Certifiable Robustness of Graph Convolutional Networks under Structure Perturbations

Daniel Zügner, Stephan Günnemann

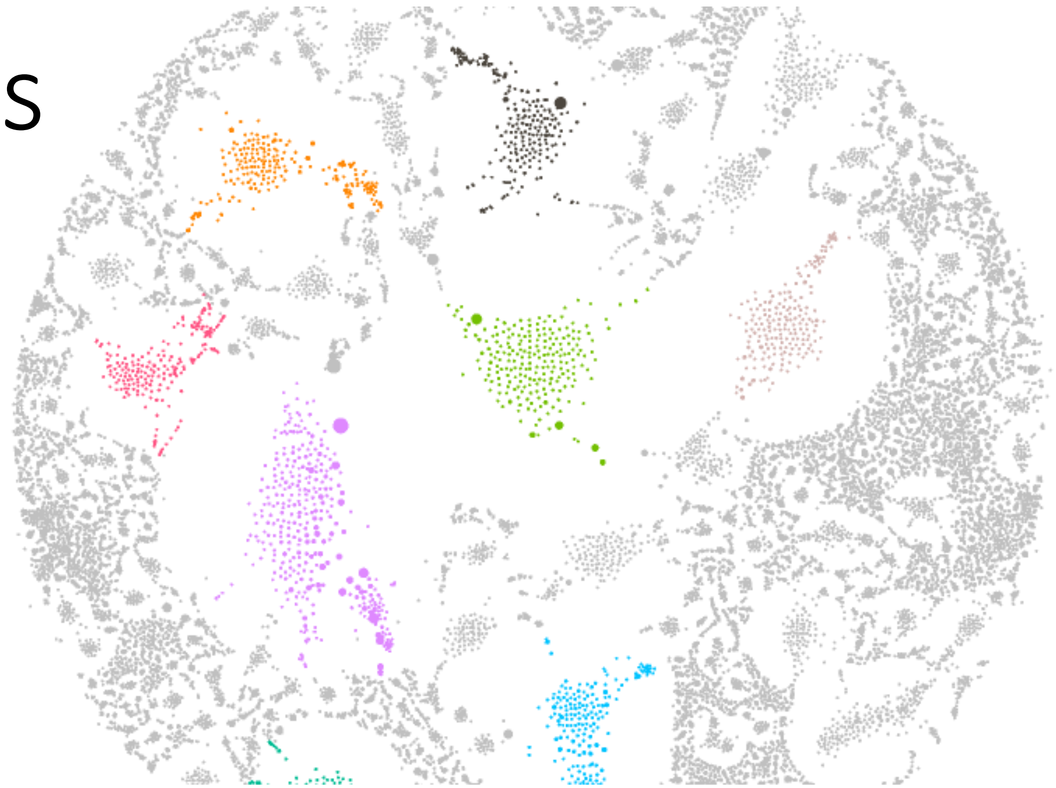
Technical University of Munich, Germany

KDD 2020

Deep Learning on Graphs

Graphs are ubiquitous

- Social networks
- Web graphs
- Knowledge graphs



[image: linkurio.us]

Graph neural networks (GNNs): state of the art on tasks such as

- **Semi-supervised node classification** (our work's focus)
- Link prediction
- Unsupervised representation learning (node embeddings)

Background: GNN Node Classification

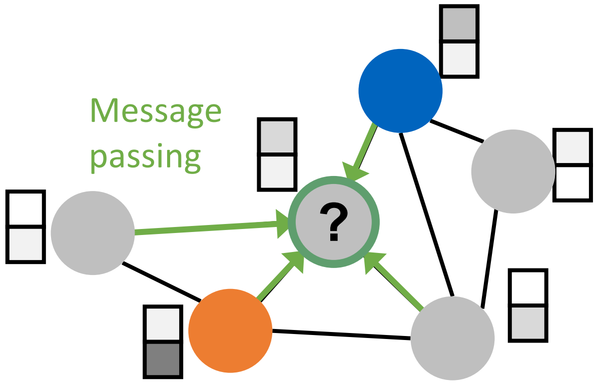
Given: Graph $G = (\boldsymbol{A}, \boldsymbol{X})$ with connectivity $\boldsymbol{A}$ and node features $\boldsymbol{X}$

Predict a **node's class** by using its **neighborhood**.

e.g., two-layer Graph Convolutional Network (**GCN**):

$$\hat{\boldsymbol{Y}} = \mathrm{softmax}\left(\hat{\boldsymbol{A}}\,\sigma\left(\hat{\boldsymbol{A}}\,\boldsymbol{X}\,\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)}\right)\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)}\right)$$

Pre-processed adjacency matrix



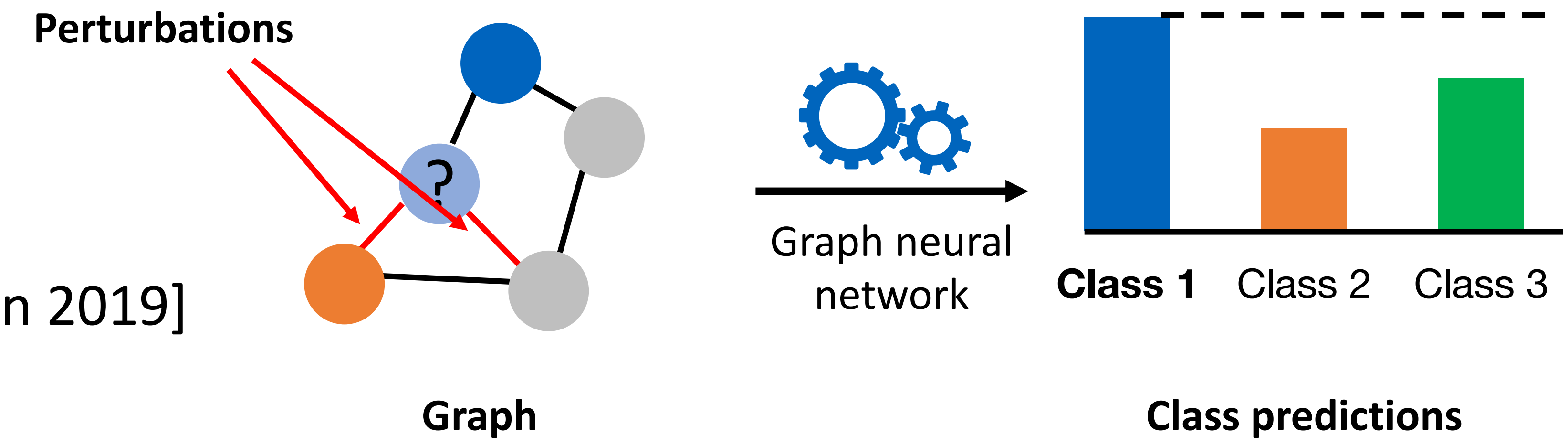
Graph Neural Networks are not robust

GNNs are not robust w.r.t. adversarial attacks.

[Dai et al. 2018]

[Wang et al. 2018]

[Zügner et al. 2018],
[Zügner and Günnemann 2019]



Adversarial attacks on GNNs:

- Node feature perturbations
- **Graph structure perturbations** (this work's focus)

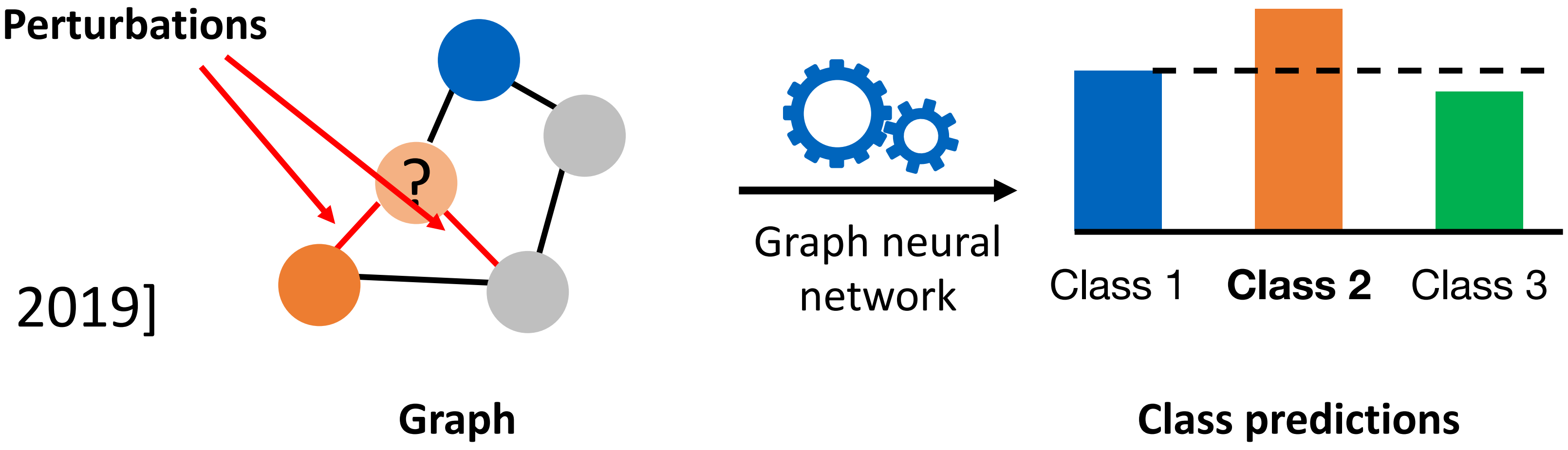
Graph Neural Networks are not robust

GNNs are not robust w.r.t. adversarial attacks.

[Dai et al. 2018]

[Wang et al. 2018]

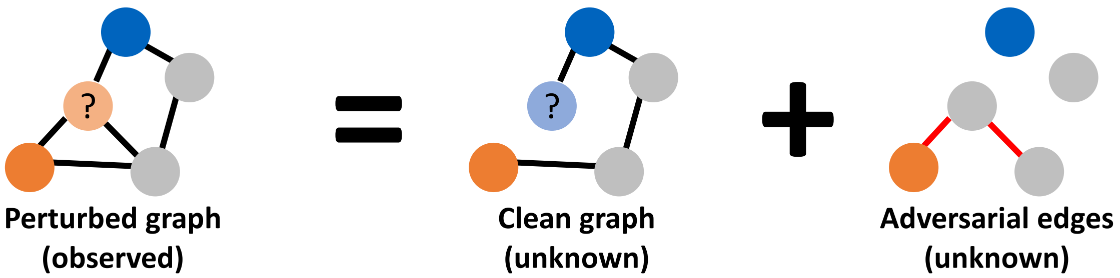
[Zügner et al. 2018],
[Zügner and Günnemann 2019]



Adversarial attacks on GNNs:

- Node feature perturbations
- **Graph structure perturbations** (this work's focus)

High-level approach



- We assume that the present graph can contain up to a certain number of adversarial edges, i.e. an attacker has **inserted edges**.
- In other words, the **clean (unknown) graph** is reachable by **removing edges** from the present graph according to the adversarial budget.
- That is, by **inserting edges** the **adversary** could have changed the **model's prediction**.

Finding the worst-case perturbation

- **Robustness certification**: check whether there exists a graph **reachable by removing edges** for which the node has a different predicted label.

➔ **Certify** robustness if it is guaranteed that the prediction **does not change**.

- Since **enumerating** all reachable graphs is **intractable**, we propose to **bound** the maximum change in logits (“**worst-case**” perturbation).

- L_0-**bounded** perturbations on the graph structure
  - At most q adversarial edges per node
  - At most Q adversarial edges in total (over all nodes)

Optimization problem view

- Finding the "worst-case" perturbation as an **optimization problem**:

$$\tilde{m}^t(y^*, y) \coloneqq \min_{\boldsymbol{A}} \; \underbrace{\overbrace{f(\boldsymbol{A}, \boldsymbol{X})_{ty^*}}}_{\text{Logit of the observed class}} - \underbrace{f(\boldsymbol{A}, \boldsymbol{X})_{ty}}_{\text{Logit of a different class}} \qquad \textit{subject to} \text{ budget constraints}$$

GCN

Optimize over **adjacency matrix**

- $\tilde{m}^t(y^*, y) > 0$ for **all** classes y: predicted class **cannot be changed (robust).**
- $\tilde{m}^t(y^*, y) < 0$ for **any** class y: prediction is **not robust.**

Optimization problem challenges

Challenges for optimization:

- f has **nonlinear activation functions**

➔ can be addressed via linear relaxation.

$$\tilde{m}^t(y^*, y) \coloneqq \min_{\boldsymbol{A}} \underbrace{f(\boldsymbol{A}, \boldsymbol{X})_{ty^*}}_{\text{Logit of the observed class}} - \underbrace{f(\boldsymbol{A}, \boldsymbol{X})_{ty}}_{\text{Logit of a different class}}$$

GCN

$\textit{subject to}$ budget constraints

- Optimization over **binary variable** ($A \in \{0,1\}^{N \times N}$)
- Problem contains **nonconvex products** of variables

Addressed in the following

Challenge: binary variables

Addressing the challenge of optimization over **binary variable** ($\boldsymbol{A} \in \{0,1\}^{N\times N}$)

- Naïve solution: **continuous relaxation**, i.e. $\boldsymbol{A} \in [0,1]^{N\times N}$
- However, GCN performs **preprocessing** on $\boldsymbol{A}$:

Pre-processed matrix: $\hat{\boldsymbol{A}}_{ij} = \begin{cases} \boxed{\frac{1}{\sqrt{(\deg(i)+1)\cdot(\deg(j)+1)}}} & \text{If nodes i and j are connected (or i=j)} \\ 0 & \text{else} \end{cases}$

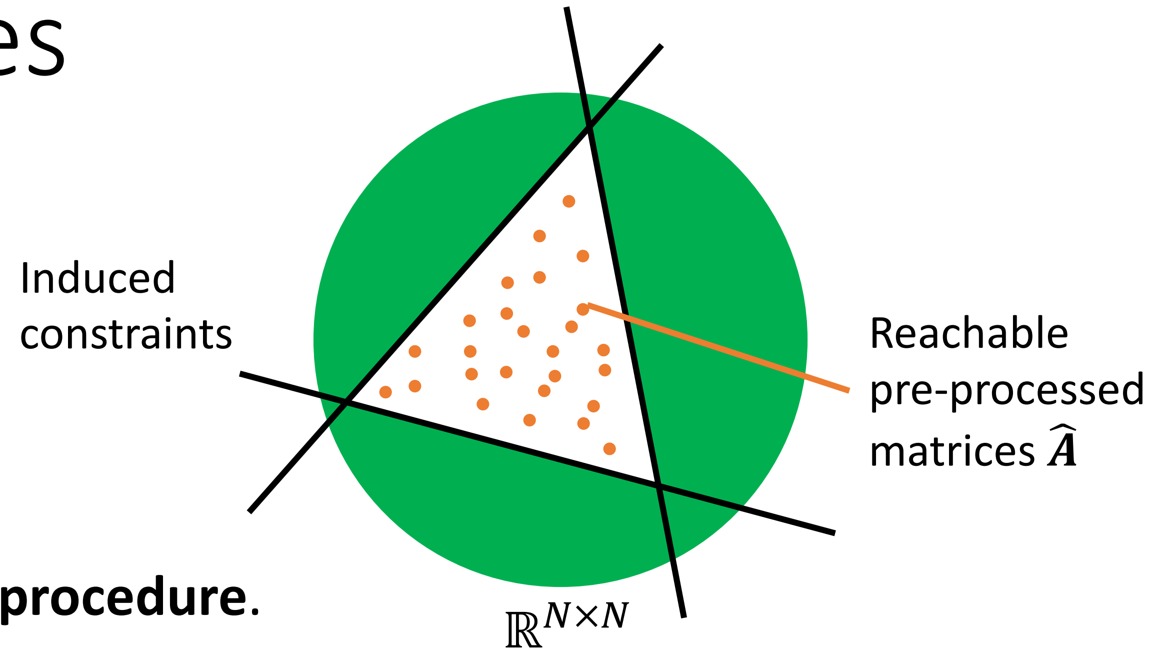
Values $\hat{\boldsymbol{A}}_{ij}$ are **not convex** in the variables ($\boldsymbol{A}_{ij}$)

Challenge: binary variables

Our approach:

- directly optimize the continuous-valued message passing matrix $\widehat{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$
- Derive **constraints** on $\widehat{\boldsymbol{A}}$ that are **induced** by the **budget constraints** on $\boldsymbol{A}$ and the **preprocessing procedure**.
- E.g.: node degrees cannot **increase.**

➔ $\widehat{\boldsymbol{A}}_{ij}$ must be in $\left[0, \frac{1}{\sqrt{(\deg(i)+1-q) \cdot (\deg(j)+1-q)}}\right]$



- In the paper we derive **induced constraints** on an element-, row-, and global level.

Challenge: nonconvex variable products

- At each **GCN layer** we perform message passing in the form of multiplying by $\hat{\boldsymbol{A}}$:

$$\boldsymbol{H}^{(l)} = \sigma\big(\overbrace{\hat{\boldsymbol{A}}\,\underbrace{\boldsymbol{H}^{(l-1)}}_{\text{depends on } \hat{\boldsymbol{A}}}}^{\text{bilinear terms}}\,\boldsymbol{W}^{(L)} + \boldsymbol{b}^{(L)}\big)$$

- This introduces **non-convex bilinear terms** of the variables.

- A **naïve solution** is to relax these terms using the reformulation-linearization (RLT) technique. However, this leads to loose approximations.

Challenge: nonconvex variable products

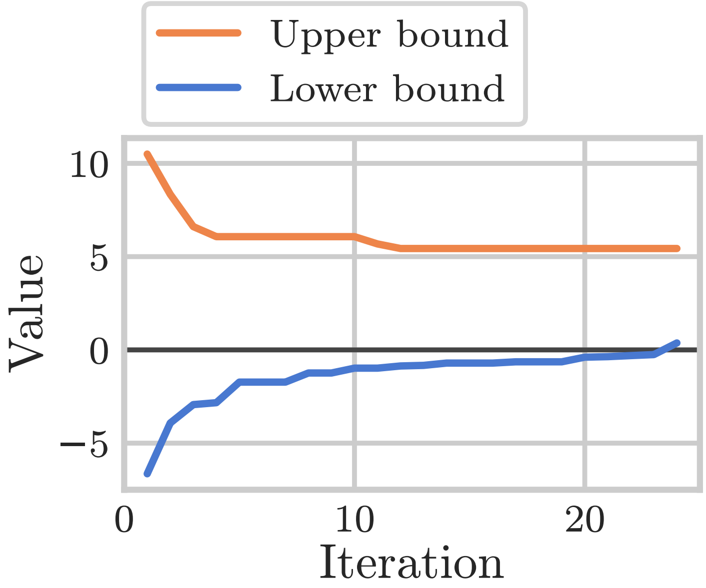
- We show how to phrase the problem as a **jointly constrained bilinear program**:

$$\min_{\mathbf{x},\mathbf{z}} \underbrace{\mathbf{x}^T\mathbf{z}}_{\text{bilinear objective}} \qquad \textit{subject to}\ \underbrace{\mathbf{x}+\mathbf{z} \le \mathbf{c}}_{\text{jointly constrained variables}}$$

(and other constraints)

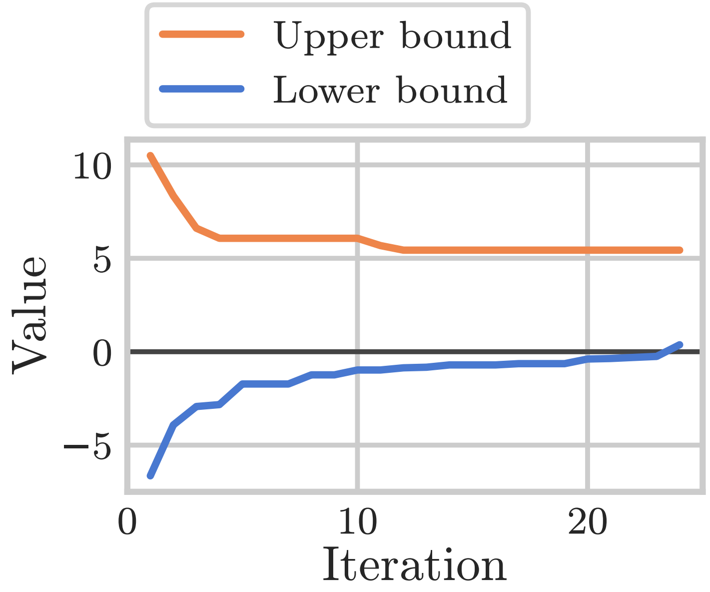
- Can be solved **exactly** via **branch-and-bound (B&B)** techniques.
- We design a custom **B&B-scheme** for **provably robust GCN:**
  - converges **faster** and
  - produces **tighter bounds** (compared to standard B&B).

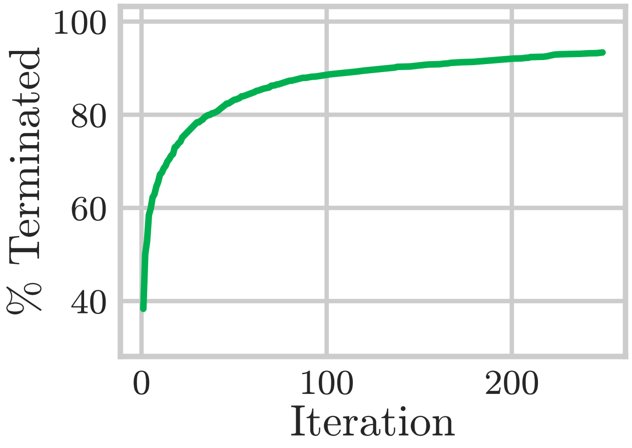
Optimization properties



- At each B&B iteration we solve a **linear program**, for which we can use highly-optimized **off-the-shelf solvers**.
- We **refine** upper and lower bounds at each iteration
- Finding the **global optimum** can take an infinite number of iterations
- Since we are only interested in the **sign of the optimum**, we can stop when either the upper or lower bound **crosses zero**.

Optimization





Once the upper or lower bound **crosses zero**, we can stop optimization.

Lower bound > 0: **provably robust**

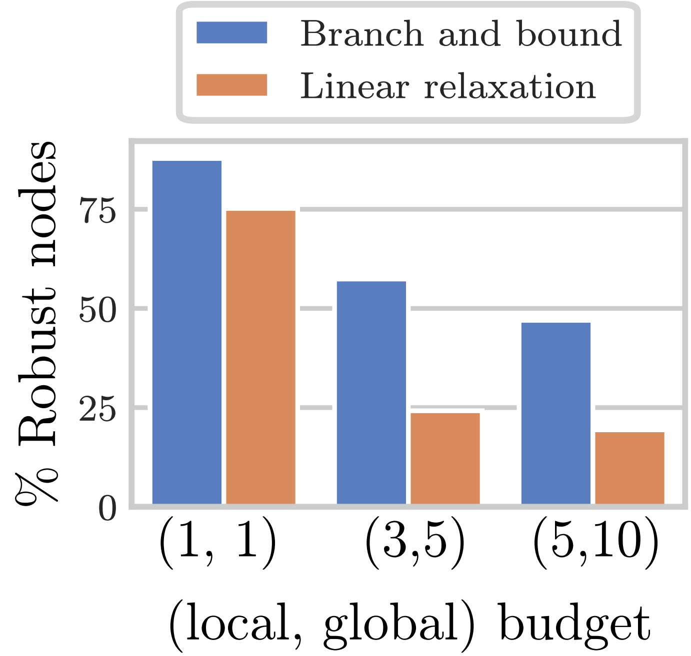
Upper bound < 0: no decision

In about **80%** of the cases, our algorithm terminates within 50 iterations.

Dataset: Cora-ML

Comparison to linear relaxation

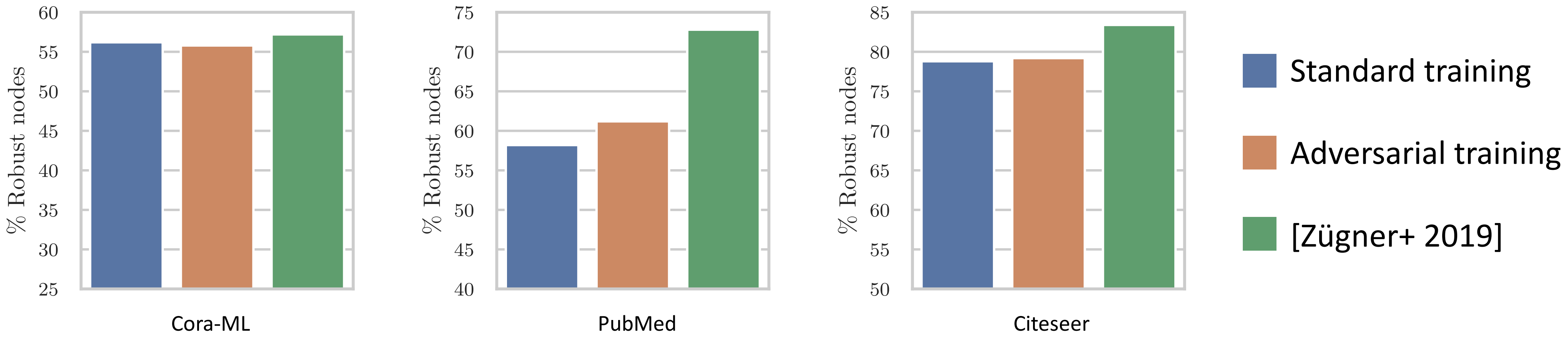
- Our **branch-and-bound** method can prove robustness for many more nodes than a linear relaxation baseline.
- Even a **single edge perturbation** can change the predicted labels of up to 11% of nodes.



Dataset: Cora-ML

Robust training comparison

- We **compare** various **robust training schemes** in terms of their robustness share.
- In line with [Dai et al 2018], **adversarial training** does not seem to help.
- Interestingly, the only **consistently more robust** training method is [Zügner+ 2019]'s method for improved robustness for attribute attacks.



Summary

- **Robustness certification** of GCN for perturbations of the graph structure
- Our novel **branch-and-bound** algorithm substantially outperforms a linear relaxation approach
- **Adversarial training** does not seem to improve provable robustness. This highlights the need for **novel robust training** schemes.
- Paper, code & more: www.daml.in.tum.de/robust-gcn