Single-Node Attacks for Fooling Graph Neural Networks

Ben Finkelshtein[a,*], Chaim Baskin [a,**], Evgenii Zheltonozhskii[a], Uri Alon[b]

[a] *Technion, Haifa, Israel*
[b] *Carnegie Mellon University*

Abstract

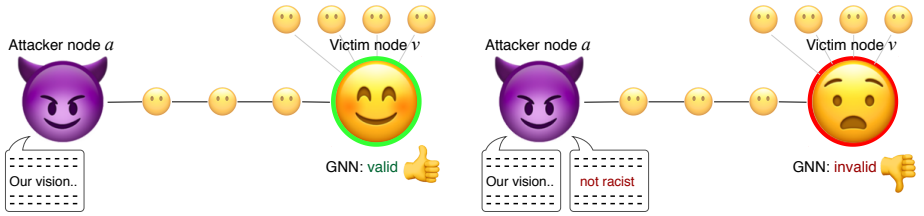
Graph neural networks (GNNs) have shown broad applicability in a variety of domains. These domains, e.g., social networks and product recommendations, are fertile ground for malicious users and behavior. In this paper, we show that GNNs are vulnerable to the extremely limited (and thus quite realistic) scenarios of a single-node adversarial attack, where the perturbed node cannot be chosen by the attacker. That is, an attacker can force the GNN to classify any target node to a chosen label, by only slightly perturbing the features or the neighbors list of another single arbitrary node in the graph, even when not being able to select that specific attacker node. When the adversary is allowed to *select the attacker node*, these attacks are even more effective. We demonstrate empirically that our attack is effective across various common GNN types (e.g., GCN, GraphSAGE, GAT, GIN) and robustly optimized GNNs (e.g., Robust GCN, SM GCN, GAL, LAT-GCN), outperforming previous attacks across different real-world datasets both in a targeted and non-targeted attacks. Our code is available anonymously at https://github.com/gnnattack/SINGLE.

Keywords: Graph neural networks, adversarial robustness, node classification

1. Introduction

Graph neural networks (GNNs) (Scarselli et al., 2009; Micheli, 2009) are becoming increasing popular due to their generality and computation efficiency (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Xu et al., 2019b). Graph-structured data underlie a plethora of domains such as citation networks (Sen et al., 2008), social networks (Ribeiro et al., 2017, 2018), knowledge graphs (Trivedi et al., 2017; Schlichtkrull et al., 2018), and product

*Equal contribution
**Corresponding author
Email addresses: benfin@campus.technion.ac.il (Ben Finkelshtein), chaimbaskin@campus.technion.ac.il (Chaim Baskin), evgeniizh@campus.technion.ac.il (Evgenii Zheltonozhskii), ualon@cs.cmu.edu (Uri Alon)



(a) Before attacking: The victim node (v) is classified as valid.

(b) After attacking: The victim node (v) is classified as invalid.

Figure 1: An partial adversarial example from the test set of the Twitter dataset. An adversarially-crafted post perturbs the representation of the attacker node. This perturbation causes a misclassification of the target victim node, although the two nodes are not even direct neighbors.

recommendations (Shchur et al., 2018). Clearly, GNNs are useful in a variety of real-world data.

Most work in this field has focused on designing new GNN variants and applying them to a growing number of domains. Very few past works, however, have explored the vulnerability of GNNs to realistic adversarial examples. Consider the following scenario: a malicious user or a bot joins a social network such as Twitter or Facebook. The malicious user mocks the behavior of a benign user, establishes connections with other users, and submits benign posts. After some time, the user submits a new adversarially crafted post, that might seem irregular but overall appears benign. If the social network uses a GNN-based model to detect malicious users, the new adversarial post changes the representation of the user as seen by the GNN. As a result, another specific benign user gets blocked from the network; alternatively, another malicious user submits an inciteful or racist post – but does not get blocked. This scenario is illustrated in Fig. 1. In this paper, we show the feasibility of such a troublesome scenario: a single attacker node can perturb its own representation, such that another node will be misclassified as a label of the attacker's choice.

Prior work that explored adversarial attacks on GNNs required the perturbation to span *multiple* aspects of the graph: changing features of multiple nodes (Zügner et al., 2018), adding and removing multiple edges (Dai et al., 2018; Bojchevski & Günnemann, 2019; Li et al., 2020), or both (Zügner et al., 2018; Wu et al., 2019b). In this paper, we demonstrate the surprising effectiveness of a *single*-node attack.

When the attacker node is chosen randomly among nodes within the victim node's reach, our single-node attack reduces test accuracy by (absolute) 7% on average, across multiple datasets. If the attacker node can be *chosen*, for example, by hacking into an existing social network account, the efficiency of the attack significantly increases, and reduces test accuracy by 35%. We also present a new single-edge adversarial attacks on GNNs, where the attacker node is chosen randomly and is limited to either inserting or removing a single edge. In both single-node and single-edge attacks, we present a white-box gradient-based

approach for selecting the attacker. Further, we present a black-box, model-free approach that chooses the attacker node using the graph topology. Finally, we perform an extensive experimental evaluation of our approach on multiple datasets and GNN architectures.

To summarize, our contributions are:

- We present several single-node adversarial attacks, which perturb a few features of the attacker node.
- We present several single-*edge* attacks that add or remove a single edge in the graph.
- We extend our attacks and allow the attacker node to be chosen according to a black-box model-free approach, or a white-box gradient-based approach, which increases the effectiveness of our attack significantly.
- We show experimentally that our approaches and their variations significantly outperform existing attacks on standard, robust, and adversarially trained GNNs.
- Finally, we perform an extensive ablation study, showing the trade-off between the effectiveness of the attack and its unnoticeability.

2. Related work

Works on adversarial attacks on GNNs differ in several main aspects. In this section, we discuss the main criteria, to clarify the settings that we address.

Single vs. multiple node perturbations All previous works allowed perturbing *multiple* nodes, or edges that are covered by multiple nodes: Zügner et al. (2018) perturb features of a *set* of attacker nodes; Zang et al. (2020) assume "a few bad actors". In contrast, we address the extremely limited and more realistic scenario of a *single* attacker node.

Edge perturbations Most adversarial attacks on GNNs perturb the input graph by allowing the insertion or deletion of *multiple* edges (Zügner & Günnemann, 2019; Xu et al., 2019a; Chen et al., 2018; Li et al., 2020; Zügner et al., 2018; Chang et al., 2020). Some previous works dealt with the topic of a single-edge attack briefly (Bojchevski & Günnemann, 2019). Sun et al. (2020) and (Dai et al., 2018) tried to guide a reinforcement learning agent to reduce the GNN node classification performance. Waniek et al. (2018) and Chang et al. (2020) allowed the insertion and deletion of edges, using attacks that are based on correlations and eigenvalues, and not on gradients.The attack of Dai et al. (2018) managed to reduce accuracy by only 10% at most, because it could only *remove* edges. Unlike previous works, we focus on single-edge attacks, where the choice of the edge is gradient-based and a single attacker is chosen randomly or via a white-box approach.

Direct vs. influence attacks Prior works also differ by focusing on either *direct attacks* or *influence attacks*. In direct attacks, the attacker *perturbs the*

victim node itself. For example, the attack of Zügner et al. (2018) is the most effective when the attacker and the target are the same node. In influence attacks, the perturbed nodes are at least one hop away from the victim node. In this paper, we show that the unrealistic *direct* assumption is not required (*Single-Node-direct* in Section 5.3), and that our attack is effective *when the attacker and the target are not even direct neighbors*, i.e., they are at least *two* hops away (*Single-Node-hops* in Section Section 4.5).

Poisoning vs. evasion attacks In a related scenario, some work (Zügner & Günnemann, 2019; Bojchevski & Günnemann, 2019; Li et al., 2020; Zhang & Zitnik, 2020) focused on *poisoning* attacks that perturb examples before training. Contrarily, we focus on the standard *evasion* scenario of adversarial examples (Szegedy et al., 2013; Goodfellow et al., 2014), where the attack operates at test time, after the model is trained.

3. Preliminaries

Let $\mathcal{G} = \{G_i\}_{i=1}^{N_G}$ be a set of graphs. Each graph $G = (\mathcal{V}, \boldsymbol{X}, \mathcal{E}) \in \mathcal{G}$ has a set of N nodes $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where $(u, v) \in \mathcal{E}$ denotes an edge from a node $u \in \mathcal{V}$ to a node $v \in \mathcal{V}$. $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ is a matrix of D-dimensional given node features. The i-th row of $\boldsymbol{X}$ is the feature vector of the node $v_i \in \mathcal{V}$ and is denoted as $\boldsymbol{x}_i = \boldsymbol{X}_{i,:} \in \mathbb{R}^D$.

Graph neural networks. GNNs operate by iteratively propagating neural messages between neighboring nodes. Every GNN layer updates the representation of every node by combining its current representation with the aggregated current representations of its neighbors.

Formally, each node is associated with an initial representation $\boldsymbol{x}_v^{(0)} = \boldsymbol{h}_v^{(0)} \in \mathbb{R}^D$. This representation is considered as the given features of the node. Then, a GNN layer updates each node's representation given its neighbors, yielding $\boldsymbol{h}_v^{(1)} \in \mathbb{R}^D$ for every $v \in \mathcal{V}$. In general, the ℓ-th layer of a GNN is a function that updates a node's representation by combining it with its neighbors:

$$\boldsymbol{h}_v^{(\ell)} = \text{COMBINE}\left(\boldsymbol{h}_v^{(\ell-1)}, \{\boldsymbol{h}_u^{(\ell-1)} \mid u \in \mathcal{N}_v\}; \theta_\ell\right)$$

where $\mathcal{N}_v$ is the set of direct neighbors of v: $\mathcal{N}_v = \{u \in \mathcal{V} \mid (u, v) \in \mathcal{E}\}$.

The COMBINE function is what mostly distinguishes GNN types. For example, graph convolutional networks (GCN) (Kipf & Welling, 2017) define a layer as:

$$\boldsymbol{h}_v^{(\ell)} = \text{ReLU}\left(\sum\nolimits_{u \in \mathcal{N}_v \cup \{v\}} \frac{1}{c_{u,v}} \boldsymbol{W}^{(\ell)} \boldsymbol{h}_u^{(\ell-1)}\right)$$

where $c_{u,v}$ is a normalization factor usually set to $\sqrt{|\mathcal{N}_v| \cdot |\mathcal{N}_u|}$. After ℓ such aggregation iterations, every node representation captures aggregated information from all nodes within its ℓ-hop neighborhood. The total number of layers L is usually determined empirically as a hyperparameter. The final representation $\boldsymbol{h}_v^{(L)}$ is usually used in predicting properties of the node v.

	Cora	CiteSeer	PubMed	Twitter
Clean (no attack)	80.5 ± 0.8	68.5 ± 0.7	78.5 ± 0.6	89.1 ± 0.2
Single-Node	71.5 ± 0.8	47.7 ± 0.8	74.3 ± 0.3	85.5 ± 2.1
Single-Node+GradChoice	64.4 ± 1.0	37.5 ± 0.8	68.5 ± 0.1	53.9 ± 9.8
Single-Node+Topology	61.2 ± 0.9	34.5 ± 1.4	66.0 ± 0.5	55.8 ± 11.2
Single-Node-hops	76.5 ± 0.8	61.2 ± 0.8	75.5 ± 0.2	87.8 ± 1.6
Single-Node-two attackers	67.8 ± 0.8	42.2 ± 0.6	69.9 ± 0.3	84.8 ± 3.3
Single-Node-direct	53.4 ± 1.1	23.4 ± 1.3	46.3 ± 1.1	80.1 ± 2.3

Table 1: Test accuracy (lower is better) under the different variations of a *Single-Node* attack, when the attacker node is chosen *randomly*.

For brevity, we focus our definitions on the common semi-supervised transductive node classification goal, where the dataset contains a single graph G, and the split into training and test sets is across nodes in the same graph. Nonetheless, these definitions can be trivially generalized to the inductive setting, where the dataset contains multiple graphs, the split into training and test sets is between graphs, and the test nodes are unseen during training.

We associate each node $v \in \mathcal{V}$ with a class $y_v \in \mathcal{Y} = \{1, ..., Y\}$. Labels of training nodes are given during training; test nodes are seen during training but without their labels. Given a training subset $\mathcal{D} = \left(\boldsymbol{X}, \mathcal{E}, \{(v_i, y_i)\}_{i=0}^{N_\mathcal{D}}\right)$, the goal is to learn a model $f_\theta : (\boldsymbol{X}, \mathcal{E}, \mathcal{V}) \rightarrow \mathcal{Y}$ that will classify the rest of the nodes correctly. During training, the model f_θ thus minimizes the loss over the given labels, using $J(\cdot, \cdot)$, which is typically the cross-entropy loss:

$$\begin{aligned} \theta^* &= \text{argmin}_\theta \mathcal{L}(f_\theta, \mathcal{D}) = \\ &= \text{argmin}_\theta \frac{1}{N_\mathcal{D}} \sum_{i=0}^{N_\mathcal{D}} J\left(f_\theta\left(\boldsymbol{X}, \mathcal{E}, v_i\right), y_i\right) \end{aligned} \tag{1}$$

4. Method

In this section, we describe our *Single-Node indirect gradient adversariaL evasion* (dubbed *Single-Node*) attack. This is the first influence attack that perturbs nodes, which works with an *arbitrary single attacker node* (in contrast to to multiple (Zügner et al., 2018) and "direct" attacks (Li et al., 2020)). In Section 4.4, we propose a *Single-Edge gradient adversariaL evasion* (dubbed *Single-Edge*) attack. In Section 4.5, we present a white-box (GradChoice) variation for the selection of the attacker for both *Single-Node* and *Single-Edge* along with a black-box (Topology) variation that chooses the attacker node in *Single-Node*, following a heuristic.

	Cora	CiteSeer	PubMed	Twitter
Clean (no attack)	80.5 ± 0.8	68.5 ± 0.7	78.5 ± 0.6	89.1 ± 0.2
R_{ND} (Zügner et al., 2018)	61.0	60.0	-	-
NETTACK-IN (Zügner et al., 2018)	67.0	62.0	-	-
GF-Attack (Chang et al., 2020)	72.6	64.7	72.4	-
Single-Edge (ours)	70.5 ± 0.6	48.2 ± 0.9	59.7 ± 0.7	82.7 ± 0.1
Single-Edge+GradChoice (ours)	$\mathbf{29.7} \pm 2.4$	$\mathbf{11.9} \pm 0.8$	$\mathbf{15.3} \pm 0.4$	82.0 ± 1.4

Table 2: Test accuracy (lower is better) under different edge-based attacks.

4.1. Problem Definition

Given a graph $G = (\mathcal{V}, \boldsymbol{X}, \mathcal{E})$, a trained model f_θ, a "victim" node v from the test set along with its classification by the model $\hat{y}_v = f_\theta(v, \boldsymbol{X}, \mathcal{E})$, we assume that an adversary controls another node a in the graph. The goal of the adversary is to modify its own feature vector $\boldsymbol{x}_a$ by adding a perturbation vector $\boldsymbol{\eta} \in \mathbb{R}^D$ of its choice, such that the model's classification of v will change. We denote by $\boldsymbol{X}_a^{\boldsymbol{\eta}}$ the node feature matrix, where vector $\boldsymbol{\eta}$ was added with to row of $\boldsymbol{X}$ that corresponds to node a. In a non-targeted attack, the goal of the attacker is to find a perturbation vector $\boldsymbol{\eta}$ that will change the classification to *any* other class, i.e., $f_\theta(\mathcal{E}, \boldsymbol{X}_a^{\boldsymbol{\eta}}, v) \neq f_\theta(\mathcal{E}, \boldsymbol{X}, v)$. In a *targeted* attack, the adversary chooses a specific label $y_{adv} \in \mathcal{Y}$ and the adversary's goal is to force $f_\theta(\mathcal{E}, \boldsymbol{X}_a^{\boldsymbol{\eta}}, v) = y_{adv}$.

In this work, we focus on gradient-based attacks. These attacks assume that the attacker can access a similar model to the model under attack and compute gradients. As recently shown by Wallace et al. (2020), this is a reasonable assumption: an attacker can query the original model, imitate the model under attack by training an imitation model, find adversarial examples using the imitation model, and transfer these adversarial examples back to the original model. Under this assumption, these attacks are general and are applicable to any GNN and dataset.

4.2. Limited Perturbations

Our main challenge is to find a **realistic** adversarial perturbation in a way that allows us to constrain its magnitude. In images, this is usually attained by constraining the l_∞-norm of the perturbation vector $\boldsymbol{\eta}$. It is, however, unclear how one can constrain a graph.

In most GNN datasets, a node's features are a bag-of-words representation of the words that are associated with the node. For example, in Cora (McCallum et al., 2000; Sen et al., 2008), every node is annotated by a many-hot feature vector of words that appear in the paper. We denote such datasets as *discrete datasets*, because the given feature vector of every node contains only discrete values. In contrast, in PubMed (Namata et al., 2012), node vectors are TF-IDF word frequencies; in Twitter-Hateful-Users (Ribeiro et al., 2017), node features are averages of GloVe embeddings, which can be viewed as word frequency

vectors multiplied by a (frozen) embedding matrix. We denote such datasets as *continuous datasets*, because the initial feature vector of every node is continuous.

In continuous datasets, the number of times a word has been added or removed by the perturbation should not seem anomalous. Therefore, we constrain the perturbation vector $\boldsymbol{\eta}$ by requiring $\|\boldsymbol{\eta}\|_\infty \leqslant \epsilon_\infty$ – the absolute value of the elements in the perturbation vector is bounded by $\epsilon_\infty \in \mathbb{R}^+$. However, imagine a random post on twitter, solely constraining the l_∞-norm of our perturbation vector. As the average post uses a small set of words from the corpus, our perturbed post, which make use of a larger part of the corpus (with a low frequency) would be easily detected.

To prevent easy detection of the attack, we want the perturbed sample to be in the domain of the training data distribution, preferably with a high likelihood. At the very least, for any choice of norm function, the norm of the perturbed sample features should be comparable with the norm of training samples features.

We limit the number of words (features) we perturb in a node $\|\boldsymbol{\eta}\|_0$ to be strictly smaller than the average number of non-zero entries in the dataset. In this way, the number of words added or removed by the perturbation would not seem anomalous and our perturbation would be indiscernible.

In summary, for continuous datasets, we limit the value of the features we perturb to be strictly smaller than the average over the value of the non-zero entries in the dataset $\|\boldsymbol{\eta}\|_\infty \leqslant \epsilon_\infty$ while also requiring that $\|\boldsymbol{\eta}\|_0/D \leqslant \epsilon_0$: the fraction of non-zero elements in the perturbation vector is bounded by $\epsilon_0 \in [0, 1]$.

For discrete datasets the perturbed vector $\boldsymbol{x}_a+\boldsymbol{\eta}$ must be discrete as well – if every node is given as a many-hot vector $\boldsymbol{x}_a$, the perturbed vector $\boldsymbol{x}_a+\boldsymbol{\eta}$ must remain many-hot as well. Hence, we constrain the l_0-norm of our perturbation vector. Where for discrete datasets, measuring the ℓ_0 norm of $\boldsymbol{\eta}$ is equivalent to the ℓ_1 norm: $\|\boldsymbol{\eta}\|_0 = \|\boldsymbol{\eta}\|_1$.

4.3. Finding the Perturbation Vector

To find the perturbation vector, our general approach is to iteratively differentiate the desired loss of v with respect to the perturbation vector $\boldsymbol{\eta}$, and update $\boldsymbol{\eta}$ according to the gradient, similarly to the general approach in adversarial examples of computer vision models (Goodfellow et al., 2014). In non-targeted attacks, we take the positive gradient of the loss of the undesired label to increase the loss; in targeted attacks, we take the negative gradient of the loss of the adversarial label y_{adv}:

$$\boldsymbol{\eta}_{t+1} = \begin{cases} \boldsymbol{\eta}_t + \gamma \nabla_{\boldsymbol{\eta}} J\left(f_\theta\left(\boldsymbol{X}_a^{\boldsymbol{\eta}_t}, \mathcal{E}, v\right), \hat{y}_v\right) & \text{non-targeted} \\ \boldsymbol{\eta}_t - \gamma \nabla_{\boldsymbol{\eta}} J\left(f_\theta\left(\boldsymbol{X}_a^{\boldsymbol{\eta}_t}, \mathcal{E}, v\right), y_{adv}\right) & \text{targeted} \end{cases}$$

where $\gamma \in \mathbb{R}^+$ is a learning rate. We repeat this process for a predefined number of K iterations, or until the model predicts the desired label.

In continuous datasets, after each update, we clip perturbation vector $\boldsymbol{\eta}_{t+1}$ according to the ϵ_∞ constraint: $\|\boldsymbol{\eta}_{t+1}\|_\infty \leqslant \epsilon_\infty$ and set all attributes of perturbation vector $\boldsymbol{\eta}_{t+1}$ to zero, except for the largest $\epsilon_0 \cdot D$ attributes, according to the ℓ_0 constraint: $\|\boldsymbol{\eta}_{t+1}\|_0/D \leqslant \epsilon_0$.

In discrete datasets, where node features are many-hot vectors, the only possible perturbation to every feature is "flipping" it from 0 to 1 or vice versa. In every update iteration, we thus "flip" the vector attribute with the largest gradient out of the vector attributes that have not been yet flipped. We repeat this process as long as the ℓ_0 constraint holds: $\|\boldsymbol{\eta}_{t+1}\|_0/D \leqslant \epsilon_0$, or until the model predicts the desired label.

Differentiate by frequencies, not by embeddings. When taking the gradient $\nabla_{\boldsymbol{\eta}}$, there is a subtle, but crucial, difference between the way that node representations are provided in the dataset: (a) *indicative* datasets provide initial node representations $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ...]$ that are word indicator vectors (many-hot) or frequencies such as (weighted) bag-of-words (Sen et al., 2008; Shchur et al., 2018); (b) in contrast, in *encoded* datasets, initial node representations are already encoded, e.g., as an average of word2vec vectors (Hamilton et al., 2017). *Indicative* datasets can be converted to *encoded* by multiplying every vector by an embedding matrix; *encoded* datasets *cannot* be converted to *indicative*, without the authors releasing the textual data that was used to create the *encoded* dataset.

In *indicative* datasets, a perturbation of a node vector *can* be realized as a perturbation of the original text from which the *indicative* vector was derived. That is, adding or removing words in the text can result in the perturbed node vector. In contrast, a few-indices perturbation in *encoded* datasets might be an effective attack, but is *not* be realistic because there is no perturbation of the original text that results in that perturbation of the vector. In other words, realistic adversarial examples require *indicative* datasets, or converting *encoded* datasets to *indicative* representation (as we do in Section 5) using their original text.

4.4. Single-Edge GNN Attack

Single-Edge is an attack that either inserts or removes a single edge according to the gradient,[1] of an *arbitrary single attacker node*. We denote the insertion or removal of an edge as a toggle operator that "flips" the current status of the edge.

Finding an edge to flip For each node under attack v and a corresponding attacker node u, we add to the original topology of the graph $\mathcal{E}$ edges between the attacker node u and every node in the (k-1)-hop vicinity of the attacked node $\mathcal{N}_{k-1}(v)$, as nodes that are further away would not influence the attacked node:

$$\mathcal{E}(v) = \{(u, w) | \forall w \in \mathcal{V} \text{ s.t. } (u, w) \in \mathcal{E} \cup w \in \mathcal{N}_{k-1}(v)\} \tag{2}$$

We also add an edge weight vector $W(v)$. To preserve the original topology we initialize the weight of the existing edges in $\mathcal{E}$ to 1 and all other edges to 0.

[1] This can be implemented easily using *edge weights*: training the GNN with weights of 1 for existing edges, adding all possible edges with weights of 0, and taking the gradient with respect to the vector of weights.

We then choose the edge $e(v) \in \mathcal{E}(v)$ with the largest loss gradient and flip it:

$$e^*(v) = \text{argmax}_{e(v)} \begin{cases} -J\left(f_\theta\left(X, \mathcal{E}(v), W(v), v\right), \hat{y}_v\right) & \text{non-targeted} \\ J\left(f_\theta\left(X, \mathcal{E}(v), W(v), v\right), y_{adv}\right) & \text{targeted} \end{cases}$$

Similarly to our single-node approach, for non-targeted attacks we take the negative gradient of the loss of the true label to decrease the loss; for targeted attacks, we take the positive gradient of the loss of the adversarial label y_{adv}.

4.5. Attacker Choice

If the attacker could *choose* the node it will use for the attack, e.g., by hijacking a specifically chosen existing account in a social network, could they increase the effectiveness of the attack? We examine the following approaches of choosing the attacker node.

Single-Node *Gradient Attacker Choice* (*Single-Node+GradChoice*) chooses the attacker node according to the largest gradient with respect to the node representations (for a non-targeted attack): $a^* = \text{argmax}_{a_i \in \mathcal{V}} \|\nabla_{\boldsymbol{x}_i} J\left(f_\theta\left(G, v\right), \hat{y}_v\right)\|_\infty$. The chosen attacker node is never the victim node itself.

Single-Node *Topological Attacker Choice* (*Single-Node+Topology*) chooses the attacker node according to the graph's topological properties. As an example, we choose the neighbor of the victim node v with the smallest number of neighbors as we expect a node with less neighbor to have more influence over the small amount of neighbors he does have (same intuition stands behind GNN normalization): $a^* = \text{argmin}_{a \in \mathcal{N}_v} |\mathcal{N}_a|$. In this approach, the attacker choice is *model-free*: if the attacker cannot compute gradients, they can at least choose the most harmful attacker node, and then perform the perturbation itself using other non-gradient approaches (Waniek et al., 2018; Chang et al., 2020).

Edge Gradient Attacker Choice (GradChoice) is a modification where the edge that is either inserted or removed is sampled from the entire graph, according to the gradient. We compare our two *Single-Edge* approaches with additional approaches from the literature. As in *Single-Node*, we report the means and standard deviations.

5. Evaluation

In this section, we evaluate the effectiveness of our *Single-Node* and *Single-Edge* attacks. In Section 5.1, we demonstrate the effectiveness of our limited perturbation *Single-Node* attack, including its white-box (GradChoice) and black-box (Topology) variations. We also show the effectiveness of our *Single-Edge* attack, which is higher than some multi-edge attacks. In Section 5.2 we show that *Single-Node* is robust to adversarial training and robust GNNs (e.g., Robust GCN, SM GCN, GAL, LAT-GCN). In Sections 5.4 and 5.5 we analyze the effects of ϵ_0 and ϵ_∞ , respectively, and in the supplementary material we examine their trade-off.

	Robust GCN		SM GCN		
	Cora	**CiteSeer**	**Cora**	**CiteSeer**	**PubMed**
Clean	79.7 ± 0.8	58.0 ± 1.9	78.8 ± 0.3	68.2 ± 0.5	78.2 ± 0.6
Single-Node	74.4 ± 0.7	44.5 ± 0.5	48.8 ± 1.4	22.1 ± 1.6	65.7 ± 0.4
Single-Node+GradChoice	69.5 ± 0.5	33.8 ± 0.7	42.3 ± 1.0	18.9 ± 0.3	63.7 ± 0.3
Single-Node+Topology	66.5 ± 0.8	29.5 ± 1.1	38.7 ± 1.0	16.4 ± 0.7	62.3 ± 0.3
Single-Node-hops	79.4 ± 0.8	56.8 ± 1.9	52.4 ± 1.6	29.3 ± 0.6	65.9 ± 0.7
Single-Node-two attackers	69.8 ± 1.0	37.2 ± 1.5	$45.4 \pm 1,3$	21.8 ± 0.8	64.2 ± 0.3
Single-Node-direct	54.2 ± 1.3	18.6 ± 2.8	32.4 ± 0.4	13.8 ± 0.6	29.8 ± 0.5

Table 3: Test accuracy of a robustly trained GCN model (Zügner & Günnemann, 2019) and a GCN with a Soft Medoid as the aggregation function (Geisler et al., 2020)

Setup. We trained each standard GNN type with two layers ($L = 2$), using the Adam optimizer, early stopped according to the validation set, and applied a dropout of 0.5 between layers. We trained each robust GNN according to the author' implementation. We used up to $K = 20$ attack iterations. All experiments described in Section 5 were performed with GCN, except for Section 5.4, where additional GNN types (GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019b)) are used. In the supplementary material, we show consistent results across all GNN types mentioned above as well as SGC (Wu et al., 2019a). We ran each approach five times with different random seeds for each dataset, and report the means and standard deviations. Our PyTorch Geometric (Fey & Lenssen, 2019) implementation is available anonymously at `https://github.com/gnnattack/SINGLE`.

Data. We used Cora and CiteSeer (Sen et al., 2008), which are discrete datasets, i.e., the given node vectors are many-hot vectors. We also used PubMed (Sen et al., 2008) and the Twitter-Hateful-Users (or, shortly, Twitter) (Ribeiro et al., 2017) datasets, which are continuous, and node features represent frequencies of words. As explained in Section 4.2, we limit the fraction of perturbed attributes ϵ_0 for all datasets and the absolute change in each element ϵ_∞ for continuous datasets, which allows finer control over the intensity of the perturbation. The ϵ_0 and ϵ_∞ values for each dataset are provided in the supplementary material. In practice, the attack usually uses *fewer* node attributes. An analysis of values of ϵ_0 and ϵ_∞ is presented in Sections 5.4 and 5.5, and in supplementary material.

The Twitter dataset is originally provided as an *encoded* dataset, where every node is an average of GloVe vectors (Pennington et al., 2014). We reconstructed this dataset using the original text from Ribeiro et al. (2017), to be able to compute gradients with respect to the weighted histogram of words rather than the embeddings. We took the most frequent 10,000 words as node features and used GloVe-Twitter embeddings to multiply by the node features. We thus converted this dataset to indicative rather than encoded. Statistics of all datasets are provided in the supplementary material.

5.1. Main Results

Table 1 presents our main results for non-targeted attacks. Even under the heavy limitations of a single node perturbation, which is limited by the ℓ_0 and the ℓ_∞ norms, *Single-Node* is effective across all datasets.

Single-Node-hops, which is more indiscernible than attacking with a neighboring node, reduces test accuracy by an average of only 5% (absolute), whereas *Single-Node*, which attacks using either a neighboring or non-neighboring node, reduces test accuracy by an average of 11% (absolute).

Choosing the attacker node, whether by using gradients (white-box attack) or topology (black-box), significantly increases the effectiveness of our *Single-Node* attack: for example, in Cora, from 80.5% (Table 1) to 71.5% test accuracy. Furthermore, *Single-Node+Topology* outperforms *Single-Node+GradChoice* across all of the datasets, apart from the Twitter dataset. We believe that white box attack is less effective due to the iterative nature of the attack: it is difficult to differentiate between multiple harmful attackers based on a single gradient step.

Table 2 shows our results for *Single-Edge* non-targeted attacks compared to the previous *multiple*-edge attacks: R_{ND} (Zügner et al., 2018), NETTACK-IN (Zügner et al., 2018) and GF-Attack (Chang et al., 2020) . *Single-Edge* is more effective than previous methods over CiteSeer and PubMed, reducing the test accuracy by 20.3% and 18.8%, respectively. Furthermore, allowing the attacker to perturb an edge from the entire graph using *Single-Edge+GradChoice* significantly increases the effectiveness of our *Single-Edge* attack. As a result, *Single-Edge+GradChoice* is the most effective edge-based attack across all datasets: for example, in PubMed, test accuracy drops from 59.7% (Table 2) to 15.3%. In the supplementary material, we show that allowing *Single-Edge+GradChoice* to insert and remove *multiple* edges of the same attacker node does not lead to a significant improvement.

5.2. Effectiveness of the Attack Facing Defense

In this section, we investigate to what extent defensive training approaches can defend against *Single-Node*.

Adversarial Training. We experimented with attacking models that were adversarially trained (Madry et al., 2018). In each training step we used *Single-Node* or *Single-Node+Topology* on each labeled training node. The model is then trained to minimize the original cross-entropy loss and the adversarial loss:

$$\mathcal{L}(f_\theta, \mathcal{D}) = \frac{1}{2N_{\mathcal{D}}} \sum_{i=0}^{N_{\mathcal{D}}} \left[J\left(f_\theta\left(\boldsymbol{X}, \mathcal{E}, v_i\right), y_i\right) + J\left(f_\theta\left(\boldsymbol{X}_{a_i}^{\boldsymbol{\eta}_i}, \mathcal{E}, v_i\right), y_i\right) \right] \quad (3)$$

The main difference from Eq. (1) is the adversarial term $J\left(f_\theta\left(\boldsymbol{X}_{a_i}^{\boldsymbol{\eta}_i}, \mathcal{E}, v_i\right), y_i\right)$, where a_i is the randomly sampled attacker for node v_i in *Single-Node* or the node chosen according to the topological properties of the graph in *Single-Node+Topology*.

After the model is trained, we attack the model with the different variations of a *Single-Node* attack. This is similar to the approach of Feng et al. (2019) and

Deng et al. (2019). Instead of using adversarial training as a regularization to improve the accuracy of a model on clean data, here we use adversarial training to defend a model against an attack at test time.

As shown in the supplementary material, adversarial training does not improve the model robustness against the different *Single-Node* attacks. Since, the adversarially trained model is much less susceptible to adversarial attacks, and due to the small size of training set, it appears that adversarially trained models are not able to generalize to unseen nodes.

Robust Training. We also experimented with attacking robust models such as robustly trained GCN (Zügner & Günnemann, 2019), where the architecture and the training scheme of GNNs are optimized for robustness, Soft Medoid GCN (Geisler et al., 2020), which employs a robust aggregation function, GAL (Liao et al., 2021), which locally filters out pre-determined sensitive attributes via adversarial training with the total variation and the Wasserstein distance and LAT-GCN (Jin & Zhang, 2020), which perturbs the latent representation of a GNN. We used the publicly available code of Zügner & Günnemann (2019), Geisler et al. (2020) and Liao et al. (2021). For LAT-GCN (Liao et al., 2021), we used our re-implementation. Table 3 shows that robust GNNs are as vulnerable to our *Single-Node* attack as a standard GCN, demonstrating the effectiveness of our attack and indicating that there is still much room for novel ideas and improvements to the robustness of current GNNs. Additional results for robust GNN architectures are presented in the supplementary material.

5.3. Scenario Ablation

The main scenario that we focus on in this paper is a *Single-Node* approach that always perturbs a *single* node, which is not the victim node ($a \neq v$). For each victim node, the attacker node is selected randomly and the attack perturbs the chosen attacker node's features according to the ϵ_∞ and ϵ_0 values. We now examine our *Single-Node* attack in other, easier but less realistic, scenarios:

Single-Node-*hops* is a modification of *Single-Node* where the attacker node is sampled *only among nodes that are not direct neighbors*, i.e., the attacker and the victim are not directly connected ($(a, v) \notin \mathcal{E}$). The idea in Single-Node-*hops* is to evaluate a variant of Single-Node that is more indiscernible in reality.

Single-Node-two attackers follows Zügner et al. (2018) and Zang et al. (2020). It randomly samples two attacker nodes and perturbs their features using the same approach as *Single-Node*.

Single-Node-direct perturbs the victim node directly (i.e., $a = v$), an approach that was found to be the most efficient by Zügner et al. (2018). Table 1 shows the test accuracy of these ablations. Expectantly, perturbing two attacker nodes or perturbing the victim node itself is more effective, albeit less realistic.

Larger number of attackers. We performed additional experiments with up to five randomly sampled attacker nodes simultaneously and perturb their features using the same approach as *Single-Node*, presented in supplementary material.

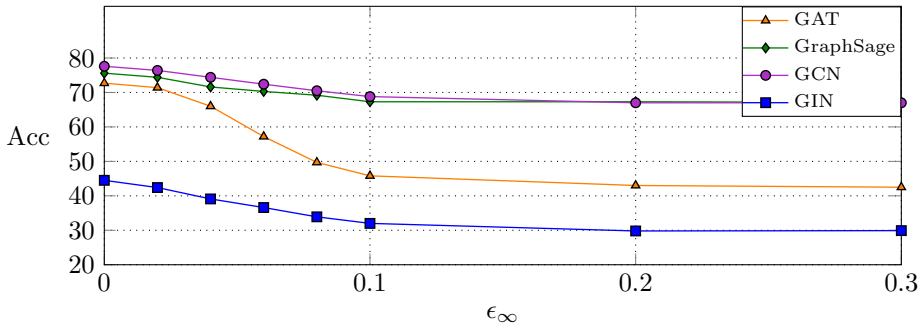


Figure 2: Effectiveness of our *Single-Node* attack compared to the allowed ϵ_∞ (on PubMed).

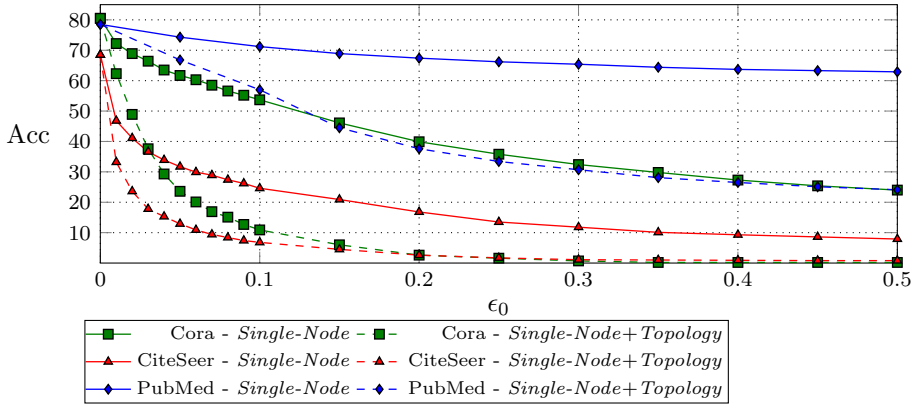


Figure 3: Test accuracy under our *Single-Node* attack, compared to ϵ_0, the maximal ratio of perturbed features.

5.4. Sensitivity to ϵ_∞

How does the norm of the adversarial perturbation affect the attack? Intuitively, the less we restrict the perturbation (i.e., the larger the value of ϵ_∞ is), the more powerful the attack. We examine whether this holds in practice.

In our experiments described in Sections 5.1 to 5.3, we used $\epsilon_\infty = 0.1$ for the continuous datasets (PubMed and Twitter). Here, we vary the value of ϵ_∞ across different GNN types. Fig. 2 shows the results on PubMed and demonstrates that smaller values of ϵ_∞ are effective as well. As the value of ϵ_∞ increases, GAT (Veličković et al., 2018) demonstrates a large drop in test accuracy. In contrast, GCN, GraphSage and and GIN (Xu et al., 2019b) are more robust to an increased norm of perturbations.

5.5. Sensitivity to ϵ_0

In Section 5.4, we analyzed the effect of ϵ_∞, the maximal allowed perturbation in each vector attribute, on the performance of the attack. In Cora and CiteSeer, the input features are discrete (i.e., the given input node vector is

many-hot). In such datasets, the interesting analysis focuses on the value of ϵ_0, the maximal *fraction* of allowed perturbed vector elements, on the performance of the attack. Here, we vary the value of ϵ_0 across different datasets. We also included an analysis of ϵ_0 for the PubMed dataset, with a constant $\epsilon_\infty = 0.04$. We experimented with limiting ϵ_0 and measuring the resulting test accuracy for both *Single-Node* and *Single-Node+Topology*. The results appear in Fig. 3.

As we increase ϵ_0, the test accuracy naturally decreases for all datasets, whether they are discrete or continuous and for both *Single-Node* and *Single-Node+Topology*.

6. Conclusion

In this paper, we show that GNNs are susceptible to the extremely limited scenario of a *Single-node INdirect Gradient adversariaL Evasion* (*Single-Node*) attack. Furthermore, we show that even robustly optimized GNNs and adversarial training fail in defending against a limited single-node attack. We also present a new *Single-Edge gradient adversariaL evasion* (*Single-Edge*) attack that is stronger than its predecessors.

We perform a thorough experimental evaluation across multiple variations of the *Single-Node* attack, datasets, GNN types and also an extensive ablation study. The practical consequences of these findings are that a single attacker can force a GNN to classify any other node as the attacker's chosen label by slightly perturbing some of the attacker's features. Furthermore, if the attacker can choose its attacker node, the effectiveness of the attack significantly increases.

We believe that this work will drive research toward exploring novel defense approaches for GNNs. Such defenses can be crucial for real-world systems that are modeled using GNNs. We also believe that this work's surprising results motivate a search for a better theoretical understanding of the expressiveness and generalization of GNNs.

References

Bojchevski, A., & Günnemann, S. (2019). Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning* (pp. 695–704).

Chang, H., Rong, Y., Xu, T., Huang, W., Zhang, H., Cui, P., Zhu, W., & Huang, J. (2020). A restricted black-box adversarial framework towards attacking graph embedding models. In *AAAI* (pp. 3389–3396).

Chen, J., Wu, Y., Xu, X., Chen, Y., Zheng, H., & Xuan, Q. (2018). Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*, .

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., & Song, L. (2018). Adversarial attack on graph structured data. In *International Conference on Machine Learning* (pp. 1115–1124).

Deng, Z., Dong, Y., & Zhu, J. (2019). Latent adversarial training of graph convolution networks. In *ICML Workshop on Learning and Reasoning with Graph-Structured Representations*.

Feng, F., He, X., Tang, J., & Chua, T.-S. (2019). Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering*, .

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Geisler, S., Zügner, D., & Günnemann, S. (2020). Reliable graph neural networks via robust aggregation. *arXiv preprint arXiv:2010.15651*, .

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, .

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (pp. 1024–1034). Curran Associates, Inc. volume 30.

Jin, H., & Zhang, X. (2020). Robust training of graph convolutional networks via latent perturbation. In *ECML/PKDD (3)* (pp. 394–411).

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Li, J., Xie, T., Chen, L., Xie, F., He, X., & Zheng, Z. (2020). Adversarial attack on large scale graph. *arXiv preprint arXiv:2009.03488*, .

Liao, P., Zhao, H., Xu, K., Jaakkola, T., Gordon, G. J., Jegelka, S., & Salakhutdinov, R. (2021). Information obfuscation of graph neural networks. In *International Conference on Machine Learning* (pp. 6600–6610). PMLR.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

McCallum, A. K., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, *3*, 127–163.

Micheli, A. (2009). Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, *20*, 498–511.

Namata, G. M., London, B., Getoor, L., & Huang, B. (2012). Query-driven active surveying for collective classification. In *Workshop on Mining and Learning with Graphs*.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543).

Ribeiro, M. H., Calais, P. H., Santos, Y. A., Almeida, V. A. F., & Meira Jr, W. (2017). "Like sheep among wolves": Characterizing hateful users on twitter. *arXiv preprint arXiv:1801.00317*, .

Ribeiro, M. H., Calais, P. H., Santos, Y. A., Almeida, V. A. F., & Meira Jr, W. (2018). Characterizing and detecting hateful users on twitter. *arXiv preprint arXiv:1803.08977*, .

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*, 61–80.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference* (pp. 593–607). Springer.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*, 93.

Shchur, O., Mumme, M., Bojchevski, A., & Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop, NeurIPS 2018*, .

Sun, Y., Wang, S., Tang, X., Hsieh, T.-Y., & Honavar, V. (2020). Non-target-specific node injection attacks on graph neural networks: A hierarchical reinforcement learning approach. In *Proc. WWW*. volume 3.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, .

Trivedi, R., Dai, H., Wang, Y., & Song, L. (2017). Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *International Conference on Machine Learning* (pp. 3462–3471).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Wallace, E., Stern, M., & Song, D. (2020). Imitation attacks and defenses for black-box machine translation systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 5531–5546).

Waniek, M., Michalak, T. P., Wooldridge, M. J., & Rahwan, T. (2018). Hiding individuals and communities in a social network. *Nature Human Behaviour*, *2*, 139–147.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019a). Simplifying graph convolutional networks. In *International Conference on Machine Learning* (pp. 6861–6871).

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., & Zhu, L. (2019b). Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (pp. 4816–4823). AAAI Press.

Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., & Lin, X. (2019a). Topology attack and defense for graph neural networks: an optimization perspective. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (pp. 3961–3967). AAAI Press.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019b). How powerful are graph neural networks? In *International Conference on Learning Representations*.

Zang, X., Xie, Y., Chen, J., & Yuan, B. (2020). Graph universal adversarial attacks: A few bad actors ruin graph learning models. *arXiv preprint arXiv:2002.04784*, .

Zhang, X., & Zitnik, M. (2020). GNNGuard: Defending graph neural networks against adversarial attacks. *arXiv preprint arXiv:2006.08149*, .

Zügner, D., Akbarnejad, A., & Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zügner, D., & Günnemann, S. (2019). Certifiable robustness and robust training for graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zügner, D., & Günnemann, S. (2019). Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations*.

Table A.1: Dataset statistics.

	#Training	#Val	#Test	#Unlabeled Nodes	#Classes	Avg. Node Degree
Cora	140	500	1000	2708	7	3.9
CiteSeer	120	500	1000	3327	6	2.7
PubMed	60	500	1000	19717	3	4.5
Twitter	4474	248	249	95415	2	45.6

Table A.2: Default ℓ_0 and ℓ_∞.

	#Features	Avg. ratio of non-zero features	ℓ_0	Avg. Amplitude of non-zero feature	ℓ_∞
Cora	1433	0.013	0.01	-	-
CiteSeer	3703	0.009	0.01	-	-
PubMed	500	0.1	0.05	0.04	0.04
Twitter	10000	0.052	0.1	0.0009	0.01

Appendix A. Dataset Statistics

Statistics of the datasets are shown in Table A.1.

Table A.2 displays the default settings for the ℓ_0 norm and ℓ_∞ norm of each dataset, where the ℓ_0 and ℓ_∞ norms are influenced by the average ratio of non-zero attributes and the average amplitude of the non-zero attributes, respectively.

Appendix A.1. Additional GNN Types

Tables A.3 to A.5 present the test accuracy of different attacks applied on GAT (Veličković et al., 2018), GIN (Xu et al., 2019b), GraphSAGE (Hamilton et al., 2017), and SGC (Wu et al., 2019a), showing the effectiveness of *Single-Node* across different GNN types.

	Cora	CiteSeer	PubMed
Clean	78.2 ± 1.4	65.6 ± 1.4	75.0 ± 0.3
Single-Node	67.9 ± 1.1	45.4 ± 3.4	68.9 ± 3.7
Single-Node+GradChoice	58.2 ± 1.6	37.0 ± 4.7	52.0 ± 2.1
Single-Node+Topology	59.4 ± 2.2	36.6 ± 4.3	51.5 ± 5.5
Single-Node-hops	72.4 ± 0.9	54.7 ± 3.0	70.5 ± 3.5
Single-Node-two attackers	64.4 ± 1.0	41.2 ± 3.9	62.6 ± 4.9
Single-Node-direct	55.4 ± 3.7	31.1 ± 3.9	47.7 ± 6.9
Single-Edge	70.5 ± 0.6	49.4 ± 1.4	64.9 ± 1.0
Single-Edge+GradChoice	67.8 ± 4.9	48.3 ± 5.1	63.5 ± 4.6

Table A.3: Test accuracy of GAT under different non-targeted attacks

	Cora	**CiteSeer**	**PubMed**
Clean	57.7 ± 1.4	39.5 ± 1.9	55.0 ± 4.4
Single-Node	14.9 ± 1.8	14.9 ± 1.0	36.8 ± 8.1
Single-Node+GradChoice	26.8 ± 1.7	8.8 ± 1.1	24.9 ± 7.3
Single-Node+Topology	25.9 ± 1.7	8.1 ± 1.5	21.2 ± 7.2
Single-Node-hops	42.1 ± 1.9	18.9 ± 0.7	37.7 ± 8.2
Single-Node-two attackers	32.5 ± 2.0	11.9 ± 1.4	31.1 ± 8.3
Single-Node-direct	20.4 ± 1.9	5.2 ± 2.0	41.1 ± 0.7
Single-Edge	32.9 ± 3.1	18.5 ± 3.0	33.3 ± 1.7
Single-Edge+GradChoice	10.7 ± 2.8	4.8 ± 2.1	10.3 ± 1.0

Table A.4: Test accuracy of GIN under different non-targeted attacks

	Cora	**CiteSeer**	**PubMed**
Clean	78.7 ± 0.3	66.0 ± 0.5	75.9 ± 0.5
Single-Node	70.7 ± 2.0	44.7 ± 2.3	72.2 ± 0.5
Single-Node+GradChoice	56.2 ± 2.9	28.1 ± 1.7	46.5 ± 0.4
Single-Node+Topology	57.5 ± 2.9	29.8 ± 1.9	43.6 ± 0.5
Single-Node-hops	76.3 ± 1.9	60.2 ± 2.6	72.2 ± 0.5
Single-Node-two attackers	65.8 ± 1.7	40.8 ± 1.5	67.6 ± 0.7
Single-Node-direct	47.7 ± 1.4	21.9 ± 2.0	41.1 ± 0.7
Single-Edge	62.9 ± 1.9	45.9 ± 3.4	64.2 ± 1.6
Single-Edge+GradChoice	48.9 ± 2.7	40.4 ± 3.3	64.7 ± 1.1

Table A.5: Test accuracy of GraphSAGE under different non-targeted attacks

	Cora	**CiteSeer**
Clean	79.9 ± 0.6	67.9 ± 0.2
Single-Node	70.4 ± 0.3	46.6 ± 0.5
Single-Node+GradChoice	60.4 ± 0.5	35.8 ± 0.4
Single-Node+Topology	55.9 ± 0.5	33.4 ± 0.4
Single-Node-hops	75.9 ± 0.3	59.5 ± 0.4
Single-Node-two attackers	64.5 ± 0.5	40.8 ± 0.5
Single-Node-direct	45.1 ± 0.4	22.6 ± 0.5
Single-Edge	71.3 ± 1.0	55.0 ± 1.6
Single-Edge+GradChoice	29.7 ± 1.7	13.5 ± 2.0

Table A.6: Test accuracy of SGC (Wu et al., 2019a) under different non-targeted attacks.

	Cora	CiteSeer	PubMed
Single-Node	9.1 ± 0.4	21.6 ± 0.8	12.7 ± 0.8
Single-Node+GradChoice	14.5 ± 1.2	27.9 ± 1.1	15.9 ± 0.8
Single-Node+Topology	17.0 ± 1.5	30.9 ± 1.0	17.2 ± 0.7
Single-Node-hops	4.2 ± 0.3	8.8 ± 1.0	12.4 ± 0.9
Single-Edge	8.0 ± 0.7	14.8 ± 0.5	20.1 ± 0.6
Single-Edge+GradChoice	59.4 ± 0.9	78.7 ± 0.9	80.1 ± 0.6

Table B.7: Success rate (higher is better) of different *targeted* attacks on a GCN network.

Appendix B. Targeted Attacks

Tables B.7 to B.10 show the results of *targeted* attacks across datasets and approaches. Differently from other tables that show test accuracy, Tables B.7 to B.10 present the targeted attack's *success rate*, which is the fraction of test examples that the attack managed to force to make *a specific label prediction* (in these results, higher is better).

	Cora	CiteSeer	PubMed
Single-Node	13.7 ± 1.2	28.7 ± 3.6	15.5 ± 1.7
Single-Node+GradChoice	25.8 ± 2.2	38.6 ± 4.7	20.6 ± 2.3
Single-Node+Topology	24.9 ± 2.6	18.4 ± 5.3	22.1 ± 1.6
Single-Node-hops	7.2 ± 0.9	14.4 ± 2.0	14.7 ± 1.8
Single-Edge	6.1 ± 0.4	12.5 ± 1.2	17.9 ± 1.5
Single-Edge+GradChoice	6.0 ± 1.4	14.6 ± 2.8	22.3 ± 3.6

Table B.8: Success rate (higher is better) of different *targeted* attacks on a GAT network.

	Cora	CiteSeer	PubMed
Single-Node	23.6 ± 0.4	50.9 ± 4.5	39.4 ± 9.2
Single-Node+GradChoice	36.0 ± 0.7	64.5 ± 5.0	49.2 ± 7.2
Single-Node+Topology	36.9 ± 1.3	64.8 ± 5.5	52.2 ± 7.1
Single-Node-hops	17.1 ± 1.0	34.9 ± 3.3	38.5 ± 9.3
Single-Edge	16.8 ± 1.2	25.6 ± 1.0	37.9 ± 2.6
Single-Edge+GradChoice	44.7 ± 4.7	55.0 ± 7.0	64.9 ± 11.8

Table B.9: Success rate (higher is better) of different *targeted* attacks on a GIN network.

Appendix B.1. Adversarial Training

Table B.11 presents the test accuracy on PubMed of a model that was trained adversarially on the *Single-Node* attack or the *Single-Node+Topology* attack. It

	Cora	**CiteSeer**	**PubMed**
Single-Node	9.8 ± 0.7	25.2 ± 1.1	14.2 ± 0.9
Single-Node+GradChoice	20.5 ± 2.1	40.5 ± 1.5	23.3 ± 1.4
Single-Node+Topology	20.1 ± 1.4	39.1 ± 1.6	26.8 ± 1.1
Single-Node-hops	4.6 ± 0.6	8.6 ± 0.7	12.9 ± 0.9
Single-Edge	7.6 ± 0.3	16.3 ± 1.7	19.1 ± 1.4
Single-Edge+GradChoice	9.3 ± 0.9	14.7 ± 1.1	19.6 ± 0.8

Table B.10: Success rate (higher is better) of different *targeted* attacks on a GraphSAGE network.

	Std.	**Adv.**	**Adv.+Top.**
Clean (no attack)	78.5 ± 0.6	76.0 ± 1.8	76.0 ± 1.9
Single-Node	74.3 ± 0.3	73.5 ± 1.9	72.9 ± 1.6
Single-Node-hops	75.5 ± 0.2	73.5 ± 1.8	73.7 ± 1.9
Single-Node+GradChoice	68.5 ± 0.1	66.5 ± 2.6	66.5 ± 2.7
Single-Node+Topology	66.0 ± 0.5	64.8 ± 1.9	64.7 ± 1.8
Single-Node-two attackers	69.9 ± 0.3	69.1 ± 1.6	69.0 ± 1.9
Single-Node-direct	46.3 ± 1.1	47.1 ± 0.7	46.8 ± 0.6

Table B.11: Test accuracy on PubMed of a model that was trained adversarially on the *Single-Node* attack or the *Single-Node+Topology* attack.

	GAL (unique train/val/test split)		LAT-GCN		
	CiteSeer	**PubMed**	**Cora**	**CiteSeer**	**PubMed**
Clean	78.4 ± 1.3	74.9 ± 3.5	80.1 ± 0.3	69.4 ± 0.5	73.7 ± 1.8
Single-Node	25.1 ± 2.9	71.0 ± 2.4	62.2 ± 0.8	35.2 ± 0.7	43.8 ± 5.4
Single-Node+GradChoice	10.1 ± 2.8	63.7 ± 2.7	55.3 ± 0.8	27.5 ± 1.1	40.0 ± 6.9
Single-Node+Topology	10.5 ± 2.0	60.7 ± 1.8	52.9 ± 0.7	27.0 ± 0.5	40.1 ± 6.9
Single-Node-hops	42.3 ± 4.0	72.5 ± 2.4	67.0 ± 0.8	46.5 ± 0.6	44.9 ± 5.2
Single-Node-two attackers	26.1 ± 2.9	67.2 ± 2.4	57.1 ± 0.5	34.0 ± 0.9	40.5 ± 5.7
Single-Node-direct	7.9 ± 2.3	54.5 ± 1.7	44.8 ± 0.8	17.9 ± 1.3	31.0 ± 5.0

Table C.12: Test accuracy of Graph Adversarial Networks (GAL) (Liao et al., 2021) with a unique unique train/val/test split and LAT-GCN (Jin & Zhang, 2020) under different non-targeted attacks

shows that adversarial training does not improve the model robustness against the different *Single-Node* attacks.

Appendix C. Additional Robust GNN Types

Table C.12 presents the test accuracy of different attacks applied on *GAL* (Liao et al., 2021) and *LAT-GCN* (Jin & Zhang, 2020). It shows the effectiveness of *Single-Node* across different robust GNN types.

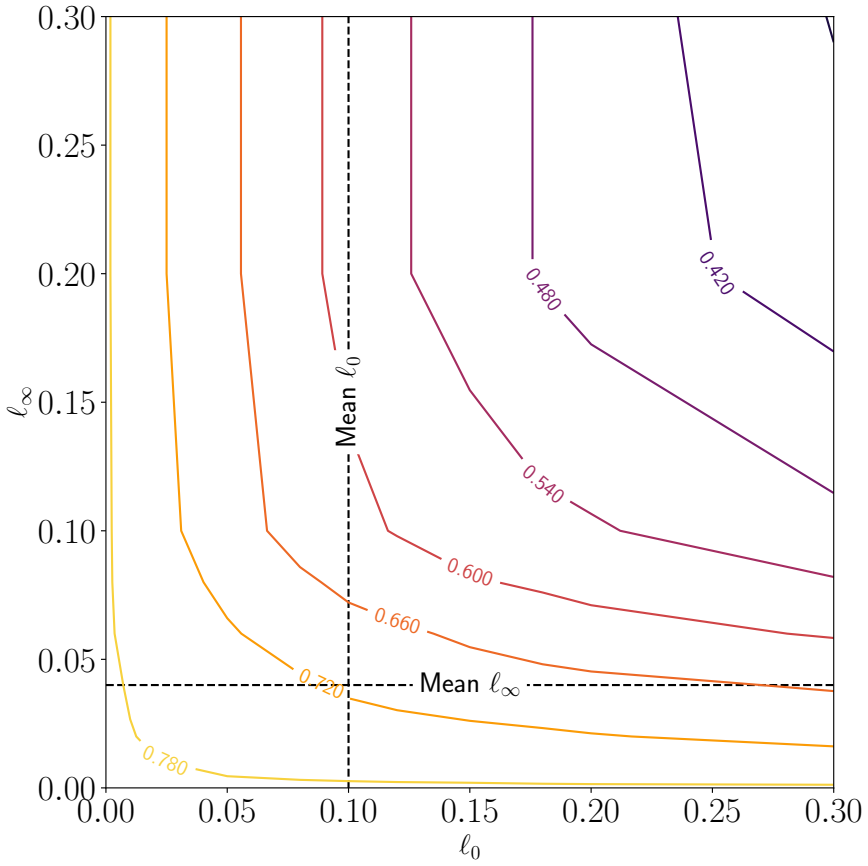


Figure C.1: Constant accuracy lines for ϵ_0 and ϵ_∞, interpolated based on grid with $\epsilon_0 \in \{0, 0.01, 0.02, 0.05, 0.08, 0.1, 0.12, 0.15, 0.18, 0.2, 0.3\}$ and $\epsilon_\infty \in \{0, 0.01, 0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.3\}$.

Appendix C.1. The Trade-Off Between ϵ_0 and ϵ_∞

Fig. C.1 shows a contour plot of accuracy as a function of ϵ_∞ and ϵ_0. We attacked the PubMed dataset with out *Single-Node* with $\epsilon_0 \in \{0, 0.01, 0.02, 0.05, 0.08, 0.1, 0.12, 0.15, 0.18, 0.2, 0.3\}$ and $\epsilon_\infty \in \{0, 0.01, 0.02, 0.04, 0.06, 0.08, 0.1, 0.15, 0.2, 0.3\}$. These contours detail the trade-off between ϵ_∞ and ϵ_0 values for a constant desired accuracy.

Appendix D. Additional Baselines

Appendix D.1. Zero-Features Approach

We experimented with an additional baseline where we set $\eta = -x_a$ as the feature perturbation. The objective of experimenting with such an attack is to

illustrate that *Single-Node* can find better perturbations than simply canceling the node feature vector, making the new vector a vector of zeros.

	PubMed
Clean	78.5 ± 0.6
Single-Node	74.3 ± 0.3
Zero features	77.8 ± 0.1

Table D.13: Test accuracy of our zero features attack on a GCN network.

As shown, *zero features* is barely effective (compared to "Clean"), and *Single-Node* can find much better perturbations.

Appendix D.2. Injection attacks

We also studied an additional type of a realistic attack that is based on node injection. In this approach, we inserted a new node into the graph with a single edge attached to our victim node. The attack was performed by perturbing the injected node's attributes. This attack is very powerful, reducing the test accuracy to $10.1 \pm 0.9\%$ on PubMed.

Appendix E. Distance Between Attacker and Victim

In Section 5.1, we found that *Single-Node* performs similarly to *Single-Node-hops*, although *Single-Node-hops* samples attacker node a whose distance from victim node v is at least 2. We further question whether the effectiveness of the attack depends on the distance between the attacker and the victim. We trained a new model for each dataset using $L = 8$ layers. Then, for each test victim node, we sampled attackers according to their distance to the test node.

As shown in Fig. E.2, the effectiveness of the attack *increases* as the distance between the attacker and the victim *decreases*. At distance of 4, the test accuracy saturates. A possible explanation is that apparently more than a few layers (e.g., $L = 2$ in Kipf & Welling (2017)) are not needed in most datasets. Thus, the rest of the layers can theoretically learn *not* to pass much of their input, starting from the redundant layers, excluding adversarial signals as well.

Appendix F. Larger number of attackers

We performed additional experiments with up to five randomly sampled attacker nodes simultaneously and perturb their features using the same approach as *Single-Node* (Table F.14). As expected, allowing a larger number of attackers reduces the test accuracy. The main observation in this paper, however, is that even a single attacker node is surprisingly effective.

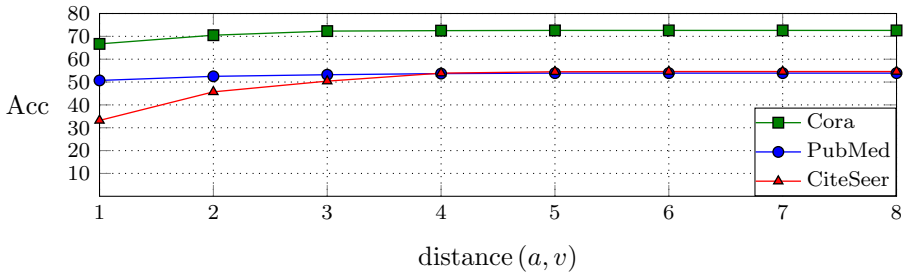


Figure E.2: Test accuracy compared to the distance between the attacker and the victim, when trained with $L = 8$ layers.

Number of attackers	**PubMed**
1	74.3 ± 0.3
2	69.9 ± 0.3
3	66.9 ± 0.4
4	63.2 ± 0.9
5	59.7 ± 0.9

Table F.14: Test accuracy for *Single-Node* with a different number of attackers on PubMed.

Appendix G. *Multi-Edge* attacks

We strengthened our *Single-Edge* attack by allowing it to add and remove multiple edges that are connected to the attacker node, thereby creating *Multi-Edge*. Accordingly, *Multi-Edge+GradChoice* adds and removes multiple edges from the entire graph.

	PubMed
Clean	78.5 ± 0.6
Single-Edge	65.1 ± 1.3
Multi-Edge	64.5 ± 0.2
Single-Edge+GradChoice	15.3 ± 0.4
Multi-Edge+GradChoice	15.3 ± 0.5

Table G.15: Test accuracy of GCN using Multi-Edge attacks

As shown in Table G.15, allowing the attacker node to add and remove multiple edges (*Multi-Edge* and *Multi-Edge+GradChoice*) results in a very minor improvement compared to *Single-Edge* and *Single-Edge+GradChoice*.