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 of Graph Neural Networks (GNNs) has become exceedingly important due to the popularity and diverse applications of GNNs. Specifically, structure perturbations are very effective, but designing attacks that add or remove edges is difficult because of the discrete optimization domain. 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 address the question of how to scale adversarial attacks for evaluating the robustness for such applications. First, we show that the widely used surrogate losses are not well-suited for global attacks on GNNs at scale and propose two-alternatives that overcome these limitations. Second, we propose three attacks based on first-order optimization that do not require a dense adjacency matrix. Hence, we use our methods for global attacks on graphs more than 100 times larger than previously evaluated and scale local attacks to a graph 500 times larger than before. Moreover, one of the proposed attacks considers the very practical case of node injection. Last, we leverage recent advances in differentiable sorting for robust aggregation in message passing that scales linearly with the neighborhood size. We, therefore, improve one of the most effective defense strategies relying on a robust message-passing aggregation. Consequently, we also show how to improve the robustness of a graph neural network at scale.

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, 12 pages. https://doi.org/TBD

Unpublished working draft. Not for distribution.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '21, August 14–18, 2021, Online
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/TBD

1 INTRODUCTION

The adversarial robustness from the perspective of attack and defenses had been widely studied in recent research (Todo: cite). However, most of these works analyzed graphs with less then 20,000 nodes. In particular from the perspective of real-world citation networks or internet-scale applications those graphs are tiny. 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. Our first, experiments show that the adversarial robustness of a GNN depends on the graph size.

We show that contemporary surrogate losses are problematic for global attacks on large graphs. Especially in combination with realistic budgets, previous surrogate losses lead to weak attacks. We propose two new surrogate losses to overcome these limitations. On the traditional graphs, the new losses easily improve the strength of the attack by 33% and for larger datasets we observe gains of more than 100%. Hence, GNNs are even more fragile than previously believed.

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 (19,717 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 3.2, we describe how one might add/remove edges between existing nodes based on Randomized Block Coordinate Descent (R-BCD). Thereafter in Section 3.3 we propose an attack that adds adversarial nodes and was one of the top 5 attacks of the KDD Cup 2020 [1]. In this work, we focus on the important task of node classification. Since we cover the task of global attacks our attacks can easily be generalized to graph classification.

The recent defense of Geisler et al. [10], based on robust aggregations in the message passing step showed supervisor performance over the other compared defenses. We use the very recent advancements in differential sorting (Todo: cite), to propose a computationally less demanding robust, differentiable aggregations. We call this aggregation Soft Median, observe a similar robustness to [10] and

Unpublished working draft. Not for distribution.

can leverage the lower memory footprint in GNNs when memory is at premium.

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 two novel losses for global attacks on graph neural networks that overcome 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.

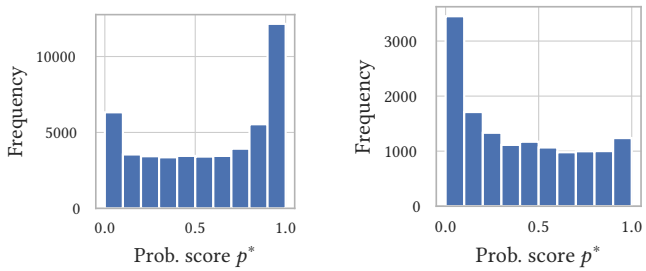
2 CROSS ENTROPY IS A BAD SURROGATE

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 it is often sufficient to maximize the cross entropy

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

Many *global* attacks [5, 24, 26, 30] also perform a global attack on a GNN via maximizing the cross entropy $\max_{\boldsymbol{A}} \mathrm{CE}(f_\theta(\boldsymbol{A}, \boldsymbol{X}))$. However, in our experiments, especially on large graphs, we have often observed that the CE loss increased while the accuracy did not decline. This can be explained by a bias of CE towards nodes which had a low confidence score (misclassified). Maximizing the CE is equivalent to minimizing the data likelihood and poorly correlates with the accuracy given this limited budget. This is apparent in Figure 1.

In contrast of attacking a single image/node, a global structure attack has to 1) keep house with the budget Δ and 2) find edges that degrade the accuracy maximally (i.e. potentially target "fragile" nodes). Without additional information, intuitively, one would first attack low confidence nodes close to the boundary. Analogously, a first-order optimization algorithm focuses on the nodes where loss surface is steep. In general, attacking a GNN is non-convex and presumably NP hard and hence it is hard to make any statements that hold in general. We propose to study surrogate losses for applying first-order optimization to the collective attack of multiple nodes with a global budget Δ based on a greedy scheme. This scheme attacks one node at a time given its confidence scores $\boldsymbol{p}$ or margin $\psi = \min_{c \neq c^*} p_{c^*} - p_c$. Note that such a greedy algorithm is equivalent to a scheduling algorithm where we have the jobs $i \in \mathbb{V}$ of length Δ_i, with the set of nodes $\mathbb{V}$.



(a) Clean graph **(b) Attacked nodes**

Figure 1: In (a) we show the distribution of confidence scores for the correct class p^* over all test nodes on the clean graph. We observe a large fraction of very confident nodes. In stark contrast, in (b) we analyze the distribution of the directly attacked test nodes before the evasion attack started (i.e. if the attack would randomly attack nodes this distribution should match (a)). Here we small budget of one percent of edges ($\Delta = \epsilon = 0.01$) on the arXiv dataset (see Tab.3)

Unpublished working draft. Not for distribution.

Theorem 2.1. *Let $f_\theta(\boldsymbol{A}, \boldsymbol{X})$ be a node classification algorithm applied to a graph with the adjacency matrix $\boldsymbol{A}$, $\mathcal{L}$ is the negative 0/1 loss, and let Δ_i be the budget to move the prediction of any arbitrary node i (independently for each node) over the decision boundary is given by $\Delta_i = |\psi_i| + \eta$ (with an arbitrary small constant η). We can obtain the global optimum of*

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

via greedily attacking the 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 $\Delta < \sum_{i=0}^{l+1} \Delta_i$. $\mathcal{L}'$ denotes the monotonically decreasing surrogate loss and its derivative is (a) $\partial \mathcal{L}'/\partial p^*|_{p^*<0} \leq 0$, has (b) its minimum for $\psi \to 0^+$, and (c) $\partial \mathcal{L}'/\partial p^*|_{p^*>0}$ is monotonically decreasing.

Proof. An optimal solution is obtained by the greedy algorithm that executes the tasks in order $\mathcal{L}(-\eta) - \mathcal{L}(\psi_0)/\psi_0 \geq \mathcal{L}(-\eta) - \mathcal{L}(\psi_1)/\psi_1 \geq \cdots \geq \mathcal{L}(-\eta) - \mathcal{L}(\psi_l)/\psi_l$ until the budget is exceeded: $\Delta < \sum_{i=0}^{l+1} \Delta_i$. 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. In case σ^* would contain one or more tasks for that $w > l$ instead of $b \leq l$. We know that $\psi_w > \psi_b$ and hence $\Delta_w > \Delta_b$. Thus, replacing ψ_b by ψ_w would either lead to the an equal good solution or would be even better (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 $\mathcal{L}'$ must maintain the order above: $\mathcal{L}'(-\eta) - \mathcal{L}'(\psi_0) \geq \mathcal{L}'(-\eta) - \mathcal{L}'(\psi_1) \geq \cdots \geq \mathcal{L}'(-\eta) - \mathcal{L}'(\psi_l)$ in order to reach the global optimum as well (possibly allowing for equivalently good exchanged tasks). The order is preserved if (a) $\mathcal{L}'(-a) \leq \mathcal{L}'(-\eta)$ for every $a \in (\eta, 1]$, (c) $\partial \mathcal{L}'/\partial p^*|_{p^*>0}$ is strictly monotonically decreasing or in other words $\mathcal{L}'$ is strictly concave

for positive inputs. From this it follows that (b) $\partial\mathcal{L}'/\partial p^*$ is minimal for $\psi \to 0^+$. (Todo: Shorten? Reference?) □

Corollary 2.2. *The cross entropy surrogate loss $\mathcal{L}$ = Accuracy ≈ CE (Eq. 1) does not obtain the global optimum since it violates properties (a) and (b).*

Corollary 2.3. *The margin loss $\mathcal{L}$ = Accuracy ≈ Margin Loss = $min(0, \psi)$ does not obtain the global optimum, since its gradient is constant for $\psi > 0$ and therefore violates (b) and (c).*

We argue that the surrogate is certainly not well suited if it even does not work in such a basic scenario. More generally, we state the subsequent conjectures a well-suited, monotonically decreasing surrogate loss $\mathcal{L}'(y, \boldsymbol{p})$ should obey:

Conjecture 2.4. *A proper surrogate loss $\mathcal{L}'(y, \boldsymbol{p})$ should saturate for low confidence values of the correct class:* $\lim_{\psi \to -1^+} \mathcal{L}(y, \boldsymbol{p}) = k < \infty$.

Conjecture 2.5. *A proper surrogate loss $\mathcal{L}'(y, \boldsymbol{p})$ should favour points close to the decision boundary:* $\partial\mathcal{L}(y,\boldsymbol{p})/\partial p_{c^*}|_{\psi>0} > \partial\mathcal{L}(y,\boldsymbol{p})/\partial p_{c^*}|_{\psi\to 0^+}$

In this work we will study several choices for surrogate loss functions:

(1) Cross Entropy: CE = $\log(p_{c^*})$
(2) Carlini-Wagner [4]: 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 = $\tanh(\min_{c\neq c^*} z_{c^*} - z_c)$

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. To simplify notation, we define the losses for a single node (except for MCE) and denote the correct class with c^*.

We choose the losses (1-3) since they are alleged natural choices or have been used in the literature [5, 24, 26, 30]. Loss (1) violates Conjecture 2.4 and losses (2) and (3) violate Conjecture 2.4. (4) and (5) are both losses that obeys both conjectures. Note that (5) would not pass the test of Theorem 2.1. We study it nevertheless since a surrogate loss that saturates already for small negative values could negatively impact the learning dynamics for projected gradient descent.

(Todo: Should we already teaser that thos losses are much better?)

3 ATTACKS

(Todo: Explain chapter)

3.1 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 machine learning and large scale optimization [23]. Nesterov [16] 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 [17]. Constrained optimization with first-order methods can be solved with methods such as Projected Gradient Descent (PGD) or Fast Gradient Sign Method (FGSM) [11]. Similarly, one can extend R-BCD to constrained optimization [16].

Algorithm 1 R-BCD [16]

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**

Adversarial attacks. Beginning with [7, 29], many adversarial attacks on the graph structure have been proposed [3, 20, 21, 24, 25, 30]. 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. Further, we make a distinction between local attacks that attack a single node or small group and global attacks that attack the whole graph. It is apparent that local attacks are much easier to scale than global attacks.

First-order optimization attacks such as Metattack [30] or integrated gradients [24] 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 adjacency matrix $\boldsymbol{A}$, node features $\boldsymbol{X}$, loss $\mathcal{L}$, and the trained network f_θ.

Dai et al. [7] scale their local reinforcement learning approach to a very sparse, large-scale graph for financial transactions. In contrast to our work, they scale their local attack only using a very small budget Δ (their time complexity scales linearly with Δ). Wang and Gong [21] 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 3.3. (Todo: MGA: Momentum Gradient Attack on Network)

3.2 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 $O(n^2)$ (on top of the complexity of

Unpublished working draft. Not for distribution.

the attacked model; in the following we will neglect this fact). 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). We give a formal definition of Projected and Randomized Block Coordinate Descent (PR-BCD) in Algorithm 2. PR-BCD comes with a space complexity of $\Theta(m)$, as we typically choose Δ to be a fraction of m.

As proposed by Xu et al. [25], $\boldsymbol{p}$ is interpreted as the probability for flipping 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 $e \in \{1, 2, \dots, E\}$, this probability mass $\boldsymbol{p}$ is used as edge weight. In this case we overload $\oplus$ s.t. $\boldsymbol{A}_{ij} \oplus p_{ij} = \boldsymbol{A}_{ij} + p_{ij}$ if $\boldsymbol{A}_{ij} = 1$ and $\boldsymbol{A}_{ij} - p_{ij}$ otherwise. 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).

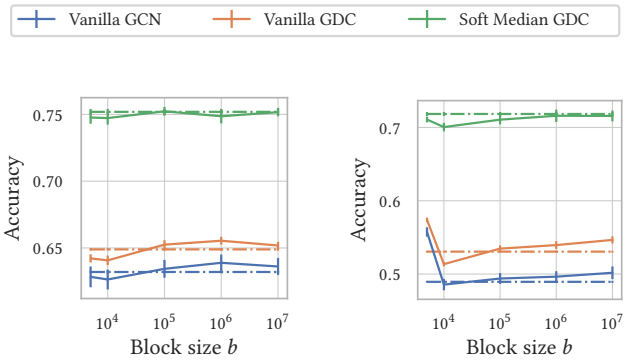
Note that the projection of the perturbation set $\Pi_{\mathbb{E}[\text{Bernoulli}(\boldsymbol{p})]=\Delta}(\boldsymbol{p})$ contains many zero elements, but is not guaranteed to be sparse. If $\boldsymbol{p}$ has more than 50% non-zero entries, we remove the entries with the lowest probability mass 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).

We keep the random block fixed and run K_{resample} epochs. Thereafter, we decay the learning rate as in [25]. 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 $\mathcal{L}$.

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. 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 [22], it is not necessary to test every possible edge to obtain an effective attack. In Fig. 2, 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. For larger graphs, we observe that depending on the block size we observe a greater influence of the block size b. However, as shown in Fig. 3, one might increase the number of epochs for an even improved attack strength. We argue that this indicates that PRBCD successfully spots the right edges to keep.

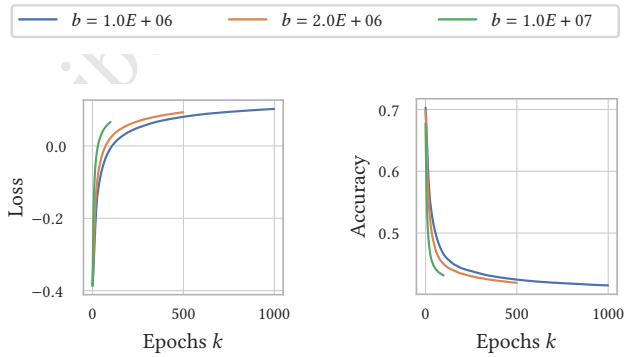
We also compare this approach to a Greedy R-BCD (GR-BCD), that greedily flips the entries with largest gradient in the random search space, such that after K iterations the budget requirements are met.

(Todo: Local version)



(a) $\epsilon = 0.1$ **(i.e.** $\Delta = 798$ **(b)** $\epsilon = 0.25$ **(i.e.** $\Delta = 1995$

Figure 2: We perform the proposed PR-BCD (solid lines) to obtain a perturbed adjacency matrix with the fraction of perturbed edges $\epsilon = 0.1$ and $\epsilon = 0.25$ with loss (5) of Sec. 2. 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 [25]. We report the mean and its three-sigma error over five random seeds.



(a) **(b)**

Figure 3: The perturbed loss and perturbed accuracy over the epochs k on the arXiv dataset (see Tab. 3) using PR-BCD for different block sizes b with loss (5) of Sec. 2. The number of epochs k is chosen such that $kb =$ const.. Hence, approximately the same amount of edges has been "visited". At $k = 100$ the accuracy varies in the range of around 5% despite the very different choices of b. Furthermore, it seems like that the attacks that ran more epochs are slightly stronger.

3.3 Adding Adversarial Nodes

In this section we discuss how solve the attack optimization problem via adding nodes

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

with the space complexity of $O(m)$. With $\boldsymbol{A}|\boldsymbol{A}'$ we denote the addition of rows & columns and with $\boldsymbol{X}|\boldsymbol{X}'$ the concatenation of

Unpublished working draft. Not for distribution.

Algorithm 2 Projected and Randomized Block Coordinate Descent (PR-BCD)

Input: Adj. $\boldsymbol{A}$, feat. $\boldsymbol{X}$, labels $\boldsymbol{y}$, GNN $f_\theta(\boldsymbol{A}, \boldsymbol{X})$, loss $\mathcal{L}$
Parameter: budget Δ, block size b, epochs K, heur. $h(\dots)$
Draw random indices $\boldsymbol{i}_0 \in \{0, 1, \dots, N\}^b$
Initialize zeros for $\boldsymbol{p}_0 \in \mathbb{R}^b$
for $k \in \{1, 2, \dots, K\}$ **do**
  $\hat{\boldsymbol{y}} \leftarrow f_\theta(\boldsymbol{A} \oplus \boldsymbol{p}_{k-1}, \boldsymbol{X})$
  $\boldsymbol{p}_k \leftarrow \boldsymbol{p}_{k-1} + \alpha_{k-1} \nabla_{\boldsymbol{i}_{k-1}} \mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$
  Projection $\boldsymbol{p}_k \leftarrow \Pi_{\mathbb{E}[\text{Bernoulli}(\boldsymbol{p}_k)]=\Delta}(\boldsymbol{p}_k)$
  $\boldsymbol{i}_k \leftarrow \boldsymbol{i}_{k-1}$
  if $k \leq K_{\text{resample}}$ **then**
    $\text{mask}_{\text{resample}} \leftarrow h(\boldsymbol{p}_k)$
    $\boldsymbol{p}_k[\text{mask}_{\text{resample}}] \leftarrow \boldsymbol{0}$
    Resample $\boldsymbol{i}_k[\text{mask}_{\text{resample}}] \in \{0, 1, \dots, N\}^{|\text{mask}_{\text{resample}}|}$
  end if
end for
$\boldsymbol{P} \sim \text{Bernoulli}(\boldsymbol{p}_k)$ s.t. $\sum \boldsymbol{P} \leq \Delta$
Return $\boldsymbol{A} \oplus \boldsymbol{P}$

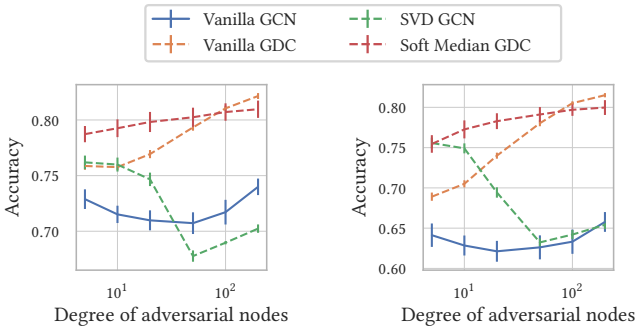


Figure 4: Influence of the degree of the added nodes on the perturbed accuracy on Cora ML. (a) shows the perturbed accuracy with a budget of changing $\epsilon = 0.1$ edges, and (b) for $\epsilon = 0.25$. The number of nodes is determined as $\Delta_n = \epsilon/\Delta_e$. We report the mean perturbed accuracy and its three-sigma error over five random seeds.

the respective attributes ($\boldsymbol{A}$ & $\boldsymbol{X}$ 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/PRBCD 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. 4 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 [13]. Surprisingly, we find that especially GDC [14] (i.e. personalized

Algorithm 3 Greedy Adversarial Node Generation (GANG)

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

PageRank) and a low-rank SVD approximation [9] 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 [10], seems to be effective regardless of the node degree.

4 DEFENSE

We build upon the very recent defence using a robust message passing aggregation that they call Soft Medoid Geisler et al. [10]. Our method *Soft 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. For this we rely on the recent advancements in differentiable sorting Prillo and Martin Eisenschlos [18].

Related work. Many defenses have been proposed, often observing specific characteristics of some attacks. We can classify those defenses into categories such as (1) preprocessing [9, 24], (2) robust training [25, 30], and (3) modifications of the architecture [10, 27, 28]. In this section, we improve the Soft Medoid of [10]. They suggest to interpret a GNNs such as:

$$\mathbf{h}_v^{(l)} = \sigma^{(l)}\left[\text{AGG}^{(l)}\left\{\left(\boldsymbol{A}_{vu}, \mathbf{h}_u^{(l-1)}\boldsymbol{W}^{(l)}\right), \forall\, u \in \mathbb{N}'(v)\right\}\right] \quad (6)$$

with the neighborhood $\mathbb{N}'(v) = \mathbb{N}(v) \cup v$ including the node itself, some message passing aggregation $\text{AGG}^{(l)}$ 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 $\sigma^{(l)}$. And they propose to use the differentiable robust aggregation for $\text{AGG}^{(l)}$ which they call Soft Medoid (for simplicity we only present the unweighted version):

$$t_{\text{SoftMedoid}}(\boldsymbol{X}) = \text{s}\left(-\frac{1}{T}\boldsymbol{d}\right)^\top \boldsymbol{X}, \text{ with } d_v = \sum_{u \in \mathbb{N}'(v)} \|\boldsymbol{X}_{u,:} - \boldsymbol{X}_{v,:}\| \quad (7)$$

where $s(\boldsymbol{z})_i = \exp\left(-T^{-1}z_i\right)/\sum_{j=1}^n \exp\left(-T^{-1}z_j\right)$ is the i-th element of the softmax with temperature. Note that calculating $\boldsymbol{d}$ is equivalent to a row/column sum over the distance matrix with respect to a nodes neighborhood. Hence this operation has a quadratic complexity w.r.t. the neighborhood size and comes with a recognizable memory overhead during training and inference. (Todo: Scalable defenses)

Unpublished working draft. Not for distribution.

Our novel, robust, differentiable aggregation. For improving the previous aggregation we leverage two key facts. First, the Medoid is a multivariate generalization of the median Median and 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 in the space of distances to the dimension-wise Median $\bar{\boldsymbol{x}}$:

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

Intuitively, our proposed aggregation relies on a robust version of the Mahalanobis distance on a spherical Gaussian. To recover the Mahalanobis distance, we would simply need to replace dimension-wise Median with the sample mean. Equivalently, we use the standardized Euclidean distance. Due to the weighting of the samples with the softmax with temperature this also has connections to the density of a bell shaped distribution.

The temperature hyperparameter. Temperature parameter T controls the steepness of the weight distribution $\hat{\boldsymbol{s}}$ between the neighbors and corresponds to the twice the standard deviation in the interpretation as Mahalanobis distance. In the extreme case as $T \to 0$ we recover the point which is closes to the dimension-wise Median (i.e. $\arg\min_{\boldsymbol{x}' \in \mathbb{X}} \|\bar{\boldsymbol{x}} - \boldsymbol{x}'\|$). In the other extreme temperature as $T \to \infty$, the Soft Median is equivalent to the sample mean. We observe a similar behavior as Geisler et al. [10] and by grid search decide for a temperature value of $T = 0.2$ which is a good compromise between clean accuracy and robustness (similar to $T = 0.5$ for the Soft Medoid).

Robustness. Naturally, the question arises if this estimator is robust since in one extreme it recovers the sample mean which is non 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 [8] (here m denotes the number of perturbed examples):

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

Following Theorem 1 of Geisler et al. [10], our proposed Soft Median comes with the best possible breakdown point as we state formally in Theorem 4.1.

Theorem 4.1. *Let $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a collection of points in $\mathbb{R}^d$ with finite coordinates and temperature $T \in [0, \infty)$. Then the Soft Median location estimator (Eq. 8) has the finite sample breakdown point of $\epsilon^*(t_{\text{Soft Median}}, \boldsymbol{X}) = 1/n \lfloor (n+1)/2 \rfloor$ (asymptotically $\lim_{n\to\infty} \epsilon^*(t_{\text{SoftMedian}}, \boldsymbol{X}) = 0.5$).*

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 $\lim_{\tilde{d}_v \to \infty} \|t_{\text{SoftMedian}}(\tilde{\boldsymbol{X}}_\epsilon)\| < \infty$ 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:

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

Instead of $\lim_{\|\tilde{\boldsymbol{x}}_v\| \to \infty} \hat{\mathbf{s}}_v \boldsymbol{x}_v$, we can equivalently derive the limit for the enumerator 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 $\lim_{\tilde{\boldsymbol{x}}_v \| \to \infty} \|\bar{\boldsymbol{x}} - \tilde{\boldsymbol{x}}_v\| = 0$. Consequently, $\|t(\boldsymbol{X}) - t(\tilde{\boldsymbol{X}}_\epsilon)\| = \infty$ can only be true if $m \ge n$ for $T \in [0, \infty)$. □

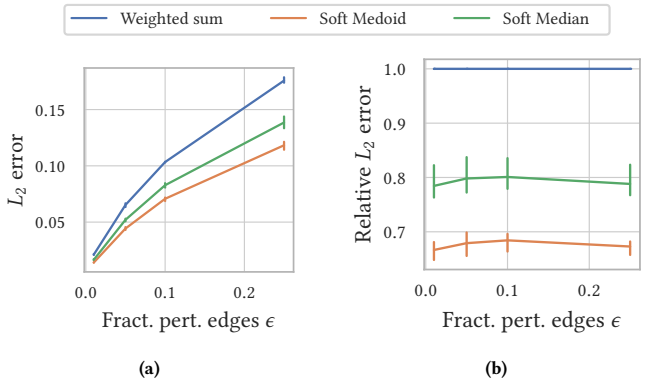


Figure 5: Empirical bias $B(\epsilon)$ for the second layer of a GDC [14] network. (a) shows the absolute bias for a PGD attack, with loss (5) of Sec. 2, and a budget of changing $\epsilon = 0.25$ edges. (b) shows the relative bias over the weighted mean of a GDC. We use for all estimator a temperature of $T = 0.2$

(Todo: Add weighted version)

Empirical robustness. The optimal breakdown point does not necessarily imply that the proposed aggregation is more robust for finite perturbations. In Fig. 5, 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.řeftab:losscompare and reftab:global. An important and interesting fact is, that similarily to the Soft Medoid

Unpublished working draft. Not for distribution.

Table 1: Perturbed accuracy comparing the conventional losses with our loss. We report the mean over three different seeds. ϵ denotes the fraction of edges perturbed (relative to the clean graph). We use random split with 20 nodes per class. For each architecture and budget we embolden the better loss. For details about the set up we refer to Section 5.

Attack	Frac. edges ϵ	Loss	Cora ML [2]							Citeseer [15]						
			Vanilla GCN	Vanilla GDC	SVD GCN	Jaccard GCN	RGCN	Soft Medoid GDC	Soft Median GDC	Vanilla GCN	Vanilla GDC	SVD GCN	Jaccard GCN	RGCN	Soft Medoid GDC	Soft Median GDC
greedy FGSM	0.01	CE	0.8087	0.8144	0.7576	0.8066	0.7864	0.8061	0.8092	0.7052	0.7000	0.6401	0.7091	0.6389	0.7045	0.7061
		CW	0.8079	0.8145	0.7573	0.8041	0.7843	0.8149	0.8167	0.6966	0.6916	0.6394	0.7018	0.6312	0.7070	0.7077
		MCE	**0.7859**	**0.7953**	0.7573	**0.7871**	**0.7730**	0.8070	0.8078	**0.6850**	**0.6832**	**0.6376**	**0.6959**	**0.6305**	0.7037	0.7046
		SCE	0.8124	0.8204	0.7580	0.8084	0.7872	0.8157	0.8170	0.7009	0.6975	0.6387	0.7046	0.6328	0.7075	0.7089
		tanh Margin	0.7920	0.7953	**0.7567**	0.7905	0.7756	**0.8033**	**0.8025**	0.6895	0.6856	0.6383	0.6970	0.6335	**0.7029**	**0.7021**
	0.05	CE	0.7577	0.7586	0.7414	0.7605	0.7428	0.7722	0.7722	0.6693	0.6594	0.6244	0.6799	0.6077	**0.6852**	0.6831
		CW	0.7378	0.7531	0.7445	0.7465	0.7318	0.8054	0.8016	0.6332	0.6385	0.6225	0.6560	0.5818	0.7029	0.7039
		MCE	**0.6908**	0.7045	0.7426	0.7116	0.7004	0.7885	0.7850	**0.6064**	**0.6036**	0.6212	0.6410	**0.5815**	0.6952	0.6911
		SCE	0.7623	0.7750	0.7444	0.7647	0.7466	0.8083	0.8062	0.6519	0.6565	0.6205	0.6693	0.5879	0.7061	0.7059
		tanh Margin	0.7011	**0.7007**	**0.7389**	**0.7071**	**0.6993**	**0.7713**	**0.7639**	0.6082	0.6094	**0.6168**	**0.6348**	0.5927	0.6889	**0.6802**
	0.10	CE	0.7188	0.7179	0.7153	0.7221	0.7080	0.7426	0.7410	0.6316	0.6223	0.6025	0.6476	0.5766	**0.6665**	**0.6590**
		CW	0.6751	0.7001	0.7265	0.7005	0.6874	0.7924	0.7874	0.5692	0.5763	0.5884	0.6112	0.5348	0.6984	0.6968
		MCE	**0.6086**	0.6385	0.7215	0.6441	0.6387	0.7773	0.7693	0.5335	**0.5346**	0.5991	0.5922	**0.5323**	0.6893	0.6791
		SCE	0.7042	0.7235	0.7271	0.7166	0.7065	0.7947	0.7881	0.6004	0.6096	0.5866	0.6298	0.5455	0.7012	0.6998
		tanh Margin	0.6337	**0.6350**	**0.7107**	**0.6430**	**0.6370**	**0.7415**	**0.7320**	**0.5323**	0.5417	**0.5848**	**0.5763**	0.5492	0.6797	0.6668
	0.25	CE	0.6353	0.6391	0.6399	0.6427	0.6312	**0.6785**	0.6746	0.5401	0.5330	**0.3626**	0.5729	0.5130	**0.6201**	**0.6073**
		CW	0.4987	0.5788	0.6344	0.5635	0.5613	0.7738	0.7635	0.4431	0.4804	0.4998	0.5355	0.4369	0.6895	0.6807
		MCE	**0.4599**	0.5275	0.6374	0.5245	**0.5101**	0.7632	0.7524	0.3898	0.4128	0.5036	0.4973	**0.4244**	0.6820	0.6674
		SCE	0.5364	0.5931	**0.4644**	0.5833	0.5785	0.7679	0.7569	0.4558	0.4783	0.4854	0.5373	0.4449	0.6879	0.6768
		tanh Margin	0.5128	**0.5165**	0.6195	**0.5233**	0.5173	0.6841	**0.6744**	**0.3897**	**0.4116**	0.4973	**0.4590**	0.4360	0.6560	0.6369
PGD	0.01	CE	0.8047	0.8107	0.7568	0.8020	0.7848	0.8053	0.8069	0.7032	0.6980	**0.6378**	0.7057	0.6373	**0.7030**	**0.7041**
		CW	0.8124	0.8204	**0.7563**	0.8087	0.7904	0.8142	0.8161	0.7011	0.6975	0.6378	0.7053	**0.6339**	0.7064	0.7086
		MCE	0.8245	0.8314	0.7606	0.8191	0.8000	0.8167	0.8195	0.7121	0.7094	0.6408	0.7137	0.6456	0.7073	0.7093
		SCE	0.8245	0.8314	0.7606	0.8191	0.8000	0.8167	0.8195	0.7121	0.7094	0.6408	0.7137	0.6456	0.7073	0.7093
		tanh Margin	**0.7892**	**0.7960**	0.7567	**0.7903**	**0.7758**	**0.8050**	**0.8053**	**0.6873**	**0.6840**	0.6383	**0.6957**	0.6355	0.7045	0.7045
	0.05	CE	0.7547	0.7555	0.7348	0.7564	0.7390	**0.7735**	**0.7742**	0.6595	0.6551	0.6144	0.6718	0.6068	**0.6852**	**0.6829**
		CW	0.7719	0.7838	0.7369	0.7764	0.7617	0.8083	0.8074	0.6677	0.6613	0.4257	0.6797	**0.5998**	0.7039	0.7050
		MCE	0.8245	0.8314	0.7606	0.8191	0.8000	0.8167	0.8195	0.7121	0.7094	0.6408	0.7137	0.6456	0.7073	0.7093
		SCE	0.8245	0.8314	0.7606	0.8191	0.8000	0.8167	0.8195	0.7121	0.7094	0.6408	0.7137	0.6456	0.7073	0.7093
		tanh Margin	**0.7065**	**0.7165**	**0.7253**	**0.7153**	**0.7155**	0.7816	0.7762	**0.6264**	**0.6205**	**0.4137**	**0.6419**	0.6020	0.6884	0.6868
	0.10	CE	0.7053	0.7045	0.6957	0.7080	0.6991	**0.7432**	**0.7402**	0.6184	0.6102	0.5925	0.6392	0.5836	**0.6708**	**0.6617**
		CW	0.7411	0.7540	0.7043	0.7472	0.7325	0.7988	0.7950	0.6440	0.6376	0.4009	0.6640	0.5733	0.7012	0.6989
		MCE	0.8245	0.8314	0.7606	0.8191	0.8000	0.8167	0.8195	0.7121	0.7094	0.6408	0.7137	0.6456	0.7073	0.7093
		SCE	0.8245	0.8314	0.7606	0.8191	0.8000	0.8167	0.8195	0.7121	0.7094	0.6408	0.7137	0.6456	0.7073	0.7093
		tanh Margin	**0.6379**	**0.6481**	**0.6905**	**0.6577**	**0.6560**	0.7603	0.7532	**0.5592**	**0.5620**	**0.3947**	**0.5932**	**0.5554**	0.6838	0.6713
	0.25	CE	0.5901	0.5953	0.6028	0.6011	0.5942	**0.6830**	**0.6775**	0.5260	0.5175	0.4904	0.5542	0.5039	**0.6283**	**0.6086**
		CW	0.6819	0.7198	0.6144	0.7078	0.6856	0.7848	0.7846	0.5982	0.5907	0.4902	0.6357	0.5217	0.6900	0.6836
		MCE	0.8245	0.8314	0.7606	0.8191	0.8000	0.8167	0.8195	0.7121	0.7094	0.6408	0.7137	0.6456	0.7073	0.7093
		SCE	0.8245	0.8314	0.7606	0.8191	0.8000	0.8167	0.8195	0.7121	0.7094	0.6408	0.7137	0.6456	0.7073	0.7093
		tanh Margin	**0.4993**	**0.5254**	**0.5791**	**0.5332**	**0.5311**	0.7246	0.7175	**0.4283**	**0.4342**	**0.4713**	**0.5043**	**0.4599**	0.6674	0.6487

Unpublished working draft. Not for distribution.

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 scale.

Defenses. We also report the results on state of the art defenses of [9, 10, 24, 28]. 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 [9] uses a (dense) low-rank approximation (here rank 50) of the adjacency matrix to filter adversarial perturbations. RGCN [28] models the neighborhood aggregation via Gaussian distribution to filter outliers, and Jaccard GCN [24] filters edges based on attribute dissimilarity (here threshold 0.01).

Attacks. We compare our GANG, PR-BCD, and GR-BCD attacks (see Sections 3.3-3.2) with the global DICE [22], PGD [25], and greedy FGSM attacks Geisler et al. [10]. 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 [15], and PubMed [19] for comparing against the other state of the art attacks and defenses. For large scale experiments, we use two graphs of the

Table 2: Perturbed accuracy for the proposed attacks (see Sections 3.3-3.2) and baselines on all datasets (see Table 3). 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	
	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	
	Architecture																									
Cora ML	Vanilla GCN	0.822	0.813	0.803	0.765	0.809	0.766	0.732	0.658	0.792	0.701	0.634	0.513	0.790	*0.699*	0.627	0.506	0.825	0.825	0.825	0.825	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	*0.701*	0.635	**0.516**	0.798	0.709	0.640	0.542	0.831	0.831	0.831	0.831	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	0.739	0.711	0.619	0.757	0.743	0.722	0.633	0.761	0.761	0.761	0.761	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	0.791	*0.707*	0.643	**0.523**	0.789	0.716	0.655	0.557	0.819	0.819	0.819	0.819	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	0.776	*0.699*	0.637	**0.517**	0.774	0.706	0.643	0.529	0.800	0.800	0.800	0.800	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	0.803	0.771	0.742	**0.684**	0.806	0.788	0.775	0.755	0.817	0.817	0.817	0.817	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	0.803	*0.764*	0.732	**0.674**	0.808	0.782	0.767	0.742	0.819	0.819	0.819	0.819	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	0.689	0.608	0.532	0.390	0.682	*0.602*	0.528	**0.368**	0.712	0.712	0.712	0.712	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	0.686	0.609	0.542	0.412	0.681	*0.602*	0.537	0.407	0.709	0.709	0.709	0.709	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	0.617	0.585	0.497	0.639	0.624	0.593	0.484	0.641	0.641	0.641	0.641	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	0.697	*0.635*	0.576	**0.459**	0.694	0.636	0.589	0.494	0.714	0.714	0.714	0.714	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	0.634	0.593	0.549	**0.436**	0.634	*0.590*	0.550	0.452	0.646	0.646	0.646	0.646	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	0.703	*0.689*	0.680	**0.656**	0.702	0.694	0.689	0.678	0.707	0.707	0.707	0.707	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	0.702	*0.680*	0.667	**0.637**	0.704	0.695	0.684	0.663	0.709	0.709	0.709	0.709	0.702	0.685	0.669	0.643	0.709
PubMed	Vanilla GCN	0.780	0.767	0.753	0.712	0.758	0.695	0.631	0.493	-	-	-	-	0.752	*0.654*	0.575	**0.457**	-	-	-	-	0.755	0.678	0.607	0.459	0.783
	Vanilla GDC	0.781	0.767	0.754	0.713	0.761	0.722	0.685	0.623	-	-	-	-	0.753	*0.661*	0.588	**0.496**	-	-	-	-	0.756	0.686	0.633	0.550	0.784
	Soft Medoid GDC	0.770	0.763	0.757	0.734	0.726	0.724	0.723	0.719	-	-	-	-	0.756	*0.717*	0.687	**0.660**	-	-	-	-	0.758	0.728	0.708	0.679	0.770
	Soft Median GDC	0.770	0.762	0.757	0.736	0.727	0.725	0.724	0.720	-	-	-	-	0.755	*0.714*	0.683	**0.653**	-	-	-	-	0.758	0.725	0.703	0.671	0.772
arXiv	Vanilla GCN	0.699	0.685	0.661	0.613	0.618	0.489	0.377	0.180	-	-	-	-	0.599	0.473	0.394	0.297	-	-	-	-	0.597	*0.421*	0.304	**0.175**	0.686
	Vanilla GDC	0.635	0.634	0.609	0.552	0.697	0.689	0.663	0.646	-	-	-	-	0.497	0.381	0.305	0.247	-	-	-	-	0.559	*0.379*	0.277	**0.176**	0.677
	Soft Medoid GDC	0.574	0.562	0.552	0.530	0.564	0.557	0.553	0.548	-	-	-	-	0.502	*0.446*	0.429	0.428	-	-	-	-	0.527	0.456	0.420	**0.373**	0.585
	Soft Median GDC	0.656	0.641	0.628	0.593	0.658	0.650	0.644	0.634	-	-	-	-	0.572	0.465	0.422	0.414	-	-	-	-	0.584	*0.463*	0.389	**0.298**	0.665
Products	Vanilla GCN	0.709	0.674	0.636	0.545	0.708	0.659	0.594	0.390	-	-	-	-	0.604	*0.508*	0.441	**0.321**	-	-	-	-	0.619	0.516	0.533	0.480	0.719
	Vanilla GDC	0.701	0.677	0.657	0.618	0.706	0.700	0.695	0.685	-	-	-	-	0.610	0.575	0.560	**0.528**	-	-	-	-	0.628	*0.564*	0.572	0.539	0.709
	Soft Median GDC	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.382	*0.376*	0.373	**0.174**	0.391

Table 3: 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.

	#Nodes n	#Edges e	#Features d	Size (dense)	Size (sparse)
Cora ML [2]	2,995	8,416	2,879	35.88 MB	168.32 kB
Citeseer [15]	3,312	4,715	3,703	43.88 MB	94.30 kB
PubMed [19]	19,717	88,648	500	1.56 GB	1.77 MB
arXiv [12]	169,343	1,166,243	128	114.71 GB	23.32 MB
Products [12]	2,449,029	123,718,280	100	23.99 TB	2.47 GB
Papers 100M [12]	111,059,956	1,615,685,872	128	49.34 PB	32.31 GB

recent Open Graph Benchmark [12]. 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 Table 3). For Cora, Citeseer, PubMed, arXiv as well as Papers 100M we use an 11GB GeForce GTX 1080 Ti. 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.

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. [6], 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. [10] in their evaluation and for models on OGB we follow Hu et al. [12], 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 4), 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.

Evaluation of Losses. In Table 1, we compare the convectional CE loss with our newly proposed losses. For the admittedly large budget of $\epsilon = 0.25$, we see gains of up to 40% on the perturbed accuracy. Moreover, if we compare the accuracy drop (i.e. clean minus perturbed accuracy) we also achieve for low budgets like $\epsilon = 0.01$ an improvements of more then 100%. And those are only the numbers for the small datasets. On larger graphs such as arXiv we even observe gains of more than 100% directly on the perturbed accuracy.

Results overview. In Table 2 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 2, we even outperform the dense PGD on Cora ML. We

Unpublished working draft. Not for distribution.

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 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 3). Naively attacking the dense adjacency matrix would certainly require more than 1 TB.

(Todo: PPRGo / PRBCD vs. NETTACK)

6 CONCLUSION

(Todo: 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] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. *IEEE Symposium on Security and Privacy* (2017), 39–57. https://doi.org/10.1109/SP.2017.49 arXiv:1608.04644

[5] 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

[6] 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

[7] 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

[8] 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.

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

[14] 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

[15] 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

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

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

[18] Sebastian Prillo and Julian Martin Eisenschlos. 2020. SoftSort: A Continuous Relaxation for the argsort Operator. *arXiv* (2020). arXiv:2006.16038

[19] 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

[20] 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

[21] 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

[22] 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

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

[24] 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

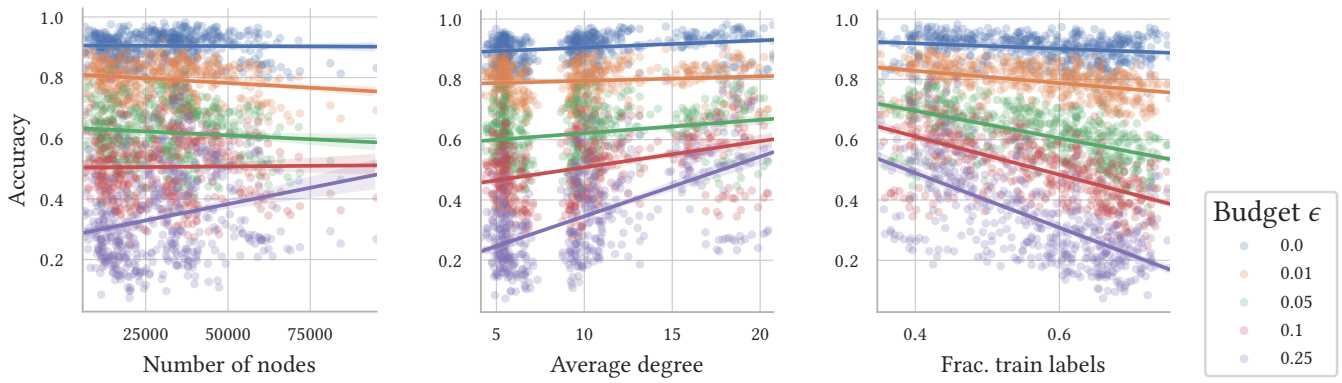
[25] Kaidi Xu, Hongge Chen, Sijia Liu, Pin Yu Chen, Tsui Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology attack and defense for graph neural networks: An optimization perspective. *IJCAI International Joint Conference on Artificial Intelligence* 2019-Augus (2019), 3961–3967. https://doi.org/10.24963/ijcai.2019/550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

[26] 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

[27] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. 2019. Bayesian Graph Convolutional Neural Networks for Semi-Supervised Classification. *AAAI Conference on Artificial Intelligence* 33 (2019), 5829–5836. https://doi.org/10.1609/aaai.v33i01.33015829 arXiv:1811.11103

[28] 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

Unpublished working draft. Not for distribution.



(a) Clean graph **(b)** $\epsilon = 0.01$ **(c)** $\epsilon = 0.1$

Figure 6: For each of the 500 experiments, we sample 8 classes out of the 40 classes of the arXiv dataset and then take the largest connected component. We argue that the difficulty of a classification task is strongly influenced by the number of classes which is apparent by the lower random chance for more classes. In (a) we see, that there is some variance in the results, but the slope is not significant. For low perturbation budgets (b), we see a clear trend that the large graph is less robust (i.e. the drop in accuracy is stronger for large graphs). Around a budget of ϵ the slope becomes insignificant and (d) with a large budget the smaller graphs are less robust. In (b) we observe a correlation of $\rho_b = 0.36$ with a significance of $\alpha_b = 2e-16$ and in (d) $\rho_b = -0.28$ with a significance of $\alpha_b = 2e-10$

[29] 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

[30] 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

Unpublished working draft. Not for distribution.

(Todo: table of hyperparameters)

A ALTERNATIVE MOTIVATION: CROSS ENTROPY IS A BAD SURROGATE

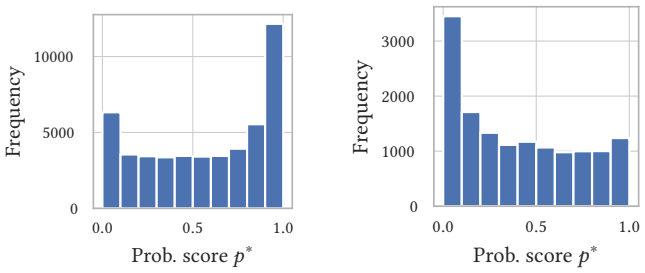
In the context of images, typically a single sample is attacked. In the context of graph neural networks this corresponds to a local attack. For such a scenario an untargeted attack it is often sufficient to maximize the cross entropy

$$\mathrm{CE}^{(n)}(y, \boldsymbol{p}) = \sum_{c \in \mathbb{C}} \mathbb{I}[y^{(n)} = c] \log(p_c)^{(n)} . \tag{10}$$

Many *global* attacks Chen et al. [5], Wu et al. [24], Xu et al. [26], Zügner and Günnemann [30] also attack via maximizing the cross entropy $\max_{\boldsymbol{A}} \mathrm{CE}(f_\theta(\boldsymbol{A}, \boldsymbol{X}))$. However, in our experiments, while attacking GNNs on large graphs, we have often observed that the CE loss increased while the accuracy did not decline. This can be explained by a bias of CE towards nodes which had a low confidence score (misclassified). This is apparent in Figure 7. Intuitively, in contrast of attacking a single image/node, a global structure attack has to 1) keep house with the budget Δ and 2) find edges that degrade the accuracy maximally.

In Theorem A.1, we propose. According to Corollary A.2 and Corollary A.3 it follows that maximizing the CE or the margin loss do not work well for the constructed scenario. For the following discussion, we define the classification margin as $\psi = \min_{c \text{ s.t. } c \neq c^*} p_{c^*} - pc$.

Theorem A.1. *Let $f_\theta(\boldsymbol{A}, \boldsymbol{X})$ be a GNN applied to a large (possibly infinite), diverse graph with the adjacency matrix $\boldsymbol{A}$. Due to the size and diversity of the graph, the attack can move any node's predicted probability it chooses by exactly d and there exists a node for every possible confidence score p^* for the correct class on the clean graph.*



(a) Clean graph **(b) Attacked nodes**

Figure 7: In (a) we show the distribution of confidence scores for the correct class p^* over all test nodes on the clean graph. We observe a large fraction of very confident nodes. In stark contrast, in (b) we analyze the distribution of the directly attacked test nodes before the evasion attack started (i.e. if the attack would randomly attack nodes this distribution should match (a)). Here we small budget of one percent of edges ($\Delta = \epsilon = 0.01$)

The budget Δ is chosen such that the global optimum of

$$\max_{\tilde{\boldsymbol{A}} \text{ s.t. } \|\tilde{\boldsymbol{A}} - \boldsymbol{A}\|_0 < \Delta} \mathcal{L}(f_\theta(\tilde{\boldsymbol{A}}, \boldsymbol{X})) \tag{11}$$

can be obtained by moving a single node's prediction such that the decision boundary is just crossed. That is, after attacking $d \leq \tilde{\psi} < 0$ with an arbitrary small constant d. The derivative of the surrogate loss $\partial \mathcal{L}'/\partial p^*$ that maximizes Eq. 11, must have its unique global minimum s.t. for $\psi \to 0^+$.

Proof. We know that $\mathcal{L}'(\psi - d) > \mathcal{L}'(\psi)$ or equivalently we can analyze the normalized gain $g(\psi)$ and let $d \to 0$:

$$g(\psi) = \lim_{d \to 0^+} -\frac{\mathcal{L}'(\psi - d) - \mathcal{L}'(\psi)}{d} = -\frac{\partial \mathcal{L}'(\psi)}{\partial \psi}$$

Further, to make sure that $\arg\max_\psi \mathcal{L}' = \arg\max_\psi \mathcal{L}$ we must perturb a node that is within $0 \leq \psi < d$ of the decision boundary. Hence,

$$\frac{\partial \mathcal{L}'(\psi)}{\partial \psi}\Big|_{\psi \to 0^+} < \frac{\partial \mathcal{L}'(\psi)}{\partial \psi}\Big|_r \quad \forall r \in \mathbb{R}_*^-$$

and

$$\frac{\partial \mathcal{L}'(\psi)}{\partial \psi}\Big|_{\psi \to 0^+} < \lim_{d \to 0^+} \frac{\partial \mathcal{L}'(\psi)}{\partial \psi}\Big|_{d+r} \quad \forall r \in \mathbb{R}_*^+$$

must hold. Or equivalently,

$$\arg\max_\psi \mathcal{L}' = \arg\min_\psi \frac{\partial \mathcal{L}'(\psi)}{\partial \psi} = \frac{\partial \mathcal{L}'(\psi)}{\partial \psi}\Big|_{\psi \to 0^+}$$

with the unique global minima for $\psi \to 0^+$. □

Corollary A.2. *The cross entropy surrogate loss $\mathcal{L}$ = Accuracy $\approx$ CE (Eq. 10) does not obtain the global optimum since the loss is maximal for nodes with $p^* \to 0^+$.*

Corollary A.3. *The margin loss $\mathcal{L}$ = Accuracy $\approx$ Margin Loss = $min(0, \psi)$ does not obtain the global optimum, since its gradient is constant for $\psi > 0$.*

Of course, the formal statements up to now just covered a very basic scenario where we do not worry about effects such as dependencies between the nodes. However, we argue that the surrogate is certainly not well suited if it even does not work in such a basic scenario. More generally, we state the subsequent conjectures a well-suited, monotonically decreasing surrogate loss $\mathcal{L}^*(y, \boldsymbol{p})$ should obey for globally attacking a node-classification algorithm.

Conjecture A.4. *The loss $\mathcal{L}^*(y, \boldsymbol{p})$ should saturate for low confidence values of the correct class: $\lim_{\psi \to -1^+} \mathcal{L}(y, \boldsymbol{p}) = k < \infty$.*

Conjecture A.5. *The loss should favour points close to the decision boundary: $\partial \mathcal{L}(y,\boldsymbol{p})/\partial p_{c^*}|_{\psi=1} > \partial \mathcal{L}(y,\boldsymbol{p})/\partial p_{c^*}|_{\psi \to 0^+}$.*

One natural choice that obeys the restrictions of the conjectures and the theorem is the masked cross entropy

$$\text{MCE} = \frac{1}{|\mathbb{V}^+|} \sum_{n \in \mathbb{V}^+} \sum_{c \in \mathbb{C}} \mathbb{I}[y^{(n)} = c] \log(p_c^{(n)}) \tag{12}$$

where $\mathbb{V}^+$ is the set of correctly classified nodes.

However, MCE is not a good choice for a projected gradient descent method like the one we are going to propose in Section 3.2. 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

$$\text{tanh Margin} = \frac{1}{|\mathbb{V}|} \sum_{n \in \mathbb{V}} \tanh(\psi^{(n)}) \tag{13}$$

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

In Table 4, we compare the convectional CE loss with our newly proposed losses. For the admittedly large budget of $\epsilon = 0.25$, we see gains of up to 40% on the perturbed accuracy. Moreover, if we compare the accuracy drop (i.e. clean minus perturbed accuracy) we also achieve for low budgets like $\epsilon = 0.01$ an improvements of more then 100%. And those are only the numbers for the small datasets. On larger graphs such as arXiv we even observe gains of more than 100% directly on the perturbed accuracy.

Unpublished working draft. Not for distribution.

Table 4: Perturbed accuracy comparing the conventional losses with our loss. We report the mean over three different seeds. ϵ denotes the fraction of edges perturbed (relative to the clean graph). We use random split with 20 nodes per class. For each architecture and budget we embolden the better loss. For details about the set up we refer to Section 5.

Attack	Frac. edges ϵ	Loss	Cora ML [2]							Citeseer [15]						
			Vanilla GCN	Vanilla GDC	SVD GCN	Jaccard GCN	RGCN	Soft Medoid GDC	Soft Median GDC	Vanilla GCN	Vanilla GDC	SVD GCN	Jaccard GCN	RGCN	Soft Medoid GDC	Soft Median GDC
greedy FGSM	0.01	CE	0.8087	0.8144	0.7576	0.8066	0.7864	0.8061	0.8092	0.7052	0.7000	0.6401	0.7091	0.6389	**0.7045**	0.7061
		MCE	**0.7859**	**0.7953**	**0.7573**	**0.7871**	**0.7730**	**0.8070**	**0.8078**	**0.6850**	**0.6832**	**0.6376**	**0.6959**	**0.6305**	0.7037	**0.7046**
	0.05	CE	0.7577	0.7586	0.7414	0.7605	0.7428	**0.7722**	**0.7722**	0.6693	0.6594	0.6244	0.6799	0.6077	**0.6852**	**0.6831**
		MCE	**0.6908**	**0.7045**	**0.7426**	**0.7116**	**0.7004**	0.7885	0.7850	**0.6064**	**0.6036**	**0.6212**	**0.6410**	**0.5815**	0.6952	0.6911
	0.10	CE	0.7188	0.7179	0.7153	0.7221	0.7080	**0.7426**	**0.7410**	0.6316	0.6223	0.6025	0.6476	0.5766	**0.6665**	**0.6590**
		MCE	**0.6086**	**0.6385**	**0.7215**	**0.6441**	**0.6387**	0.7773	0.7693	**0.5335**	**0.5346**	**0.5991**	**0.5922**	**0.5323**	0.6893	0.6791
	0.25	CE	0.6353	0.6391	0.6399	0.6427	0.6312	**0.6785**	**0.6746**	0.5401	0.5330	**0.3626**	0.5729	0.5130	**0.6201**	**0.6073**
		MCE	**0.4599**	**0.5275**	**0.6374**	**0.5245**	**0.5101**	0.7632	0.7524	**0.3898**	**0.4128**	0.5036	**0.4973**	**0.4244**	0.6820	0.6674
PGD	0.01	CE	0.8047	0.8107	**0.7568**	0.8020	0.7848	**0.8053**	**0.8069**	0.7032	0.6980	0.6378	0.7057	0.6373	**0.7030**	**0.7041**
		tanh Margin	**0.7892**	**0.7960**	0.7567	**0.7903**	**0.7758**	0.8050	0.8053	**0.6873**	**0.6840**	**0.6383**	**0.6957**	**0.6355**	0.7045	0.7045
	0.05	CE	0.7547	0.7555	**0.7348**	0.7564	0.7390	**0.7735**	**0.7742**	0.6595	0.6551	**0.6144**	0.6718	0.6068	**0.6852**	**0.6829**
		tanh Margin	**0.7065**	**0.7165**	0.7253	**0.7153**	**0.7155**	0.7816	0.7762	**0.6264**	**0.6205**	0.4137	**0.6419**	**0.6020**	0.6884	0.6868
	0.10	CE	0.7053	0.7045	**0.6957**	0.7080	0.6991	**0.7432**	**0.7402**	0.6184	0.6102	**0.5925**	0.6392	0.5836	**0.6708**	**0.6617**
		tanh Margin	**0.6379**	**0.6481**	0.6905	**0.6577**	**0.6560**	0.7603	0.7532	**0.5592**	**0.5620**	0.3947	**0.5932**	**0.5554**	0.6838	0.6713
	0.25	CE	0.5901	0.5953	**0.6028**	0.6011	0.5942	**0.6830**	**0.6775**	0.5260	0.5175	**0.4904**	0.5542	0.5039	**0.6283**	**0.6086**
		tanh Margin	**0.4993**	**0.5254**	0.5791	**0.5332**	**0.5311**	0.7246	0.7175	**0.4283**	**0.4342**	0.4713	**0.5043**	**0.4599**	0.6674	0.6487

Unpublished working draft. Not for distribution.