain an algorithm with constant complexity w.r.t. the graph size.

Defense (3). We propose *Soft Median* – a computationally less demanding, robust, differentiable aggregation function inspired by Geisler et al. [13], by taking advantage of recent advancements in differentiable sorting [21]. Using Soft Median we observe similar robustness to [13], but with a significantly lower memory footprint, which enables us to defend GNNs at scale.

2 SURROGATE LOSSES

Most GNNs for node classification are soft classifiers and predict a probability or confidence score $\boldsymbol{p}_i = f(\boldsymbol{A}, \boldsymbol{X})_i$ for node i instead of the discrete classes. During training, we ideally wish to optimize a target metric which comes with a discontinuous loss (e.g. accuracy and the 0/1 loss $\mathcal{L}_{0/1}$). We typically use gradient methods to optimize our models and therefore commonly substitute the actual target loss by a differentiable *surrogate* $\mathcal{L}' \approx \mathcal{L}$, e.g., cross entropy for the 0/1 loss. The same is true for attacking a model to assess its *adversarial robustness*. In the context of images, typically a single example is attacked in isolation, which corresponds to a *local* attack for GNNs. In such a scenario it is often sufficient for an untargeted attack to *maximize* the cross entropy for the attacked node/image:

$$\mathrm{CE}^{(n)}(y, \boldsymbol{p}) = \sum_{c \in \mathbb{C}} \mathbb{I}[y^{(n)} = c] \log(p_c)^{(n)} = \log(p_{c^*}^{(n)}) \,. \qquad (1)$$

which corresponds to a minimization of the likelihood of the target class c^*. Many *global* attacks for GNNs [7, 28, 29, 33] maximize the cross entropy $\max_{\boldsymbol{A}} \mathrm{CE}(f_\theta(\boldsymbol{A}, \boldsymbol{X}))$ on the *whole dataset*, subject to a global budget constraint Δ. However, especially on large graphs, we often observed that the CE loss increases even though the accuracy does not decline. This can be explained by a bias of CE towards nodes which have a low confidence score, i.e. with CE we primarily attack nodes that are already misclassified. In other word those points have already a negative classification margin $\psi = \min_{c \neq c^*} p_{c^*} - p_c$ as we can see in Fig. 2.

In contrast to attacking a single image/node, a global attack has to (1) keep house with the budget Δ and (2) find edges that degrade the overall accuracy maximally (i.e. potentially target "fragile" nodes).

[1]The method in Sec. 3.1 was partially discussed at a non-archival workshop [1]. For double-blind rules we do not disclose the author list.

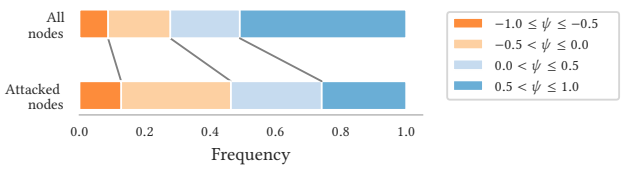


Figure 2: Relative distribution of classification margins ψ over all test nodes on the clean graph and the distribution of ψ among the nodes which where attacked when using CE. That is, we plot the confidence ψ of the nodes adjacent to the added/removed edge before the attack has even started. We use a small budget of one percent of edges ($\Delta = \epsilon = 0.01$) on the arXiv dataset (see Tab.2). As shown, among the attacked nodes, misclassified nodes (i.e., with negative margins) are highly overrepresented.

Without additional information, intuitively, one would first attack low-confidence nodes close to the decision boundary. To focus on nodes close to the decision boundary the surrogate loss should have a maximal gradient at $\psi = 0$. Moreover, if we solely want to attack the accuracy, then we can stop attacking a node once it is misclassified. More generally, we argue a *well-suited* surrogate loss for global attacks should have the subsequent properties:

(1) A *well-suited* surrogate loss only incentivizes perturbing nodes that are correctly classified: $\partial \mathcal{L}'/\partial p^*|_{p^*<0} \leq 0$.
(2) A *well-suited* surrogate loss favours points close to the decision boundary: $\partial \mathcal{L}'(y,\boldsymbol{p})/\partial p_{c^*}|_{\psi>0} > \partial \mathcal{L}'(y,\boldsymbol{p})/\partial p_{c^*}|_{\psi \to 0^+}$.

To overcome these limitations, we propose the Masked Cross Entropy: $\mathrm{MCE} = \frac{1}{|\mathbb{V}^+|} \sum_{n \in \mathbb{V}^+} \log(p_{c^*}^{(n)})$ enforces this by masking already wrongly classified nodes. Under certain conditions a *well-suited* surrogate loss (fulfills both properties) such the MCE even obtains the global optimum. On large graphs with small budgets, $\Delta/n \to 0$, most nodes become approximately independent since the receptive field becomes insignificant in comparison to the rest of the graph. Or in other words, which node to attack is rather a local instead of a global decision and, therefore, assume perfect independence for the following analysis. Likely, the budget required to change the prediction depends on the margin $\Delta_i = g(\psi_i)$, where here we assume an increasing linear function $g(\psi_i)$ (i.e. the larger the margin, the harder to attack). Under these assumptions, with a *well-suited* surrogate loss we will obtain the global optimum by the following greedy scheme:

Proposition 2.1. *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 joint budget Δ. Let $\Delta_i = g(|\psi_i|)$ be the budget required to move the prediction node i over the decision boundary and an attack of node i does not influence other nodes. Suppose we greedily attack nodes in order of $\mathcal{L}'(-\eta) - \mathcal{L}'(\psi_0) \geq \mathcal{L}'(-\eta) - \mathcal{L}'(\psi_1) \geq \cdots \geq \mathcal{L}'(-\eta) - \mathcal{L}'(\psi_l)$ until the budget is exceeded $\Delta < \sum_{i=0}^{l+1} \Delta_i$ with an arbitrarily small constant η. In this case, we obtain the global optimum of*

$$\max_{\tilde{\boldsymbol{A}} \text{ s.t. } \|\tilde{\boldsymbol{A}} - \boldsymbol{A}\|_0 < \Delta} \mathcal{L}_{0/1}(f_\theta(\tilde{\boldsymbol{A}}, \boldsymbol{X})) \qquad (2)$$

if $\mathcal{L}'$ has the properties (a) $\partial\mathcal{L}'/\partial p^*|_{p^*<0} \leq 0$ and (b) $\partial\mathcal{L}'(y,\boldsymbol{p})/\partial p_{c^*}|_{\psi>0} > \partial\mathcal{L}'(y,\boldsymbol{p})/\partial p_{c^*}|_{\psi\to 0^+}$.

Proof. We can easily see that this greedy solution obtains the optimal solution by an exchange argument. Lets suppose we are given the optimal plan σ^* and the greedy solution has the plan σ. Suppose σ^* would contain one or more tasks for that $w > l$ instead of $b \leq l$. We know that $\psi_w \geq \psi_b$ and hence $\Delta_w \geq \Delta_b$. Thus, replacing b by w would either lead to the an equally good or even better solution (contradiction!). Hence, the greedy plan σ is at least as good as the optimal plan σ^*. Moreover, the maximum is unique except for ties s.t. $\psi_i = \psi_j \geq 0, \forall i, j \in \mathbb{V}$.

Consequently, a surrogate loss $\mathcal{L}$' that leads to the order above will yield the global optimum as well. The order is preserved if (a) $\mathcal{L}'(-a) \leq \mathcal{L}'(-\eta)$ for every $a \in (\eta, 1]$ and (b) $\partial\mathcal{L}'/\partial p^*|_{p^*>0}$ is strictly monotonically decreasing or in other words $\mathcal{L}'$ is strictly concave for positive inputs. From this it follows that $\partial\mathcal{L}'/\partial p^*$ is minimal for $\psi \to 0^+$. □

Note, that even under those simplifying assumptions, the Cross Entropy (CE) violates property (2). CE is not guaranteed to obtain the global optimum since we possibly only perturb nodes that are already misclassified (see Fig. 2). Obviously, the Carlini Wagner (CW) [5] loss CW = $(\min_{c\neq c^*} z_{c^*} - z_c)+$ violates property (1). It is also not guaranteed to obtain the global optimum, since the CW loss does not focus on nodes close to the decision boundary. In the worst case, an attack with CW possibly spends all its budget on confident nodes—without even flipping one.

On the other hand, the MCE fulfills both properties and reaches the global optimum. Empirically, for a greedy gradient-based attack the MCE comes with gains of up to 200% in strength (see Sec. 5). Surprisingly, if we apply it to an attack using Projected Gradient Descent (PGD), 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. Second, the assumption about independence between the attacked nodes may be too strong in practice. We 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. For this we relax property (1):

(3) A *well-suited* surrogate loss should saturate for confidently misclassified nodes: $\lim_{\psi\to -1^+} \mathcal{L}'(y, \boldsymbol{p}) = k < \infty$.

Additionally propose to use the tanh on the margin in logit space, i.e. tanh Margin = $\tanh(\min_{c\neq c^*} z_{c^*} - z_c)$ which obeys properties (2) and (3).

3 SCALABLE ATTACKS

Beginning with [10, 34], many adversarial attacks on the graph structure have been proposed [3, 23, 24, 28, 30, 33]. Gradient-based attacks such as Metattack [33] or integrated gradients [28] rely on the gradient towards all possible entries in the *dense* adjacency matrix $\boldsymbol{A}$ (quadratic space complexity) to solve the optimization problem for structure perturbations:

$$\max_{\boldsymbol{A}} \mathcal{L}(f_\theta(\boldsymbol{A}, \boldsymbol{X})) \tag{3}$$

with the trained network f_θ and (surrogate) loss function $\mathcal{L}$ (or $\mathcal{L}'$). Since most attacks come with very limited scalability (e.g. see Fig. 1), GNNs robustness on larger graphs has barely been studied so far. In Sec. 3.1, 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. We show in Sec. 3.2 how we can even maintain this scalability for a scalable GNN called PPRGo [4] and, hence, are almost unlimited by memory restrictions.

Related work. Dai et al. [10] scale their local reinforcement learning approach to a sparse graph for financial transactions with roughly 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 a 111M nodes graph and allow large budgets Δ. Li et al. [17] analyze their *local* adversarial attack on mini-batch techniques such as Cluster-GCN applied to a graph with around 200k nodes. We consider a wider class of Graph Neural Networks and we even scale our *global* attack to a graph ten times larger. With PPRGo we even outscale them by factor of 500.

Large scale optimization. In a big data setting, 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 [27]. Nesterov [19] 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 [20].

3.1 Projected Randomized Block Coordinate Descent (PR-BCD)

In this section we discuss attacking the existing, binary graph structure via additions and deletions of edges:

$$\max_{\boldsymbol{P} \text{ s.t. } \sum \boldsymbol{P} \leq \Delta} \mathcal{L}(f_\theta(\boldsymbol{A} \oplus \boldsymbol{P}, \boldsymbol{X}))\,. \tag{4}$$

Here, $\oplus$ stands for an element-wise exclusive or, Δ denotes the edge budget (i.e. the number of altered entries in the perturbed adjacency matrix) and $\boldsymbol{P} \in \{0,1\}^{n\times n}$ denotes edge flips at the respective location where $\boldsymbol{P}_{ij} = 1$. 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 $O(n^2)$ 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 Algorithm 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 we typically choose Δ to be a fraction of m and $b > \Delta$, in practice, we have a linear overhead.

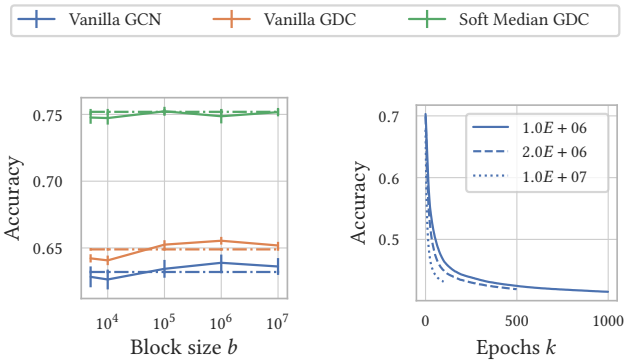
PR-BCD. For L_0-norm PGD we relax the discrete edge perturbations $\boldsymbol{P}$ from $\{0,1\}^{(n\times n)}$ to $[0,1]^{(n\times n)}$ as proposed by Xu et al. [30]. Each entry of $\boldsymbol{P}$ denotes the probability for flipping it. In each epoch we only look at a randomly sampled block of $\boldsymbol{P}$ of size b (line 3, line 10-13). In each epoch $e \in \{1,2,\dots\}$, $\boldsymbol{p}$ is added/subtracted from the discrete edge weight (line 6). 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 index $\boldsymbol{p}$ via its corresponding index of $\boldsymbol{P}$. 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 sample $\boldsymbol{P} \in \{0,1\}^{(n\times n)}$ via $\boldsymbol{P} \sim \text{Bernoulli}(\boldsymbol{p})$ (line 16).

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

1: **Input:** Adj. $\boldsymbol{A}$, feat. $\boldsymbol{X}$, labels $\boldsymbol{y}$, GNN $f_\theta(\boldsymbol{A},\boldsymbol{X})$, loss $\mathcal{L}$
2: **Parameter:** budget Δ, block size b, epochs K, heur. $h(\dots)$
3: Draw random indices w/o replacement $\boldsymbol{i}_0 \in \{0,1,\dots,n^2-1\}^b$
4: Initialize zeros for $\boldsymbol{p}_0 \in \mathbb{R}^b$
5: **for** $k \in \{1,2,\dots,K\}$ **do**
6: $\quad \hat{\boldsymbol{y}} \leftarrow f_\theta(\boldsymbol{A} \oplus \boldsymbol{p}_{k-1}, \boldsymbol{X})$
7: $\quad \boldsymbol{p}_k \leftarrow \boldsymbol{p}_{k-1} + \alpha_{k-1} \nabla_{\boldsymbol{i}_{k-1}} \mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$
8: $\quad$ Projection $\boldsymbol{p}_k \leftarrow \Pi_{\mathbb{E}[\text{Bernoulli}(\boldsymbol{p}_k)]=\Delta}(\boldsymbol{p}_k)$
9: $\quad \boldsymbol{i}_k \leftarrow \boldsymbol{i}_{k-1}$
10: $\quad$ **if** $k \le K_{\text{resample}}$ **then**
11: $\quad\quad \text{mask}_{\text{resample}} \leftarrow h(\boldsymbol{p}_k)$
12: $\quad\quad \boldsymbol{p}_k[\text{mask}_{\text{resample}}] \leftarrow \boldsymbol{0}$
13: $\quad\quad$ Resample $\boldsymbol{i}_k[\text{mask}_{\text{resample}}] \in \{0,1,\dots,n^2-1\}^{|\text{mask}_{\text{resample}}|}$
14: $\quad$ **end if**
15: **end for**
16: $\boldsymbol{P} \sim \text{Bernoulli}(\boldsymbol{p}_k)$ s.t. $\sum \boldsymbol{P} \le \Delta$
17: Return $\boldsymbol{A} \oplus \boldsymbol{P}$

Note that the projection of the perturbation $\Pi_{\mathbb{E}[\text{Bernoulli}(\boldsymbol{p})]=\Delta}(\boldsymbol{p})$ likely contains many zero elements, but is not guaranteed to be sparse. 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 $h(\boldsymbol{p})$ (see line 11). We keep the random block size b fixed and run K_{resample} epochs. Thereafter, we decay the learning rate as in [30]. We also employ early stopping for both stages ($k \le K_{\text{resample}}$ and $k > K_{\text{resample}}$ with the epoch k) such that we take the result of the epoch with highest loss $\mathcal{L}$.

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 [26], 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 such a small dataset and over a wide range of block sizes b, our method performs comparably 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 even improved attack strength. We argue that this indicates that PR-BCD successfully spots the right edges to keep.



(a) Cora ML **(b) arXiv**

Figure 3: We perturb the graph with our PR-BCD attack (solid lines) using the tanh Margin loss (Sec. 2) and $\epsilon = 0.1$. In (a) we report the mean and its three-sigma error over five random seeds on Cora ML (Tab. 2) for different block sizes b. We run 50 epochs where we resample the search space and then fine-tune for 250 epochs. The dashed lines show the performance of vanilla PGD [30]. In (b) we show the perturbed accuracy over epochs k on arXiv (170k nodes, Tab. 2) using PR-BCD for three different block sizes b shown with different lines. We choose k such that kb = const., i.e. approximately the same number of edges are "visited".

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 K iterations the budget is met.

3.2 Attacking Scalable GNNs

Up to now, we implicitly assumed that there is enough memory for the GNN to compute both the prediction for a given node, as well as obtain the gradient towards the edges. Such GNNs that typically process the whole graph "at once", are inherently limited in their scalability. Our attack PR-BCD is even applicable when operating at those limits (see experiments on the Product dataset in Sec. 5). However, to assess the adversarial robustness at scale, we must consider more scalable GNNs. Scaling traditional GNNs to massive graphs is difficult. Several approaches have been proposed to scale GNNs further. They either sample subgraphs [6, 9] or simplify the message passing operation [4].

To scale to massive graphs, it is desirable to obtain an attack with sublinear complexity w.r.t. the number of graph nodes. This severely restricts the possibilities of how one might approach a global attack. Therefore, we 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. In contrast, PPRGo [4] leverages the Personalized Page Rank (PPR) matrix $\Pi = \alpha(\boldsymbol{I} - (1-\alpha)\boldsymbol{D}^{-1}\boldsymbol{A})^{-1}$ (here with row normalization) to reduce the number of explicit message passing steps to one. This allows us to efficiently

attack a single node if we are able to update PPR scores efficiently (inserting and removing edges). As it turns out, we can leverage PR-BCD to obtain an attack on a single node (i.e. a local attack) for PPRGo with constant memory (assuming $b \ll n$) (i.e. it is sublinear). This approach does not restrict us on how we can sample other edges and allows us to obtain the gradient towards the input cheaply. Since all components are fully differentiable we can omit the use of a surrogate model as proposed by many previous approaches [25, 34]. Note that our novel PR-BCD algorithm is general and with only minor modifications we can apply it to other scalable GNNs than PPRGo.

To update the PPR scores for a given node in Π, we use the Sherman-Morrison formula

$$(\boldsymbol{B} + \boldsymbol{u}\boldsymbol{v}^\top)^{-1} = \boldsymbol{B}^{-1} - \frac{\boldsymbol{B}^{-1}\boldsymbol{u}\boldsymbol{v}^\top\boldsymbol{B}^{-1}}{1 + \boldsymbol{v}^\top\boldsymbol{B}^{-1}\boldsymbol{u}} \tag{5}$$

for rank one update $\boldsymbol{u}\boldsymbol{v}^\top$ of the inverse of an invertible matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$. This also allows us to obtain the gradient towards the edges of the current block b, which is all we need for PR-BCD. The rank one $\boldsymbol{u}\boldsymbol{v}^\top$ update in general has shape $[n \times n]$ and therefore comes with space complexity $O(n^2)$ and the update via the Sherman-Morrison formula has $O(n^3)$. Since we use row normalization with PPRGo, we can attack the PPR scores via updating a single row $\tilde{\Pi}_i = \tilde{\pi}(i)$ of the adjacency matrix $\boldsymbol{A}$ (including normalization). For this we choose $u_j = 0\,\forall j \neq i$ and $u_i = 1$. We can write the closed-form local PPR update as:

$$\tilde{\pi}(i) = \tilde{\Pi}_i = \alpha\left[\boldsymbol{I} - (1-\alpha)\boldsymbol{D}^{-1}\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^\top\right]^{-1}_i = \alpha\left(\Pi'_i - \frac{\Pi'_{ii}\boldsymbol{v}\Pi'}{1 + \boldsymbol{v}\Pi'_{:i}}\right) \tag{6}$$

where $\Pi' = (\boldsymbol{I} - (1-\alpha)\boldsymbol{D}^{-1}\boldsymbol{A})^{-1} = \alpha^{-1}\Pi$ and $\boldsymbol{v} = (\boldsymbol{D}_{ii} + \sum \boldsymbol{p})^{-1}(\boldsymbol{A}_i + \boldsymbol{p}) - \boldsymbol{D}_{ii}^{-1}\boldsymbol{A}_i$. We optimize over the b potentially non-zero entries in $\boldsymbol{p}$. $\boldsymbol{v}$ contains only zero elements, but for the current node that is being attacked.

With dense matrices, this would leave us with a complexity of $O(bn)$ due to the vector-matrix product $\boldsymbol{v}\Pi'$. We follow Bojchevski et al. [4] and use the top-k-sparsified PPR $\Pi^{(k)}$ instead of Π with at most k entries per row. Since $\boldsymbol{v}$ has at most b non-zero entries, most columns in the slice $\Pi^{(k)}_{\boldsymbol{v}\neq 0}$ only contain zero elements. Thus, we can equivalently write $\boldsymbol{v}\Pi'$ as a dense vector matrix product of shapes $[1, b]$ and $[b, r]$, where r is the number of non-zero columns in the rows Π'_b. With randomly distributed ones, the probability of a non-zero entry is k/n. Hence we can approximate $P(\sum \Pi^{(k)}_{\boldsymbol{v}\neq 0,j}) = \mathrm{Bin}(b, k/n)$ for column j and analogously $\mathbb{E}[r] = n \cdot P(\sum \Pi^{(k)}_{\boldsymbol{v}\neq 0,j} > 0) = n[1 - P(\sum \Pi^{(k)}_{\boldsymbol{v}\neq 0,j} = 0)] = k^b/n^{b-1} = O(1)$ for $k \ll n$ and $b \ll n$. For appropriate choices of k and b the expected complexity of our local attack is $O(br) = O(bk^b/n^{b-1}) = O(b)$. Please note that in contrast to the global PR-BCD attack, this *includes* the GNN/PPRGo, and therefore is much more scalable.

To obtain this local attack we simply need to change lines 3 and 13 of Algo. 1 to sample only indices $\boldsymbol{i}_0 \in \{0, 1, \ldots, n-1\}$. Further, we either keep line 6 if we attack e.g. GCN [15] or update $\tilde{\pi}(i)$ as described in Eq. 6. We further simply use a margin loss in logit space since we also only have a local budget.

4 SCALABLE DEFENSE

We are not aware of any defense that scales to graphs significantly larger than PubMed. We conclude, defending GNNs at scale is an entirely new research area. We propose a novel, scalable defense based on a robust message-passing aggregation, relying on recent advancements in differentiable sorting [21]. Our method *Soft Median* performs similarly to Geisler et al. [13]'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.

Related work. Many defenses counteract specific, observed characteristics of some attacks. In general we classify defenses into categories such as (1) preprocessing [12, 28], (2) robust training [30, 33], and (3) modifications of the architecture [13, 31, 32]. To start with a strong and practical baseline, we avoid defenses such as robust training due to the severe overhead during training. Nonetheless, our attacks would in principle allow robust training of GNNs at scale, but we leave this investigation for future work.

Background. Geisler et al. [13] relies on an observation about the message passing framework:

$$\mathbf{h}_v^{(l)} = \sigma^{(l)}\left[\mathrm{AGG}^{(l)}\left\{\left(\boldsymbol{A}_{vu}, \mathbf{h}_u^{(l-1)}\boldsymbol{W}^{(l)}\right), \forall\, u \in \mathbb{N}'(v)\right\}\right] \tag{7}$$

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 embeddings $\mathbf{h}_v^{(l)}$, the normalized message passing matrix $\boldsymbol{A}$, the weights $\boldsymbol{W}^{(l)}$, and activations $\sigma^{(l)}$. Since common aggregations (e.g. sum or mean) in Eq. 7 are known to be non-robust, Geisler et al. [13] 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 embeddings 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. For improving the previous aggregation we leverage two key facts. 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 summary, we propose a differentiable relaxation of the Median that softly selects the embedding which is closest the dimension-wise Median $\bar{\boldsymbol{x}}$:

$$\begin{aligned} t_{\mathrm{SoftMedian}}(\boldsymbol{X}) &= \mathrm{s}\left(-\frac{1}{T\sqrt{D}}\boldsymbol{d}\right)^\top \boldsymbol{X}, \text{ with } d_v = \|\bar{\boldsymbol{x}} - \boldsymbol{X}_{v,:}\| \\ &= \boldsymbol{s}^\top \boldsymbol{X} \approx \underset{\boldsymbol{x}'\in\mathbb{X}}{\arg\min}\, \|\bar{\boldsymbol{x}} - \boldsymbol{x}'\|, \end{aligned} \tag{8}$$

We normalized by the dimensions D, so that T does not depend on the number of dimensions. Our proposed aggregation can be interpreted as the Mahalanobis distance to the median with a spherical covariance matrix, or *standardized* Euclidean distance to the median. This is in particular a good assumption in the presence of batch normalization. Due to the weighting of the samples with the softmax with temperature T, our Soft Median also has connections to the density of the Gaussian distribution.

The temperature hyperparameter. Temperature parameter T controls the steepness of the weight distribution $\hat{\boldsymbol{s}}$ between the neighbors and corresponds to twice the standard deviation in the interpretation as Mahalanobis distance. In the extreme case as $T \to 0$ we recover the point 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. [13] and we decide on a temperature value in our experiments by grid search.

Robustness. Naturally, the question arises if this estimator is robust since in one extreme it recovers the sample mean which is known to be non-robust. Many metrics have been proposed that capture robustness with different flavours. One of the most widely used properties is the break down point. The (finite-sample) breakdown point captures the minimal fraction $\epsilon = m/n$ so that the result of the location estimator $t(\boldsymbol{X})$ can be arbitrarily placed [11] (here m denotes the number of perturbed examples):

$$\epsilon^*(t, \boldsymbol{X}) = \min_{1 \le m \le n} \left\{ \frac{m}{n} : \sup_{\tilde{\boldsymbol{X}}_\epsilon} \|t(\boldsymbol{X}) - t(\tilde{\boldsymbol{X}}_\epsilon)\| = \infty \right\} \tag{9}$$

Our proposed Soft Median has the best possible breakdown point of 0.5 as we state formally in Theorem 4.1[2]. Note that despite the lower complexity of the Soft Median, we maintain the same breakdown point.

Theorem 4.1. *Let $\mathbb{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ 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. 8) has the finite sample breakdown point of $\epsilon^*(t_{Soft\ Median}, \boldsymbol{X}) = 1/n \lfloor (n+1)/2 \rfloor$ (asymptotically $\lim_{n\to\infty} \epsilon^*(t_{SoftMedian}, \boldsymbol{X}) = 0.5$).*

Proof. Let $\tilde{\mathbb{X}}_\epsilon$ be decomposable such that $\tilde{\mathbb{X}}_\epsilon = \tilde{\mathbb{X}}_\epsilon^{(\mathrm{c})} \cup \tilde{\mathbb{X}}_\epsilon^{(\mathrm{p})}$. We now have to find the minimal fraction of outliers ϵ for which $\lim_{\tilde{d}_v \to \infty} \|t_{\mathrm{SoftMedian}}(\tilde{\boldsymbol{X}}_\epsilon)\| < \infty$ does not hold anymore. According to Eq. 9, if we now want to arbitrarily perturb the Soft Median, we must $\tilde{\boldsymbol{x}_v} \to \infty$, $\exists v \in \tilde{\mathbb{X}}_\epsilon^{(\mathrm{p})}$. Next we analyze the influence of this point on Eq. 8 (w.l.o.g we omit the factor $\sqrt{D}$):

$$\hat{\boldsymbol{s}}_v \boldsymbol{x}_v = \frac{\exp\left\{-\frac{1}{T}\|\bar{\boldsymbol{x}} - \tilde{\boldsymbol{x}}_v\|\right\} \boldsymbol{x}_v}{\sum_{i \in \tilde{\mathbb{X}}_\epsilon^{(\mathrm{c})}} \exp\left\{-\frac{1}{T}\|\bar{\boldsymbol{x}} - \boldsymbol{x}_i\|\right\} + \sum_{j \in \tilde{\mathbb{X}}_\epsilon^{(\mathrm{p})}} \exp\left\{-\frac{1}{T}\|\bar{\boldsymbol{x}} - \boldsymbol{x}_j\|\right\}}$$

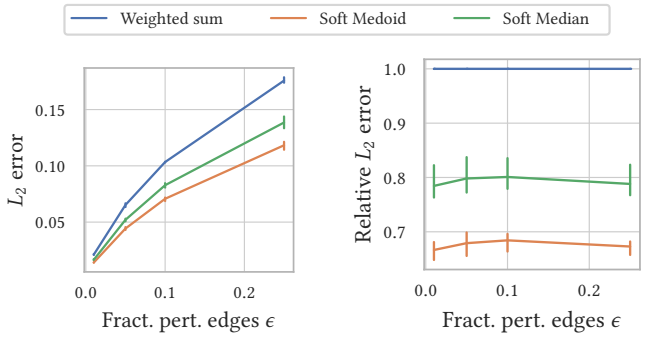
Instead of $\lim_{\|\tilde{\boldsymbol{x}}_v\| \to \infty} \hat{\boldsymbol{s}}_v \boldsymbol{x}_v$, we can equivalently derive the limit for the numerator and the denominator independently (as long as the denominator does not approach 0 and it is easy to show that the denominator is > 0 and $\le |\tilde{\mathbb{X}}_\epsilon|$):

$$\lim_{\|\tilde{\boldsymbol{x}}_v\| \to \infty} \exp\left\{-\frac{1}{T}\|\bar{\boldsymbol{x}} - \tilde{\boldsymbol{x}}_v\|\right\} \|\boldsymbol{x}_v\| = \begin{cases} 0, \text{ if } \lim_{\|\tilde{\boldsymbol{x}}_v\| \to \infty} \|\bar{\boldsymbol{x}} - \tilde{\boldsymbol{x}}_v\| = 0 \\ \infty, \text{ otherwise} \end{cases}$$

Please note that $\lim_{x \to \infty} x e^{-x/a} = 0$ for $a \in [0, \infty)$.

As long as $\epsilon < 0.5$, we know that for each dimension the perturbed dimension-wise Median must be still within the range of the clean points. Or in other words, the perturbed Median lays within the smallest possible hypercube around the original clean data $\mathbb{X}$. As long as $\epsilon < 0.5$ we have that $\lim_{\tilde{\boldsymbol{x}}_v\| \to \infty} \|\bar{\boldsymbol{x}} - \tilde{\boldsymbol{x}}_v\| = 0$.

[2]This does not prove adversarial robustness, but is a common proxy for robustness.



(a) (b)

Figure 4: Empirical bias $B(\epsilon)$ for the second layer of a GDC [16] network. (a) shows the absolute bias for a PGD attack, with loss (5) of Sec. 2, and a budget of changing $\epsilon = 0.25$ edges. (b) shows the relative bias over the weighted mean of a GDC. We use for all estimator a temperature of $T = 0.2$

Consequently, $\|t(\boldsymbol{X}) - t(\tilde{\boldsymbol{X}}_\epsilon)\| = \infty$ can only be true if $m \ge n$ for $T \in [0, \infty)$. □

Analogously to Geisler et al. [13], we define the *weighted* Soft Median as:

$$\tilde{t}_{\mathrm{WSM}}(\boldsymbol{X}, \mathbf{a}) = c\,(\boldsymbol{s} \circ \mathbf{a})^\top \boldsymbol{X} \tag{10}$$

where $\boldsymbol{s}$ is the softmax weight of Eq.8 obtained using the weighted dimension-wise Median and c is a normalization s.t. $\sum \boldsymbol{s} \circ \mathbf{a} = \sum \mathbf{a}$. It is easy to show that our proof also holds in the weighted case (see argument in [13]). Note that we recover the message passing operation of a GCN (mean) [15] for $T \to \infty$.

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). At least in the latent space, the Soft Medoid seems to be more robust. However, this is not consistent with the perturbed accuracy values in Tab. 1 and Tab. 3.

We can easily plug this aggregation into PPRGo with explicitly calculated PPR scores:

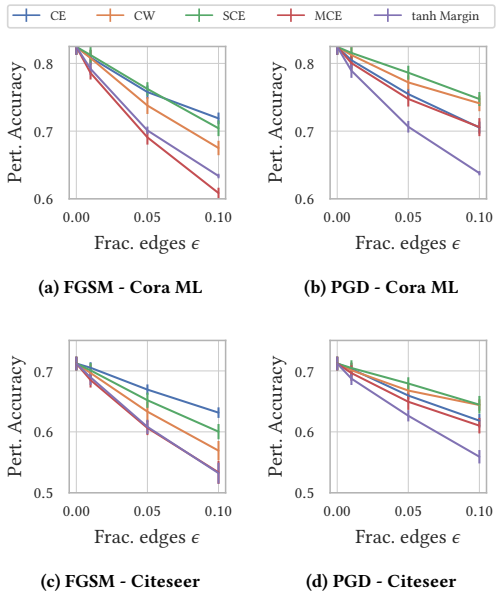
$$\boldsymbol{p}_i = \mathrm{softmax}\left[\mathrm{AGG}\left\{(\pi(v)_u, f_{\mathrm{enc}}(\boldsymbol{x}_u)), \forall\, u \in \mathbb{N}'(v)\right\}\right] \tag{11}$$

with feature encoder f_{enc} and for AGG we use the Soft Median.

5 EMPIRICAL EVALUATION

In the following, we present our experiments to show the effectiveness and scalability of our proposed attacks and the defense. We first describe our setup and then discuss the results over wide range of graphs of different scales. We will open source the code with configurations that reproduce the results. If not stated otherwise, we report the average over three random seeds/splits and 3-sigma error of the mean.

Attacks. We compare our local PR-BCD (Sec. 3.2) with Nettack [34]. We compare our global attacks PR-BCD and GR-BCD



(a) FGSM - Cora ML (b) PGD - Cora ML

(c) FGSM - Citeseer (d) PGD - Citeseer

Figure 5: Comparison of our MCE and tanh Margin with previous surrogate losses on a Vanilla GCN attacked via greedy FGSM and PGD. ϵ denotes the fraction of edges perturbed (relative to the clean graph).

Table 1: Drop in accuracy for the proposed *global* attacks (see Sec. 3.1) and baselines on Cora ML and Citeseer (see Tab. 2). The drop is relative to the clean accuracy (last column). A stronger attack results in a larger drop and a stronger defense results in a lower drop. We embolden the strongest attack per budget over all architectures and underline the strongest defense per attack and budget. The budget ϵ denotes the fraction of perturbed edges (relative to the clean graph). Our defenses Soft Median GDC and Soft Median PPRGo are consistently among the best.

	Attack	greedy FGSM			GR-BCD (ours)			PGD			PR-BCD (ours)			Acc.
	Frac. edges ϵ	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1	
	Architecture													
Cora ML	Vanilla GCN	**0.039**	**0.134**	**0.216**	0.035	0.125	0.198	0.032	0.103	0.164	0.035	0.113	0.183	0.82
	Vanilla GDC	0.036	**0.127**	**0.193**	0.033	0.122	0.191	0.031	0.1	0.155	**0.037**	0.114	0.182	0.83
	SVD GCN	0.003	0.018	0.039	0.003	0.017	0.039	**0.004**	**0.031**	**0.072**	**0.004**	0.027	0.069	0.76
	Jaccard GCN	**0.032**	**0.108**	**0.175**	0.03	0.103	0.164	0.026	0.091	0.141	0.029	0.095	0.159	0.82
	RGCN	**0.027**	**0.1**	**0.161**	0.026	0.094	0.157	0.024	0.076	0.128	0.023	0.088	0.142	0.80
	Soft Medoid GDC	0.01	0.028	0.039	**0.011**	0.029	0.042	0.008	0.028	0.046	**0.011**	**0.037**	**0.059**	0.82
	Soft Median GDC	0.012	0.035	0.05	0.012	0.038	0.052	0.011	0.034	0.054	**0.014**	**0.043**	**0.069**	0.82
	Vanilla PPRGo	0.026	0.08	0.107	0.024	0.078	0.105	0.024	0.069	0.1	**0.029**	**0.082**	**0.122**	0.83
	Soft Medoid PPRGo	**0.02**	**0.04**	0.049	0.019	0.039	0.052	0.017	0.036	0.052	0.018	**0.04**	**0.059**	0.82
	Soft Median PPRGo	0.016	0.03	0.04	0.016	0.032	0.044	**0.017**	0.031	0.045	**0.017**	**0.036**	0.055	0.82
Citeseer	Vanilla GCN	0.027	0.106	0.179	**0.03**	**0.11**	**0.184**	0.016	0.063	0.102	0.027	0.104	0.168	0.71
	Vanilla GDC	0.026	0.106	**0.175**	0.028	**0.107**	0.172	0.019	0.068	0.106	**0.031**	0.098	0.17	0.71
	SVD GCN	0.003	0.02	0.042	0.002	0.016	0.048	0.003	0.028	0.065	**0.006**	**0.033**	**0.079**	0.64
	Jaccard GCN	0.018	0.073	0.122	0.02	**0.078**	**0.125**	0.011	0.042	0.076	**0.021**	0.077	0.124	0.71
	RGCN	**0.015**	**0.064**	**0.113**	0.012	0.055	0.096	0.012	0.049	0.086	0.009	0.047	0.085	0.65
	Soft Medoid GDC	0.004	0.012	0.018	**0.005**	0.013	0.018	0.002	0.007	0.014	**0.005**	**0.016**	**0.025**	0.71
	Soft Median GDC	0.005	0.018	0.03	0.005	0.014	0.025	0.001	0.012	0.021	**0.007**	**0.024**	**0.041**	0.71
	Vanilla PPRGo	0.016	**0.058**	0.085	0.018	0.053	0.075	0.012	0.032	0.046	**0.019**	0.052	**0.086**	0.72
	Soft Medoid PPRGo	**0.009**	**0.023**	**0.032**	0.007	**0.023**	0.03	**0.009**	0.017	0.021	**0.009**	0.02	0.026	0.71
	Soft Median PPRGo	**0.014**	0.026	**0.037**	0.013	**0.027**	0.035	0.013	0.023	0.032	0.013	0.023	0.032	0.71

(Sec. 3.1) with PGD [30], and greedy FGSM (similar to Dai et al. [10]) attacks. The greedy FGSM-like attack is the dense equivalent of our GR-BCD attack with the exception of flipping one edge at a time. On the large datasets, we also compare to the global DICE [26] attack. DICE is a greedy, randomized black-box attack that flips one randomly determined entry in the adjacency matrix at a time. An edge is deleted if both nodes share the same label and an edge is added if the labels of the nodes differ. We ensure that a single node does not become disconnected. Moreover, we use 60% of the budget to add new edges and otherwise remove edges.

Defenses. We compare our Soft Median architectures with state-of-the-art defenses [12, 13, 28, 32]. The SVD GCN [12] uses a (dense) low-rank approximation (here rank 50) of the adjacency matrix to filter adversarial perturbations. RGCN [32] models the neighborhood aggregation via Gaussian distribution to filter outliers, and Jaccard GCN [28] filters edges based on attribute dissimilarity (here threshold 0.01). Following [13], we use the GDC/PPR preprocessing [16] in combinatin with out Soft Median. We set the temperature to $T = 0.2$ for all datasets, except Products and Papers 100M where we choose $T = 5.0$ via grid search. For the Soft Medoid GDC, we use the temperature $T = 0.5$ as it is a good compromise between accuracy and robustness.

Datasets. We use the common Cora ML [2] and Citeseer [18] for a comprehensive comparison of state of the art attacks and defenses. For large scale experiments, we use PubMed [22] as well as arXiv, Products and Papers 100M of the recent Open Graph Benchmark [14]. In comparison to Pubmed, we scale the global attack by more than 100 times (number of nodes), or by factor 15,000 if counting the possible adjacency matrix entries (see Tab. 2).

We scale our local attack to Papers 100M which has 111 million nodes, outscaling previous local attacks by a factor of 500. We use an 11GB GeForce GTX 1080 Ti for all our experiments. The only exception are the full-batch experiments on Products where we use a 32GB Tesla V100. We exclusively perform the calculations on GPU, but for Products where we coalesce the adjacency matrix on CPU. On Papers 100M we only load the required parts on the GPU.

Checkpointing. Empirically, almost 30 GB are required to train a three-layer GCN on Products (our largest dataset for global attacks) using sparse matrices. However, obtaining the gradient, e.g. towards the perturbation vector/matrix, requires extra memory. We notice that most operations in modern GNNs only depend on the neighborhood size (i.e. a row in the adjacency matrix). As proposed by Chen et al. [8], the gradient is obtainable with sublinear memory cost via checkpointing. The idea is to discard some intermediate results in the forward phase and recompute them in the backward

Table 2: Statistics of the used datasets. For the dense adjacency matrix we assume that each element is represented by 4 bytes. In the sparse case we use two 8 byte integer pointers and a 4 bytes float value.

Dataset	#Nodes n	#Edges e	#Features d	Size (dense)	Size (sparse)
Cora ML [2]	2.8 k	8.4 k	2,879	35.88 MB	168.32 kB
Citeseer [18]	3.3 k	4.7 k	3,703	43.88 MB	94.30 kB
PubMed [22]	19.7 k	88.6 k	500	1.56 GB	1.77 MB
arXiv [14]	169.3 k	1.2 M	128	114.71 GB	23.32 MB
Products [14]	2.4 M	123.7 M	100	23.99 TB	2.47 GB
Papers 100M [14]	111.1 M	1.6 G	128	49.34 PB	32.31 GB

Table 3: Analogous to Tab.1, we report the drop in accuracy for the *global* attacks on the large datasets. We GNNs and attacks that do not scale and report DICE as baseline for our attacks. We also embolden the strongest attack per budget over all architectures and underline the strongest defense per attack and budget.

	Attack	DICE			GR-BCD (ours)			PR-BCD (ours)			Acc.
	Frac. edges ϵ	0.01	0.05	0.1	0.01	0.05	0.1	0.01	0.05	0.1	
PubMed	Vanilla GCN	0.003	0.017	0.03	**0.032**	**0.129**	**0.209**	0.028	0.106	0.176	0.78
	Vanilla GDC	0.003	0.017	0.029	**0.031**	**0.122**	**0.196**	0.028	0.098	0.15	0.78
	Soft Medoid GDC	0.007	0.014	0.021	**0.018**	**0.061**	**0.094**	0.014	0.049	0.072	0.77
	Soft Median GDC	0.002	0.009	0.015	**0.017**	**0.058**	**0.089**	0.014	0.047	0.069	0.77
arXiv	Vanilla GCN	-0.013	0.001	0.017	**0.126**	0.277	0.357	0.12	**0.3**	**0.405**	0.69
	Vanilla GDC	0.036	0.043	0.068	0.104	0.28	0.367	**0.136**	**0.293**	**0.4**	0.68
	Soft Medoid GDC	0.011	0.022	0.032	**0.082**	**0.139**	0.156	0.057	0.128	**0.165**	0.58
	Soft Median GDC	0.009	0.024	0.037	**0.093**	0.2	0.243	0.081	**0.202**	**0.276**	0.66
Products	Vanilla GCN	0.01	0.045	0.084	0.116	0.212	0.279	**0.127**	**0.253**	**0.306**	0.72
	Vanilla GDC	0.008	0.031	0.052	0.099	0.134	0.149	**0.101**	**0.176**	**0.196**	0.71
	Soft Median GDC	0.005	0.02	0.032	0.047	0.063	0.072	**0.05**	**0.082**	**0.093**	0.66

phase. Specifically, we chunk some operations (e.g. matrix multiplication) within the message passing step to successfully scale to larger graphs. This allows us to attack a three-layer GCN on Products with full GPU acceleration.

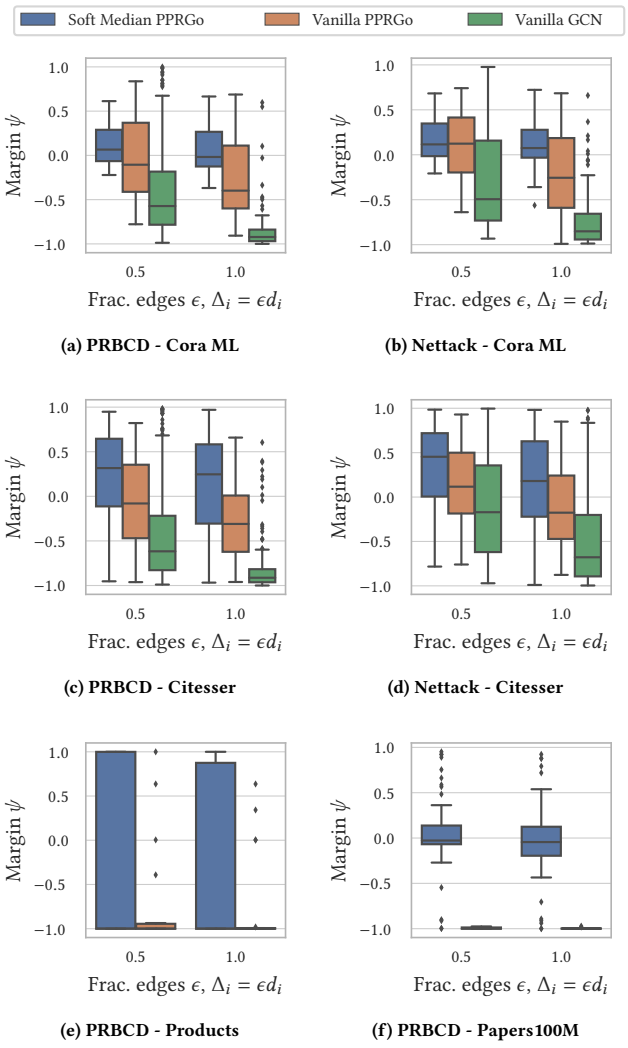
Hyperparameters. We use the same setup as Geisler et al. [13] in their evaluation and for models on OGB we follow Hu et al. [14]. For the attacks GR-BCD and PR-BCD, we run the attack for 500 epochs (100 epochs fine-tuning with PR-BCD). We choose the search space size to be at least twice the edge perturbation ratio ϵ (depends on the dataset). Further hyperparameters such as learning rates, weight decay etc. will be released with the code.

Surrogate Loss. In Fig. 5, we compare the conventional CE loss with our newly proposed losses. Overall we compare:

(1) Cross Entropy: CE = $\log(p_{c^*})$
(2) Carlini-Wagner [5]: CW = $(\min_{c \neq c^*} z_{c^*} - z_c)+$
(3) Second-most-likely CE: SCE = $-\log(\arg\max_{c \neq c^*} p_c)$
(4) Masked CE: MCE = $\frac{1}{|\mathbb{V}^+|} \sum_{n \in \mathbb{V}^+} \log(p^{(n)}_{c^*})$
(5) tanh Margin = $\tanh(\min_{c \neq c^*} z_{c^*} - z_c)$

To simplify notation, we define the losses for a single node (except for MCE) and denote the correct class with c^*. Note that $\mathbb{V}^+$ is the set of correctly classified nodes, $\boldsymbol{p}$ is the vector of confidence scores, and $\boldsymbol{z}$ is the vector with logits. We see that our losses (4) Masked CE (MCE) and (5) the tanh Margin perform equally well with FGSM. As expected and discussed in Sec. 2, for PGD the tanh Margin is consistently and significantly stronger. Even for the small datasets and over all tested ϵ, we see relative gains of more 100% on the drop of the perturbed accuracy. We omit the results on larger data due to space limitations. Comparing CE with our novel tanh Margin loss we sometimes observe attack improvements of more than 200 %.

Robustness w.r.t. global attacks. In Tab. 1, we present the experimental results for our proposed global attacks on the small datasets Cora ML and Citeseer. Our attacks are as strong as their dense equivalents despite being much more scalable. In Tab. 3, we compare our our novel attacks on all baselines that fit into memory or can be trained within 24 hours. Our defense Soft Median GDC and Soft Median PPRGo are consisently among the best models tested over all scales. 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



(a) PRBCD - Cora ML **(b) Nettack - Cora ML**
(c) PRBCD - Citesser **(d) Nettack - Citesser**
(e) PRBCD - Products **(f) PRBCD - Papers100M**

Figure 6: Perturbed classification margins $\tilde{\psi}_i$ of the attacked nodes. In (a)-(d), we compare our local PR-BCD attack with Nettack [34] on Cora ML and Citeseer. In (e) and (f), we show the results on the 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.

almost the entire memory of the 32 Gb GPU. Despite the small sacrifice in clean accuracy, we already outperform the baselines for a budget of $\epsilon = 0.01$. We faced similar scaling limitations for the Soft Medoid GDC baseline already on arXiv. This highlights the lower memory requirements for our Soft Median.

Robustness w.r.t. local attacks. In Fig. 6, we compare the results of our local PR-BCD with Nettack on Cora ML and Citeseer. Similarly to Zügner et al. [34], we define the budget $\Delta_i = \epsilon d_i$. We apply the attack only to correclty classified nodes: 10 with highest confidence, 10 with lowest and 30 random nodes. To attack a GCN with PR-BCD we do not need the PPR update via the Sherman

Morrison formula. We clearly see that our attack is stronger than Nettack on both datasets, all architectures and budgets. Nettack has the advantage of solving a discrete optimization problem. However, it performs a transfer attack with a linearized surrogate. Evidently, it hurts more to neglect the non-linearities and to transfer the attack as in Nettack, than relaxing the optimization problem as done in PR-BCD. On the large datasets Products and Papers 100M we observe that the Vanilla PPRGo is very fragile and even low budgets suffice to flip almost every nodes prediction. Our proposed defense Soft Median PPRGo on the other hand remains similary robust as on the small datasets. On Papers 100M, the Soft Median PPRGo reduces the attacker's success rate from around 100% to just 40% (80% vs. 50% on Products). On Papers 100M, equipping PPRGo with the Soft Median does degrade the clean accuracy of around 60%.

Relation between fragility and graph size. In the following, we analyze the results of PR-BCD, with a budget of $\epsilon = 0.1$ for the GCN model. With PR-BCD, we observe a relative drop in the perturbed accuracy by 22% on Cora, 59 % on arXiv, and 43% on Products. On the larger graphs the degradation of the accuracy is much larger which indicates a relationship between the adversarial robustness and the size of the graph. This relationship seems to persist for architectures other than GCN as well, though, it also depends of course on the characteristics of the dataset itself. Similarly, for local attacks we observe that even small budgets suffice to fool almost all nodes. On small graphs PPRGo already seems to be quite an effective defense (see Fig. 6). However, on large graphs it seems to be easier to successfully attack many of the nodes. We leave a detailed study of this relationship for future work.

Time and memory cost. On arXiv, we train for 500 epochs and run the global PR-BCD attack for 500 epochs. The whole training and attacking procedure requires less than 2 minutes and the peak usage of GPU memory is below 2.5 GB. Note that only loading the adjacency matrix for traditional attacks (no training etc.) would require around 115 GB (see Tab. 2). Naively attacking the dense adjacency matrix would certainly require more than 1 TB (compare with Fig. 1). One epoch on Papers 100M with the local PR-BCD attack takes less then 10 seconds and we require 6 GB at maximum while attacking the Vanilla PPRGo.

6 CONCLUSION

We study adversarial robustness of GNNs at scale. We tackle all three of the identified challenges. First, we study previous surrogate losses for global attacks on GNNs, and we propose two alternatives that overcome the problems we identified. Our new losses easily double the strength of the attacks we tested. Second, we propose a new family of first-order optimization attacks that are much more scalable than previous attacks without sacrificing strength. We apply this general framework to global attacks on the graph accuracy for different GNNs, and to local attacks on massive graphs in combination with PPRGo. Third, we propose a cheap aggregation function called Soft Median, that has the best possible breakdown point of 0.5 and is fully differentiable. We can apply this defense to both the message passing step of traditional GNNs, as well as PPRGo and we show its efficacy against different attacks.

In summary, we scale adversarial attacks and defenses to graphs of up to 111 million nodes. Our attacks and defenses are practical and allow for the first time to assess the adversarial robustness and to defend against attacks for real-world massive-scale applications.

REFERENCES

[1] Anonymous. Attacking Graph Neural Networks at Scale. *DLG workshop @ AAAI)*, 2021.
[2] A. Bojchevski and S. Günnemann. Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking. *ICLR*, 2018.
[3] A. Bojchevski and S. Günnemann. Adversarial attacks on node embeddings via graph poisoning. *ICML*, 2019.
[4] A. Bojchevski, J. Klicpera, B. Perozzi, A. Kapoor, M. Blais, B. Rózemberczki, M. Lukasik, and S. Günnemann. Scaling Graph Neural Networks with Approximate PageRank. *KDD*, 2020.
[5] N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. *IEEE Symposium on Security and Privacy*, 2017.
[6] J. Chen, T. Ma, and C. Xiao. FASTGCN: Fast learning with graph convolutional networks via importance sampling. *ICLR*, 2018.
[7] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan. Fast gradient attack on network embedding. *arXiv*, 2018.
[8] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training Deep Nets with Sublinear Memory Cost. *arXiv*, 2016.
[9] 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. *KDD*, 2019.
[10] H. Dai, H. Li, T. Tian, H. Xin, L. Wang, Z. Jun, and S. Le. Adversarial attack on graph structured data. *ICML*, 2018.
[11] 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.
[12] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis. All you need is Low (rank): Defending against adversarial attacks on graphs. *WSDM*, 2020.
[13] S. Geisler, D. Zügner, and S. Günnemann. Reliable Graph Neural Networks via Robust Aggregation. *NeurIPS*, 2020.
[14] 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.
[15] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.
[16] J. Klicpera, S. Weißenberger, and S. Günnemann. Diffusion Improves Graph Learning. *NeurIPS*, 2019.
[17] J. Li, T. Xie, L. Chen, F. Xie, X. He, and Z. Zheng. Adversarial attack on large scale graph. *arXiv*, 2020.
[18] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 2000.
[19] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 2012.
[20] Y. Nesterov and S. Stich. Efficiency of accelerated coordinate descent method on structured optimization problems. *Siam J. Optim.*, 2017.
[21] S. Prillo and J. Martin Eisenschlos. SoftSort: A Continuous Relaxation for the argsort Operator. *ICML*, 2020.
[22] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 2008.
[23] X. Tang, Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang. Transferring robustness for graph neural network against poisoning attacks. *WSDM*, 2020.
[24] B. Wang and N. Z. Gong. Attacking graph-based classification via manipulating the graph structure. *ACM CCS*, 2019.
[25] 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*, (5), 2020.
[26] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, (2), 2018.
[27] S. J. Wright. Coordinate Descent Algorithms. *Mathematical Programming*, 2015.
[28] 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*, 2019.
[29] K. Xu, C. Li, Y. Tian, T. Sonobe, K. I. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. *ICML*, 2018.
[30] 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*, 2019.
[31] Y. Zhang, S. Pal, M. Coates, and D. Ustebay. Bayesian Graph Convolutional Neural Networks for Semi-Supervised Classification. *AAAI*, 2019.
[32] D. Zhu, P. Cui, Z. Zhang, and W. Zhu. Robust graph convolutional networks against adversarial attacks. *KDD*, 2019.
[33] D. Zügner and S. Günnemann. Adversarial attacks on graph neural networks via meta learning. *ICLR*, 2019.
[34] D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. *KDD*, 2018.