Adversarial Attacks on Neural Networks for Graph Data*

Daniel Zügner†, **Amir Akbarnejad** and **Stephan Günnemann**
Technical University of Munich
{zuegnerd, akbarnej, guennemann}@in.tum.de

Abstract

Deep learning models for graphs have achieved strong performance for the task of node classification. Despite their proliferation, currently there is no study of their robustness to adversarial attacks. Yet, in domains where they are likely to be used, e.g. the web, adversaries are common. Can deep learning models for graphs be easily fooled? In this extended abstract we summarize the key findings and contributions of our work [Zügner and Günnemann, 2019a], in which we introduce the first study of adversarial attacks on attributed graphs, specifically focusing on models exploiting ideas of graph convolutions. In addition to attacks at test time, we tackle the more challenging class of poisoning/causative attacks, which focus on the training phase of a machine learning model. We generate adversarial perturbations targeting the *node's features* and the *graph structure*, thus, taking the dependencies between instances in account. Moreover, we ensure that the perturbations remain *unnoticeable* by preserving important data characteristics. To cope with the underlying discrete domain we propose an efficient algorithm NETTACK exploiting incremental computations. Our experimental study shows that accuracy of node classification significantly drops even when performing only few perturbations. Even more, our attacks are transferable: the learned attacks generalize to other state-of-the-art node classification models and unsupervised approaches, and likewise are successful given only limited knowledge about the graph.

1 Introduction

Graph data is the core of many high impact applications ranging from the analysis of social and rating networks (Facebook, Amazon), over gene interaction networks (BioGRID), to interlinked document collections (PubMed, Arxiv). One of the most frequently applied tasks on graph data is *node classification*: given a single large (attributed) graph and the class labels of a few nodes, the goal is to predict the labels of the remaining nodes. For example, one might wish to classify the role of a protein in a biological interaction graph [Hamilton *et al.*, 2017], predict the customer type of users in e-commerce networks [Eswaran *et al.*, 2017], or assign scientific papers from a citation network into topics [Kipf and Welling, 2017].

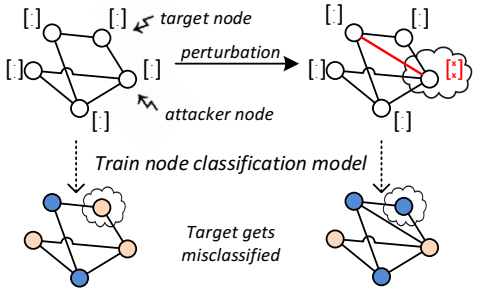


Figure 1: Adversarial attacks on graph neural networks.

While many classical approaches have been introduced in the past to tackle the node classification problem [London and Getoor, 2014; Chapelle *et al.*, 2006], the last years have seen a tremendous interest in methods for *deep learning on graphs* [Bojchevski and Günnemann, 2018; Monti *et al.*, 2017; Cai *et al.*, 2018; Klicpera *et al.*, 2019]. Specifically, approaches from the class of graph convolutional networks [Kipf and Welling, 2017; Pham *et al.*, 2017] have achieved strong performance in many graph-learning tasks.

The strength of these methods — beyond their non-linear, hierarchical nature – relies on their use of the graphs' relational information to perform classification: instead of only considering the instances individually (nodes and their features), the relationships between them are exploited as well (the edges). Put differently: the instances are not treated independently; we deal with a certain form of non-i.i.d. data where so-called network effects such as homophily (that is, nodes tend to connect to nodes similar to themselves) [London and Getoor, 2014] support the classification.

However, there is one big catch: Many researchers have noticed that deep learning architectures for classical learning tasks can easily be fooled/attacked [Szegedy *et al.*, 2014; Goodfellow *et al.*, 2015] . Even only slight, deliberate per-

*Extended abstract of the Best Research Paper of the 24th ACM International Conference on Knowledge Discovery & Data Mining.
†Contact Author

turbations of an instance – also known as *adversarial perturbations/examples* – can lead to wrong predictions. Such negative results significantly hinder the applicability of these models, leading to unintuitive and unreliable results, and they additionally open the door for attackers that can exploit these vulnerabilities. Before our study [Zügner *et al.*, 2018], however, the question of adversarial perturbations for deep learning methods on graphs had not been addressed. This is highly critical, since especially in domains where graph-based learning is used (e.g. the web) adversaries are common and false data is *easy to inject*: spammers add wrong information to social networks; fraudsters frequently manipulate online reviews and product websites [Hooi *et al.*, 2016].

In our work, we close this gap and we investigate whether such manipulations are possible. *Can deep learning models for attributed graphs be easily fooled? How reliable are their results?* This extended abstract summarizes the key findings and contributions of our work [Zügner *et al.*, 2018], where the interested reader can find more details and proofs to our theorems. Since then, there has been increasing interest in adversarial attacks [Zügner and Günnemann, 2019a; Bojchevski and Günnemann, 2019] and the first work about defenses [Zügner and Günnemann, 2019b].

The answer to this question is indeed not foreseeable: On one hand the relational effects might improve robustness since predictions are not based on individual instances only but based on various instances jointly. On the other hand, the propagation of information might also lead to cascading effects, where manipulating a single instance affects many others. Indeed, compared to the existing works on adversarial attacks, our work significantly differs in various aspects.

Opportunities. (1) Since we are operating on an attributed graph, adversarial perturbations can manifest in two different ways: by changing the nodes' features or the graph structure. Manipulating the graph, i.e. the dependency structure between instances, has not been studied so far, but is a highly likely scenario in real-life. For example, one might add or remove (fake) friendship relations to a social network. (2) While existing works were limited to manipulating an instance itself to enforce its wrong prediction[1], the relational effects give us more power: by manipulating one instance, we might specifically misguide the prediction for another one. Again, this scenario is highly realistic. Think about a fraudster who hijacks some accounts, which he then manipulates to enforce a wrong prediction for *another* account he has *not* under control. Thus, in graph-based learning scenarios we can distinguish between (i) nodes which we aim to misclassify, called *targets*, and (ii) nodes which we can directly manipulate, called *attackers*. Figure 1 illustrates the goal of our work and shows the result of our method on the Citeseer network. Clearly, compared to classical attacks to learning models, graphs enable much richer potential for perturbations. But likewise, constructing them is far more challenging.

[1]Due to the independence assumption, a misclassification for instance i can only be achieved by manipulating instance i itself for the commonly studied evasion (test-time) attacks. For the less studied poisioning attacks we might have indirect influence.

Challenges. (1) Unlike, e.g., images consisting of continuous features, the graph structure – and often also the nodes' features – is discrete. Therefore, gradient based approaches [Goodfellow *et al.*, 2015; Mei and Zhu, 2015] for finding perturbations are not suited. How to design efficient algorithms that are able to find adversarial examples in a discrete domain? (2) Adversarial perturbations are aimed to be unnoticeable (by humans). For images, one often enforces, e.g., a maximum deviation per pixel value. How can we capture the notion of 'unnoticeable changes' in a (binary, attributed) graph? (3) Last, node classification is usually performed in a *transductive* learning setting. Here, the train and test data are used jointly to learn a new classification model before the predictions are performed on the specific test data. This means, that the predominantly performed evasion attacks – where the parameters of the classification model are assumed to be static – are not realistic. The model has to be (re)trained on the manipulated data. Thus, graph-based learning in a transductive setting is inherently related to the challenging poisoning/causative attacks [Biggio *et al.*, 2014].

Given these challenges, we propose a principle for adversarial perturbations of attributed graphs that aim to fool state-of-the art deep learning models for graphs. In particular, we focus on semi-supervised classification models based on graph convolutions such as GCN [Kipf and Welling, 2017] and Column Network (CLN) [Pham *et al.*, 2017] – but we will also showcase our methods' potential on the unsupervised model DeepWalk [Perozzi *et al.*, 2014]. By default, we assume an attacker with knowledge about the full data, which can, however, only manipulate parts of it. This assumption ensures reliable vulnerability analysis in the worst case. But even when only parts of the data are known, our attacks are still successful as shown by our experiments.

2 Preliminaries

We consider the task of (semi-supervised) node classification in a single large graph having binary node features. Formally, let $G = (\boldsymbol{A}, \boldsymbol{X})$ be an attributed graph, where $\boldsymbol{A} \in \{0,1\}^{N \times N}$ is the adjacency matrix representing the connections and $\boldsymbol{X} \in \{0,1\}^{N \times D}$ represents the nodes' features. We denote with $\boldsymbol{x}_v \in \{0,1\}^D$ the D-dim. feature vector of node v. W.l.o.g. we assume the node-ids to be $\mathcal{V} = \{1, \ldots, N\}$ and the feature-ids to be $\mathcal{F} = \{1, ..., D\}$.

Given a subset $\mathcal{V}_L \subseteq \mathcal{V}$ of labeled nodes, with class labels from $\mathcal{C} = \{1, 2, \ldots, c_K\}$, the goal of node classification is to learn a function $g : \mathcal{V} \rightarrow \mathcal{C}$ which maps each node $v \in \mathcal{V}$ to one class in $\mathcal{C}$. Since the predictions are done for the *given* test instances, which are already known before (and also used during) training, this corresponds to a typical *transductive* learning scenario [Chapelle *et al.*, 2006].

In this work, we focus on node classification employing graph convolution layers. In particular, we will consider the well established work [Kipf and Welling, 2017]. Here, the hidden layer $l+1$ is defined as

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right), \qquad (1)$$

where $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$ is the adjacency matrix of the (undirected) input graph G after adding self-loops via the identity

matrix $\boldsymbol{I}_N$. $\boldsymbol{W}^{(l)}$ is the trainable weight matrix of layer l, $\widetilde{\boldsymbol{D}}_{ii} = \sum_j \widetilde{\boldsymbol{A}}_{ij}$, and $\sigma(\cdot)$ is an activation function (usually ReLU). In the first layer we have $\boldsymbol{H}^{(0)} = \boldsymbol{X}$, i.e. using the nodes' features as input. Since the latent representations $\boldsymbol{H}$ are (recursively) relying on the neighboring ones (multiplication with $\widetilde{\boldsymbol{A}}$), all instances are coupled together. Following the authors of [Kipf and Welling, 2017], we consider GCNs with a single hidden layer:

$$f_\theta(\boldsymbol{A}, \boldsymbol{X}) = \operatorname{softmax}\left(\hat{\boldsymbol{A}}\, \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}^{(1)}\right)\, \boldsymbol{W}^{(2)}\right), \quad (2)$$

where $\hat{\boldsymbol{A}} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$. Here, we use θ to denote the set of all parameters, i.e. $\theta = \{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}\}$. The parameters θ are then learned in a semi-supervised fashion by minimizing cross-entropy on the output of the labeled samples $\mathcal{V}_L$.

3 Attack Model

In this section we provide an outline of our algorithm – readers are invited to read [Zügner *et al.*, 2018] for more details.

Given the node classification setting as described in Sec. 2, our goal is to perform small perturbations on the graph $G^{(0)} = (\boldsymbol{A}^{(0)}, \boldsymbol{X}^{(0)})$, leading to the graph $G' = (\boldsymbol{A}', \boldsymbol{X}')$, such that the classification performance drops when training a classifier on the modified graph. Specifically, our goal is to attack a specific target node $v_t \in \mathcal{V}$, i.e. we aim to change v_t's prediction. This problem can be described by the following bilevel problem:

$$\max_{G' \in \Phi(G^{(0)})} \mathcal{L}_{v_t}(f_{\theta^*}(\boldsymbol{A}', \boldsymbol{X}')) \quad (3)$$
$$\text{subject to } \theta^* = \arg\max_\theta \mathcal{L}(f_\theta(\boldsymbol{A}', \boldsymbol{X}')).$$

Here, $\arg\max$ denotes the model training, e.g. via gradient descent and $\Phi(G^{(0)})$ denotes the set of admissible perturbations of the input graph (which we will describe shortly). $\mathcal{L}_{v_t}$ is the loss (typically cross entropy) of the classification of node v_t. Solving this problem exactly is intractable for multiple reasons. One of our main contributions is that we propose an alternative problem formulation that is *tractable*.

First, we propose a surrogate model which has analytic properties that we can exploit. For this we remove all non-linearities from the two-layer GCN in Eq. (2) to get

$$\hat{\boldsymbol{Z}} = \hat{f}_\theta(\boldsymbol{A}, \boldsymbol{X}) = \hat{\boldsymbol{A}}\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}^{(1)}\boldsymbol{W}^{(2)} = \hat{\boldsymbol{A}}^2\boldsymbol{X}\boldsymbol{W}, \quad (4)$$

where $\boldsymbol{W} = \boldsymbol{W}^{(1)}\boldsymbol{W}^{(2)}$. We propose the following adversarial attack formulation:

$$\max_{G' \in \Phi(G^{(0)})} \hat{\mathcal{L}}_{v_t}(\hat{f}_{\theta^*}(\boldsymbol{A}', \boldsymbol{X}')) \quad (5)$$
$$\text{subject to } \theta^* = \arg\max_\theta \mathcal{L}(f_\theta(\boldsymbol{A}, \boldsymbol{X})),$$
$$\text{where } \hat{\mathcal{L}}_{v_t}(\hat{\boldsymbol{Z}}) = \hat{\boldsymbol{Z}}_{tc'} - \hat{\boldsymbol{Z}}_{tc^*}.$$

Note here that the loss $\hat{\mathcal{L}}_{v_t}$ in the first line is the difference of the log probabilities of some class c' and the *true* class c^* output by the surrogate model. If this difference is positive, this corresponds to the model predicting a *wrong class* for the target node v_t. That is, we aim to find a perturbed graph G' that classifies v_t as c' and has maximal 'distance' (in terms of log-probabilities/logits) to c^*. Moreover, in our formulation the optimal parameters θ^* are obtained by training the surrogate model on the original (not corrupted) graph $G^{(0)}$ and not the modified graph G' as in Eq. (3). This leads to the problem being easier to optimize while, as shown by our experiments, still leading to highly destructive adversarial examples for graph neural networks. Note that for evaluation of the attacks, i.e. measuring their impact on classification performance, we follow the poisoning procedure. That is, we train the graph neural networks on graphs that were modified by our algorithm. See [Zügner and Günnemann, 2019a] for an alternative approach that explicitly operates on the bilevel problem via meta gradients.

While exactly solving Eq. (5) is still intractable because of its combinatorial nature (recall that the graph structure is discrete), we can compute the change in the loss $\hat{\mathcal{L}}_{v_t}$ given a perturbation in closed form and constant time. We make use of this fact in our greedy adversarial attack algorithm, which works as follows. At each iteration, we compute the set of admissible structure and feature perturbations; among these, our algorithm selects the perturbation which maximizes the resulting loss of the surrogate model that we would get if we performed the perturbation.

So far we have not specified the set of admissible perturbations $\Phi(G^{(0)})$. The next two sections briefly outline how we constrain the set of nodes to which the attacker can insert or remove edges or features, as well as our *unnoticeability* constraint that ensures that the perturbations remain subtle.

3.1 Target vs Attackers

Recall that our goal is to attack a specific target node $v_t \in \mathcal{V}$, i.e. we aim to change v_t's prediction. Due to the non-i.i.d. nature of the data, v_t's outcome not only depends on the node itself, but also on the other nodes in the graph. Thus, we are not limited to perturbing v_t but we can achieve our aim by also changing other nodes. Indeed, this reflects real world scenarios since it is likely that an attacker has access to a few nodes only, and not to the entire data or v_t itself. Therefore, besides the target node, we introduce the attacker nodes $\mathcal{A} \subseteq \mathcal{V}$. The perturbations on $G^{(0)}$ are constrained to these nodes. If the target $v_t \notin \mathcal{A}$, we call the attack an **influencer attack**, since v_t gets not manipulated directly, but only indirectly via some influencers. If $\{v_t\} = \mathcal{A}$, we call it a **direct attack**. To ensure that the attacker cannot modify the graph completely, we limit the number of allowed changes by a budget Δ.

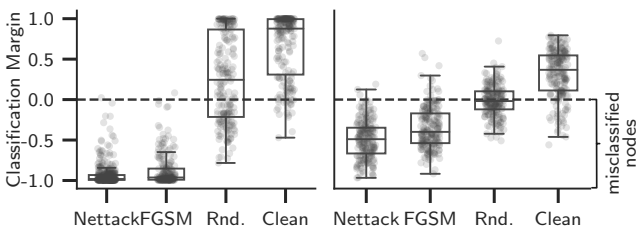
3.2 Unnoticeable Perturbations

Typically, in an adversarial attack scenario, the attacker tries to modify the input data such that the changes are *unnoticeable*. Unlike to image data, where this can easily be verified visually and by using simple constraints, in the graph setting this is harder mainly for two reasons: (i) the graph structure is discrete preventing to use infinitesimal small changes, and (ii) large graphs are not suitable for visual inspection.

How can we ensure unnoticeable perturbations in our setting? In particular, we argue that only considering the budget Δ might not be enough. Especially if a large Δ is required

	CORA-ML			CITESEER			POLBLOGS		
	GCN	CLN	DeepWalk	GCN	CLN	DW	GCN	CLN	DW
Clean	0.90	0.82	0.84	0.88	0.71	0.76	0.94	0.82	0.93
NETTACK	**0.01**	**0.16**	**0.02**	**0.02**	**0.20**	**0.01**	**0.06**	**0.46**	**0.06**
FGSM	0.03	0.18	0.10	0.07	0.23	0.05	0.41	0.54	0.37
RND	0.61	0.52	0.46	0.60	0.52	0.38	0.36	0.54	0.30
NETTACK-IN	0.64	0.67	0.65	0.62	0.56	0.48	0.86	0.62	0.91

Table 1: Classification accuracy when training on datasets poisoned by different methods ($\Delta = d_t + 2$). Lower is better.



(a) Poisoning of GCN (b) Poisoning of DeepWalk

Figure 2: Results on CORA-ML using different attack algorithms ($\Delta = d_t + 2$). Clean indicates the original data. Lower is better.

due to complicated data, we still want realistically looking perturbed graphs G'. Therefore, our core idea is to allow only those perturbations that preserve specific inherent properties of the input graph.

Graph Structure Preserving Perturbations

Undoubtedly, the most prominent characteristic of the graph structure is its degree distribution, which often resembles a power-law like shape in real networks. If two networks show very different degree distributions, it is easy to tell them apart.

For this purpose we refer to a statistical two-sample test for power-law distributions [Bessi, 2015]. That is, we estimate whether the two degree distributions of $G^{(0)}$ and G' stem from the same or from individual distributions using a likelihood ratio test. A perturbation is only admissible if, after the perturbation, the likelihood ratio cutoff is not violated.

Feature Statistics Preserving Perturbations

While the above principle could be applied to the nodes' features as well (e.g. preserving the distribution of feature occurrences), we argue that such a procedure is too limited. In particular, such a test does not reflect the correlation/co-occurrence of different features: If two features have never occurred together in $G^{(0)}$, but they do in G', the distribution of feature occurrences would still be very similar. Such a change, however, is easily noticeable – think about two words which have never been used together but are suddenly used in G'. Thus, we refer to a test based on feature co-occurrence.

4 Results

We evaluate the effectiveness of our adversarial attack algorithm on three well-known datasets: CORA-ML [McCallum *et al.*, 2000], CITESEER [Sen *et al.*, 2008], and POLBLOGS [Adamic and Glance, 2005]. On each of the datasets we first train our surrogate model once; among the test nodes that are correctly classified we select (i) the 10 nodes with highest confidence (ii) the 10 nodes with lowest confidence, (iii) 20 nodes sampled uniformly, which serve as the target nodes for the attacks. We run all our experiments on five different random splits of 20% labeled and 80% unlabeled nodes. The budget Δ is set to $d_t + 2$, i.e. the degree of the respective target node plus two. For each target node and each random split we train GCN [Kipf and Welling, 2017], Column Network (CLN) [Pham *et al.*, 2017], and DeepWalk (DW) [Perozzi *et al.*, 2014] on the graph modified by an adversarial attack for ten times, and measure how often the target node is correctly classified. If after the attacks, on average, the target nodes are misclassified more often than when training on the original data, we conclude that the attack was successful.

	[1;5]	[6;10]	[11;20]	[21;100]	[100; ∞)
Clean	0.878	0.823	1.0	1.0	1.0
NETTACK	0.003	0.009	0.014	0.036	0.05

Table 2: GCN accuracy after attack using NETTACK, by node degree

As baselines we have the performance on the original data ('Clean'), a random baseline which inserts edges from the target node to nodes from different classes ('Rnd.'), and a gradient method ('FGSM') where the attacker uses the gradient of the trained model w.r.t. the adjacency matrix for attacks.

In Table 1 we can see that NETTACK leads to a dramatic reduction in classification performance and outperforms all baselines by a large margin. In Figures 2a and 2b we show how the classification margins of the individual nodes are reduced for the different attack strategies. Again, NETTACK has the strongest impact. We further want to highlight that NETTACK-IN, the variant in which the attacker does not have direct access to the target node, also leads to a consistent decrease in classification performance across datasets. Table 2 shows that high degree nodes are harder to attack than low-degree nodes: they have higher classification accuracy both in the clean graph and in the attacked graph. We refer interested readers to [Zügner *et al.*, 2018] for more experiments, where we e.g. show that even when NETTACK can only observe a small fraction of the network, its attacks are effective.

5 Conclusion

We present the first work on adversarial attacks on graph neural networks. Our attacks target the nodes' features and the graph structure. Exploiting the relational nature of the data, we propose direct and influencer attacks. To ensure unnoticeable changes even in a discrete, relational domain, we propose to preserve the graph's degree distribution and feature co-occurrences. Our algorithm enables efficient perturbations on a discrete domain. Based on our extensive experiments we conclude that our method is successful even on the challenging class of poisoning attacks. The classification performance is consistently reduced, even when only partial knowledge of the graph is available or the attack is restricted to a few influencers. Even more, the attacks generalize to other node classification models. Studying the robustness of deep learning models for graphs is an important problem, and this work provides essential insights for deeper study. As future work we aim to derive extensions of existing models to become more robust, and to study tasks beyond node classification.

References

[Adamic and Glance, 2005] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 US election: divided they blog. In *International workshop on Link discovery*, pages 36–43, 2005.

[Bessi, 2015] Alessandro Bessi. Two samples test for discrete power-law distributions. *arXiv preprint arXiv:1503.00643*, 2015.

[Biggio *et al.*, 2014] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE TKDE*, 26(4):984–996, 2014.

[Bojchevski and Günnemann, 2018] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR*, 2018.

[Bojchevski and Günnemann, 2019] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 695–704, 2019.

[Cai *et al.*, 2018] H. Cai, V. W. Zheng, and K. C. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, Sep. 2018.

[Chapelle *et al.*, 2006] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning. Adaptive Computation and Machine Learning series*. The MIT Press, 2006.

[Eswaran *et al.*, 2017] Dhivya Eswaran, Stephan Günnemann, Christos Faloutsos, Disha Makhija, and Mohit Kumar. Zoobp: Belief propagation for heterogeneous networks. *PVLDB*, 10(5):625–636, 2017.

[Goodfellow *et al.*, 2015] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[Hamilton *et al.*, 2017] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[Hooi *et al.*, 2016] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. BIRDNEST: bayesian inference for ratings-fraud detection. In *SIAM SDM*, pages 495–503, 2016.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Klicpera *et al.*, 2019] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2019.

[London and Getoor, 2014] Ben London and Lise Getoor. Collective classification of network data. *Data Classification: Algorithms and Applications*, 399, 2014.

[McCallum *et al.*, 2000] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[Mei and Zhu, 2015] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, pages 2871–2877, 2015.

[Monti *et al.*, 2017] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5425–5434, 2017.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710, 2014.

[Pham *et al.*, 2017] Trang Pham, Truyen Tran, Dinh Q. Phung, and Svetha Venkatesh. Column networks for collective classification. In *AAAI*, pages 2485–2491, 2017.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Google Inc, Joan Bruna, Dumitru Erhan, Google Inc, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[Zügner and Günnemann, 2019a] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[Zügner and Günnemann, 2019b] Daniel Zügner and Stephan Günnemann. Certifiable robustness and robust training for graph convolutional networks. In *SIGKDD*, 2019.

[Zügner *et al.*, 2018] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *SIGKDD*, pages 2847–2856, 2018.