Robust Graph Neural Networks at Scale

Simon Geisler, Hakan Şirin, Daniel Zuegner, Tobias Schmidt, Aleksandar Bojchevski, Stephan Guennemann
{geisler,sirin,zuegnerd,schmidtt,a.bojchevski,guennemann}@in.tum.de
Technical University of Munich
Germany

ABSTRACT

Adversarial robustness for Graph Neural Networks (GNNs) has become exceedingly important due to the popularity and diverse applications of GNNs. Specifically, structure perturbations have proven to be challenging to defend against. Moreover, designing attacks is not straight-forward due to the discrete optimization domain. One of the most effective defense strategies is to use a robust estimator for the aggregation of the message passing operation, and most robust estimators are known to be computationally demanding. In this work, we leverage recent advances in differentiable sorting for robust aggregation in message passing that scales linearly with the neighborhood size. The increased scalability enables us to apply our defense to real-world graphs orders of magnitude larger than previously done. However, existing adversarial attacks typically require a dense adjacency matrix and can therefore only be applied to small graphs. For this reason, in addition to the scalable robust aggregation function, we propose two attacks based on first-order optimization that do not require a dense adjacency matrix and, hence, can be used for a global attack on graphs 5,000 times larger than previously evaluated.

KEYWORDS

Adversarial robustness, graph neural networks, scalability, semi-supervised learning

ACM Reference Format:
Simon Geisler, Hakan Şirin, Daniel Zuegner, Tobias Schmidt, Aleksandar Bojchevski, Stephan Guennemann. 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, 7 pages. https://doi.org/TBD

1 INTRODUCTION

Attacks based on combinatorial approaches easily become computationally infeasible because of the vast amount of potential adjacency matrices (2^{n^2}). With first-order optimization we can approximate the discrete optimization problem. First-order optimization attacks typically require the gradient towards the entries in the adjacency matrix and, hence, reduce the complexity to $\Theta(n^2)$. To attack a small dataset such as PubMed (19717 nodes), typically more than 20 GB are required if using a dense adjacency matrix. We argue that such memory requirements are still impractical and hinder practitioners to assess adversarial robustness. We identify the necessity to research scalable attacks for GNNs, due to the lack of such attacks for real-world graphs

We propose two strategies how one may apply first-order optimization, without the burden of a dense adjacency matrix. In Section 5 we propose an attack that adds adversarial nodes and was one of the top 5 attacks of the KDD Cup 2020 [1]. Thereafter in Section 4, we describe how one might add/remove edges between existing nodes based on Randomized Block Coordinate Descent (R-BCD). Even though our attacks can be generalized to graph classification, we focus on the important task of node classification.

In this work, we set the foundation for the study of adversarial robustness of GNNs on real-world citation/social networks. Importantly, it is unknown how the adversarial robustness/vulnerability relates to the graph size. In our experiments (Section 7), we point out that a GNN applied to a larger graph is more fragile w.r.t. structure perturbations.

Our contributions are the following:

- We show that the widely used cross entropy is not a good surrogate loss for attacking the graph structure on graph neural networks.
- We propose a novel loss for global attacks on graph neural networks that overcomes the limitations and empirically boosts the attack strength by up to 100%.
- We propose two scalable adversarial attacks that add/remove edges between the existing nodes. One relies on projected gradient descent and the other uses a greedy FGSM-like optimization scheme (space complexity of $\Theta(m)$ in the number of edges m).
- We propose one scalable adversarial attack that adds adversarial nodes (space complexity of $\Theta(n)$).
- We propose a differentiable robust aggregation that scales linearity w.r.t. the neighborhood size of the message passing aggregation and performs au par to the previous defense. This enables us to defend a GNN when memory is at premium.
- We study the adversarial robustness on graphs substantially larger than PubMed. Empirically, we find that the graph size negatively related to the adversarial robustness.

2 RELATED WORK

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

KDD '21, August 14–18, 2021, Online

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

machine learning and large scale optimization [21]. Nesterov [15] proposed (and analyzed the convergence) of Randomized Block Coordinate Descent (R-BCD). For R-BCD's pseudo code see Algorithm 1. In R-BCD only a subset 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 [16]. Constrained optimization with first-order methods can be solved with methods such as Projected Gradient Descent (PGD) or Fast Gradient Sign Method (FGSM) [10]. Similarly, one can extend R-BCD to constrained optimization [15].

Adversarial attacks. Beginning with [6, 26], many adversarial attacks on the graph structure have been proposed [3, 18, 19, 22, 23, 27]. We limit the scope to 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 asses adversarial robustness. Dai et al. [6] scale their reinforcement learning approach to a very sparse, large-scale graph for financial transactions. In contrast to our work, they only use a very small budget Δ (their time complexity scales linearly with Δ). Wang and Gong [19] also scale their attack to a larger graph than PubMed but they do not attack GNNs. (Todo: Adversarial attack on large scale graph) analyze adversarial attacks on mini-batch techniques such as Cluster-GCN (Todo: cite). However, they only scale to a dataset with around 200k nodes and also only consider a local attack. We scale to much bigger datasets, consider a wider class of Graph Neural Networks and also propose a global attack that is scalable. (Todo: Mention Fake Node Attacks) propose an attack based on GANs that inserts fake nodes and is the closest related work to our attack in Section 5.

(Todo: MGA: Momentum Gradient Attack on Network) First-order optimization attacks such as Metattack [27] or integrated gradients [22] 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:

$$\max_{\boldsymbol{A}} \mathcal{L}(f_\theta(\boldsymbol{A}, \boldsymbol{X})) \tag{1}$$

with the adjacency matrix $\boldsymbol{A}$, node features $\boldsymbol{X}$, loss $\mathcal{L}$, and the trained network f_θ.

(Todo: defenses)

3 CROSS ENTROPY IS A BAD SURROGATE

In the context of images, typically a single sample is attacked. In the context of graph neural networks this corresponds to a local attack). For such a scenario an untargeted attack may simply maximize the cross entropy $\mathrm{CE}(y, \boldsymbol{p}) = \sum_{c \in \mathbb{C}} \mathbb{I}[y = c] \log(p_c)$. Many *global* attacks Chen et al. [4], Wu et al. [22], Xu et al. [24], Zügner and Günnemann [27] also attack via maximizing the cross entropy $\max_{\boldsymbol{A}} \mathrm{CE}(f_\theta(\boldsymbol{A}, \boldsymbol{X}))$. In our preliminary experiments while attacking a GNN on large graphs, we have often observed that the loss increased while the accuracy increased to.

Algorithm 1 R-BCD [15]

1: Choose $\boldsymbol{x}_0 \in \mathbb{R}^d$
2: **for** $k \in \{1, 2, \ldots, K\}$ **do**
3: Draw random indices $\boldsymbol{i}_k \in \{0, 1, \ldots, n\}^b$
4: $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} - \alpha_k \nabla_{\boldsymbol{i}_k} \mathcal{L}(\boldsymbol{x}_{k-1})$
5: **end for**

Proof. We are given a graph neural network $f_\theta(\boldsymbol{A}, \boldsymbol{X})$ with fixed parameters θ and want to $\max_{\boldsymbol{A}} \mathrm{CE}(f_\theta(\boldsymbol{A}, \boldsymbol{X}))$ s.t. the number of edges flipped in $\boldsymbol{A}$ is limited by Δ. The gradient with softmax activation $\boldsymbol{p} = \mathrm{Softmax}(\boldsymbol{z})$ is given by $\partial \mathrm{CE} / \partial \boldsymbol{z}$ □

Despite its wide use, the surrogate loss cross entropy can be completely uncorrelated with the accuracy if the budget Δ of flipping edges is limited. The cross entropy is unbounded and if the probability of the correct class p_{c^*} approaches 0 it diverges (i.e. $\lim_{p_{c^*} \to 0} \mathrm{CE} = \infty$). Hence, for small enough budgets Δ on a large graph it is possible that the attack only focuses on nodes that are *already wrongly classified* before the attack has even started (for those nodes the gradient is the largest).

A global structure attack has to 1) keep house with the budget Δ and 2) find edges that degrade the accuracy maximally. Thus, we propose the subsequent axioms a well-suited, monotonically decreasing surrogate loss $\mathcal{L}^*(y, \boldsymbol{p})$ should obey for attacking a node-classification algorithm. We define the classification margin as $\psi = \min_{c \text{ s.t. } c \neq c^*} p_{c^*} - p_c$.

Axiom 1. *The loss $\mathcal{L}^*(y, \boldsymbol{p})$ should saturate for low confidence values of the correct class:* $\lim_{\psi \to -1^+} \mathcal{L}(y, \boldsymbol{p}) = k < \infty$.

Axiom 2. *The loss should favour points close to the decision boundary:* $\partial \mathcal{L}(y, \boldsymbol{p}) / \partial p_{c^*}|_{\psi=1} > \partial \mathcal{L}(y, \boldsymbol{p}) / \partial p_{c^*}|_{\psi \to 0^+}$.

Please note that also the widely used margin loss $\min(0, \psi)$ violates Axiom 2, since all correctly classified nodes receive exactly the same gradient. Obviously, for very low budgets Δ, it likely better to focus on nodes close to the decision boundary.

One natural choice that fulfills those axioms is the masked cross entropy

$$\mathrm{MCE} = \frac{1}{|\mathbb{V}^+|} \sum_{n \in \mathbb{V}^+} \sum_{c \in \mathbb{C}} \mathbb{I}[y = c] \log(p_c) \tag{2}$$

where $\mathbb{V}^+$ is the set of correctly classified nodes. Axiom 1 holds since for a negative margin $\psi < 0$ the node will be simply omitted. Axiom 2 holds since the gradient $\partial \mathrm{MCE} / \partial p_{c^*}$ is strictly decreasing w.r.t. ψ. This loss is well suited for greedy attacks.

However, MCE is not a good choice for a projected gradient descent method like the one we are going to propose in Section 4. For such an optimization, we typically tune the learning rate such that the budget Δ is exceeded after each gradient update and then the project operation maps the parameters back into the feasible region. If we now set the loss to zero / mask it out if wrongly classified, the contributing edges will not gain anything in the gradient update and likely loose strength in the upcoming project step. Hence, those nodes will oscillate at the decision boundary. Therefore, we use tanh of the classification margin

$$\tanh \mathrm{Margin} = \frac{1}{|\mathbb{V}|} \sum_{n \in \mathbb{V}} \tanh(\psi) \tag{3}$$

where ψ denotes the classification margin. This loss obviously fulfills both axioms.

Obviously, one could construct more complicated loss functions considering the asymmetry between correctly and wrongly classified nodes. However, in our experiments we did not achieve much better results. Please note that the reformulations already comes with a significant boost in attack strength.

(Todo: Table comparing naive and new loss)

4 ADDING AND REMOVING EDGES

In this section we discuss the case where the attack vector is to perturb the existing, binary graph structure:

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

Here, $\oplus$ stands for an element-wise exclusive or and Δ denotes the edge budget (i.e. the number of altered entries in the perturbed adjacency matrix). In the following, we use set $\mathbb{P}$ and matrix notation $\boldsymbol{P}$ for the sparse perturbations $\mathbb{P}$ interchangeably. Naively, applying R-BCD to optimize towards the dense adjacency matrix would only save some computation on obtaining the respective gradient, but still has a space complexity of $O(n^2)$. Note that in R-BCD we interpret each possible entry in the perturbation set $\mathbb{P}$ as one dimension of our optimization problem. To mitigate the quadratic complexity, we make use of the fact that the solution is going to be sparse (L_0 perturbation). As in evolutionary algorithms, in each epoch, we keep that part of the search space which is promising and resample the rest. We can view the underlying problem as a combination of L_0-norm PGD and an adaptive version of Randomized Block Coordinate Descent (R-BCD). R-BCD comes with a space complexity of $\Theta(m)$, as we typically choose Δ to be a fraction of m. We give a formal definition of Projected and Randomized Block Coordinate Descent (PR-BCD) in Algorithm 2.

As proposed by Xu et al. [23], $\boldsymbol{p}$ is interpreted as the probability mass for each potential entry in the perturbation set $\mathbb{P}$ and is used in the end to sample the final edge additions/removals. In each epoch $e \in \{1, 2, \ldots, E\}$, this probability mass $\boldsymbol{p}$ is used as edge weight. 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). If using an undirected graph, the potential edges are restricted to the the upper/lower triangular $n \times n$ matrix. In the end we sample $\mathbb{P} \sim \text{Bernoulli}(\boldsymbol{p})$ (line 16).

Note that the projection of the perturbation set $\Pi_{\mathbb{E}[\text{Bernoulli}(\boldsymbol{p})]=\Delta}(\boldsymbol{p})$ 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).

With growing n it is unrealistic that each possible entry of the adjacency matrix was part of at least one random search space of R-BCD. Apparently, with a constant search space size, the number of mutually exclusive chunks of the perturbation set grows with $\Theta(n^2)$ and this would imply a quadratic runtime. However, as evident in randomized black-box attacks [20], it is not necessary to test every possible edge to obtain an effective attack. In Fig. 1, we analyze the influence of the random block size on the perturbed accuracy. For a sufficient block size b our method performs comparably to its dense equivalent.

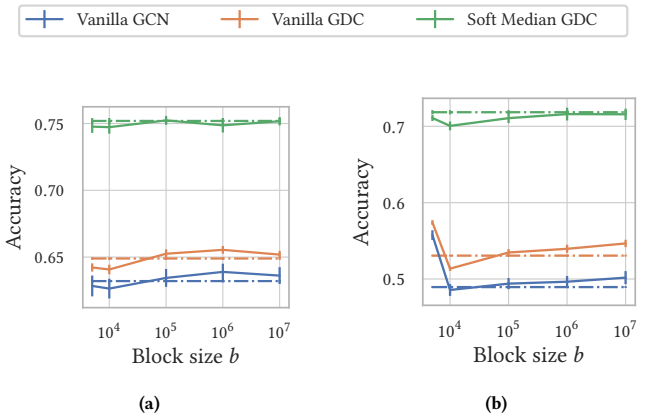

Figure 1: We perform the proposed PR-BCD (solid lines) to obtain a perturbed adjacency matrix with the fraction of perturbed edges $\epsilon = 0.25$ (i.e. $\Delta = 1995$ edges). We run 50 epochs where we resample the search space and subsequently fine-tune for 250 epochs. The dashed lines show the performance of vanilla PGD [23]. We report the mean and its three-sigma error over five random seeds.

We keep the random block fixed and run K_{resample} epochs. Thereafter, we decay the learning rate as in [23]. We also employ early stopping for both stages ($k \leq K_{\text{resample}}$ and $k > K_{\text{resample}}$ with the epoch k) such that we take the result of the epoch with highest loss $\mathcal{L}$.

Algorithm 2 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. $h(\ldots)$
3: Draw random indices $\boldsymbol{i}_0 \in \{0, 1, \ldots, N\}^b$
4: Initialize zeros for $\boldsymbol{p}_0 \in \mathbb{R}^b$
5: **for** $k \in \{1, 2, \ldots, K\}$ **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** $k \leq K_{\text{resample}}$ **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 $\boldsymbol{i}_k[\text{mask}_{\text{resample}}] \in \{0, 1, \ldots, N\}^{|\text{mask}_{\text{resample}}|}$
14: $\quad$ **end if**
15: **end for**
16: $\boldsymbol{P} \sim \text{Bernoulli}(\boldsymbol{p}_k)$ s.t. $\sum \boldsymbol{P} \leq \Delta$
17: Return $\boldsymbol{A} \oplus \boldsymbol{P}$

We also compare this approach to a Greedy R-BCD (GR-BCD), that greedily flips the entries with largest gradient in the random search space, such that after K iterations the budget requirements are met.

If we performed a targeted instead of a global attack, we could further reduce the space requirements since we only need to consider the node's "receptive field". Such an attack could be very similar to adding one node in the introduced GANG attack with further constraints (see Section 5).

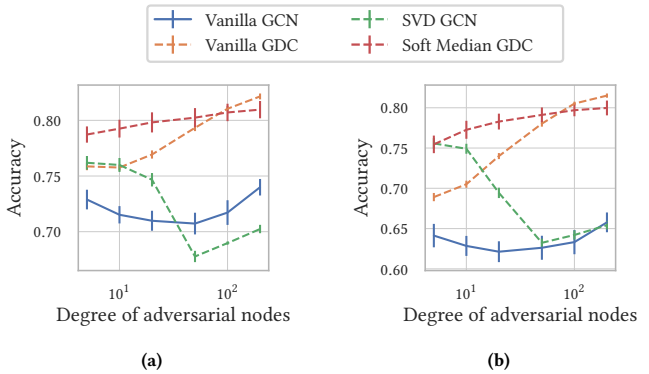

Figure 2: Influence of the degree of the added nodes on the perturbed accuracy on Cora ML. (a) shows the perturbed accuracy with a budget of changing $\epsilon = 0.1$ edges, and (b) for $\epsilon = 0.25$. The number of nodes is determined as $\Delta_n = \epsilon/\Delta_e$. We report the mean perturbed accuracy and its three-sigma error over five random seeds.

5 ADDING ADVERSARIAL NODES

We solve the attack optimization problem via adding nodes

$$\max_{A',X'} \mathcal{L}(f_\theta(A|A', X|X')) \tag{5}$$

with the space complexity of $O(m)$ (on top of the complexity of the attacked model; in the following we will neglect this fact). With $A|A'$ we denote the addition of rows & columns and with $X|X'$ the concatenation of the respective attributes (A & X are const.). For imperceptibility, we further limit the number of nodes Δ_n and their degree Δ_e.

The attacks add one node (or a small group of nodes) at a time to the sparse adjacency matrix and connect it to every other node with edge weight zero. Subsequently, we perform a constrained gradient-based optimization to determine the best edges with a given budget. We decide to add nodes in an greedy manner. For each new node, we determine the edges in s_e steps via a greedy FGSM-like procedure (PGD is an alternative). Then, the initial features (randomly sampled) are optimized via PGD (s_x epochs). In Algorithm 3, we give a formal definition of GANG.

In Fig. 2 we analyze the influence of the degree of the added nodes via GANG. Interestingly, low degree nodes seem to be more effective than high degree nodes. This could be due to the normalization by the square root of the inverse node degree for a GCN [12]. Surprisingly, we find that especially GDC [13] (i.e. personalized PageRank) and a low-rank SVD approximation [8] are effective defenses (prepossessing techniques of the adjacency matrix). SVD is a strong defense against low-degree nodes and personalized page rank is particularly strong against high degree nodes. The recent defense Soft Medoid GDC [9], seems to be effective regardless of the node degree.

6 DEFENSE

We build upon the very recent defence using a robust message passing aggregation that they call Soft Medoid [9]. Our method Soft

Algorithm 3 Greedy Adversarial Node Generation (GANG)

Input: Adj. A, feat. X, labels y, GNN $f_\theta(A, X)$, loss $\mathcal{L}$
Parameter: budgets Δ_n & Δ_e, step size s_e, steps features s_x
Initialize empty A' and X'
for $k \in \{1, \dots, \Delta_n\}$ **do**
  $A' \leftarrow$ concatenate new node to A' (empty row and column)
  $X' \leftarrow$ concatenate X' vector $\tilde{x}_k \sim \Pi(\mathcal{N}(0, \sigma_n^2))$
  for $j \in \{0, \dots, \Delta_e/s_e\}$ **do**
    $\hat{y} \leftarrow f_\theta(A|A', X|X')$
    $g \leftarrow \nabla_{A'_k} \mathcal{L}(\hat{y}, y)$ for all nodes where $\hat{y} = y$
    $A' \leftarrow$ add the top s_e edges to A' w.r.t. g
  end for
  for $j \in \{1, \dots, s_x\}$ **do**
    $X' \leftarrow \Pi(\tilde{X} + \alpha_x \nabla_{X'} \mathcal{L}(f_\theta(A|A', X|X'))$
  end for
end for

Median performs similarly to Soft Medoid with better complexity w.r.t. the neighborhood size lower memory footprint and enables us to scale bigger graphs. Our aggregation utilizes a distance based heuristic to discard outliers where we weight each sample $x \in \mathbb{R}^d$ depending on its euclidean distance to the component-wise median of the sample vector x. This limits the influence of outlier samples, resulting in a robust aggregation function. In our method, we propose the aggregation function $t_{\text{Soft Median}}$ as:

$$t_{\text{Soft Median}}(X) = \hat{s}^\top X = \sum_{i=1}^{n} \hat{s}_i X_{i,:} \approx \arg\min_{x \in \mathbb{X}} \|x - \bar{x}\|, \tag{6}$$

where component-wise median $\bar{x} = \arg\min_{\bar{x} \in \mathbb{R}^d} \sum_{i=1}^n \|\bar{x} - X_{i,:}\|_1$ and the weights $0 \leq \hat{s}_i \leq 1, \sum_i \hat{s}_i = 1$ are obtained via softmax based method where we are utilizing the euclidean distances to the sample's component-wise median $\bar{x}$ as:

$$\hat{s}_i = \text{softmax}\left(-\frac{1}{T}d\right)_i = \frac{\exp\left(-\frac{1}{T}d_i\right)}{\sum_{j=1}^n \exp\left(-\frac{1}{T}d_j\right)}, \text{ where } d_k = \|X_{k,:} - \bar{x}\| \tag{7}$$

In $t_{\text{Soft Median}}$, we calculate a weighted mean of the neighbouring nodes' features $x \in \mathbb{X}$ where the weightings $\hat{s}$ on $x \in \mathbb{X}$ depends on their distances to the sample mean $\bar{x}$. Whereas the closer to the mean samples with smaller residual have bigger weights, the more distant ones with bigger residuals have weights close to zero which results in a aggregation function that is that is robust to distant samples.

Temperature parameter T controls the steepness of the weight distribution $\hat{s}$ between the neighbors. In the extreme case as $T \to 0$, the weight for the smallest residual sample go towards one, resulting with $\arg\min_{x \in \mathbb{X}} \|x - \bar{x}\|_p$. In the other extreme temperature as $T \to \text{inf}$, we have the equally distributed weights where we output $\bar{x}$, and have an aggregation function equivalent to a mean.

Our suggested relaxation for $\arg\min_{x \in \mathbb{X}} \|x - \bar{x}\|$ is motivated by a very recent work of (Todo: prillo et al 2020 softsort paper) where they propose a softsort operator that uses softmax of negative pairwise distances resulting with a approximate permutation matrix. Whereas the suggested softsort was operating on scalars, in Soft Residuals we extended their work on multidimensional features

using the euclidean distance between the vectors, and we avoided quadratic complexity of the pairwise distances by only calculating the distances to $\bar{\boldsymbol{x}}$.

Robustness. Naturally, the question arises what are the properties especially w.r.t. robustness of our proposed aggregation for deep learning. 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 [7] (here m denotes the number of perturbed examples):

$$\epsilon^*(t, \boldsymbol{X}) = \min_{1 \le m \le n} \left\{ \frac{m}{n} : \sup_{\tilde{\boldsymbol{X}}_\epsilon} \|t(\boldsymbol{X}) - t(\tilde{\boldsymbol{X}}_\epsilon)\| = \infty \right\} \tag{8}$$

Following Theorem 1 of Geisler et al. [9], our proposed Soft Median comes with the best possible breakdown point as we state formally in Theorem 1. Our proof includes 2 parts where in the first part we show the sample's component-wise median is bounded by the n dimensional hypercube created by the clean points. And in the second part we show the output of the aggregation is bounded by the sample's component-wise median.

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 (Equation 6) has the finite sample breakdown point of $\epsilon^*(t_{\text{Soft Median}}, \boldsymbol{X}) = 1/n \lfloor (n+1)/2 \rfloor$ (asymptotically $\lim_{n\to\infty} \epsilon^*(t_{\text{Soft Median}}, \boldsymbol{X}) = 0.5$).*

Proof. *Proof* For clean set we have the component-wise median as the dimension wise median of d dimension where we have $\lceil n/2 \rceil$ values lower and higher than the median element. By adding m elements each element wise median can move at most by $\lceil m/2 \rceil$ element and since we have $m < n$ we conclude the resulting component-wise median $\bar{\boldsymbol{x}}$ is still inside the hypercube that is constructed using the clean points. Moreover, the farther the perturbed points are away from the clean points the greater the distance to the median (as long as $\epsilon < 0.5$) and, hence, the lesser their weights. Due to the properties of the exponential function, if the perturbed points move to infinity the weights of the perturbed points approaches zero, i.e. $\lim_{p\to\infty} p \exp(-p/T) = 0$. On the contrary, the distance of the clean points to the median does not change for a sufficiently large magnitude of the perturbed samples. □

7 EMPIRICAL EVALUATION

In the following, we present our experiments to show the effectiveness and scalability of our proposed attacks. We first describe our setup and then discuss the results over wide range of graphs of different scale.

Defenses. We also report the results on state of the art defenses of [8, 9, 22, 25]. For the Soft Medoid GDC, we use the temperature $T = 0.5$ as it is a good compromise between accuracy and robustness. The SVD GCN [8] uses a (dense) low-rank approximation (here rank 50) of the adjacency matrix to filter adversarial perturbations. RGCN [25] models the neighborhood aggregation via Gaussian distribution to filter outliers, and Jaccard GCN [22] filters edges based on attribute dissimilarity (here threshold 0.01).

Attacks. We compare our GANG, PR-BCD, and GR-BCD attacks (see Sections 5-4) with the global DICE [20], PGD [23], and greedy FGSM attacks Geisler et al. [9]. The greedy FGSM-like attack is the dense equivalent of our GR-BCD attack with the exception of flipping one edge at a time. 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.

Datasets. We use the common Cora ML [2], Citeseer [14], and PubMed [17] for comparing against the other state of the art attacks. For large scale experiments, we use two graphs of the recent Open Graph Benchmark [11]. In comparison to Pubmed, we scale by 2.5 orders of magnitude (number of nodes), or by factor 15,000 if counting the possible adjacency matrix entries (see Table 2). For Cora, Citeseer, as well as PubMed we use an 11GB GeForce GTX 1080 Ti and a 32GB Tesla V100 otherwise. We exclusively perform the calculations on GPU, but for products where we coalesce the adjacency matrix on CPU.

Checkpointing. Empirically, almost 30 GB are required to train a three-layer GCN on Products (our largest dataset) using sparse matrices. However, obtaining the gradient, e.g. towards the perturbation set, 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. [5], 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 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. [9] in their evaluation and for models on OGB we follow Hu et al. [11], 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). Since the edge budget Δ_e in GANG influences the perturbed accuracy (see Figure 2), we decide for a relatively low value of 250. As these are preliminary results, the only exception is Products on which we report the results with the budget of $\Delta_e = 25,000$. Moreover with GANG we binarize the attributes on Cora ML, Citeseer and PubMed and use L_0-norm PGD analogously to PR-BCD.

Results overview. In Table 1 we present the preliminary experimental results for our proposed attacks. We do not observe that sampling the search space harms the attack strength. Similarly to Figure 1, we even outperform the dense PGD on Cora ML. We conclude that our attacks are as effective as the other state of the art attacks.

GNNs' fragility on large graphs. In the following, we analyze the results of PR-BCD, with a budget of $\epsilon = 0.25$, and the GCN. We observe a relative drop in the perturbed accuracy by 20% on Cora, 25% on PubMed, 33 % on arXiv. On products with the lower budget of $\epsilon = 0.1$, we already see a drop of the perturbed accuracy of 31%. We conclude that there is likely some relationship between the

Table 1: Perturbed accuracy for the proposed attacks (see Sections 5-4) and baselines on all datasets (see Table 2). ϵ denotes the fraction of edges perturbed (relative to the clean graph). The last column contains the clean accuracy. As this a work-in-progress report, the experiments for the defenses on the large datasets are due and on Products we did not optimize the hyperparameters for GANG. For each architecture we italicize the strongest attack where $\epsilon = 0.05$, underline where $\epsilon = 0.1$, and embolden where $\epsilon = 0.25$. From an attack perspective, a lower perturbed accuracy is better. We rerun the experiments with three different seeds. For OGB we use the provided data splits and otherwise we use random split with 20 nodes per class.

	Attack	DICE				GANG (ours)				greedy FGSM				GR-BCD (ours)				PGD				PR-BCD (ours)				Accuracy
	Frac. edges ϵ	0.01	0.05	0.1	0.25	0.01	0.05	0.1	0.25	0.01	0.05	0.1	0.25	0.01	0.05	0.1	0.25	0.01	0.05	0.1	0.25	0.01	0.05	0.1	0.25	
Cora ML	Vanilla GCN	0.822	0.813	0.803	0.765	0.809	0.766	0.732	0.658	0.786	*0.691*	0.609	**0.460**	0.790	0.699	0.627	0.506	0.789	0.706	0.638	0.499	0.790	0.711	0.641	0.498	0.825
	Vanilla GDC	0.829	0.820	0.807	0.774	0.822	0.788	0.762	0.712	0.795	*0.704*	0.638	0.528	0.798	0.709	0.640	0.542	0.796	0.716	0.648	**0.525**	0.794	0.717	0.649	0.526	0.831
	SVD GCN	0.758	0.754	0.741	0.696	0.770	0.764	0.760	0.722	0.757	0.743	0.721	0.637	0.757	0.743	0.722	0.633	0.757	*0.725*	0.691	0.579	0.757	0.733	0.691	**0.570**	0.761
	Jaccard GCN	0.817	0.810	0.801	0.769	0.808	0.788	0.768	0.737	0.787	*0.712*	0.644	**0.525**	0.789	0.716	0.655	0.557	0.790	0.715	0.658	0.533	0.790	0.724	0.660	0.532	0.819
	RGCN	0.799	0.794	0.785	0.756	0.732	0.701	0.674	0.603	0.773	*0.700*	0.639	**0.510**	0.774	0.706	0.643	0.529	0.776	0.716	0.656	0.531	0.777	0.712	0.658	0.528	0.800
	Soft Medoid GDC	0.816	0.813	0.806	0.793	0.772	*0.769*	0.765	0.755	0.807	0.789	0.777	0.763	0.806	0.788	0.775	0.755	0.805	0.782	0.760	**0.725**	0.806	0.780	0.758	**0.725**	0.817
	Soft Median GDC	0.819	0.814	0.811	0.797	0.791	0.789	0.785	0.773	0.808	0.785	0.769	0.752	0.808	0.782	0.767	0.742	0.805	*0.776*	0.753	0.718	0.805	*0.776*	0.750	**0.711**	0.819
Citeseer	Vanilla GCN	0.710	0.702	0.691	0.663	0.700	0.675	0.644	0.593	0.685	0.606	0.534	0.390	0.682	*0.602*	0.528	**0.368**	0.687	0.626	0.559	0.428	0.685	0.608	0.544	0.410	0.712
	Vanilla GDC	0.706	0.694	0.682	0.649	0.701	0.686	0.662	0.630	0.683	0.604	0.535	0.413	0.681	*0.602*	0.537	0.407	0.684	0.620	0.562	0.434	0.679	0.611	0.539	**0.405**	0.709
	SVD GCN	0.637	0.625	0.606	0.566	0.639	0.627	0.420	0.539	0.638	0.621	0.599	0.504	0.639	0.624	0.593	0.484	0.638	*0.414*	0.395	0.471	0.635	0.608	0.562	**0.464**	0.641
	Jaccard GCN	0.712	0.707	0.699	0.676	0.710	0.705	0.698	0.691	0.696	0.641	0.592	0.497	0.694	*0.636*	0.589	0.494	0.696	0.642	0.593	0.504	0.693	0.637	0.590	**0.481**	0.714
	RGCN	0.643	0.634	0.624	0.597	0.641	0.624	0.601	0.543	0.630	*0.581*	0.532	**0.424**	0.634	0.590	0.550	0.452	0.635	0.602	0.555	0.460	0.637	0.599	0.560	0.462	0.646
	Soft Medoid GDC	0.707	0.703	0.701	0.694	0.706	0.704	0.700	0.695	0.704	0.695	0.689	0.682	0.702	0.694	0.689	0.678	0.704	*0.688*	0.684	0.667	0.702	0.691	0.682	**0.661**	0.707
	Soft Median GDC	0.709	0.708	0.701	0.693	0.710	0.707	0.703	0.697	0.705	0.691	0.679	0.667	0.704	0.695	0.684	0.663	0.704	0.687	0.671	0.649	0.702	*0.685*	0.669	**0.643**	0.709
arXiv	Vanilla GCN	0.702	0.678	0.661	0.609	0.647	0.464	0.393	0.211	-	-	-	-	0.600	0.484	0.412	0.312	-	-	-	-	0.603	*0.429*	0.277	**0.168**	0.705
	Vanilla GDC	0.615	0.574	0.538	0.462	0.622	0.617	0.611	0.606	-	-	-	-	0.493	*0.341*	0.270	**0.191**	-	-	-	-	0.515	0.373	0.285	0.197	0.617
	Soft Medoid GDC	0.596	0.586	0.576	0.554	0.593	0.585	0.580	0.574	-	-	-	-	0.528	0.472	0.457	0.459	-	-	-	-	0.534	*0.454*	0.413	**0.367**	0.599
Products	Vanilla GCN	-	-	-	-	0.712	0.661	0.601	0.391	-	-	-	-	0.604	*0.508*	0.441	**0.321**	-	-	-	-	0.677	0.624	0.592	0.542	0.719
	Vanilla GDC	-	-	-	-	0.706	0.700	0.694	0.684	-	-	-	-	0.610	*0.575*	0.560	**0.528**	-	-	-	-	0.670	0.627	0.602	0.564	0.709

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	2,995	8,416	2,879	8.970E+06	35.88 MB	168.32 kB
Citeseer	3,312	4,715	3,703	1.097E+07	43.88 MB	94.30 kB
PubMed	19,717	88,648	500	3.888E+08	1.56 GB	1.77 MB
arXiv	169,343	1,166,243	128	2.868E+10	114.71 GB	23.32 MB
Products	2,449,029	123,718,280	100	5.998E+12	23.99 TB	2.47 GB
Papers 100M	111,059,956	1,615,685,872	128	1.233E+16	49.34 PB	32.31 GB

fragility and the graph size. This relationship is similar for GR-BCD but much stronger for GANG. However, for GANG we have to consider that we may choose the attributes as well. arXiv as well as Products have *dense* continuous features, and all the other datasets have *sparse* continuous features. This relationship seems to persist for architectures other than GCN as well. Please note that further experiments are required to confirm this hypothesis. For example, on arXiv and products we use a three-layer GCN (to achieve state of the art accuracy) and for the other datasets we use just two layers. Moreover, the datasets have a different number of classes.

(Todo: Dedicated Experiment)

Time and memory cost. On arXiv, we train for 500 epochs and run the 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 Table 2). Naively attacking the dense adjacency matrix would certainly require more than 1 TB.

(Todo: PPRGo)

8 CONCLUSION

We propose three new attacks that all have the potential to scale to much larger graphs. We are this first to study adversarial attacks on graphs of practical size and, hence, set the cornerstone for the important evaluation of adversarial robustness at scale. We give some intriguing insights. For example, our experiments suggest that adversarial robustness seems to decrease with the size of the graph.

Moreover, it seems to be very different to defend against adversarially added nodes than edge additions or deletions within the existing graph structure. For most applications, we believe that adding new nodes is more realistic than adding edges between existing nodes and, hence, we argue that this setting should be studied more in future work.

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

REFERENCES

[1] Biendata. 2020. KDD Cup 2020: Graph Adversarial Attack and Defense. https://www.biendata.xyz/competition/kddcup{_}2020{_}formal/

[2] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking. *6th International Conference on Learning Representations, ICLR* (2018), 1–13. arXiv:1707.03815

[3] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial attacks on node embeddings via graph poisoning. *36th International Conference on Machine Learning, ICML* 2019-June (2019), 1112–1123. arXiv:1809.01093

[4] Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, and Qi Xuan. 2018. Fast gradient attack on network embedding. *arXiv* (2018), 1–12. arXiv:1809.02797

[5] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training Deep Nets with Sublinear Memory Cost. *arXiv preprint arXiv:1604.06174* (2016). arXiv:1604.06174 http://arxiv.org/abs/1604.06174

[6] Hanjun Dai, Hui Li, Tian Tian, Huang Xin, Lin Wang, Zhu Jun, and Song Le. 2018. Adversarial attack on graph structured data. *35th International Conference on Machine Learning, ICML* 3 (2018), 1799–1808. arXiv:1806.02371

[7] David Donoho and Peter J. Huber. 1983. The notion of breakdown point. In *A Festschrift For Erich L. Lehmann*. Wadsworth Statist./Probab. Ser., Wadsworth, Belmont, CA, 1983, 157–184.

[8] Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. 2020. All you need is Low (rank): Defending against adversarial attacks on graphs. *International Conference on Web Search and Data Mining, WSDM* (2020), 169–177. https://doi.org/10.1145/3336191.3371789

[9] Simon Geisler, Daniel Zügner, and Stephan Günnemann. 2020. Reliable Graph Neural Networks via Robust Aggregation. *Neural Information Processing Systems, NeurIPS* NeurIPS (2020). arXiv:2010.15651 http://arxiv.org/abs/2010.15651

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *3rd International Conference on Learning Representations, ICLR* (2015), 1–11. arXiv:1412.6572

[11] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. (2020), 33 pages. arXiv:2005.00687 http://arxiv.org/abs/2005.00687

[12] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations, ICLR* (2017), 1–14. arXiv:1609.02907

[13] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion Improves Graph Learning. *Neural Information Processing Systems, NeurIPS* (2019). arXiv:1911.05485 http://arxiv.org/abs/1911.05485

[14] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* (2000). https://doi.org/10.1023/A:1009953814988

[15] Yu Nesterov. 2012. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.* 22 (2012), 341–362.

[16] Yu Nesterov and S Stich. 2017. Efficiency of accelerated coordinate descent method on structured optimization problems. *Siam J. Optim.* 27 (2017), 110–123.

[17] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine* (2008). https://doi.org/10.1609/aimag.v29i3.2157

[18] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. 2020. Transferring robustness for graph neural network against poisoning attacks. *Conference on Web Search and Data Mining, WSDM* (2020), 600–608. https://doi.org/10.1145/3336191.3371851 arXiv:1908.07558

[19] Binghui Wang and Neil Zhenqiang Gong. 2019. Attacking graph-based classification via manipulating the graph structure. *ACM Conference on Computer and Communications Security* (2019), 2023–2040. https://doi.org/10.1145/3319535.3354206 arXiv:1903.00553

[20] Marcin Waniek, Tomasz P. Michalak, Michael J. Wooldridge, and Talal Rahwan. 2018. Hiding individuals and communities in a social network. *Nature Human Behaviour* 2, 2 (2018), 139–147. https://doi.org/10.1038/s41562-017-0290-3 arXiv:1608.00375

[21] Stephen J Wright. 2015. Coordinate Descent Algorithms. *Mathematical Programming* 151 (2015), 3–34.

[22] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples for graph data: Deep insights into attack and defense. *IJCAI International Joint Conference on Artificial Intelligence* 2019-Augus (2019), 4816–4823. https://doi.org/10.24963/ijcai.2019/669 arXiv:arXiv:1903.01610v3

[23] Kaidi Xu, Hongge Chen, Sijia Liu, Pin Yu Chen, Tsui Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology attack and defense for graph neural networks: An optimization perspective. *IJCAI International Joint Conference on Artificial Intelligence* 2019-Augus (2019), 3961–3967. https://doi.org/10.24963/ijcai.2019/550 arXiv:1906.04214

[24] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken Ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. *35th International Conference on Machine Learning, ICML* 12 (2018), 8676–8685. arXiv:1806.03536

[25] Dingyuan Zhu, Peng Cui, Ziwei Zhang, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. *International Conference on Knowledge Discovery and Data Mining, KDD* (2019), 1399–1407. https://doi.org/10.1145/3292500.3330851

[26] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. *International Conference on Knowledge Discovery and Data Mining, KDD* (2018), 2847–2856. https://doi.org/10.1145/3219819.3220078 arXiv:1805.07984

[27] Daniel Zügner and Stephan Günnemann. 2019. Adversarial attacks on graph neural networks via meta learning. *7th International Conference on Learning Representations, ICLR* (2019), 1–15. arXiv:1902.08412

(Todo: table of hyperparameters)