Chapter 8
Graph Neural Networks: Adversarial Robustness

Stephan Günnemann

Abstract Graph neural networks have achieved impressive results in various graph learning tasks and they have found their way into many applications such as molecular property prediction, cancer classification, fraud detection, or knowledge graph reasoning. With the increasing number of GNN models deployed in scientific applications, safety-critical environments, or decision-making contexts involving humans, it is crucial to ensure their reliability. In this chapter, we provide an overview of the current research on adversarial robustness of GNNs. We introduce the unique challenges and opportunities that come along with the graph setting and give an overview of works showing the limitations of classic GNNs via adversarial example generation. Building upon these insights we introduce and categorize methods that provide provable robustness guarantees for graph neural networks as well as principles for improving robustness of GNNs. We conclude with a discussion of proper evaluation practices taking robustness into account.

8.1 Motivation

The success story of graph neural networks is astonishing. Within a few years, they have become a core component of many deep learning applications. Nowadays they are used in scientific applications such as drug design or medical diagnoses, are integrated in human-centered applications like fake news detection in social media, get applied in decision-making tasks, and even are studied in safety-critical environments like autonomous driving. What unites these domains is their crucial need for reliable results; misleading predictions are not only unfortunate but indeed might lead to dramatic consequences – from false conclusions drawn in science to harm for people. However, can we really trust the predictions resulting from graph neural

Stephan Günnemann
Department of Informatics, Technical University of Munich, e-mail: guennemann@in.tum.de

networks? What happens when the underlying data is corrupted or even becomes deliberately manipulated?

Indeed, the vulnerability of classic machine learning models to (deliberate) perturbations of the data is well known (Goodfellow et al, 2015): even only slight changes of the input can lead to wrong predictions. Such instances, for humans nearly indistinguishable from the original input yet wrongly classified, are also known as *adversarial examples*. One of the most well-known and alarming examples is an image of a stop sign, which is classified as a speed limit sign by a neural network with only very subtle changes to the input; though, for us as humans it still clearly looks like a stop sign (Eykholt et al, 2018). Examples like these illustrate how machine learning models can dramatically fail in the presence of adversarial perturbations. Consequently, adopting machine learning for safety-critical or scientific application domains is still problematic. To address this shortcoming, many researchers have started to analyze the robustness of models in domains like images, natural language, or speech. Only recently, however, GNNs have come into focus. Here, the first work studying GNNs' robustness (Zügner et al, 2018) investigates one of the most prominent tasks, node-level classification, and demonstrated the susceptibility of GNNs to adversarial perturbations as well (see Figure 8.1). Since then, the field of adversarial robustness on graphs has been rapidly expanding, with many works studying diverse tasks and models, and exploring ways to make GNNs more robust.

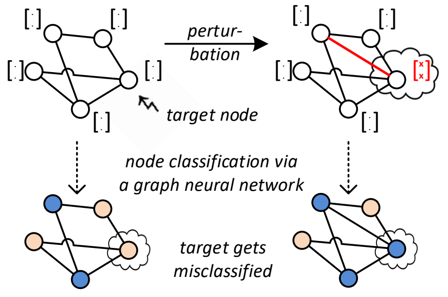

Fig. 8.1 The upper left graph is the original input. On the right is the graph after performing a small change (e.g. adding an edge or changing some node attributes). The lower part illustrates the predicted classes for each node obtained from a GNN. Is it possible to change the predictions? Are GNNs robust?

To some degree it is surprising that graphs were not in the focus even earlier. Corrupted data and adversaries are common in many domains where graphs are analyzed, e.g., social media and e-commerce systems. Take for example a GNN-based model for detecting fake news in a social network (Monti et al, 2019; Shu et al, 2020). Adversaries have a strong incentive to fool the system in order to avoid being detected. Similarly, in credit scoring systems, fraudsters try to disguise themselves by creating fake connections. Thus, robustness is an important concern for graph-based learning.

It is important to highlight, though, that adversarial robustness is not only a topic in light of security concerns, where intentional changes, potentially crafted by humans, are used to try to fool the predictions. Instead, adversarial robustness considers worst-case scenarios in general. Especially in safety-critical or scientific applications where reliability is key, understanding the robustness of GNNs to worst-case noise is important, as nature itself might be the adversary. The construction of gene interaction networks, for example, often leads to corrupted graphs containing spurious edges (Tian et al, 2017). Thus, to make sure that graph neural networks work reliably in all these scenarios, we need to investigate robustness under *worst-case/adversarial* corruptions of the data.

Moreover, non-robustness of GNNs shows conceptual gaps: while neural networks are hypothesized to learn meaningful representations that capture the semantics of the domain and task, a non-robust model clearly violates this property. Since the small changes leading to an adversarial example do not alter the meaning, a reasonable representation should also not change the prediction. Thus, understanding adversarial robustness means understanding generalization performance.

Unique Challenges in the Graph Domain

In contrast to other application domains of deep learning, robustness analysis for graphs is especially challenging for multiple reasons:

1. Complex perturbation space: Changes can manifest in various ways including perturbations in the graph structure and the node attributes, leading to a vast space to explore. Importantly, unlike other fields this often means operating in a *discrete data domain* such as adding or removing edges, leading to hard discrete optimization problems as we will see later.
2. Interdependent data: The core feature of GNNs is to exploit the interdependence between instances, for example, in the form of message passing or graph convolution. Perturbations to the graph structure change the message passing scheme, modifying how learned representation are propagated. Specifically, changes to one part of the graph, e.g. one node, might affect many other instances.
3. Notion of similarity: We expect GNN models to be robust to small changes in the graph. If the graphs are almost indistinguishable, the predictions should be the same. However, defining the notion of similarity between graphs itself is a hard problem and unlike, e.g., images, manual inspection by a human is not practical.

Given these challenges, in the following Section 8.2 we first introduce the principle of adversarial attacks on GNNs and highlight some non-robustness results. In Section 8.3, we give an overview of robustness certificates, providing ways for proving the reliability of predictions, followed by Section 8.4 where approaches for improving GNNs' robustness are introduced. We conclude in Section 8.5 with discussing aspects of proper evaluation.

8.2 Limitations of Graph Neural Networks: Adversarial Examples

To understand the (non-)robustness of GNNs, we can try to construct worst-case perturbations – finding a *small* change of the data, which in consequence leads to a *strong* change in the GNN's output. This is also known as performing an adversarial attack and the resulting perturbed data is often called an adversarial example.[1] While random perturbations of the data often have minor effect, specific perturbations, in contrast, can be dramatic. Accordingly, an attack is often phrased as an optimization problem with the goal to *find* a perturbation of the data which maximizes some attack objective (e.g., maximize the predicted probability of some incorrect class).

8.2.1 *Categorization of Adversarial Attacks*

Before providing a general definition of adversarial attacks, it is helpful to distinguish two very different notions, called *poisoning vs. evasion scenarios*. The difference lies in the stage of the learning process in which the data perturbation is performed. In a poisoning scenario, the perturbation is injected *before* the training of the model; the perturbed data, thus, also affects the learning and the final model we obtain. In contrast, an evasion scenario assumes the model to be given, i.e., already trained and fixed, and the perturbation is applied to *future* data during the application/test phase of the GNN. It is worth to highlight, that for the frequently considered *transductive learning setting* of GNNs – where we have *no* future test data, but only the given (un)labeled data – a poisoning scenario is the more natural choice. Though, in principle any combination of learning (transductive vs. inductive) and attack scenario (poisoning vs. evasion) is worth to be studied.

Given this basic distinction, performing a poisoning adversarial attack can be generally formulated as a bi-level optimization problem

$$\max_{\hat{\mathcal{G}}\in\Phi(\mathcal{G})} \quad \mathcal{O}_{\text{atk}}(f_{\theta^*}(\hat{\mathcal{G}})) \quad s.t. \quad \theta^* = \arg\min_{\theta} \quad \mathcal{L}_{\text{train}}(f_{\theta}(\hat{\mathcal{G}})) \tag{8.1}$$

Here $\Phi(\mathcal{G})$ denotes the set of all graphs we are treating as indistinguishable to the given graph $\mathcal{G}$ at hand, and $\hat{\mathcal{G}}$ denotes a specific perturbed graph from this set. For example, $\Phi(\mathcal{G})$ could capture all graphs which differ from $\mathcal{G}$ in at most ten edges or in a few node attributes. The attacker's goal is to find a graph $\hat{\mathcal{G}}$ that, when passed through the GNN f_{θ^*}, maximizes a specific objective $\mathcal{O}_{\text{atk}}$, e.g., increasing the predicated probability of a certain class for a specific node. Importantly, in a poisoning setting, the weights θ^* of the GNN are not fixed but learned based on the perturbed data, leading to the inner optimization problem that corresponds to the usual training procedure on the (now perturbed) graph. That is, θ^* is obtained

[1] Again it is worth highlighting that such 'attacks' are not always due to human adversaries. Thus, the terms 'change' or 'perturbation' might be better suited and have a more neutral connotation.

Poisoning & evasion

by minimizing some training loss $\mathscr{L}_{\text{train}}$ on the graph $\hat{\mathscr{G}}$. This nested optimization makes the problem specifically hard.

To define an evasion attack, the above equation can simply be changed by assuming the parameter θ^* to be fixed. Often it is assumed to be given by minimizing the training loss w.r.t. the given graph $\mathscr{G}$ (i.e. $\theta^* = \arg\min_\theta \mathscr{L}_{\text{train}}(f_\theta(\mathscr{G}))$). This makes the above scenario a single-level optimization problem.

This general form of an attack enables us to provide a categorization along different aspects and illustrates the space to explore for robustness characteristics of GNNs in general. While this taxonomy is general, for ease of understanding, it helps to think about an intentional attacker.

Aspect 1: Property under Investigation (Attacker's Goal)

What is the robustness property we want to analyze? For example, do we want to understand how robust the classification of an individual node is? Will it change when perturbing the data? The property under investigation is modeled via $\mathscr{O}_{\text{atk}}$. It intuitively represents the attacker's goal. If $\mathscr{O}_{\text{atk}}$ for example measures the difference between a node's ground truth label and the currently predicted one, maximizing this difference in Eq. equation 8.1 tries to enforce a misclassification.

The attacker's goal is highly task-dependent. The majority of existing works has focused on the robustness of node-level classification based on GNNs, where we have to distinguish two scenarios. Works such as (Zügner et al, 2018; Dai et al, 2018a; Wang and Gong, 2019; Wu et al, 2019b; Chen et al, 2020f; Wang et al, 2020c) investigate how the prediction of an *individual* target node changes under perturbations – also called *local* attack. In contrast, Zügner and Günnemann (2019); Wu et al (2019b); Liu et al (2019c); Ma et al (2020b); Geisler et al (2021); Sun et al (2020d) have investigated how the overall performance on an *entire set* of nodes can drop – called a *global* attack.[2] This seemingly subtle difference between the two scenarios is crucial: In the latter case one has to find a single perturbed graph $\hat{\mathscr{G}} \in \Phi(\mathscr{G})$ which simultaneously changes many predictions, taking into account that all node-level predictions are indeed done jointly based on one input. In the former case, for each individual target node v_i a different perturbation $\hat{\mathscr{G}}_i \in \Phi(\mathscr{G})$ can be selected. Both views are reasonable; they simply model different aspects.

Beyond node-level classification, further works have investigated robustness of graph-level classification (Chen et al, 2020j), link prediction (Chen et al, 2020h; Lin et al, 2020d), and node embeddings (Bojchevski and Günnemann, 2019; Zhang et al, 2019e). The last one is worth mentioning since it targets an unsupervised learning setting, aiming to be task-agnostic. Unlike the other examples, the goal is not to target one specific task but to perturb the quality of the embeddings in general such that one or multiple downstream tasks are hindered. Since it is not known a priori which tasks (classification, link prediction, etc.) will be performed based on the

[2] Local attacks have also been called targeted attacks, while global ones untargeted. Since this, however, leads to a name clash with categorizations used in other communities (Carlini and Wagner, 2017) we decided to use local/global here.

node embeddings, defining the objective $\mathscr{O}_{\text{atk}}$ is challenging. As a proxy measure, Bojchevski and Günnemann (2019) for example uses the training loss itself, setting $\mathscr{O}_{\text{atk}} = \mathscr{L}_{\text{train}}$.

Aspect 2: The Perturbation Space (Attacker's Capabilities)

What changes are allowed to the original graph? What do we expect the perturbations to look like? For example, do we want to understand how deleting a few edges influences the prediction? The space of perturbations under consideration is modeled via $\Phi(\mathscr{G})$. It intuitively represents the attacker's capabilities; what and how much they are able to manipulate. The complexity of the perturbation space for graphs represents one of the biggest differences to classical robustness studies and stretches along two dimensions.

(1) What can be changed? Unique to the graph domain are perturbations of the graph structure. In this regard, most publications have studied the scenarios of removing or adding edges to the graph (Dai et al, 2018a; Wang and Gong, 2019; Zügner et al, 2018; Zügner and Günnemann, 2019; Bojchevski and Günnemann, 2019; Zhang et al, 2019e; Zügner et al, 2018; Tang et al, 2020b; Chen et al, 2020f; Chang et al, 2020b; Ma et al, 2020b; Geisler et al, 2021). Focusing on the node level, some works (Wang et al, 2020c; Sun et al, 2020d; Geisler et al, 2021) have considered adding or removing nodes from the graph. Beyond the graph structure, GNN robustness has also been explored for changes to the node attributes (Zügner et al, 2018; Wu et al, 2019b; Takahashi, 2019) and the labels used in semi-supervised node classification (Zhang et al, 2020b).

An intriguing aspect of graphs is to investigate how the interdepenence of instances plays a role in robustness. Due to the message passing scheme, changes to one node might affect (potentially many) other nodes. Often, for example, a node's prediction depends on its k-hop neighborhood, intuitively representing the node's receptive field. Thus, it is not only important what type of change can be performed but also where in the graph this can happen. Consider for example Figure 8.1: to analyze whether the prediction for the highlighted node can change, we are not limited to perturbing the node's own attributes and its incident edges but we can also achieve our aim by perturbing other nodes. Indeed, this reflects real world scenarios much better since it is likely that an attacker has access to a few nodes only, and not to the entire data or the target node itself. Put simply, we also have to consider which nodes can be perturbed. Multiple works (Zügner et al, 2018; Zhang et al, 2019e; Takahashi, 2019) investigate what they call *indirect attacks* (or sometimes influencer attacks), specifically analyzing how an individual node's prediction can change when only perturbing other parts of the graph while leaving the target node untouched.

(2) How much can be changed? Typically, adversarial examples are designed to be nearly indistinguishable to the original input, e.g., changing the pixel values of an image so that it stays visually the same. Unlike image data, where this can easily be verified by manual inspection, this is much more challenging in the graph setting.

Technically, the set of perturbations can be defined based on any graph distance function D measuring the (dis)similarity between graphs. All graphs similar to the given graph $\mathscr{G}$ then define the set $\Phi(\mathscr{G}) = \{\hat{\mathscr{G}} \in \mathbb{G} \mid D(\mathscr{G}, \hat{\mathscr{G}}) \leq \Delta\}$, where $\mathbb{G}$ denotes the space of all potential graphs and Δ the largest acceptable distance.

Defining what are suitable graph distance functions is in itself a challenging task. Beyond that, computing these distances and using them within the optimization problem of Eq. equation 8.1 might be computationally intractable (think, e.g., about the graph edit distance which itself is NP-hard to compute). Therefore, existing works have mainly focused on so called budget constraints, limiting the *number of changes* allowed to be performed. Technically, such budgets correspond to the L_0 pseudo-norm between the clean and perturbed data, e.g., relating to the graphs' adjacency matrix A or its node attributes X.[3] To enable more fine-grained control, often such budget constraints are used locally per node (e.g., limiting the maximal number of edge deletions per node; Δ_i^{loc}) as well as globally (e.g., limiting the overall number of edge deletions; Δ^{glob}). For example

$$\Phi(\mathscr{G}) = \{\hat{\mathscr{G}} = (\hat{A}, \hat{X}) \in \mathbb{G} \mid ||A - \hat{A}||_0 \leq \Delta^{\text{glob}} \wedge \forall i : ||A_i - \hat{A}_i||_0 \leq \Delta_i^{\text{loc}} \wedge X = \hat{X}\}, \tag{8.2}$$

where the graphs $\mathscr{G} = (A, X)$ and $\hat{\mathscr{G}} = (\hat{A}, \hat{X})$ are assumed to have the same size and the node attributes, X resp. $\hat{X}$, to stay unchanged; A_i denotes the ith row of A.

Beyond these budget constraints, it might be useful to preserve further characteristics of the data. In particular for real-world networks many patterns such as specific degree distributions, large clustering coefficients, low diameter, and more are known to hold (Chakrabarti and Faloutsos, 2006). If two graphs show very different patterns, it is easy to tell them apart – and a different prediction could be expected. Therefore, in (Zügner et al, 2018; Zügner and Günnemann, 2019; Lin et al, 2020d) only perturbed graphs are considered which follow similar power-law behavior in the degree distribution. Similarly, one can impose constraints on the attributes considering, e.g., the co-occurrence of specific values.

Aspect 3: Available Information (Attacker's Knowledge)

What information is available to find a harmful perturbation? What is the attacker's knowledge about the system? Considering a human-like adversary, the more knowledge is available, the stronger are the potential attacks.

In general, we have to distinguish between knowledge about the data/graph and knowledge about the model. For the first, either the full graph could be known or only parts of it as, e.g., investigated in (Zügner et al, 2018; Dai et al, 2018a; Chang et al, 2020b; Ma et al, 2020b). While for worst-case analysis we often assume that the attacker has full knowledge, for practical scenarios it is indeed realistic to assume that an attacker only observes subsets of the data. For supervised learning settings,

[3] This is a similar approach to image data, where often we take a certain radius as measured by, e.g., an L_p norm around the original input as the allowed perturbation set, assuming that for small radii the semantic meaning of the input does not change.

the ground-truth labels of the target node(s) could additionally be hidden from the attacker. The knowledge about the model includes many aspects such as knowledge about the used GNN architecture, the model's weights, or whether only the output predictions or the gradients are known. Given all these variations, the most common ones are white-box settings, where full information is available, and black-box settings, which usually mean that only the graph and potentially the predicted outputs are available.

Among the three aspects above, the attacker's knowledge seems to be the one which most strongly links to human-like adversaries. It should be highlighted, though, that worst-case perturbations in general are best reflected by the fully white-box setting, making it the preferred choice for strong robustness results. If a model performs robustly in a white-box setting, it will also be robust under the limited scenarios. Moreover, as we will see in Section 8.2.2.1, the transferability of attacks implies that knowledge about the model is not really required.

Aspect 4: The Algorithmic View

Besides the above categorization that focuses on the properties of the attack, another, more technical, view can be taken by considering the algorithmic approach how the (bi-level) optimization problem is solved. In the discussion of the perturbation space we have seen that graph perturbations often relate to the addition/removal of edges or nodes — these are discrete decisions, making Eq. equation 8.1 a discrete optimization problem. This is in stark contrast to other data domains where infinitesimal changes are possible. Thus besides adapting gradient-based approximations, various other techniques can be used to tackle Eq. equation 8.1 for GNNs such as reinforcement learning (Sun et al, 2020d; Dai et al, 2018a) or spectral approximations (Bojchevski and Günnemann, 2019; Chang et al, 2020b). Moreover, the attacker's knowledge has also implications on the algorithmic choice. In a black-box setting where, e.g., only the input and output are observed, we cannot use the true GNN f_θ to compute gradients but have to use other principles like first learning some surrogate model.

8.2.2 *The Effect of Perturbations and Some Insights*

The above categorization shows that various kinds of adversarial perturbations under different scenarios can be investigated. Summarizing the different results obtained in the literature so far, the trend is clear: standard GNNs trained in the standard way are not robust. In the following, we given an overview of some key insights.

Figure 8.2 illustrates one of the results of the method Nettack as introduced in (Zügner et al, 2018). Here, *local attacks* in an evasion setting focusing on graph structure perturbations are analyzed for a GCN (Kipf and Welling, 2017b). The figure shows the classification margin, i.e., the difference between the predicted

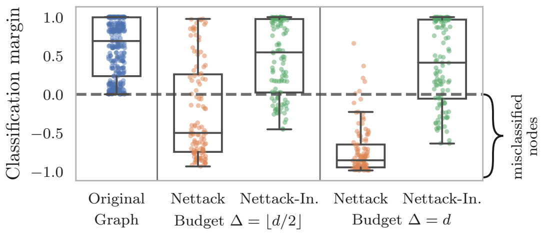

Fig. 8.2 Performing local structure attacks on a GCN model and the Cora ML data with the Nettack (Zügner et al, 2018) approach. If a node is below the dashed line it is misclassified w.r.t. the ground truth label. As shown, almost any node's prediction can be changed.

probability of the node's true class minus the one of the second highest class. The left column shows the results for the unperturbed graph where most nodes are correctly classified as illustrated by the predominantly positive classification margin. The second column shows the result after perturbing the graph based on the perturbation found by Nettack using a global budget of $\Delta = \lfloor d_v/2 \rfloor$ and making sure that no singletons occur where d_v is the degree of the node v under attack. Clearly, the GCN model is not robust: almost every node's prediction can be changed. Moreover, the third column shows the impact of indirect attacks. Recall that in these scenarios the performed perturbations cannot happen at the node we aim to misclassify. Even in this setting, a large fraction of nodes is vulnerable. The last two columns show results for an increased budget of $\Delta = d_v$. Not surprisingly, the impact of the attack becomes even more pronounced.

Considering *global attacks* in the poisoning setting similar behavior can be observed. For example, when studying the effect of node additions, the work (Sun et al, 2020d) reports a relative drop in accuracy by up to 7 percentage points with a budget of 1% of additional nodes, without changing the connectivity between existing nodes. For changes to the edge structure, the work (Zügner and Günnemann, 2019) reports performance drops on the test sets by around 6 to 16 percentage points when perturbing 5% of the edges. Noteworthy, on one dataset, these perturbations lead to a GNN obtaining *worse* performance than a logistic regression baseline operating only on the node attributes, i.e., ignoring the graph altogether becomes the better choice.

The following observation from (Zügner and Günnemann, 2019) is important to highlight: One core factor for the obtained lower performance on the perturbed graphs are indeed the learned GNN weights. When using the weights $\theta_{\hat{\mathscr{G}}}$ trained on the perturbed graph $\hat{\mathscr{G}}$ obtained by the poisoning attack, not only the performance on $\hat{\mathscr{G}}$ is low but even the performance on the unperturbed graph $\mathscr{G}$ suffers dramatically. Likewise, when applying weights $\theta_{\mathscr{G}}$ trained on the unperturbed graph $\mathscr{G}$ to the graph $\hat{\mathscr{G}}$, the classification accuracy barely changes. Thus, the poisoning attack performed in (Zügner and Günnemann, 2019) indeed derails the training procedure, i.e., leads to 'bad' weights. This result emphasizes the importance of the training procedure for the performance of graph models. If we are able to find appropriate weights, even perturbed data might be handled more robustly. We will encounter this aspect again in Section 8.4.2.

8.2.2.1 Transferability and Patterns

An interesting question to investigate is the adversarial examples' transferability. Transferability relates to the fact that a harmful perturbation for one model (e.g. a GCN) is also harmful for another model (e.g. GAT (Veličković et al, 2018)). Thus, one can simply reuse one perturbation to fool many models. The transferability of GNN attacks has been investigated in multiple works (Zügner et al, 2018; Zügner and Günnemann, 2019; Lin et al, 2020d; Chen et al, 2020f) and seems to hold across many models. For example, local attacks computed based on Nettack's GCN-like surrogate model in an evasion scenario are also harmful for the original GCN and the Column Network (Pham et al, 2017) model; for evasion and poisoning alike. Interestingly, the performance gets detoriated even for unsupervised node embeddings such as DeepWalk (Perozzi et al, 2014), combined with a subsequent logistic regression to obtain predictions.

The wide transferability of adversarial perturbations could be an indicator that they follow general patterns. There seems to be some systematic change of the graph which hinders many GNN models to perform well. If we can find out what makes, for example, an edge insertion a strong adversarial change, we can use this knowledge to detect adversarial attacks and/or make graph neural networks more robust (see Section 8.4). However, it is yet still not fully understood what makes these adversarial attacks harmful to a variety of models.

In (Zhang et al, 2019b) the predicted categorical distributions over classes for perturbed and unperturbed instances after performing a Nettack attack has been analyzed. Inspecting the average KL-divergence of the predicted categorical distributions of a node and its neighbors, perturbed nodes seem to show higher divergences, i.e., the attacks appear to be aiming to violate the homophily assumption in the graph. Relatedly, Wu et al (2019b) compared the Jaccard similarity between adjacent node's attributes and noticed a change in distribution from the clean and perturbed graph. The work (Zügner et al, 2020) investigated various graph properties, including aspects such as node degree, closeness centrality, PageRank (Brin and Page, 1998) scores, or attribute similarity. They focused on structure attacks using Nettack, allowing only edge insertions and deletions to the target node.

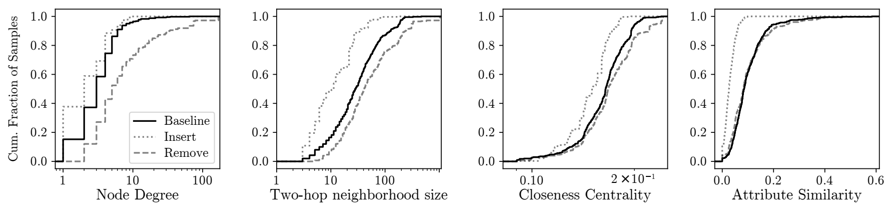

Fig. 8.3: Cumulative distributions of properties of nodes connected to (*Insert*) or disconnected from (*Remove*) the target node by the Nettack method. *Baseline* is the distribution in the entire graph.

Figure 8.3 compares the distribution of such a property (e.g. node degree) when considering all nodes of the unperturbed graph with the distribution of the property when considering only the nodes incident to the inserted/removed adversarial edges. The comparison indicates a statistically significant difference between the distributions. For example, in Figure 8.3 (left) we can see that the Nettack method tends to connect a target node to low-degree nodes. This could be due to the degree-normalization performed in GCN, where low-degree nodes have a higher weight (i.e., influence) on the aggregation of a node. Likewise, considering nodes incident to edges removed by the adversary we can observe that the Nettack method tends to disconnect high-degree nodes from the target node. In Figure 8.3 (second and third plot) we can see that the attack tends to connect the target node with peripheral nodes, as evidenced by small two-hop neighborhood size and low closeness centrality of the adversarially connected nodes. In Figure 8.3 (right) we can see that the adversary tends to connect a target node to other nodes which have dissimilar attributes. As also shown in other works, the adversary appears to try to counter the homophily property in the graph – which is not surprising, since the GNN has likely learned to partly infer a node's class based on its neighbors.

To understand whether such detected patterns are universal, they can be used to design attack principles itself — indeed, this even leads to black-box attacks since the analyzed properties usually relate to the graph only and not the GNN. In (Zügner et al, 2020) a prediction model was learned estimating the potential impact of a perturbation on unseen graphs using the above mentioned properties as input features. While this often resulted in finding effective adversarial perturbations, thus, highlighting the generality of the regularities uncovered, the attack performance was not on par with the original Nettack attack. Similarly, in (Ma et al, 2020b) PageRank-like scores have been used to identify potential harmful perturbations.

8.2.3 Discussion and Future Directions

The aspects along which adversarial attacks on graphs can be studied allow for a huge variety of scenarios. Only a few of them have been thoroughly investigated in the literature. One important aspect to consider, for example, is that in real applications the cost of perturbations differ: while changing node attributes might be relatively easy, injecting edges might be harder. Thus, designing improved perturbation spaces can make the attack scenarios more realistic and better captures the robustness properties one might want to ensure. Moreover, many different data domains such as knowledge graphs or temporal graphs need to be investigated.

Importantly, while first steps have been made to understand the patterns that makes these perturbations harmful, a clear understanding with a sound theoretical backing is still missing. In this regard, it is also worth repeating that all these studies have focused on analyzing perturbations obtained by Nettack; other attacks might potentially lead to very different patterns. This also implies that exploiting the resulting patterns to design more robust GNNs (see Section 8.4.1) is not necessarily a good solution. Moreover, finding reliable patterns also requires more research on how to compute adversarial perturbations in a scalable way (Wang and Gong, 2019; Geisler et al, 2021), since such patterns might be more pronounced on larger graphs.

8.3 Provable Robustness: Certificates for Graph Neural Networks

Adversarial attack approaches are heuristics to highlight potential vulnerabilities of a GNN. However, they do not provide formal guarantees on the reliability of the methods. In particular, an *unsuccessful* attack does *not* imply the robustness of the GNN. It might just be that the attack approach could simply not find an/the adversarial example since it does not solve Eq. equation 8.1 exactly. Attacks, when successful, only provide results about non-robustness. For a safe use of GNNs, however, we need the opposite: we need principles for provable robustness. These methods provide so called *robustness certificates*, giving formal guarantees that no perturbation regarding a specific perturbation model $\Phi(\mathscr{G})$ will change the prediction.

Considering, for example, the task of node-level classification, the problem these certification approaches are aiming to solve is: Given a graph $\mathscr{G}$, a perturbation set $\Phi(\mathscr{G})$, and a GNN f_θ. Verify that the predicted class for node v stays the same for all $\hat{\mathscr{G}} \in \Phi(\mathscr{G})$. If this holds, we say that v is *certifiably robust* w.r.t. $\Phi(\mathscr{G})$.

Only few robustness certificates so far have been proposed for GNNs. They can mainly be categorized into two principles: model-specific and model-agnostic.

8.3.1 Model-Specific Certificates

Model-specific certificates are designed for a specific class of GNN models (e.g., 2-layer GCNs) and a specific task such as node-level classification. A common theme is to phrase certification as a constrained optimization problem: Recall that in a classification task, the final prediction is usually obtained by taking the class with the largest predicted probability or logit. Let $c^* = \arg\max_{c\in\mathscr{C}} f_\theta(\mathscr{G})_c$ denote the predicted class[4] obtained on the unperturbed graph $\mathscr{G}$, where $\mathscr{C}$ is the set of classes and $f_\theta(\mathscr{G})_c$ denotes the logit obtained for class c. This specifically implies, that the *margin* $f_\theta(\mathscr{G})_{c^*} - f_\theta(\mathscr{G})_c$ between class c^* and any other class c is positive.

A particularly useful quantity for robustness certification is the *worst-case margin*, i.e., the smallest margin possible under any perturbed data $\hat{\mathscr{G}}$:

$$\hat{m}(c^*, c) = \min_{\hat{\mathscr{G}}\in\Phi(\mathscr{G})} [f_\theta(\hat{\mathscr{G}})_{c^*} - f_\theta(\hat{\mathscr{G}})_c] \tag{8.3}$$

[4] This could either be the predicted class for a specific target node v in case of node-level classification; or for the entire graph in case of graph-level classification. We drop the dependency on v since it is not relevant for the discussion. For simplicity, we assume the maximizer c^* to be unique.

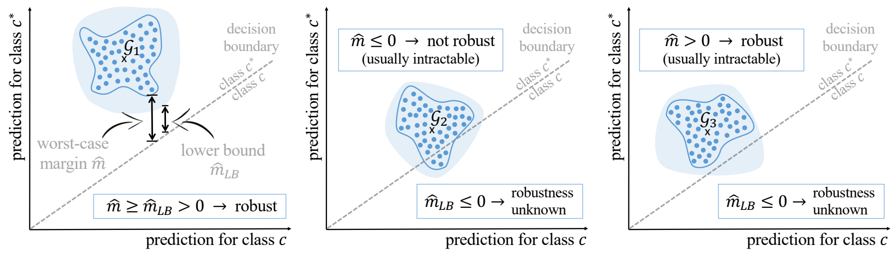
prediction for class c^*
$\mathcal{G}_1$
decision boundary
class c^*
class c
worst-case margin $\hat{m}$
lower bound $\hat{m}_{LB}$
$\hat{m} \geq \hat{m}_{LB} > 0 \rightarrow$ robust
prediction for class c
$\hat{m} \leq 0 \rightarrow$ not robust (usually intractable)
$\mathcal{G}_2$
$\hat{m}_{LB} \leq 0 \rightarrow$ robustness unknown
$\hat{m} > 0 \rightarrow$ robust (usually intractable)
$\mathcal{G}_3$

Fig. 8.4: Obtaining robustness certificates via the worst-case margin: The prediction obtained from the unperturbed graph $\mathcal{G}_i$ is illustrated with a cross, while the predictions for the perturbed graphs $\Phi(\mathcal{G}_i)$ are illustrated around it. The worst-case margin measures the shortest distance to the decision boundary. If it is positive (see $\mathcal{G}_1$), all predictions are on the same side of the boundary; robustness holds. If it is negative (see $\mathcal{G}_2$), some predictions cross the decision boundary; the class prediction will change under perturbations, meaning the model is not robust. When using lower bounds — the shaded regions in the figure — robustness is ensured for positive values (see $\mathcal{G}_1$) since the exact worst-case margin can only be larger. If the lower bound becomes negative, no statement can be made (see $\mathcal{G}_2$ and $\mathcal{G}_3$; robustness unknown). Both $\mathcal{G}_2$ and $\mathcal{G}_3$ have a negative lower bound, while the (not tractable to compute) exact worst-case margin differs in sign.

If this term is positive, c can never be the predicted class for node v. And if the worst-case margin $\hat{m}(c^*, c)$ stays positive for all $c \neq c^*$, the prediction is certifiably robust since the logit for class c^* is always the largest – for all perturbed graphs in $\Phi(\mathcal{G})$. This idea is illustrated in Figure 8.4.

As shown, obtaining a certificate means solving the (constrained) optimization problem in Eq. equation 8.3 for every class c. Not surprisingly, however, solving this optimization problem is usually intractable – for similar reasons as computing adversarial attacks is hard. So how can we obtain certificates? Just heuristically solving Eq. equation 8.3 is not helpful since we aim for guarantees.

Lower Bounds on the Worst-Case Margin

The core idea is to obtain tractable lower bounds on the worst-case margin. That is, we aim to find functions $\hat{m}_{LB}$ that ensure $\hat{m}_{LB}(c^*, c) \leq \hat{m}(c^*, c)$ and are more efficient to compute. One solution is to consider relaxations of the original constrained minimization problem, replacing, for example, the model's nonlinearities and hard discreteness constraints via their convex relaxation. For example, instead of requiring that an edge is perturbed or not, indicated by the variables $e \in \{0,1\}$, we can use $e \in [0,1]$. Intuitively, using such relaxations leads to supersets of the actually reachable predictions, as visualized in Figure 8.4 with the shaded regions.

Overall, if the lower bound $\hat{m}_{LB}$ stays positive, the robustness certificate holds — since $\hat{m}$ is positive by transitivity as well. This is shown in Figure 8.4 for graph $\mathscr{G}_1$. If $\hat{m}_{LB}$ is negative, no statement can be made since it is only a lower bound of the original worst-case margin $\hat{m}$, which thus can be positive or negative. Compare the two graphs $\mathscr{G}_2$ and $\mathscr{G}_3$ in Figure 8.4: While both have a negative lower bound (i.e., both shaded regions cross the decision boundary), their actual worst-case margins $\hat{m}$ differ. Only for graph $\mathscr{G}_2$ the actually reachable predictions (which are not efficiently computable) also cross the decision boundary. Thus, if the lower bound is negative, the actual robustness remains unknown – similar to an unsuccessful attack, where it remains unclear whether the model is actually non-robust or the attack simply not strong enough. Therefore, besides being efficient to compute, the function $\hat{m}_{LB}$ should be as close as possible to $\hat{m}$ to avoid cases where no answer can be given despite the model being robust.

The above idea, using convex relaxations of the model's nonlinearities and the admissible perturbations, is used in the works (Zügner and Günnemann, 2019; Zügner and Günnemann, 2020) for the class of GCNs and node-level classification. In (Zügner and Günnemann, 2019), the authors consider perturbations to the node attributes and obtain lower bounds via a relaxation to a linear program. The work (Zügner and Günnemann, 2020) considers perturbations in the form of edge deletions and reduces the problem to a jointly constrained bilinear program. Similarly, also using convex relaxations, Jin et al (2020a) has proposed certificates for graph-level classification under edge perturbations using GCNs. Beyond GCNs, model-specific certificates for edge perturbations have also been devised for the class of GNNs using PageRank diffusion (Bojchevski and Günnemann, 2019), which includes label/feature propagation and (A)PPNP (Klicpera et al, 2019a). The core idea of (Bojchevski and Günnemann, 2019) is to treat the problem as a PageRank optimization task which subsequently can be expressed as a Markov decision process. Using this connection one can indeed show that in scenarios where only local budgets are used (see Section 8.2; Eq. equation 8.2) the derived certificates are exact, i.e., no lower bound, while we can still compute them in polynomial time w.r.t. the graph size. In general, all models above consider local and global budget constraints on the number of changes.

Besides providing certificates, being able to efficiently compute (a differentiable lower bound on) the worst-case margin as in Eq. equation 8.3 also enables to improve GNN robustness by incorporating the margin during training, i.e. aiming to make it positive for all nodes. We will discuss this in detail in Section 8.4.2.

Overall, a strong advantage of model-specific certificates is their explicit consideration of the GNN model structure within the margin computation. However, the white-box nature of these certificates is simultaneously their limitation: The proposed certificates capture only a subset of the existing GNN models and any GNN yet to be developed likely requires a new certification technique as well. This limitation is tackled by model-agnostic certificates.

8.3.2 Model-Agnostic Certificates

Model-agnostic certificates treat the machine learning model as a black-box. For example, the work (Bojchevski et al, 2020a) provides certificates for any classifier operating on discrete data, including GNNs. Most importantly, it is sufficient to consider *only* the output of the classifier for different samples to obtain the certificate. This is precisely what makes it particularly appealing for certifying GNNs since it allows us to sidestep a complex analysis of the message-passing dynamics and the non-linear interactions between the nodes. So far, model-agnostic certificates are mainly based on the idea of randomized smoothing (Lecuyer et al, 2019; Cohen et al, 2019), originally proposed for continuous data. To handle graphs, extensions to discrete data have been proposed.

The core idea is to base the certificate on a *smoothed classifier*, which aggregates the output of the original (base) GNN when applied to randomly perturbed versions of the input graph $\mathscr{G}$. For example, the smoothed classifier might report the most likely (majority) class on these randomized samples. While different variants of this approach are possible, we provide one intuitive setting in the following to convey the main idea.

Let $f : \mathbb{G} \to \mathscr{C}$ denote a function (e.g., a GNN) that takes a graph $\mathscr{G} \in \mathbb{G}$ as input and predicts a single class $f(\mathscr{G}) = c \in \mathscr{C}$ as output, e.g. a node's prediction. Let τ be a smoothing distribution, also called randomization scheme, that adds random noise to the input graph. For example, τ might randomly add Bernoulli noise to the adjacency matrix of $\mathscr{G}$, corresponding to randomly adding or deleting edges. Technically, τ assigns probability mass/density $\Pr(\tau(\mathscr{G}) = \mathscr{Z})$ to each graph $\mathscr{Z} \in \mathbb{G}$. We can construct a *smoothed* (ensemble) classifier g from the *base* classifier f as follows:

$$g(\mathscr{G}) = \arg\max_{c \in \mathscr{C}} \Pr(f(\tau(\mathscr{G})) = c) \tag{8.4}$$

In other words, $g(\mathscr{G})$ returns the most likely class obtained by first randomly perturbing the graph $\mathscr{G}$ using τ and then classifying the resulting graphs $\tau(\mathscr{G})$ with the base classifier f.

As in Section 8.3.1, the goal is to assess whether the prediction does not change under perturbations: denoting with $c^* = g(\mathscr{G})$ the class predicted by the smoothed classifier on $\mathscr{G}$, we want $g(\hat{\mathscr{G}}) = c^*$ for all $\hat{\mathscr{G}} \in \Phi(\mathscr{G})$. Considering for simplicity the case of binary classification, this is equivalent to ensure that $\Pr(f(\tau(\hat{\mathscr{G}})) = c^*) \geq 0.5$ for all $\hat{\mathscr{G}} \in \Phi(\mathscr{G})$; or short: $\min_{\hat{\mathscr{G}} \in \Phi(\mathscr{G})} \Pr(f(\tau(\hat{\mathscr{G}})) = c^*) \geq 0.5$.

Since, unsurprisingly, the term is intractable to compute, we refer again to a lower bound to obtain the certificate:

$$\min_{\hat{\mathscr{G}} \in \Phi(\mathscr{G})} \min_{h \in \mathscr{H}_f} \Pr(h(\tau(\hat{\mathscr{G}})) = c^*) \leq \min_{\hat{\mathscr{G}} \in \Phi(\mathscr{G})} \Pr(f(\tau(\hat{\mathscr{G}})) = c^*) \tag{8.5}$$

Here, $\mathscr{H}_f$ is the set of *all* classifiers sharing some properties with f, e.g., often that the smoothed classifier based on h and f would return the same probability for $\mathscr{G}$, i.e., $\Pr(h(\tau(\mathscr{G})) = c^*) = \Pr(f(\tau(\mathscr{G})) = c^*)$. Since $f \in \mathscr{H}_f$, the inequality holds trivially. Accordingly, if the left hand side of Eq. equation 8.5 is larger than 0.5, also the right hand side is guaranteed to be so, implying that $\mathscr{G}$ would be certifiably robust.

What does Eq. equation 8.5 intuitively mean? It aims to find a base classifier h which minimizes the probability that the perturbed sample $\hat{\mathscr{G}}$ is assigned to class c^*. Thus, h represents a kind of *worst-case base classifier* which, when used within the smoothed classifier, tries to obtain a different prediction for $\hat{\mathscr{G}}$. If even this worst-case base classifier leads to certifiable robustness (left hand side of Eq. equation 8.5 larger than 0.5), then surely the actual base classifier at hand has well.

The most important part to make this all useful, however, is the following: given a set of classifiers $\mathscr{H}_f$, finding the worst-case classifier h and minimizing over the perturbation model $\Phi(\mathscr{G})$ is often tractable. In some cases, the optima can even be calculated in closed-form. This shows some interesting relation to the previous section: There, the intractable minimization over $\Phi(\mathscr{G})$ in Eq. equation 8.3 was replaced by some tractable lower bound, e.g., via relaxations. Now, by finding a worst-case classifier h we not only obtain a lower bound but minimization over $\Phi(\mathscr{G})$ becomes often also immediately tractable. Note, however, that in Section 8.3.1 we obtain a certificate for the base classifier f, while here we obtain a certificate for the smoothed classifier g.

Putting Model-Agnostic Certificates into Practice

As said, given a set of classifiers $\mathscr{H}_f$, finding the worst-case classifier h and minimizing over the perturbation model $\Phi(\mathscr{G})$ is often tractable. The main computational challenge in practice lies in determining $\mathscr{H}_f$. Let's consider our previous example where we enforced all classifiers h to ensure $\Pr(h(\tau(\mathscr{G})) = c^*) = \Pr(f(\tau(\mathscr{G})) = c^*)$. To determine $\mathscr{H}_f$, one needs to compute $\Pr(f(\tau(\mathscr{G})) = c^*)$. Clearly, doing this exactly is again usually intractable. Instead, the probability can be estimated using sampling. To ensure a tight approximation, the base classifier has to be fed a large number of samples from the smoothing distribution. This becomes increasingly expensive as the size and complexity of the GNN model increases. Furthermore, the resulting estimates only hold with a certain probability. Accordingly, also the derived guarantees have the same probability, i.e., one obtains only *probabilistic robustness certificates*. Despite these practical limitations, randomized smoothing has become widely popular, as it is often still more efficient than model-specific certificates.

This general idea of model-agnostic certificates has been investigated for discrete data in (Lee et al, 2019a; Dvijotham et al, 2020; Bojchevski et al, 2020a; Jia et al, 2020), with the latter two focusing also on graph-related tasks. In (Jia et al, 2020), the authors investigate the robustness of community detection. In (Bojchevski et al, 2020a), the main focus is on node-level and graph-level classification w.r.t. graph structure and/or attribute perturbations under global budget constraints. Specifically, Bojchevski et al (2020a) overcomes critical limitations of the other approaches in two regards: it explicitly accounts for sparsity in the data as present in many graphs, and it obtains strong certificates with a dramatically reduced computational complexity. Both aspects are core to making certification useful and possible for graph data. Since the approach of (Bojchevski et al, 2020a) is agnostic to the underlying classifier – it can be used as long as the input is discrete – it has been applied to various GNNs including GCN, GAT, (A)PPNP (Klicpera et al, 2019a), RGCN (Zhu et al, 2019a), and Soft Medoid (Geisler et al, 2020) as well as node-level and graph-level classification.

8.3.3 Advanced Certification and Discussion

Research on robustness certificates for GNNs is still in a very early stage. As we have seen in Section 8.2, the space of attacks is vast with different properties to study and perturbation models to consider. The methods discussed above cover only a few of these scenarios.

One step forward to more powerful certificates is the work of (Schuchardt et al, 2021). Like in local attacks to individual nodes, existing robustness certificates aim to certify each prediction independently. Thus, they assume that an adversary can use different perturbed inputs to attack different predictions. Alternatively, and similar to a global attack, the work (Schuchardt et al, 2021) introduces *collective robustness certificates* which compute the number of predictions which are *simultaneously* guaranteed to remain stable under perturbation. That is, it exploits the fact that a GNN simultaneously outputs multiple predictions based on a single shared input. Given a fixed perturbation budget, using this idea, the number of certifiable predictions can be increased by orders of magnitudes compared to certifying each prediction independently. The work, however, can not handle perturbation models with edge additions. As mentioned before, both views – local and global – are reasonable and it depends on the application which robustness guarantee is more relevant.

To cover the full spectrum of GNN applications, surely further certificates for other scenarios and tasks are required. Specifically, so far, all certificates assume an evasion attack scenario. It is also worth repeating that in the randomized smoothing approaches discussed above, we are actually certifying the smoothed (ensemble) classifier, and not the underlying base classifier. From a practitioner's point of view this means that obtaining a single prediction always requires to feed a large amount of samples through the GNN, leading to a scalability bottleneck which needs to be tackled in the future.

8.4 Improving Robustness of Graph Neural Networks

As we have established, standard GNNs trained in the usual way are not robust to even small changes to the graph, thus, using them in sensitive and critical applications might be risky. Certificates can provide us guarantees about their performance.

However, as a consequence of the non-robustness, the certificates rarely hold for standard models, i.e., only few predictions can be certified. To tackle this limitation, methods aiming to improve robustness have been investigated, i.e. making the models less susceptible to perturbations.[5] In this regard, three broad, not mutually exclusive, categories can be identified.

8.4.1 Improving the Graph

One seemingly clear direction to improve robustness is to remove perturbations from the data, i.e., to revert the performed malicious changes and obtain a more 'clean' graph. While this may sound simple, the inherent challenge is that adversarial perturbations are usually designed to be imperceptible, which makes their identification difficult. Still, as seen in Section 8.2.2.1, some patterns might be present.

Works such as (Zhang et al, 2019b) exploit this idea to perform a 'cleaning' of the graph before it is used as input to the GNN, relying on observations that, for example, the predicted class distribution changes for attacked nodes. Similarly, for attributed graphs, Wu et al (2019b) removes potential adversarial edges based on the Jaccard similary between the nodes' attributes. Such pre-processing steps are not limited to be 'attack detection' approaches that try to spot individual suspicious nodes are edges; they can also be thought of as a kind of denoising. Indeed, the work (Entezari et al, 2020) analyzed that perturbations performed by Nettack affect mainly the high-rank (low-valued) singular components of the graph's adjacency matrix. Thus, to improve robustness they compute a low-rank approximation of the graph which aims to remove the (adversarial) noise in a pre-processing procedure. The limitation is that the resulting graph becomes dense. Overall, such graph cleaning can be used in poisoning as well as evasion scenarios. Note, though, that an approach that has shown to perform well in one scenario, does not imply the success in another.

More generally, while these approaches have shown to be effective in specific scenarios, one has to be aware of one crucial limitation: the exploited patterns are often based on specific attacks like Nettack. Thus, the resulting detections might be limited to certain perturbations and potentially do not generalize to other scenarios.

Improving the graph is not restricted to happen before the training or the inference step, i.e. we do not need to follow a sequential approach of first cleaning and then learning a prediction model. Instead, the cleaning can be interwoven with the learning approach itself. Intuitively speaking, in order to minimize the corresponding training loss, one jointly learns the GNN parameters and also how to clean the graph itself. The benefit of this joint learning approach is that the specific model and task at hand can be taken into account, while the conditions enforced on the clean graph can be rather weak, e.g., only requiring that perturbations should be sparse.

[5] In some works, such approaches are called *(heuristic) defenses* to highlight their increased resilience to attacks. Similarly, some works use the term *provable defense* when referring to certificates since they provably prevent attacks to be harmful that are within a certified set $\Phi(\mathscr{G})$.

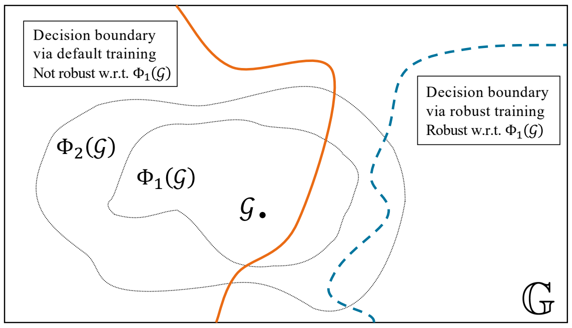

Fig. 8.5 Illustration of robust training: The classifier corresponding to the orange/-solid decision boundary is not robust to perturbations in $\Phi_1(\mathscr{G})$: some graphs cross the boundary and, thus, are assigned a different class. The classifier obtained through robust training (blue/dashed), assigns the same class to all graphs in $\Phi_1(\mathscr{G})$: it is robust w.r.t. $\Phi_1(\mathscr{G})$ – but not $\Phi_2(\mathscr{G})$.

Interestingly, even before the rise of graph neural networks, such joint approaches have been investigated, e.g., in (Bojchevski et al, 2017) to improve the robustness of spectral embeddings. For GNNs, such graph structure learning has been proposed in (Jin et al, 2020e; Luo et al, 2021) where certain properties like low-rank graph structure and attribute similarity are used to define how the clean graph should preferably look like.

8.4.2 *Improving the Training Procedure*

As discussed in Section 8.2.2, one further reason for the non-robustness of GNNs are the parameters/weights learned during training. Weights resulting from standard training often lead to models that do not generalize well to slightly perturbed data. This is illustrated in Figure 8.5 with the orange/solid decision boundary. Note that the figure shows the input space, i.e., the space of all graphs $\mathbb{G}$; this is in contrast to Figure 8.4 which shows the predicted probabilities. If we were able to improve our training procedure to find 'better' parameters – taking into account that the data is or might become potentially perturbed – the robustness of our model would improve as well. This is illustrated in Figure 8.5 with the blue/dashed decision boundary. There, all perturbed graphs from $\Phi_1(\mathscr{G})$ get the same prediction. As seen, in this regard robustness links to the generalization performance of prediction models in general.

8.4.2.1 Robust Training

Robust training refers to training procedures that aim at producing models that are robust to adversarial (and/or other) perturbations. The common theme is to optimize a *worst-case loss* (also called robust loss), i.e. the loss achieved under the worst-case perturbation. Technically, the training objective becomes:

$$\theta^* = \arg\min_{\theta} \max_{\hat{\mathscr{G}} \in \Phi(\mathscr{G})} \mathscr{L}_{\text{train}}(f_\theta(\hat{\mathscr{G}})) \tag{8.6}$$

where f_θ is the GNN with its trainable weights. As shown, we do not evaluate the loss at the unperturbed graph but instead use the loss achieved in the worst case (compare this to the standard training where we simply minimize $\mathscr{L}_{\text{train}}(f_\theta(\mathscr{G}))$). The weights are steered to obtain low loss under these worst scenarios as well, thus obtaining better generalization.

Not surprisingly, solving Eq. equation 8.6 is usually not tractable for the same reasons as finding attacks and certificates is hard: we have to solve a discrete, highly complex (minmax) optimization problem. In particular, for training, e.g., via gradient based approaches, we also need to compute the gradient w.r.t. the inner maximization. Thus, for feasibility, one usually has to refer to various surrogate objectives, substituting the worst-case loss and the resulting gradient by simpler ones.

Data Augmentation during Training

In this regard, the most naïve approach is to randomly draw samples from the perturbation set $\Phi(\mathscr{G})$ during each training iteration. That is, during training the loss and the gradient are computed w.r.t. these randomly perturbed samples; with different samples drawn in each training iteration. If the perturbation set, for example, contains graphs where up to x edge deletions are admissible, we would randomly create graphs with up to x edges dropped out. Such edge dropout has been analyzed in various works but does not improve adversarial robustness substantially (Dai et al, 2018a; Zügner and Günnemann, 2020); a possible explanation is that the random samples simply do not represent the worst-case perturbations well.

Thus, more common is the approach of *adversarial training* (Xu et al, 2019c; Feng et al, 2019a; Chen et al, 2020i). Here, we do not randomly sample from the perturbation set, but in each training iteration we create adversarial examples $\hat{\mathscr{G}}$ and subsequently compute the gradient w.r.t. these. As these samples are expected to lead to a higher loss, the result of the inner max-operation in Eq. equation 8.6 is much better approximated. Instead of perturbing the input graph, the work (Jin and Zhang, 2019) has investigated a robust training scheme which perturbs the latent embeddings.

It is interesting to note that adversarial training in its standard form requires labeled data since the attack aims to steer towards an incorrect prediction. In the typical transductive graph-learning tasks, however, large amounts of unlabeled data are available. As a solution, virtual adversarial training has also been investigated (Deng et al, 2019; Sun et al, 2020d), operating on the unlabeled data as well. Intuitively, it treats the currently obtained predictions on the unperturbed graph as the ground truth, making it a kind of self-supervised learning. The predictions on the perturbed data should not deviate from the clean predictions, thus enforcing smoothness.

Using (virtual) adversarial training has empirically shown some improvements in robustness, but not consistently. In particular, to well approximate the max term in the robust loss of Eq. equation 8.6, we need powerful adversarial attacks, which are typically costly to compute for graphs (see Section 8.2). Since here attacks need to be computed in every training iteration, the training process is slowed down substantially.

Beyond Data Augmentation - Certificate-Based Loss Functions

At the end of the day, the techniques above perform a costly data augmentation during training, i.e., they use altered versions of the graph. Besides being computationally expensive, there is no guarantee that the adversarial examples are indeed good proxies for the max term in Eq. equation 8.6. An alternative approach, e.g., followed by (Zügner and Günnemann, 2019; Bojchevski and Günnemann, 2019) relies on the idea of certification as discussed previously. Recall that these techniques compute a lower bound $\hat{m}_{LB}$ on the worst-case margin. If it is positive, the prediction is robust for this node/graph. Thus, the lower bound itself acts like a robustness loss $\mathscr{L}_{rob}$, for example instantiated as a hinge loss: $\max(0, \delta - \hat{m}_{LB})$. If the lower-bound is above δ, then the loss is zero; if it is smaller, a penalty occurs. Combining this loss function with, e.g., the usual cross-entropy loss, forces the model not only to obtain good classification performance but also robustness.

Crucially, $\mathscr{L}_{rob}$ and, thus, the lower bound need to be differentiable since we need to compute gradients for training. This, indeed, might be challenging since usually the lower bound itself is still an optimization problem. While in some special cases the optimization problem is directly differentiable (Bojchevski and Günnemann, 2019), another general idea is to relate to the principle of duality. Recall that the worst-case margin $\hat{m}$ (or a potential corresponding lower bound $\hat{m}_{LB}$) is the result of a (primal) *minimization* problem (see Eq. equation 8.3). Based on the principle of duality, the result of the dual *maximization* problem provides, as required, a lower bound to this value. Even more, *any* feasible solution of the dual problem provides a lower bound on the optimal solution. Thus, we actually do not need to solve the dual program. Instead, it is sufficient to compute the objective function of the dual at any single feasible point to obtain an (even lower, thus looser) lower bound; no optimization is required and computing gradients often becomes straightforward. This principle of duality has been used in (Zügner and Günnemann, 2019) to perform robust training in an efficient way.

8.4.2.2 Further Training Principles

Robust training is not the only way to obtain 'better' GNN weights. In (Tang et al, 2020b), for example, the idea of transfer learning (besides further architecture changes; see next section) is exploited. Instead of purely training on a perturbed target graph, the method adopts clean graphs with artificially injected perturbations to first learn suitable GNN weights. These weights are later transferred and fine-tuned to the actual graph at hand. The work (Chen et al, 2020i) exploits smoothing distillation where one trains on predicted soft labels instead of ground-truth labels to enhance robustness. The work (Jin et al, 2019b) argues that graph powering enhances robustness and proposes to minimize the loss not only on the original graph but on a set of graphs consisting of the different graph powers. Lastly, the authors of (You et al, 2021) use a contrastive learning framework using different (graph) data augmentations. Albeit adversarial robustness is not their focus, they report increased adversarial robustness against the attacks of (Dai et al, 2018a). In general, changing the loss function or regularization terms leads to different training, though the effects on robustness for GNNs are not fully understood yet.

8.4.3 Improving the Graph Neural Networks' Architecture

The final category of methods improving robustness is concerned with designing novel GNN architectures itself. Architecture engineering is one core component of neural network research in general, with many advancements in the last years. While traditionally focusing on improving prediction performance, a likewise important property becomes the methods' robustness – both being potentially opposing goals.

8.4.3.1 Adaptively Down-Weighting Edges

Inspired by the idea of graph cleaning as discussed before, a natural idea is to enhance the GNN by mechanisms to reduce the impact of perturbed edges. An obvious choice for this are edge attention principles. However, it is a false conclusion to assume that standard attention-based GNNs like GAT are immediately suitable for this task. Indeed, as shown in (Tang et al, 2020b; Zhu et al, 2019a) such models are non-robust. The problem is that these models still assume clean data to be given; they are not aware that the graph might be perturbed.

Thus, other attention approaches try to incorporate more information in the process. In (Tang et al, 2020b) the attention mechanism is enhanced by taking clean graphs into account for which perturbations have been artificially injected. Since now ground truth information is available (i.e., which edges are harmful), the attention can try to learn down-weighing these while retaining the non-perturbed ones. An alternative idea is used in (Zhu et al, 2019a). Here, the representations of each node in each layer are no longer represented as vectors but as Gaussian distribution. They hypothesize that attacked nodes tend to have large variances, thus using this information within the attention scores. Further attention mechanism considering, e.g., the model and data uncertainty or the neighboring nodes' similarity have been proposed in (Feng et al, 2021; Zhang and Zitnik, 2020).

Isnt it filtering still?

An alternative to edge attention is to enhance the aggregation used in message passing. In a GNN message passing step, a node's embedding is updated by aggregating over its neighbors' embeddings. In this regard, adversarially inserted edges add additional data points to the aggregation and therefore perturb the output of the message passing step. Aggregation functions such as sum, weighted mean, or the max operation used in standard GNNs can be arbitrarily distorted by only a single outlier. Thus, inspired by the principle of robust statistics, the work (Geisler et al, 2020) proposes to replace the usual GNN's aggregation function with a differentiable version of the Medoid, a provably robust aggregation operation. The idea of enhancing the robustness of the aggregation function used during message passing has further been investigated in (Wang et al, 2020o; Zhang and Lu, 2020). connecting adverserial nodes

Overall, all these methods down-weight the relevance of edges, with one crucial difference to the methods discussed in Section 8.4.1: they are adaptive in the sense that the relevance of each edge might vary between, e.g., the different layers of the GNN. Thus, an edge might be excluded/down-weighted in the first layer but included in the second one, depending on the learned intermediate representation. This allows a more fine-grained handling of perturbations. In contrast, the approaches in Section 8.4.1 derive a single cleaned graph that is used in the entire GNN.

8.4.3.2 Further Approaches

Many further ideas to improve robustness have been proposed, which do not all entirely fit into the before mentioned categories. For example, in (Shanthamallu et al, 2021) a surrogate classifier is trained which does not access the graph structure but is aimed to be aligned with the predictions of the GNN, both being jointly trained. Since the final predictor is not using the graph but only the node's attributes, higher robustness to structure perturbations is hypothesized. The work (Miller et al, 2019) proposes to select the training data in specific ways to increase robustness, and Wu et al (2020d) uses the principle of information bottleneck, an information theoretic approach to learn representations balancing expressiveness and robustness. Finally, also randomized smoothing (Section 8.3.2) can be interpreted as a technique to improve adversarial robustness by using an ensemble of predictors on randomized inputs.

8.4.4 Discussion and Future Directions

Considering the current state of research, a surprising observation is that robustness to graph structure perturbations is not well achieved via adversarial training. This is in stark contrast to, e.g., the image domain where robust training (in the form of adversarial training) can be considered one of the highly suitable techniques to improves robustness (Tramer et al, 2020). Focusing on perturbations of the node attributes, in contrast, robust training indeed performs very well as shown in (Zügner and Günnemann, 2019). Surprisingly, such robust training (targeting attributes) also improves robustness under graph structure perturbations (Zügner and Günnemann, 2020) – and, even more, outperforms several adversarial training strategies performing edge dropout. The question remains if structure perturbations have special property that diminishes the effect of robust training or whether the generated adversarial perturbations are not capturing the worst-case; showcasing again the hardness of the problem. This might also explain why the majority of works have focused on principles of weighting/filtering out edges.

In this regard, it is again important to remember that all approaches are typically designed with a specific perturbation model $\Phi(\mathcal{G})$ in mind. Indeed, down-weighting/filtering edges implicitly assumes that adversarial edges had been added to the graph. Adversarial edge deletions, in contrast, would require to identify potential edges to (re)add. This quickly becomes intractable due to the large number of possible edges and has not been investigated so far. Moreover, only a few methods so far have provided theoretical guarantees on the methods' robustness behavior.

8.5 Proper Evaluation in the View of Robustness

Progress in the field of GNN robustness requires sound evaluation of the proposed techniques. Importantly, we have to be aware of the potential trade-off between prediction performance (e.g., accuracy) and robustness. For example, we can easily obtain a highly robust classification model by simply always predicting the same class. Clearly, such a model has no use at all. Thus, the evaluation always involves two aspects: (1) Evaluation of the prediction performance. For this, one can simply refer to the established evaluation metrics such as accuracy, precision, recall, or similar, as known for the various supervised and unsupervised learning tasks. (2) Evaluation of the robustness performance.

Perturbation set and radius. Regarding the latter, the first noteworthy point is that robustness always links to a specific perturbation set $\Phi(.)$ that defines the perturbations the model should be robust to. To enable a proper evaluation, existing works therefore usually define some parametric form of the perturbation set, e.g., denoted $\Phi_r(\mathcal{G})$ where r is the maximal number of changes – the budget – we are allowed to perform (e.g., maximal number of edges to add). The variable r is often referred to as the *radius*. This is because the budget usually coincides with a certain maximal norm/distance we are willing to accept between graph $\mathcal{G}$ and perturbed ones. A generalization of the above form to consider multiple budgets/radii is straightforward. Varying the radius enables us to analyze the robustness behavior of the models in detail. Depending on the radius, different robustness results are expected. Specifically, for a large radius low robustness is expected – or even desired – and accordingly, the evaluation should also include these cases showing the limits of the models.

Recall that using the methods discussed in Section 8.2 and Section 8.3 together, we are able to obtain one of the following answers about a prediction's robustness: **(R)** It is robust; the certificate holds since, e.g., the lower bound on the margin is positive. **(NR)** It is non-robust; we are able to find an adversarial example. **(U)** Unknown; no statement possible since, e.g., the lower bound is negative but the attack was not successful either.

Figure 8.6 shows such an example analysis providing insights about the robustness properties of a GCN in detail. Here, local attacks and certificates are computed on standard (left) and robustly (right) trained GCNs for the task of node classification. As the result shows, robust training indeed increases the robustness of a GCN with fewer attacks being successful and more nodes being certifiable.

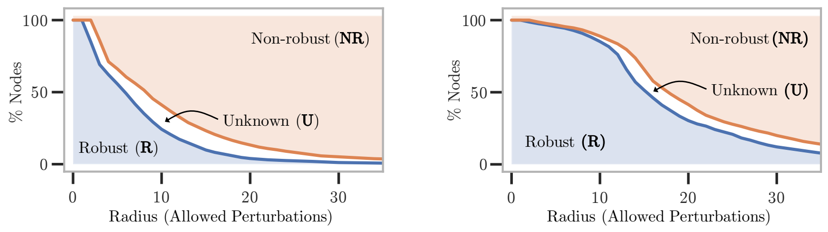

Fig. 8.6: Share of nodes which are provably robust (blue; **R**), non-robust via adversarial example construction (orange; **NR**), or whose robustness is unknown ("gap"; **U**), for increasing perturbation radii. For a given radius, the shares of **(R)**+**(NR)**+**(U)**= 100%. **Left**: Standard training; **Right**: robust training as proposed in (Zügner and Günnemann, 2019). Citeseer data and perturbations of node attributes.

It is worth highlighting that case **(U)** – the white gap in Figure 8.6 – occurs only due to the algorithmic inability to solve the attack/certificate problems exactly. Thus, case **(U)** does not give a clear indication about the GNN's robustness but rather about the performance of the attack/certificate.[6] Given this set-up, in the following we distinguish between two evaluation directions, which are reflected in frequently used measures.

Empirical Robustness Evaluation

In an empirical robustness evaluation, we perform an attack on the graph and observe the effects. Common measures are:

- The *drop in performance of the downstream task* (e.g., node classification accuracy), monitoring its decrease after the attack. This metric is typically used in combination with global attacks where a single perturbation is considered that aims to jointly change multiple predictions (see Section 8.2.1, Aspect 1).

[6] A large gap indicates that the attacks/certificates are rather loose. The gap might become smaller when improved attacks/certificates become available. Thus, attacks/certificates itself can be evaluated by analyzing the size of the gap since it shows what the maximal possible improvement in either direction is (e.g., the true share of robust predictions can never exceed 100%-NR for a specific radius).

- The *attack success rate*, measuring how many predictions were successfully changed by the attack(s). This simply corresponds to the case **(NR)**, the orange region shown in Fig 8.6. This metric is typically used in combination with local attacks where for each prediction a different perturbation can be used. Naturally, the local attacks' success rate is higher than the overall performance drop due to the flexibility in picking different perturbations.
- In the case of classification, the *classification margin*, i.e., the difference between the predicted probability of the 'true' class minus the second-highest class, and its drop after the attack. See again Figure 8.2 for an example.

The crucial limitation of this evaluation is its dependence on a specific attack approach. The power of the attack strongly affects the result. Indeed, it can be regarded as an *optimistic evaluation* of robustness since a non-successful attack is treated as seemingly robust. However, the conclusion is dangerous since a GNN might only perform well for one type of attack but not another. Thus, the above metrics rather evaluate the power of the attack but only weakly the robustness of the model. *Interpreting the results has to be done with care.* Consequently, when referring to empirical robustness evaluation, it is imperative to use multiple different and powerful attack approaches. Indeed, as also discussed in (Tramer et al, 2020), each robustification principle should come with its own specifically suited attack method (also called adaptive attack) to showcase its limitations.

Provable Robustness Evaluation

A potentially more suitable direction to analyze the robustness behavior of GNNs is to consider provable robustness. As discussed above, case **(U)** corresponds to unclear predictions for which no robustness statement can be given. Since we care about worst-case robustness, we have to assume that these predictions are non-robust as well. In short: **(NR)** and **(U)** should be rare, while case **(R)** should dominate: the number of certifiably robust predictions. Given this idea, the following evaluation metrics are often considered:

- *Certified ratio:* It corresponds to the number of predictions that can be certified as robust given a specific radius r in relation to the number of all predictions. Again take note whether for each prediction a different perturbation can be chosen from $\Phi_r(\mathscr{G})$ (local) or only a joint single one (global). Clearly, the global certified ratio is necessarily (and often significantly) larger than the local one.
- *Certified correctness:* In cases like classification, a prediction can be correct or incorrect. If it is correct and can be certified as well, the prediction is called certified correct. The other, highly undesired, extreme are predictions that are certified incorrect; they are very reliably misclassified.
- *Certified performance:* Based on the idea of certified correct predictions we can also derive a certified version of the original performance metrics, e.g., certified accuracy. Here, only those predictions are treated as correct for the metric if they are 'certified correct'. All other predictions, either incorrect or non-certifiable

are treated as wrong. The certified performance gives a provable lower bound on the performance of the GNN under any admissible perturbation w.r.t. the current perturbation set $\Phi_r(\mathscr{G})$ and the given data.

- *Certified radius:* While the above metrics assume a fixed $\Phi_r(\mathscr{G})$, i.e., a fixed radius r, we can also take another view. For a specific prediction, the largest radius r^* for which the prediction can still be certified as robust is called its certified radius. Given the certified radius of a single prediction, one can easily calculate the *average certifiable radius* over multiple predictions.

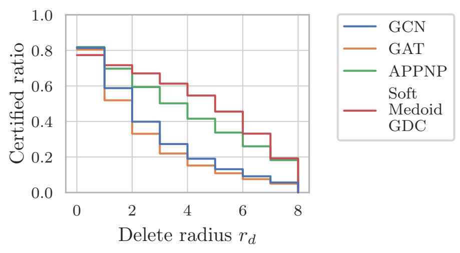

Fig. 8.7 Certified ratio of the smoothed classifier obtained from different GNN models using the certificate of (Bojchevski et al, 2020a) where $\Phi_r(\mathscr{G})$ consists of edge deletion perturbations. The model-agnostic nature of the certificate allows to compare the robustness across models.

Figure 8.7 shows the certified ratio for different GNN architectures for the task of node-classification when perturbing the graph structure. The smoothed classifier uses 10,000 randomly drawn graphs and the probabilistic certification is based on a confidence level of $\alpha = 0.05$ analogously to the set-up in (Geisler et al, 2020). Since local attacks are considered, the certified ratio is naturally rather low. Still, as shown, there is a significant difference between the models' robustness performance.

Provable robustness evaluation provides strong guarantees in the sense that the evaluation is more pessimistic. E.g. if the certified ratio is high, we know that the actual GNN can only be better. Note again, however, that we still also implicitly evaluate the certificate; with new certificates the result might become even better. Also recall that certificates based on randomized smoothing (Section 8.3.2), evaluate the robustness of the smoothed classifier, thus, not providing guarantees for the base classifier itself. Still, a robust prediction of the smoothed classifier entails that the base classifier predicts the respective class with a high probability w.r.t. the randomization scheme.

As it becomes apparent, evaluating robustness is more complex than evaluating usual prediction performance. To achieve a detailed understanding of the robustness properties of GNNs it is thus helpful to analyze all aspects introduced above.

8.6 Summary

Along with the increasing relevance of graph neural networks in various application domains, comes also an increasing demand to ensure their reliability. In this regard, adversarial robustness plays a central role since perturbed data is omnipresent. As we have seen, standard GNN architectures and training principles – as predominantly used in today's applications – lead to non-robust models, with all the undesired consequences included. However, there is hope: First, various principles to improve robustness of GNNs have started to emerge. The obtained results are already promising giving a first indication that improved robustness can be achieved without giving up too much of the GNNs' prediction performance. Second, robustness certificates provide us ways to even assess certain robustness properties in a formal way. That is, one does not need to rely on heuristics but instead obtains guarantees of the GNN's behavior. In all these directions, one has just started to explore the vast possibilities and many challenges still need to be tackled. Thus, in the upcoming years, various further insights can be expected, pursuing one common goal: to continue the success story of graph neural networks by enabling their reliable use in even sensitive and safety-critical domains.

Acknowledgements

A special thanks to my amazing PhD students Aleksandar Bojchevski, Simon Geisler, Jan Schuchardt, and Daniel Zügner who not only provided valuable feedback to this article but also made many of the research results in this field possible.

Editor's Notes: Adversarial Robustness is one of the hottest topics in Machine Learning/Deep Learning today. This wave of research starts from the robustness of Convolutional Neural Networks in computer vision domain and has rapidly influenced other ML/DL network architectures in other applications domains like NLP and Graphs. Adversarial robustness of Graph Neural Networks is a very important research area, which has a fundamental impact on many other learning tasks, including graph classification task (Chapter 9), link prediction (Chapter 10), graph generation-related tasks (Chapter 11 and Chapter 12), graph matching networks (Chapter 13), and so on. Some chapters (like Chapter 14) can be treated one of potential ways to help alleviate the effect of adversarial robustness by learning a graph structure beyond its intrinsic graph structure.