# Provable Robustness of (Graph) Neural Networks Against Data Poisoning and Backdoor Attacks

**Lukas Gosch**[1,2,3*], **Mahalakshmi Sabanayagam**[1*],
**Debarghya Ghoshdastidar**[1,2], **Stephan Günnemann**[1,2]
[1] School of Computation, Information and Technology, [2] Munich Data Science Institute
Technical University of Munich, Germany
[3] Munich Center for Machine Learning (MCML), Germany
{l.gosch, m.sabanayagam, d.ghoshdastidar, s.guennemann}@tum.de

## Abstract

Generalization of machine learning models can be severely compromised by data poisoning, where adversarial changes are applied to the training data. This vulnerability has led to interest in certifying (i.e., proving) that such changes up to a certain magnitude do not affect test predictions. We, for the *first* time, certify Graph Neural Networks (GNNs) against poisoning attacks, including backdoors, targeting the node features of a given graph. Our certificates are white-box and based upon $(i)$ the *neural tangent kernel*, which characterizes the training dynamics of sufficiently wide networks; and $(ii)$ a novel reformulation of the bilevel optimization describing poisoning as a mixed-integer linear program. We note that our framework is more general and constitutes the *first* approach to derive white-box poisoning certificates for NNs, which can be of independent interest beyond graph-related tasks.

## 1 Introduction

Numerous works showcase the vulnerability of modern machine learning models to data poisoning, where adversarial changes are made to the training data [Biggio et al., 2012, Muñoz-González et al., 2017, Zügner and Günnemann, 2019, Wan et al., 2023], as well as backdoor attacks affecting both training and test sets [Goldblum et al., 2023]. Empirical defenses against such threats are continually at risk of being compromised by future attacks [Koh et al., 2022, Suciu et al., 2018]. This motivates the development of *robustness certificates*, which provide formal guarantees that the prediction for a given test data point remains unchanged under an assumed perturbation model.

Robustness certificates can be categorized as providing deterministic or probabilistic guarantees, and as being white box, i.e. developed for a particular model, or black box (model-agnostic). While each approach has its strengths and applications [Li et al., 2023], we focus on *white-box* certificates as they can provide a more direct understanding into the worst-case robustness behavior of commonly used models and architectural choices [Tjeng et al., 2019, Mao et al., 2024, Banerjee et al., 2024]. The literature on poisoning certificates is less developed than certifying against test-time (evasion) attacks (see App. N) and we provide an overview and categorization in Table 2. Notably, white-box certificates are currently available only for decision trees [Drews et al., 2020], nearest neighbor algorithms [Jia et al., 2022], and naive Bayes classification [Bian et al., 2024]. In the case of Neural Networks (NNs), the main challenge in white-box poisoning certification comes from capturing their complex training dynamics. As a result, the current literature reveals that deriving white-box poisoning certificates for NNs, and by extension Graph Neural Networks (GNNs), is still an *unsolved* problem, raising the question if such certificates can at all be practically computed.

[*] Equal contribution.

AdvML-Frontiers'24: The 3nd Workshop on New Frontiers in Adversarial Machine Learning@NeurIPS'24, Vancouver, CA.

In this work, we give a positive answer to this question by developing the first approach towards white-box certification of NNs against data poisoning and backdoor attacks, and instantiate it for common convolution-based and PageRank-based GNNs. Concretely, poisoning can be modeled as a bilevel optimization problem over the training data $\mathcal{D}$ that includes training on $\mathcal{D}$ as its inner subproblem. To overcome the challenge of capturing the complex training dynamics of NNs, we consider the Neural Tangent Kernel (NTK) that characterizes the training dynamics of sufficiently wide NNs under gradient flow [Jacot et al., 2018, Arora et al., 2019]. In particular, we leverage the equivalence between NNs trained using the soft-margin loss and standard soft-margin Support Vector Machines (SVMs) with the NN's NTKs as kernel matrix [Chen et al., 2021]. Using this equivalence, we introduce a novel reformulation of the bilevel optimization problem as a mixed-integer linear program (MILP) that allows to certify test datapoints against poisoning as well as backdoor attacks for sufficiently wide NNs (see Fig. 1). Although our framework applies to wide NNs in general, solving the MILP scales with the number of labeled training samples. Thus, it is a natural fit for semi-supervised learning tasks, where one can take advantage of the low labeling rate. As a result, we focus on semi-supervised node classification in graphs and use our framework with corresponding NTKs [Sabanayagam et al., 2023] of various GNNs. Our **contributions** are:

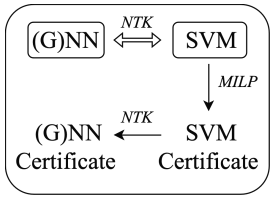


Figure 1: Illustration of our poisoning certificate.

**(i)** We are the first to certify GNNs in node-classification tasks against poisoning and backdoor attacks targeting node features. Our certification framework called QPCert is introduced in Sec. 3 and leverages the NTK to capture the complex training dynamics of GNNs. Further, it can be applied to NNs in general and thus, it represents the first approach on *white-box* poisoning certificates for NNs.

**(ii)** We contribute a reformulation of the bilevel optimization problem describing poisoning as a MILP when instantiated with kernelized SVMs, allowing for white-box certification of SVMs. While we focus on the NTK as kernel, our strategy can be transferred to arbitrary kernel choices.

**Notation.** We represent matrices and vectors with boldfaced upper and lowercase letters, respectively. $v_i$ and $M_{ij}$ denote $i$-th and $ij$-th entries of the vector $\mathbf{v}$ and matrix $\mathbf{M}$, respectively. $i$-th row of matrix $\mathbf{M}$ is $\mathbf{M}_i$. $\mathbf{I}_n$ is the identity matrix of size $n$ and $\mathbf{1}_{n\times n}$ is the matrix of all 1s with size $n \times n$. $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes scalar product between $\mathbf{a}$ and $\mathbf{b}$. We use $\|.\|_2$ for vector Euclidean norm and matrix Frobenius norm, $\mathbb{1}[.]$ for indicator function, $\mathbb{E}[.]$ for expectation, $\odot$ for the Hadamard product and the ceil operator $\lceil z \rceil$ for the smallest integer $\geq z$. $[n]$ denotes $\{1, 2, \ldots, n\}$.

## 2 Preliminaries

We are given a partially-labeled graph $\mathcal{G} = (\boldsymbol{S}, \boldsymbol{X})$ with $n$ nodes and a graph structure matrix $\boldsymbol{S} \in \mathbb{R}^{n\times n}_{\geq 0}$, representing for example, a normalized adjacency matrix. Each node $i \in [n]$ has features $\boldsymbol{x}_i \in \mathbb{R}^d$ collected in a node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$. We assume labels $y_i \in \{1, \ldots, K\}$ are given for the first $m \leq n$ nodes. Our goal is to perform node classification in transductive (labels of the remaining $n - m$ nodes should be inferred) or inductive (newly added nodes at test time should be classified) settings. $\mathcal{V}_L$ and $\mathcal{V}_U$ denote the set of labeled and unlabeled nodes, respectively.

**Perturbation model.** We assume that at training time the adversary $\mathcal{A}$ has control over the features of an $\epsilon$-fraction of nodes and that $\lceil (1-\epsilon)n \rceil$ nodes are clean. For backdoor attacks, the adversary can also change the features of a test node of interest. Following the *semi-verified learning* setup introduced in [Charikar et al., 2017], we assume that $k < n$ nodes are known to be uncorrupted. We denote the verified nodes by set $\mathcal{V}_V$ and the nodes that can be potentially corrupted as set $\mathcal{U}$. We further assume that the strength of $\mathcal{A}$ to poison training or modify test nodes is bounded by a budget $\delta \in \mathbb{R}_+$. More formally, $\mathcal{A}$ can choose a perturbed $\tilde{\boldsymbol{x}}_i \in \mathcal{B}_p(\boldsymbol{x}_i) := \{\tilde{\boldsymbol{x}} \mid \|\tilde{\boldsymbol{x}} - \boldsymbol{x}_i\|_p \leq \delta\}$ for each node $i$ under control. We denote the set of all perturbed node feature matrices constructible by $\mathcal{A}$ from $\boldsymbol{X}$ as $\mathcal{A}(\boldsymbol{X})$ and $\mathcal{A}(\mathcal{G}) = \{(\boldsymbol{S}, \tilde{\boldsymbol{X}}) \mid \tilde{\boldsymbol{X}} \in \mathcal{A}(\boldsymbol{X})\}$. In data poisoning, the *goal* of $\mathcal{A}$ is to maximize misclassification in the test nodes. For backdoor attacks $\mathcal{A}$ aims to induce misclassification only in test nodes that it controls.

**Learning setup.** GNNs are functions $f_\theta$ with (learnable) parameters $\theta \in \mathbb{R}^q$ and $L$ number of layers taking the graph $\mathcal{G} = (\boldsymbol{S}, \boldsymbol{X})$ as input and outputting a prediction for each node. We consider linear output layers with weights $\boldsymbol{W}^{L+1}$ and denote by $f_\theta(\mathcal{G})_i \in \mathbb{R}^K$ the (unnormalized) logit output associated to node $i$. Note for binary classification $f_\theta(\mathcal{G})_i \in \mathbb{R}$. We define the architectures such as

MLP, GCN [Kipf and Welling, 2017], SGC [Wu et al., 2019], (A)PPNP [Gasteiger et al., 2019] and others in App. A. We focus on binary classes $y_i \in \{\pm 1\}$ and refer to App. E for the multi-class case. Following Chen et al. [2021], the parameters $\theta$ are learned using the soft-margin loss

$$\mathcal{L}(\theta, \mathcal{G}) = \min_{\theta} \frac{1}{2}\|\mathbf{W}^{(L+1)}\|_2^2 + C\sum_{i=1}^{m} \max(0, 1 - y_i f_\theta(\mathcal{G})_i) \tag{1}$$

where the second term is the Hinge loss weighted by a regularization $C \in \mathbb{R}_+$. Note that due to its non-differentiability, the NN is trained by subgradient descent. Furthermore, we consider NTK parameterization [Jacot et al., 2018] in which parameters $\theta$ are initialized from a standard Gaussian $\mathcal{N}(0, 1/\text{width})$. Under NTK parameterization and sufficiently large width limit, the training dynamics of $f_\theta(\mathcal{G})$ are precisely characterized by the NTK defined between nodes $i$ and $j$ as $Q_{ij} = \boldsymbol{Q}(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{E}_\theta[\langle \nabla_\theta f_\theta(\mathcal{G})_i, \nabla_\theta f_\theta(\mathcal{G})_j \rangle] \in \mathbb{R}$.

**Equivalence of NN to soft-margin SVM with NTK.** Chen et al. [2021] show that training NNs in the infinite-width limit with Eq. (1) is equivalent to training a soft-margin SVM with (sub)gradient descent using the NN's NTK as kernel. Thus, both methods converge to the same solution. More formally, let the SVM be defined as $f_\theta(\mathcal{G})_i = f_\theta^{SVM}(\boldsymbol{x}_i) = \langle \boldsymbol{\beta}, \Phi(\boldsymbol{x}_i) \rangle$ where $\Phi(\cdot)$ is the feature transformation associated to the used kernel and $\theta = \boldsymbol{\beta}$ are the learnable parameters obtained by minimizing $\mathcal{L}(\theta, \mathcal{G})$. Following Chen et al. [2021], we do not include a bias term. To find the optimal $\boldsymbol{\beta}^*$, instead of minimizing Eq. (1) with (sub)gradient descent, we work with the equivalent dual

$$\mathrm{P}_1(\boldsymbol{Q}): \min_{\boldsymbol{\alpha}} -\sum_{i=1}^{m} \alpha_i + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j Q_{ij} \text{ s.t. } 0 \leq \alpha_i \leq C \ \forall i \in [m] \tag{2}$$

with the Lagrange multipliers $\boldsymbol{\alpha} \in \mathbb{R}^m$ and kernel $Q_{ij} = \boldsymbol{Q}(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathbb{R}$ computed between all labeled nodes $i \in [m]$. and $j \in [m]$. The optimal dual solution may not be unique and we denote by $\mathcal{S}(\boldsymbol{Q})$ the set of $\boldsymbol{\alpha}$ solving $\mathrm{P}_1(\boldsymbol{Q})$. However, any $\boldsymbol{\alpha}^* \in \mathcal{S}(\boldsymbol{Q})$ corresponds to the same unique $\boldsymbol{\beta}^* = \sum_{i=1}^m y_i \alpha_i^* \Phi(\mathcal{G})_i$ minimizing Eq. (1) [Burges and Crisp, 1999]. Thus, the prediction of the SVM for a test node $t$ using the dual is given by $f_\theta^{SVM}(\mathbf{x}_t) = \sum_{i=1}^m y_i \alpha_i^* Q_{ti}$ for any $\boldsymbol{\alpha}^* \in \mathcal{S}(\boldsymbol{Q})$, where $Q_{ti}$ is the kernel between a test node $t$ and training node $i$. By choosing $\mathbf{Q}$ to be the NTK of a GNN $f_\theta$, the prediction equals $f_\theta(\mathcal{G})_t$ if the width of the GNN's hidden layers goes to infinity. Thus, a certificate for the SVM directly translates to a certificate for infinitely-wide GNNs. In the finite-width case, where the smallest GNN's layer width is $h$, the output difference between both methods can be bounded with high probability by $\mathcal{O}(\frac{\ln h}{\sqrt{h}})$ (the probability $\to 1$ as $h \to \infty$). Thus, the certificate translates to a high probability guarantee for sufficiently wide finite networks.

## 3 QPCert: Our certification framework

Poisoning a clean training graph $\mathcal{G}$ can be described as a bilevel problem where an adversary $\mathcal{A}$ tries to find a perturbed $\widetilde{\mathcal{G}} \in \mathcal{A}(\mathcal{G})$ that results in a model $\theta$ minimizing an attack objective $\mathcal{L}_{att}(\theta, \widetilde{\mathcal{G}})$:

$$\min_{\widetilde{\mathcal{G}}, \theta} \mathcal{L}_{att}(\theta, \widetilde{\mathcal{G}}) \quad \text{s. t.} \quad \widetilde{\mathcal{G}} \in \mathcal{A}(\mathcal{G}) \ \wedge \ \theta \in \arg\min_{\theta'} \mathcal{L}(\theta', \widetilde{\mathcal{G}}) \tag{3}$$

Eq. (3) is called an upper-level problem and $\min_{\theta'} \mathcal{L}(\theta', \widetilde{\mathcal{G}})$ the lower-level problem. Now, a sample-wise poisoning certificate can be obtained by solving Eq. (3) with an $\mathcal{L}_{att}(\theta, \widetilde{\mathcal{G}})$ chosen to describe if the prediction for a test node $t$ changes compared to the prediction of a model trained on the clean graph. However, this approach is challenging as even the simplest bilevel problems given by a linear lower-level problem embedded in an upper-level linear problem are NP-hard [Jeroslow, 1985]. Thus, in this section, we develop a general methodology to reformulate the bilevel (sample-wise) certification problem for kernelized SVMs as a mixed-integer linear program, making certification tractable through the use of highly efficient modern MILP solvers such as Gurobi [Gurobi Optimization, LLC, 2023] or CPLEX [Cplex, 2009]. Our approach can be divided into three steps: **(1)** The bilevel problem is reduced to a single-level problem by exploiting properties of the quadratic dual $\mathrm{P}_1(\boldsymbol{Q})$; **(2)** We model $\widetilde{\mathcal{G}} \in \mathcal{A}(\mathcal{G})$ by assuming a bound on the effect any $\widetilde{\mathcal{G}}$ can have on the elements of the kernel matrix $\boldsymbol{Q}$. This introduces a relaxation of the bilevel problem from Eq. (3) and allows us to fully express certification as a MILP; **(3)** In App. D, we choose the NTK of different GNNs as kernel and derive bounds on the kernel elements to use in the certificate. In the following, we present our certificate for binary classification where $y_i \in \{\pm 1\}$ $\forall i \in [n]$ and transductive learning, where the test node is already part of $\mathcal{G}$. We generalize it to a multi-class and inductive setting in App. E.

**A single-level reformulation.** Given an SVM $f_\theta^{SVM}$ trained on the clean graph $\mathcal{G}$, its class prediction for a test node $t$ is given by $\text{sgn}(\hat{p}_t) = \text{sgn}(f_\theta^{SVM}(\boldsymbol{x}_t))$. If for all $\widetilde{\mathcal{G}} \in \mathcal{A}(\mathcal{G})$ the sign of the prediction does not change if the SVM should be retrained on $\widetilde{\mathcal{G}}$, then we know that the prediction for $t$ is certifiably robust. Thus, the attack objective reads $\mathcal{L}_{att}(\theta, \widetilde{\mathcal{G}}) = \text{sgn}(\hat{p}_t) \sum_{i=1}^m y_i \alpha_i \widetilde{Q}_{ti}$, where $\widetilde{Q}_{ti}$ denotes the kernel computed between nodes $t$ and $i$ on the perturbed graph $\widetilde{\mathcal{G}}$, and indicates robustness if greater than zero. Now, notice that the perturbed graph $\widetilde{\mathcal{G}}$ only enters the training objective Eq. (2) through values of the kernel matrix $\widetilde{\boldsymbol{Q}} \in \mathbb{R}^{n \times n}$. Thus, we introduce the set $\mathcal{A}(\boldsymbol{Q})$ of all kernel matrices $\widetilde{\boldsymbol{Q}}$, constructable from $\widetilde{\mathcal{G}} \in \mathcal{A}(\mathcal{G})$. Furthermore, we denote with $\mathcal{S}(\widetilde{\boldsymbol{Q}})$ the optimal solution set to $\text{P}_1(\widetilde{\boldsymbol{Q}})$. As a result, we can rewrite Eq. (3) for kernelized SVMs as

$$\text{P}_2(\boldsymbol{Q}): \min_{\boldsymbol{\alpha}, \widetilde{\boldsymbol{Q}}} \text{sgn}(\hat{p}_t) \sum_{i=1}^m y_i \alpha_i \widetilde{Q}_{ti} \quad s.t. \quad \widetilde{\boldsymbol{Q}} \in \mathcal{A}(\boldsymbol{Q}) \ \wedge \ \boldsymbol{\alpha} \in \mathcal{S}(\widetilde{\boldsymbol{Q}}) \tag{4}$$

and certify robustness if the optimal solution to $\text{P}_2(\boldsymbol{Q})$ is greater than zero. Crucial in reformulating $\text{P}_2(\boldsymbol{Q})$ into a single-level problem are the Karush–Kuhn–Tucker (KKT) conditions of the lower-level problem $\text{P}_1(\widetilde{\boldsymbol{Q}})$. Concretely, the KKT conditions of $\text{P}_1(\widetilde{\boldsymbol{Q}})$ are

$$\forall i \in [m]: \quad \sum_{j=1}^m y_i y_j \alpha_j \widetilde{Q}_{ij} - 1 - u_i + v_i = 0 \qquad \textit{(Stationarity)} \tag{5}$$

$$\alpha_i \geq 0, \ C - \alpha_i \geq 0, \ u_i \geq 0, \ v_i \geq 0 \qquad \textit{(Primal and Dual feasibility)} \tag{6}$$

$$u_i \alpha_i = 0, \ v_i (C - \alpha_i) = 0 \qquad \textit{(Complementary slackness)} \tag{7}$$

where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$ are Lagrange multipliers. Now, we can state Proposition 1 proved in App. F.

**Proposition 1.** *Problem* $\text{P}_1(\widetilde{\boldsymbol{Q}})$ *given by Eq.* (2) *is convex and satisfies strong Slater's constraint. Consequently, the single-level optimization problem* $\text{P}_3(\boldsymbol{Q})$ *arising from* $\text{P}_2(\boldsymbol{Q})$ *by replacing* $\boldsymbol{\alpha} \in \mathcal{S}(\widetilde{\boldsymbol{Q}})$ *with Eqs.* (5) *to* (7) *has the same globally optimal solutions as* $\text{P}_2(\boldsymbol{Q})$.

**A mixed-integer linear reformulation.** The computational bottleneck of $\text{P}_3(\widetilde{\boldsymbol{Q}})$ are the non-linear product terms between continuous variables in the attack objective as well as in Eqs. (5) and (7), making $\text{P}_3(\widetilde{\boldsymbol{Q}})$ a bilinear problem. To transform $\text{P}_3(\widetilde{\boldsymbol{Q}})$ to a MILP, first, the complementary slackness constraints can be linearized by recognizing that they have a combinatorial structure. In particular, $u_i = 0$ if $\alpha_i > 0$ and $v_i = 0$ if $\alpha_i < C$. Thus, introducing binary integer variables $\mathbf{s}$ and $\mathbf{t} \in \{0,1\}^m$, we reformulate the constraints in Eq. (7) with big-$M$ constraints as

$$\forall i \in [m]: \quad u_i \leq M_{u_i} s_i, \ \alpha_i \leq C(1 - s_i), v_i \leq M_{v_i} t_i, \ \alpha_i \geq C t_i, \quad s_i, t_i \in \{0,1\} \tag{8}$$

where $M_{u_i}$ and $M_{v_i}$ are positive constants. In general, verifying that a certain choice of big-$M$s results in a valid (mixed-integer) reformulation of the complementary constraints Eq. (7), i.e., such that no optimal solution to the original bilevel problem is cut off, is at least as hard as solving the bilevel problem itself [Kleinert et al., 2020]. This is problematic as heuristic choices can lead to suboptimal solutions to the original problem [Pineda and Morales, 2019]. However, additional structure provided by $\text{P}_1(\widetilde{\boldsymbol{Q}})$ and $\text{P}_3(\boldsymbol{Q})$ together with insights into the optimal solution set allow us to derive valid and small $M_{u_i}$ and $M_{v_i}$ for all $i \in [m]$.

Concretely, the adversary $\mathcal{A}$ can only make a bounded change to $\mathcal{G}$. Thus, the element-wise difference of any $\widetilde{\boldsymbol{Q}} \in \mathcal{A}(\boldsymbol{Q})$ to $\boldsymbol{Q}$ will be bounded. As a result, there exist element-wise upper and lower bounds $\widetilde{Q}_{ij}^L \leq \widetilde{Q}_{ij} \leq \widetilde{Q}_{ij}^U$ for all $i, j \in [m] \cup \{t\}$ and valid for any $\widetilde{\boldsymbol{Q}} \in \mathcal{A}(\boldsymbol{Q})$. In App. D we derive concrete lower and upper bounds for the NTKs corresponding to different common GNNs. This, together with $0 \leq \alpha_i \leq C$, allows us to lower and upper bound $\sum_{j=1}^m y_i y_j \alpha_j \widetilde{Q}_{ij}$ in Eq. (5). Now, given an optimal solution $(\boldsymbol{\alpha}^*, \widetilde{\boldsymbol{Q}}^*, \mathbf{u}^*, \mathbf{v}^*)$ to $\text{P}_3(\boldsymbol{Q})$, observe that either $u_i^*$ or $v_i^*$ are zero, or can be freely varied between any positive values as long as Eq. (5) is satisfied without changing the objective value or any other variable. As a result, one can use the lower and upper bounds on $\sum_{j=1}^m y_i y_j \alpha_j \widetilde{Q}_{ij}$ to find the minimal value range necessary and sufficient for $u_i$ and $v_i$, such that Eq. (5) can always be satisfied for any $\boldsymbol{\alpha}^*$ and $\widetilde{\boldsymbol{Q}}^*$. Consequently, only redundant solutions regarding large $u_i^*$ and $v_i^*$ will be cut off and the optimal solution value stays the same as for $\text{P}_3(\boldsymbol{Q})$, not affecting the certification. The exact $M_{u_i}$ and $M_{v_i}$ depend on the signs of the involved $y_i$ and $y_j$ and are derived in App. G.

Now, the remaining non-linearities come from the product terms $\alpha_i \widetilde{Q}_{ij}$. We approach this by first introducing new variables $Z_{ij}$ for all $i, j \in [m] \cup \{t\}$ and set $Z_{ij} = \alpha_j \widetilde{Q}_{ij}$. Then, we replace all product terms $\alpha_j \widetilde{Q}_{ij}$ in Eq. (5) and in the objective in Eq. (4) with $Z_{ij}$. This alone has not changed the fact that the problem is bilinear, only that the bilinear terms have now moved to the definition of $Z_{ij}$. However, we have access to lower and upper bounds on $\widetilde{\boldsymbol{Q}}_{ij}$. Thus, replacing $Z_{ij} = \alpha_j \widetilde{Q}_{ij}$ with linear constraints $Z_{ij} \leq \alpha_j \widetilde{Q}^U_{ij}$ and $Z_{ij} \geq \alpha_j \widetilde{Q}^L_{ij}$ results in a relaxation to $\mathrm{P}_3(\boldsymbol{Q})$. This resolved all non-linearities and we can write the following theroem.

**Theorem 1** (MILP Formulation)**.** *Node $t$ is certifiably robust against adversary $\mathcal{A}$ if the optimal solution to the following MILP denoted by $P(\boldsymbol{Q})$ is greater zero*

$$
\begin{aligned}
&\min_{\boldsymbol{\alpha},\mathbf{u},\mathbf{v},\mathbf{s},\mathbf{t},\mathbf{Z}} \operatorname{sgn}(\hat{p}_t) \sum_{i=1}^{m} y_i Z_{ti} \quad \textit{s.t.} \\
&\qquad Z_{ij} \leq \alpha_j \widetilde{Q}^U_{ij},\ Z_{ij} \geq \alpha_j \widetilde{Q}^L_{ij} \qquad \forall i \in [m] \cup \{t\}, j \in [m] \\
&\forall i \in [m]: \quad \sum_{j=1}^{m} y_i y_j Z_{ij} - 1 - u_i + v_i = 0, u_i \leq M_u s_i,\ \alpha_i \leq C(1 - s_i),\ s_i \in \{0,1\}, \\
&\qquad \alpha_i \geq 0,\ C - \alpha_i \geq 0,\ u_i \geq 0,\ v_i \geq 0, v_i \leq M_v t_i,\ \alpha_i \geq C t_i,\ t_i \in \{0,1\}.
\end{aligned}
$$

$\mathrm{P}(\boldsymbol{Q})$ includes backdoor attacks through the bounds $\widetilde{Q}^L_{tj}$ and $\widetilde{Q}^U_{tj}$ for all $j \in [m]$. On computational aspects, $\mathrm{P}(\boldsymbol{Q})$ involves $(m+1)^2 + 5m$ variables out of which $2m$ are binary. Thus, the number of binary variables that mainly define the time complexity of MILP-solvers scales with the number of labeled samples. Finally, we require the element-wise bounds on the NTKs corresponding to the GNNs to apply QPCert. We derive the bounds for perturbation models $\mathcal{B}_p(\mathbf{x})$ considering $p = \infty$ and $p = 2$, and prove that they are tight in the worst-case, in App. D.

## 4 Experimental results

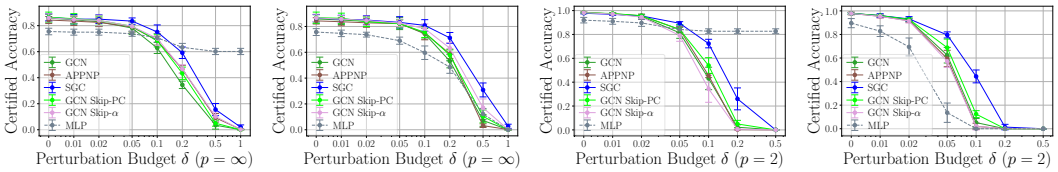
In this section, we present the effectiveness of QPCert in certifying different GNNs using their NTKs against node feature poisoning and backdoor attacks. In App. I, we provide insights into the role of graph data and architectural components. Code is at `https://github.com/saper0/qpcert`.

**Dataset.** We use graphs generated from Contextual Stochastic Block Models (CSBM) [Deshpande et al., 2018] and the real graph dataset *Cora-ML* [Bojchevski and Günnemann, 2018], where we generate continuous 384-dim. embeddings of the abstracts with a sentence transformer[2]. Furthermore, from Cora-ML's 7 classes, we extract the subgraph defined by the two largest classes and call the resulting binary-classification dataset *Cora-MLb*. We give dataset statistics and the details of graph generation using CSBM in H.1. For the CSBM graphs we sample 200 nodes and choose 40 nodes per class for training, leaving 120 unlabeled nodes. For Cora-MLb, we choose 10 nodes per class for training, leaving 1215 unlabeled nodes, and for Cora-ML, 20 train nodes per class. All results are averaged over 5 seeds (Cora-ML: 3 seeds) and reported with the standard deviation.

**GNNs.** We evaluate GCN, SGC, (A)PPNP, MLP, and the skip connection variants GCN Skip-$\alpha$ and GCN Skip-PC (see App. A). All results concern the infinite-width limit and thus, are obtained by training an SVM with the respective GNN's NTK and, if applicable, applying QPCert using Gurobi to solve the MILP from Theorem 1. Architectural details with the hyperparameters are provided in App. H.2. We fix the hidden layers to $L = 1$, and the results for $L = \{2, 4\}$ are provided in App. K.2.

**Adversarial evaluation settings.** We categorize four settings of interest. **(1)** *Poison Labeled (PL)*: The adversary $\mathcal{A}$ can potentially poison the labeled data $\mathcal{V}_L$. **(2)** *Poison Unlabeled (PU):* Especially interesting in a semi-supervised setting is the scenario when $\mathcal{A}$ can poison the unlabeled data $\mathcal{V}_U$, while the labeled data, usually representing a small curated set of high quality, is known to be clean [Shejwalkar et al., 2023]. **(3)** *Backdoor Labeled (BL):* Like (1) but the test node is also controlled by $\mathcal{A}$. **(4)** *Backdoor Unlabeled (BU):* Like (2) but again, the test node is controlled by $\mathcal{A}$. Settings (1) and (2) are evaluated transductively, i.e. on the unlabeled nodes $\mathcal{V}_U$ already known at training time. Note that this means for (2) that some test nodes may be corrupted. For the backdoor attack settings (3) and (4) the test node is removed from the graph during training and

[2] all-MiniLM-L6-v2 from `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

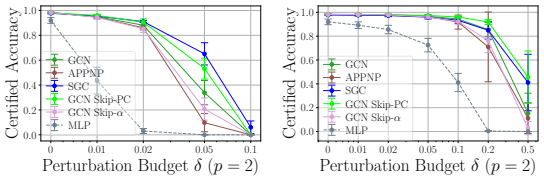



(a) CSBM: $PU$ (b) CSBM: $BU$ (c) Cora-MLb: $PU$ (d) Cora-MLb: $BU$

Figure 3: Certifiable robustness for different (G)NNs in Poisoning Unlabeled $(PU)$ and Backdoor Unlabeled $(BU)$ setting. $p_{adv} = 0.2$ for CSBM and $p_{adv} = 0.1$ for Cora-MLb.

added inductively at test time. The size of the untrusted potential adversarial node set $\mathcal{U}$ is set in percentage $p_{adv} \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$ of the scenario-dependent attackable node set and resampled for each seed. We consider node feature perturbations $\mathcal{B}_p(\mathbf{x})$ with $p = \infty$ and $p = 2$. In the case of CSBM, $\delta$ is set in percentage of $2\boldsymbol{\mu}$ of the underlying distribution, and for real data to absolute values. Our main evaluation metric is *certified accuracy*, referring to the percentage of correctly classified nodes without attack that are provably robust against data poisoning / backdoor attacks of the assumed adversary $\mathcal{A}$. We note as we are the first work to study *white-box* certificates for clean-label attacks on node features in graphs in general, there is no baseline prior work.

**Non-trivial certificates or On the importance of graph information.** We evaluate the effectiveness of our certificates in providing non-trivial robustness guarantees. Consider the $PL$ setting where $\mathcal{A}$ can poison *all* labeled nodes ($p_{adv} = 1$) for which a trivial certificate would return $0\%$ certified accuracy. Fig. 2a proves that QPcert returns non-trivial guarantees. Further, it highlights an interesting insight: All GNNs have *significantly better* worst-case robustness behavior than the certified accuracy of an MLP. Thus, leveraging the graph connectivity, significantly improves their certified accuracy, even when faced with perturbations on all labeled nodes. App. K.1 shows a similar result for CSBM establishing that this behavior is *not* dataset-specific. In Fig. 2b we show that this observation stays consistent also for other $p_{adv}$.




(a) $PL$, $p_{adv} = 1$ (b) $PL$, $p_{adv} = 0.1$

Figure 2: Poison Labeled $(PL)$: Cora-MLb

In Fig. 3 we evaluate the poison unlabeled $(PU)$ and backdoor unlabeled $(BU)$ settings for CSBM and Cora-MLb with $p_{adv} = 0.2$ and $p_{adv} = 0.1$, respectively. When poisoning only unlabeled data, the MLP is not affected by the adversary providing a good baseline for our certificate to study GNNs. Again, QPCert provides non-trivial certified robustness beyond the MLP baseline. All GNNs show certified accuracy beyond the one of an MLP persisting until relatively strong perturbations ($\delta \geq 0.2$), with a similar picture for Cora-MLb in Fig. 3c and investigated in depth for all GNNs in App. K.1. Concerning backdoor attacks on unlabeled nodes, Figs. 3b and 3d show that GNNs show significantly better certified robustness than an MLP, even if MLP training is not affected by $\mathcal{A}$. We observe similar results for a $BL$ setting in App. K.1 and App. L.1. These results show that leveraging graph information significantly improves certified accuracy across all considered attack settings. For Cora-ML results, we refer to App. M. Note that a comparison across architecture can be affected by the certificate's tightness and we hypothesize that the high worst-case robustness of SGC may be due to the certificate being tighter (Theorem 3). However, this still allows us to derive architectural insights for a specific GNN as presented in App. I. We explore the tightness of QPCert experimentally in App. J, leveraging a differentiable implementation of quadratic programming.

**Conclusion.** While we focus on (G)NNs for graph data, our framework enables white-box poisoning certification of NNs on any data domain. Further, it allows for certifying general kernelized SVMs for arbitrary kernel choices if respective kernel bounds as in App. D are derived. To the best of our knowledge, this makes our work not only the first white-box poisoning framework for NNs, but also the first for kernelized SVMs. Moreover, the reformulation of the bilevel problem to MILP is directly applicable to any quadratic program that satisfies strong Slater's constraint and certain bounds on the involved variables, hence the name QPCert. Thus extensions to certify quadratic programming layers in NN [Amos and Kolter, 2017] or other quadratic learners are thinkable. Therefore, we believe that our work opens up numerous new avenues of research in the area of provable robustness.

## Acknowledgment

The authors want to thank Yan Scholten and Pascal Esser for the interesting discussions and helpful feedback on the manuscript. This paper has been supported by the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the German Federal Ministry of Education and Research; by the German Research Foundation, grant GU 1409/4-1; as well as by the TUM Georg Nemetschek Institute Artificial Intelligence for the Built World.

## References

B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

A. Bambade, F. Schramm, A. Taylor, and J. Carpentier. QPLayer: efficient differentiation of convex quadratic optimization. 2023. URL `https://inria.hal.science/hal-04133055`.

D. Banerjee, A. Singh, and G. Singh. Interpreting robustness proofs of deep neural networks. In *International Conference on Learning Representations (ICLR)*, 2024.

S. Bian, X. Ouyang, Z. FAN, and P. Koutris. Naive bayes classifiers over missing data: Decision and poisoning. In *International Conference on Machine Learning (ICML)*, 2024.

B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning (ICML)*, 2012.

A. Bojchevski and S. Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

C. J. C. Burges and D. Crisp. Uniqueness of the svm solution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1999.

M. Charikar, J. Steinhardt, and G. Valiant. Learning from untrusted data. In *Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2017.

M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning (ICML)*, 2020.

R. Chen, Z. Li, J. Li, J. Yan, and C. Wu. On collective robustness of bagging against data poisoning. In *International Conference on Machine Learning*, pages 3299–3319. PMLR, 2022.

Y. Chen, W. Huang, L. Nguyen, and T.-W. Weng. On the equivalence between neural network and support vector machine. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

J. Cohen, E. Rosenfeld, and Z. Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)*, 2019.

I. I. Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 2009.

F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, 2020.

S. Dempe and J. Dutta. Is bilevel programming a special case of a mathematical program with complementarity constraints? *Math. Program.*, 131:37–48, 2012. doi: https://doi.org/10.1007/s10107-010-0342-1.

Y. Deshpande, S. Sen, A. Montanari, and E. Mossel. Contextual stochastic block models. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

S. Drews, A. Albarghouthi, and L. D'Antoni. Proving data-poisoning robustness in decision trees. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1083–1097, 2020.

J. Gasteiger, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2019.

M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 45(02):1563–1580, 2023. ISSN 1939-3539.

L. Gosch, D. Sturm, S. Geisler, and S. Günnemann. Revisiting robustness in graph machine learning. In *International Conference on Learning Representations (ICLR)*, 2023.

S. Günnemann. Graph neural networks: Adversarial robustness. In *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 149–176. Springer Singapore, 2022.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

S. Hong, N. Carlini, and A. Kurakin. Diffusion denoising as a certified defense against clean-label poisoning. *arXiv preprint arXiv:2403.11981*, 2024.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE Symposium on Security and Privacy (SP)*, 2018.

R. G. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical programming*, 1985.

J. Jia, X. Cao, and N. Z. Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7961–7969, 2021.

J. Jia, Y. Liu, X. Cao, and N. Z. Gong. Certified robustness of nearest neighbors against data poisoning and backdoor attacks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

T. Kleinert, M. Labbé, F. Plein, and M. Schmidt. There's no free lunch: On the hardness of choosing a correct big-m in bilevel optimization. *Operations Research*, 68 (6):1716–1721, 2020. doi: https://doi.org/10.1287/opre.2019.1944.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning (ICML)*, 2017.

P. W. Koh, J. Steinhardt, and P. Liang. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, 111(1):1–47, 2022.

Y. Lai, Y. Zhu, B. Pan, and K. Zhou. Node-aware bi-smoothing: Certified robustness against graph injection attacks, 2024.

A. Levine and S. Feizi. Deep partition aggregation: Provable defense against general poisoning attacks. In *International Conference on Learning Representations (ICLR)*, 2021.

L. Li, T. Xie, and B. Li. Sok: Certified robustness for deep neural networks. In *IEEE Symposium on Security and Privacy, (SP)*, 2023.

V. Lingam, M. S. Akhondzadeh, and A. Bojchevski. Rethinking label poisoning for GNNs: Pitfalls and attacks. In *International Conference on Learning Representations (ICLR)*, 2024.

S. Liu, A. C. Cullen, P. Montague, S. M. Erfani, and B. I. P. Rubinstein. Enhancing the antidote: Improved pointwise certifications against poisoning attacks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

X. Ma, Z. Wang, and W. Liu. On the tradeoff between robustness and fairness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Y. Ma, X. Zhu, and J. Hsu. Data poisoning against differentially-private learners: Attacks and defenses. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Y. Mao, M. N. Mueller, M. Fischer, and M. Vechev. Understanding certified training with interval bound propagation. In *International Conference on Learning Representations (ICLR)*, 2024.

S. Mei and X. Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

A. Meyer, A. Albarghouthi, and L. D'Antoni. Certifying robustness to programmable data bias in decision trees. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning, second edition*. The MIT Press, 2018.

L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 27–38, 2017.

S. Pineda and J. M. Morales. Solving linear bilevel problems using big-ms: Not all that glitters is gold. *IEEE Transactions on Power Systems*, 34(3):2469–2471, 2019. doi: 10.1109/TPWRS.2019.2892607.

K. Rezaei, K. Banihashem, A. Chegini, and S. Feizi. Run-off election: Improved provable defense against data poisoning attacks. In *International Conference on Machine Learning (ICML)*, 2023.

E. Rosenfeld, E. Winston, P. Ravikumar, and Z. Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, pages 8230–8241. PMLR, 2020.

M. Sabanayagam, P. Esser, and D. Ghoshdastidar. Analysis of convolutions, non-linearity and depth in graph neural networks using neural tangent kernel. *Transactions on Machine Learning Research (TMLR)*, 2023.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, Sep. 2008.

V. Shejwalkar, L. Lyu, and A. Houmansadr. The perils of learning from unlabeled data: Backdoor attacks on semi-supervised learning. In *International Conference on Computer Vision (ICCV)*, 2023.

J. M. Steele. *The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities*. Cambridge University Press, 2004.

J. Steinhardt, P. W. W. Koh, and P. S. Liang. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.

O. Suciu, R. Mărginean, Y. Kaya, H. Daumé, and T. Dumitraş. When does machine learning fail? generalized transferability for evasion and poisoning attacks. In *USENIX Conference on Security Symposium (SEC)*, 2018.

V. Tjeng, K. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming, 2019.

A. Wan, E. Wallace, S. Shen, and D. Klein. Poisoning language models during instruction tuning. In *International Conference on Machine Learning (ICML)*, 2023.

B. Wang, X. Cao, J. jia, and N. Z. Gong. On certifying robustness against backdoor attacks via randomized smoothing, 2020.

W. Wang, A. Levine, and S. Feizi. Improved certified defenses against data poisoning with (deterministic) finite aggregation. In *International Conference on Machine Learning (ICML)*, 2022.

M. Weber, X. Xu, B. Karlaš, C. Zhang, and B. Li. Rab: Provable robustness against backdoor attacks. In *IEEE Symposium on Security and Privacy (SP)*, 2023.

F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning (ICML)*, 2019.

H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning (ICML)*, pages 1689–1698. PMLR, 2015.

C. Xie, Y. Long, P.-Y. Chen, Q. Li, S. Koyejo, and B. Li. Unraveling the connections between privacy and certified robustness in federated learning against poisoning attacks. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023.

Y. Zhang, A. Albarghouthi, and L. D'Antoni. Bagflip: A certified defense against data poisoning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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

# A Architecture definitions

We consider GNNs as functions $f_\theta$ with (learnable) parameters $\theta \in \mathbb{R}^q$ and $L$ number of layers taking the graph $\mathcal{G} = (\boldsymbol{S}, \boldsymbol{X})$ as input and outputs a prediction for each node. We consider linear output layers with weights $\boldsymbol{W}^{L+1}$ and denote by $f_\theta(\mathcal{G})_i \in \mathbb{R}^K$ the (unnormalized) logit output associated to node $i$. In the following, we formally define the (G)NNs such as MLP, GCN [Kipf and Welling, 2017], SGC [Wu et al., 2019] and (A)PPNP [Gasteiger et al., 2019] considered in our study.

**Def. 1** (MLP)**.** *The $L$-layer Multi-Layer Perceptron is defined as $f_\theta(\mathcal{G})_i = f_\theta^{MLP}(\boldsymbol{x}_i) = \boldsymbol{W}^{L+1}\phi_\theta^{(L)}(\boldsymbol{x}_i)$. With $\phi_\theta^l(\boldsymbol{x}_i) = \sigma(\boldsymbol{W}^{(l)}\phi_\theta^{(l-1)}(\boldsymbol{x}_i) + \boldsymbol{b}^{(l)})$ and $\phi_\theta^{(0)}(\boldsymbol{x}_i) = \boldsymbol{x}_i$. $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_{l+1}\times d_l}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{d_l}$ are the weights/biases of the $l$-th layer with $d_0 = d$ and $d_{L+1} = K$. $\sigma(\cdot)$ is an element-wise activation function. If not mentioned otherwise, we choose $\sigma(z) = ReLU(z) = \max\{0, z\}$.*

**Def. 2** (GCN & SGC)**.** *A Graph Convolution Network $f_\theta^{GCN}(\mathcal{G})$ [Kipf and Welling, 2017] of depth $L$ is defined as $f_\theta(\mathcal{G}) = \phi_\theta^{(L+1)}(\mathcal{G})$ with $\phi_\theta^{(l)}(\mathcal{G}) = \mathbf{S}\sigma(\phi_\theta^{(l-1)}(\mathcal{G}))\boldsymbol{W}^{(l)}$ and $\phi_\theta^{(1)}(\mathcal{G}) = \mathbf{S}\mathbf{X}\mathbf{W}^{(1)}$. $\mathbf{W}^{(l)} \in \mathbb{R}^{d_{l-1}\times d_l}$ are the $l$-th layer weights, $d_0 = d$, $d_{L+1} = K$, and $\sigma(z) = ReLU(z)$ applied element-wise. A Simplified Graph Convolution Network $f_\theta^{SGC}(\mathcal{G})$ [Wu et al., 2019] is a GCN with linear $\sigma(z) = z$.*

**Def. 3** ((A)PPNP)**.** *The Personalized Propagation of Neural Predictions Network $f_\theta^{PPNP}(\mathcal{G})$ Gasteiger et al. [2019] is defined as $f_\theta(\mathcal{G}) = \mathbf{P}\mathbf{H}$ where $\mathbf{H}_{i,:} = f_\theta^{MLP}(\mathbf{x}_i)$ and $\mathbf{P} = \alpha(\mathbf{I}_n - (1-\alpha)\mathbf{S})^{-1}$. The Approximate PPNP is defined with $\mathbf{P} = (1-\alpha)^K\mathbf{S}^K + \alpha\sum_{i=0}^{K-1}(1-\alpha)^i\mathbf{S}^i$ where $\alpha \in [0, 1]$ and $K \in \mathbb{N}$ is a fixed constant.*

Along with the GNNs presented in Definitions 1 to 3, we consider two variants of popular skip connections in GNNs as given a name in Sabanayagam et al. [2023]: Skip-PC (pre-convolution), where the skip is added to the features before applying convolution [Kipf and Welling, 2017]; and Skip-$\alpha$, which adds the features to each layer without convolving with $\mathbf{S}$ [Chen et al., 2020]. To facilitate skip connections, we need to enforce constant layer size, that is, $d_i = d_{i-1}$. Therefore, the input layer is transformed using a random matrix $\mathbf{W}$ to $\mathbf{H}_0 := \mathbf{X}\mathbf{W}$ of size $n \times h$ where $\mathbf{W}_{ij} \sim \mathcal{N}(0, 1)$ and $h$ is the hidden layer size. Let $\mathbf{H}_i$ be the output of layer $i$ using which we formally define the skip connections as follows.

**Def. 4** (Skip-PC)**.** *In a Skip-PC (pre-convolution) network, the transformed input $\mathbf{H}_0$ is added to the hidden layers before applying the graph convolution $\mathbf{S}$, that is, $\forall i \in [L], \phi_\theta^{(l)}(\mathcal{G}) = \mathbf{S}\left(\sigma(\phi_\theta^{(l-1)}(\mathcal{G})) + \sigma_s(\mathbf{H}_0)\right)\boldsymbol{W}^{(l)}$, where $\sigma_s(z)$ can be linear or ReLU.*

Skip-PC definition deviates from Kipf and Welling [2017] because we skip to the input layer instead of the previous one. We define Skip-$\alpha$ as defined in Sabanayagam et al. [2023] similar to Chen et al. [2020].

**Def. 5** (Skip-$\alpha$)**.** *Given an interpolation coefficient $\alpha \in (0, 1)$, a Skip-$\alpha$ network is defined such that the transformed input $\mathbf{H}_0$ and the hidden layer are interpolated linearly, that is, $\phi_\theta^{(l)}(\mathcal{G}) = \left((1-\alpha)\,\mathbf{S}\phi_\theta^{(l-1)}(\mathcal{G}) + \alpha\sigma_s(\mathbf{H}_0)\right)\mathbf{W}_i\ \forall i \in [L]$, where $\sigma_s(z)$ can be linear or ReLU.*

# B Derivation of NTK for (A)PPNP

We derive the closed-form NTK expression for (A)PPNP $f_\theta(\mathcal{G})$ [Gasteiger et al., 2019] in this section. The learnable parameters $\theta$ are only part of $\mathbf{H}$. In practice, $\mathbf{H} = \text{ReLU}(\mathbf{X}\mathbf{W}_1 + \mathbf{B}_1)\mathbf{W}_2 + \mathbf{B}_2$ where node features $\mathbf{X}$, $\theta = \{\mathbf{W}_1 \in \mathbb{R}^{d\times h}, \mathbf{W}_2 \in \mathbb{R}^{h\times K}, \mathbf{B}_1 \in \mathbb{R}^{n\times h}, \mathbf{B}_2 \in \mathbb{R}^{n\times K}\}$. Note that in the actual implementation of the MLP, $\mathbf{B}_1$ is a vector and we consider it to be a matrix by having the same columns so that we can do matrix operations easily. Same for $\mathbf{B}_2$ as well. We give the full architecture with NTK parameterization in the following,

$$f_\theta(\mathcal{G}) = \mathbf{P}\left(\frac{c_\sigma}{\sqrt{h}}\sigma(\mathbf{X}\mathbf{W}_1 + \mathbf{B}_1)\mathbf{W}_2 + \mathbf{B}_2\right)$$

where $h \to \infty$ and all parameters in $\theta$ are initialized as standard Gaussian $\mathcal{N}(0, 1)$. $c_\sigma$ is a constant to preserve the input norm [Sabanayagam et al., 2023]. We derive for $K = 1$ as all the outputs are equivalent in expectation. The NTK between nodes $i$ and $j$ is $\mathop{\mathbb{E}}_{\theta\sim\mathcal{N}(0,1)}[\langle\nabla_\theta f_\theta(\mathcal{G})_i, \nabla_\theta f_\theta(\mathcal{G})_j\rangle]$.

Hence, we first write down the gradients for node $i$ following [Arora et al., 2019, Sabanayagam et al., 2023]:

$$\begin{aligned}
\frac{\partial f_\theta(\mathcal{G})_i}{\partial \mathbf{W}_2} &= \frac{c_\sigma}{\sqrt{h}}(\mathbf{P}_i\sigma(\mathcal{G}_1))^T && ;\mathcal{G}_1 = \mathbf{X}\mathbf{W}_1 + \mathbf{B}_1 \\
\frac{\partial f_\theta(\mathcal{G})_i}{\partial \mathbf{B}_2} &= (\mathbf{P}_i)^T\mathbf{1}_n \\
\frac{\partial f_\theta(\mathcal{G})_i}{\partial \mathbf{W}_1} &= \frac{c_\sigma}{\sqrt{h}}\mathbf{X}^T(\mathbf{P}_i^T\mathbf{1}_n\mathbf{W}_2^T \odot \dot{\sigma}(\mathcal{G}_1)) \\
\frac{\partial f_\theta(\mathcal{G})_i}{\partial \mathbf{B}_1} &= \frac{c_\sigma}{\sqrt{h}}\mathbf{P}_i^T\mathbf{1}_n\mathbf{W}_2^T \odot \dot{\sigma}(\mathcal{G}_1)
\end{aligned}$$

We note that $\mathbf{B}_2$ has only one learnable parameter for $K = 1$, but is represented as a vector of size $n$ with all entries the same. Hence, the derivative is simply adding all entries of $\mathbf{P}_i$. First, we compute the covariance between nodes $i$ and $j$ in $\mathcal{G}_1$.

$$\mathbb{E}\left[(G_1)_{ik}(G_1)_{jk'}\right] = \mathbb{E}\left[(\mathbf{X}\mathbf{W}_1 + \mathbf{B}_1)_{ik}(\mathbf{X}\mathbf{W}_1 + \mathbf{B}_1)_{jk'}\right]$$

Since the expectation is over $\mathbf{W}_1$ and $\mathbf{B}_1$ and all entries are $\sim \mathcal{N}(0, 1)$, and i.i.d, the cross terms will be 0 in expectation. Also, for $k \neq k'$, it is 0. Therefore, it gets simplified to

$$\begin{aligned}
\mathbb{E}\left[(G_1)_{ik}(G_1)_{jk}\right] &= \mathbb{E}\left[\mathbf{X}_i\mathbf{W}_1\mathbf{W}_1^T\mathbf{X}_j^T + \left(\mathbf{B}_1\mathbf{B}_1^T\right)_{ij}\right] \\
&= \left(\mathbf{X}\mathbf{X}^T\right)_{ij} + 1 = (\Sigma_1)_{ij} \qquad (9)
\end{aligned}$$

Thus, $\mathbf{\Sigma}_1 = \mathbf{X}\mathbf{X}^T + \mathbf{1}_{n\times n}$ and let $(E_1)_{ij} = \mathbb{E}\left[\sigma(\mathcal{G}_1)_i\sigma(\mathcal{G}_1)_j^T\right]$ and $\left(\dot{E}_1\right)_{ij} = \mathbb{E}\left[\dot{\sigma}(\mathcal{G}_1)_i\dot{\sigma}(\mathcal{G}_1)_j^T\right]$ computed using the definitions in Theorem 2 for ReLU non-linearity. Now, we can compute the NTK for each parameter matrix and then sum it up to get the final kernel.

$$\begin{aligned}
\left\langle \frac{\partial f_\theta(\mathcal{G})_i}{\partial \mathbf{W}_2}, \frac{\partial f_\theta(\mathcal{G})_j}{\partial \mathbf{W}_2} \right\rangle &= \frac{c_\sigma^2}{h}\mathbf{P}_i\sigma(\mathcal{G}_1)\sigma(\mathcal{G}_1)^T\mathbf{P}_j^T \\
&\overset{h\to\infty}{=} c_\sigma^2\mathbf{P}_i\mathbb{E}\left[\sigma(\mathcal{G}_1)\sigma(\mathcal{G}_1)^T\right]\mathbf{P}_j^T = c_\sigma^2\mathbf{P}_i\mathbf{E}_1\mathbf{P}_j^T \qquad (10)
\end{aligned}$$

$$\left\langle \frac{\partial f_\theta(\mathcal{G})_i}{\partial \mathbf{B}_2}, \frac{\partial f_\theta(\mathcal{G})_j}{\partial \mathbf{B}_2} \right\rangle = \mathbf{P}_i\mathbf{1}_{n\times n}\mathbf{P}_j^T \qquad (11)$$

$$\left\langle \frac{\partial f_\theta(\mathcal{G})_i}{\partial \mathbf{B}_1}, \frac{\partial f_\theta(\mathcal{G})_j}{\partial \mathbf{B}_1} \right\rangle \overset{h\to\infty}{=} c_\sigma^2\mathbf{P}_i(\mathbb{E}\left[\dot{\sigma}(\mathcal{G}_1)\dot{\sigma}(\mathcal{G}_1)\right])\mathbf{P}_j^T = c_\sigma^2\mathbf{P}_i\dot{\mathbf{E}}_1\mathbf{P}_j^T \qquad (12)$$

$$\begin{aligned}
\left\langle \frac{\partial f_\theta(\mathcal{G})_i}{\partial \mathbf{W}_1}, \frac{\partial f_\theta(\mathcal{G})_j}{\partial \mathbf{W}_1} \right\rangle &= \frac{c_\sigma^2}{h}\sum_{p,q}^{f,h}(\mathbf{X}^T(\mathbf{P}_i^T\mathbf{1}_n\mathbf{W}_2^T \odot \dot{\sigma}(\mathcal{G}_1)))_{pq}(\mathbf{X}^T(\mathbf{P}_j^T\mathbf{1}_n\mathbf{W}_2^T \odot \dot{\sigma}(\mathcal{G}_1)))_{pq} \\
&= \frac{c_\sigma^2}{h}\sum_{p=1}^{d}\sum_{q=1}^{h}\Big[\sum_{a=1}^{n}(\mathbf{X}^T)_{pa}(\mathbf{P}_i^T\mathbf{W}_2^T)_{aq}\dot{\sigma}(\mathcal{G}_1)_{aq} \\
&\qquad \sum_{b=1}^{n}(\mathbf{X}^T)_{pb}(\mathbf{P}_j^T\mathbf{W}_2^T)_{bq}\dot{\sigma}(\mathcal{G}_1)_{bq}\Big] \\
&\overset{h\to\infty}{=} c_\sigma^2\sum_{a=1,b=1}^{n,n}(\mathbf{X}\mathbf{X}^T)_{ab}\mathbf{P}_{ia}(\mathbf{P}^T)_{bj}\mathbb{E}\left[\dot{\sigma}(\mathcal{G}_1)\dot{\sigma}(\mathcal{G}_1)\right]_{ab} \\
&= c_\sigma^2\mathbf{P}_i(\mathbf{X}\mathbf{X}^T \odot \mathbb{E}\left[\dot{\sigma}(\mathcal{G}_1)\dot{\sigma}(\mathcal{G}_1)\right])\mathbf{P}_j^T = c_\sigma^2\mathbf{P}_i(\mathbf{X}\mathbf{X}^T \odot \dot{\mathbf{E}})_1\mathbf{P}_j^T \qquad (13)
\end{aligned}$$

Finally, the NTK matrix for the considered (A)PPNP is sum of Eqs. (10) to (13) as shown below.

$$\begin{aligned}\mathbf{Q} &= c_\sigma^2 \left( \mathbf{P}\mathbf{E}_1\mathbf{P}^T + \mathbf{P}\mathbf{1}_{n\times n}\mathbf{P}^T + \mathbf{P}\dot{\mathbf{E}}_1\mathbf{P} + \mathbf{P}\left(\mathbf{X}\mathbf{X}^T \odot \dot{\mathbf{E}}_1\right)\mathbf{P}^T \right) \\ &= c_\sigma^2 \left( \mathbf{P}\left(\mathbf{E}_1 + \mathbf{1}_{n\times n}\right)\mathbf{P}^T + \mathbf{P}\left(\left(\mathbf{X}\mathbf{X}^T + \mathbf{1}_{n\times n}\right) \odot \dot{\mathbf{E}}_1\right)\mathbf{P}^T \right) \\ &= c_\sigma^2 \left( \mathbf{P}\left(\mathbf{E}_1 + \mathbf{1}_{n\times n}\right)\mathbf{P}^T + \mathbf{P}\left(\mathbf{\Sigma}_1 \odot \dot{\mathbf{E}}_1\right)\mathbf{P}^T \right) \end{aligned} \tag{14}$$

Note that $c_\sigma$ is a constant, and it only scales the NTK, so we set it to 1 in our experiments. Since we use a linear output layer without bias term at the end, that is, $\mathbf{B}_2 = 0$, the NTK we use for our experiments is reduced to

$$\mathbf{Q} = \left( \mathbf{P}\mathbf{E}_1\mathbf{P}^T + \mathbf{P}\left(\mathbf{\Sigma}_1 \odot \dot{\mathbf{E}}_1\right)\mathbf{P}^T \right).$$

□

# C NTKs for GCN and SGC

We restate the NTK derived in Sabanayagam et al. [2023] for self containment. The GCN of depth $L$ with width $d_l \to \infty\ \forall l \in \{1, \ldots, L\}$, the network converges to the following kernel when trained with gradient flow.

**Theorem 2** (NTK for Vanilla GCN)**.** *For the GCN defined in Definition 2, the NTK* $\mathbf{Q}$ *at depth* $L$ *and* $K = 1$ *is*

$$\mathbf{Q}^{(L)} = \sum_{k=1}^{L+1} \underbrace{\mathbf{S}\Big( \ldots \mathbf{S}\Big(\mathbf{S}}_{L+1-k \text{ terms}} \left(\mathbf{\Sigma}_k \odot \dot{\mathbf{E}}_k\right)\mathbf{S}^T \odot \dot{\mathbf{E}}_{k+1}\Big)\mathbf{S}^T \odot \ldots \odot \dot{\mathbf{E}}_d\Big)\mathbf{S}^T. \tag{15}$$

*Here* $\mathbf{\Sigma}_k \in \mathbb{R}^{n\times n}$ *is the co-variance between nodes of layer* $k$*, and is given by* $\mathbf{\Sigma}_1 = \mathbf{S}\mathbf{X}\mathbf{X}^T\mathbf{S}^T$*,* $\mathbf{\Sigma}_k = \mathbf{S}\mathbf{E}_{k-1}\mathbf{S}^T$ *with* $\mathbf{E}_k = c_\sigma \mathop{\mathbb{E}}_{\mathbf{F}\sim\mathcal{N}(\mathbf{0},\mathbf{\Sigma}_k)} \left[\sigma(\mathbf{F})\sigma(\mathbf{F})^T\right]$*,* $\dot{\mathbf{E}}_k = c_\sigma \mathop{\mathbb{E}}_{\mathbf{F}\sim\mathcal{N}(\mathbf{0},\mathbf{\Sigma}_k)} \left[\dot{\sigma}(\mathbf{F})\dot{\sigma}(\mathbf{F})^T\right]$ *and* $\dot{\mathbf{E}}_{L+1} = \mathbf{1}_{n\times n}$*.*

$$\begin{aligned}\left(E_k\right)_{ij} &= \sqrt{\left(\Sigma_k\right)_{ii}\left(\Sigma_k\right)_{jj}}\ \kappa_1\left(\frac{\left(\Sigma_k\right)_{ij}}{\sqrt{\left(\Sigma_k\right)_{ii}\left(\Sigma_k\right)_{jj}}}\right) \\ \left(\dot{E}_k\right)_{ij} &= \kappa_0\left(\frac{\left(\Sigma_k\right)_{ij}}{\sqrt{\left(\Sigma_k\right)_{ii}\left(\Sigma_k\right)_{jj}}}\right),\end{aligned}$$

*where* $\kappa_0(z) = \frac{1}{\pi}\left(\pi - \arccos(z)\right)$ *and* $\kappa_1(z) = \frac{1}{\pi}\left(z\left(\pi - \arccos(z)\right) + \sqrt{1-z^2}\right)$.

# D QPCert for GNNs through their corresponding NTKs

To certify a specific GNN using our QPCert framework, we need to derive element-wise lower and upper bounds valid for all NTK matrices $\widetilde{\boldsymbol{Q}} \in \mathcal{A}(\boldsymbol{Q})$ of the corresponding network, that are constructable by the adversary. As a first step, we introduce the NTKs for the GNNs of interest before deriving the bounds. While Sabanayagam et al. [2023] provides the NTKs for GCN and SGC with and without skip connections, we derive the NTK for (A)PPNP in App. B. For clarity, we present the NTKs for $f_\theta(\mathcal{G})$ with $L = 1$ here and the general case for any $L$ in App. C. For $L = 1$, the NTKs generalize to the form $\boldsymbol{Q} = \mathbf{M}(\mathbf{\Sigma} \odot \dot{\mathbf{E}})\mathbf{M}^T + \mathbf{M}\mathbf{E}\mathbf{M}^T$ for all the networks, with the definitions of $\mathbf{M}$, $\mathbf{\Sigma}$, $\mathbf{E}$ and $\dot{\mathbf{E}}$ detailed in Table 1. Thus, it is important to note that the feature matrix $\mathbf{X}$, which the adversary can manipulate, enters into the NTKs only as a product $\mathbf{X}\mathbf{X}^T$, making this the quantity of interest when bounding the NTK matrix.

Therefore, focusing on $p = \infty$ and $p = 2$ in the perturbation model $\mathcal{B}_p(\mathbf{x})$ and $\widetilde{\mathbf{X}} \in \mathcal{A}(\mathbf{X})$, we first derive the bounds for $\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T$ by considering $\mathcal{U} := \{i : i \notin \mathcal{V}_V\}$ to be the set of all unverified nodes

Table 1: The NTKs of GNNs have the general form $\boldsymbol{Q} = \mathbf{M}(\boldsymbol{\Sigma} \odot \dot{\mathbf{E}})\mathbf{M}^T + \mathbf{M}\mathbf{E}\mathbf{M}^T$ for $L = 1$. The definitions of $\mathbf{M}$, $\boldsymbol{\Sigma}$, $\mathbf{E}$ and $\dot{\mathbf{E}}$ are given in the table. $\kappa_0(z) = \frac{1}{\pi}(\pi - \arccos(z))$ and $\kappa_1(z) = \frac{1}{\pi}\left(z(\pi - \arccos(z)) + \sqrt{1-z^2}\right)$.

| GNN | $\mathbf{M}$ | $\boldsymbol{\Sigma}$ | $E_{ij}$ | $\dot{E}_{ij}$ |
|---|---|---|---|---|
| GCN | $\mathbf{S}$ | $\mathbf{S}\mathbf{X}\mathbf{X}^T\mathbf{S}^T$ | $\sqrt{\Sigma_{ii}\Sigma_{jj}}\kappa_1\left(\frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}\right)$ | $\kappa_0\left(\frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}\right)$ |
| SGC | $\mathbf{S}$ | $\mathbf{S}\mathbf{X}\mathbf{X}^T\mathbf{S}^T$ | $\Sigma_{ij}$ | $1$ |
| (A)PPNP | $\mathbf{P}$ | $\mathbf{X}\mathbf{X}^T + \mathbf{1}_{n\times n}$ | $\sqrt{\Sigma_{ii}\Sigma_{jj}}\kappa_1\left(\frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}\right)$ | $\kappa_0\left(\frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}\right)$ |
| MLP | $\mathbf{I}_n$ | $\mathbf{X}\mathbf{X}^T + \mathbf{1}_{n\times n}$ | $\sqrt{\Sigma_{ii}\Sigma_{jj}}\kappa_1\left(\frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}\right)$ | $\kappa_0\left(\frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}\right)$ |

that the adversary can potentially control. In the following, Lemma 1, and Lemma 2 present the worst-case element-wise lower and upper bounds for $\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T = \mathbf{X}\mathbf{X}^T + \Delta$ in terms of $\Delta$. The proofs can be found in App. D.1.

**Lemma 1** (Bounds for $\Delta$, $p = \infty$)**.** *Given $\mathcal{B}_\infty(\mathbf{x})$ and any $\widetilde{\mathbf{X}} \in \mathcal{A}(\mathbf{X})$, then $\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T = \mathbf{X}\mathbf{X}^T + \Delta$ where the worst-case bounds for $\Delta$, $\Delta_{ij}^L \leq \Delta_{ij} \leq \Delta_{ij}^U$ for all $i$ and $j \in [n]$, is*

$$\begin{aligned}\Delta_{ij}^L &= -\delta\|\mathbf{X}_j\|_1\mathbb{1}[i \in \mathcal{U}] - \delta\|\mathbf{X}_i\|_1\mathbb{1}[j \in \mathcal{U}] - \delta^2 d\mathbb{1}[i \in \mathcal{U} \wedge j \in \mathcal{U} \wedge i \neq j] \\ \Delta_{ij}^U &= \delta\|\mathbf{X}_j\|_1\mathbb{1}[i \in \mathcal{U}] + \delta\|\mathbf{X}_i\|_1\mathbb{1}[j \in \mathcal{U}] + \delta^2 d\mathbb{1}[i \in \mathcal{U} \wedge j \in \mathcal{U}] \end{aligned} \tag{16}$$

**Lemma 2** (Bounds for $\Delta$, $p = 2$)**.** *Given $\mathcal{B}_2(\mathbf{x})$ and any $\widetilde{\mathbf{X}} \in \mathcal{A}(\mathbf{X})$, then $\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T = \mathbf{X}\mathbf{X}^T + \Delta$ where the worst-case bounds for $\Delta$, $\Delta_{ij}^L \leq \Delta_{ij} \leq \Delta_{ij}^U$ for all $i$ and $j \in [n]$, is*

$$\begin{aligned}\Delta_{ij}^L &= -\delta\|\mathbf{X}_j\|_2\mathbb{1}[i \in \mathcal{U}] - \delta\|\mathbf{X}_i\|_2\mathbb{1}[j \in \mathcal{U}] - \delta^2\mathbb{1}[i \in \mathcal{U} \wedge j \in \mathcal{U} \wedge i \neq j] \\ \Delta_{ij}^U &= \delta\|\mathbf{X}_j\|_2\mathbb{1}[i \in \mathcal{U}] + \delta\|\mathbf{X}_i\|_2\mathbb{1}[j \in \mathcal{U}] + \delta^2\mathbb{1}[i \in \mathcal{U} \wedge j \in \mathcal{U}] \end{aligned} \tag{17}$$

The NTK bounds $\widetilde{Q}_{ij}^L$ and $\widetilde{Q}_{ij}^U$, are then derived by simply propagating the bounds of $\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T$ through the NTK formulation since the multipliers and addends are positive. To elaborate, we compute $\widetilde{Q}_{ij}^L$ by substituting $\mathbf{X}\mathbf{X}^T = \mathbf{X}\mathbf{X}^T + \Delta^L$, and likewise for $\widetilde{Q}_{ij}^U$. Only bounding $E_{ij}$ and $\dot{E}_{ij}$ needs special care and our respective approach is discussed in App. D.2. Further, we prove that the bounds are tight in the worst-case as shown in Theorem 3 in App. D.3.

**Theorem 3** (NTK bounds are tight)**.** *The worst-case NTK bounds are tight for GNNs with linear activations such as SGC and (A)PPNP, and MLP with $\sigma(z) = z$ for both $p = \infty$ and $p = 2$ in $\mathcal{B}_p(\mathbf{x})$.*

### D.1 Derivation of NTK bounds

To derive Lemmas 1 and 2, we consider the perturbed feature matrix $\widetilde{\mathbf{X}} \in \mathcal{A}(\mathbf{X})$ and derive the worst-case bounds for $\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T$ based on the perturbation model $\mathcal{B}_p(\mathbf{x})$ where $p = \infty$, $p = 2$ and $p = 1$ in our study. Let's say $\mathcal{U}$ is the set of nodes that are potentially controlled by the adversary $\mathcal{A}(\mathbf{X})$ and $\widetilde{\mathbf{X}} = \mathbf{X} + \boldsymbol{\Gamma} \in \mathbb{R}^{n\times d}$ where $\boldsymbol{\Gamma}_i$ is the adversarial perturbations added to node $i$ by the adversary, therefore, $\|\boldsymbol{\Gamma}_i\|_p \leq \delta$ and $\boldsymbol{\Gamma}_i > 0$ for $i \in \mathcal{U}$ and $\boldsymbol{\Gamma}_i = 0$ for $i \notin \mathcal{U}$. Then

$$\begin{aligned}\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T &= (\mathbf{X} + \boldsymbol{\Gamma})(\mathbf{X} + \boldsymbol{\Gamma})^T \\ &= \mathbf{X}\mathbf{X}^T + \boldsymbol{\Gamma}\mathbf{X}^T + \mathbf{X}\boldsymbol{\Gamma}^T + \boldsymbol{\Gamma}\boldsymbol{\Gamma}^T = \mathbf{X}\mathbf{X}^T + \Delta. \end{aligned} \tag{18}$$

As a result, it suffices to derive the worst-case bounds for $\Delta$, $\Delta^L \leq \Delta \leq \Delta^U$, for different perturbations. To do so, our strategy is to bound the scalar products $\langle \boldsymbol{\Gamma}_i, \mathbf{X}_j \rangle$ and $\langle \boldsymbol{\Gamma}_i, \boldsymbol{\Gamma}_j \rangle$ element-wise, hence derive $\Delta_{ij}^L \leq \Delta_{ij} \leq \Delta_{ij}^U$. In the following, we derive $\Delta_{ij}^L$ and $\Delta_{ij}^U$ for the cases when $p = \infty$, $p = 2$ and $p = 1$ in $\mathcal{B}_p(\mathbf{x})$.

**Case $(i)$: Derivation of Lemma 1 for $p = \infty$.** In this case, the perturbation allows $\|\tilde{\mathbf{X}}_i - \mathbf{X}_i\|_\infty \leq \delta$, then by Hölder's inequality $\langle \mathbf{a}, \mathbf{b} \rangle \leq \|\mathbf{a}\|_p \|\mathbf{b}\|_q$ where $\frac{1}{p} + \frac{1}{q} = 1$ for all $p, q \in [1, \infty]$ we have

$$\begin{aligned} |\langle \mathbf{\Gamma}_i, \mathbf{X}_j \rangle| &\leq \|\mathbf{\Gamma}_i\|_\infty \|\mathbf{X}_j\|_1 \leq \delta \|\mathbf{X}_j\|_1 \\ |\langle \mathbf{\Gamma}_i, \mathbf{\Gamma}_j \rangle| &\leq \|\mathbf{\Gamma}_i\|_2 \|\mathbf{\Gamma}_j\|_2 \leq d \|\Delta_i\|_\infty \|\Delta_j\|_\infty \leq d\delta^2. \end{aligned} \tag{19}$$

Using Eq. (19), the worst-case lower bound $\Delta_{ij}^L$ is the lower bound of $\mathbf{\Gamma}\mathbf{X}^T + \mathbf{X}\mathbf{\Gamma}^T + \mathbf{\Gamma}\mathbf{\Gamma}^T$:

$$\Delta_{ij}^L = \begin{cases} 0+ & \text{if } i, j \notin \mathcal{U} \\ -\delta \|\mathbf{X}_j\|_1 + & \text{if } i \in \mathcal{U} \\ -\delta \|\mathbf{X}_i\|_1 + & \text{if } j \in \mathcal{U} \\ -\delta^2 d & \text{if } i, j \in \mathcal{U} \text{ and } i \neq j. \end{cases} \tag{20}$$

The last case in Eq. (20) is due to the fact that $\langle \mathbf{\Gamma}_i, \mathbf{\Gamma}_i \rangle \geq 0$, hence $\Delta_{ii}^L = 0$. Finally, the Eq. (20) can be succinctly written using the indicator function as

$$\Delta_{ij}^L = -\delta \|\mathbf{X}_j\|_1 \mathbb{1}[i \in \mathcal{U}] - \delta \|\mathbf{X}_i\|_1 \mathbb{1}[j \in \mathcal{U}] - \delta^2 d \mathbb{1}[i \in \mathcal{U} \wedge j \in \mathcal{U} \wedge i \neq j],$$

deriving the lower bound in Lemma 1. Similarly, applying the Hölder's inequality for the worst-case upper bound, we get

$$\Delta_{ij}^U = \begin{cases} 0+ & \text{if } i, j \notin \mathcal{U} \\ \delta \|\mathbf{X}_j\|_1 + & \text{if } i \in \mathcal{U} \\ \delta \|\mathbf{X}_i\|_1 + & \text{if } j \in \mathcal{U} \\ \delta^2 d & \text{if } i, j \in \mathcal{U}. \end{cases} \tag{21}$$

Thus, we derive Lemma 1 by succinctly writing it as

$$\Delta_{ij}^U = \delta \|\mathbf{X}_j\|_1 \mathbb{1}[i \in \mathcal{U}] + \delta \|\mathbf{X}_i\|_1 \mathbb{1}[j \in \mathcal{U}] + \delta^2 d \mathbb{1}[i \in \mathcal{U} \wedge j \in \mathcal{U}].$$

□

**Case $(ii)$: Derivation of Lemma 2 for $p = 2$.** The worst-case lower and upper bounds of $\Delta_{ij}$ for $p = 2$ is derived in the similar fashion as $p = \infty$. Here, the perturbation allows $\|\tilde{\mathbf{X}}_i - \mathbf{X}_i\|_2 \leq \delta$. Hence,

$$\begin{aligned} |\langle \mathbf{\Gamma}_i, \mathbf{X}_j \rangle| &\leq \|\mathbf{\Gamma}_i\|_\infty \|\mathbf{X}_j\|_2 \leq \delta \|\mathbf{X}_j\|_2 \\ |\langle \mathbf{\Gamma}_i, \mathbf{\Gamma}_j \rangle| &\leq \|\mathbf{\Gamma}_i\|_2 \|\mathbf{\Gamma}_j\|_2 \leq \delta^2. \end{aligned} \tag{22}$$

Using Eq. (22), we derive the lower and upper bounds of $\Delta_{ij}$:

$$\Delta_{ij}^L = \begin{cases} 0+ & \text{if } i, j \notin \mathcal{U} \\ -\delta ||x_j||_2 + & \text{if } i \in \mathcal{U} \\ -\delta ||x_i||_2 + & \text{if } j \in \mathcal{U} \\ -\delta^2 & \text{if } i, j \in \mathcal{U} \end{cases} \qquad \Delta_{ij}^U = \begin{cases} 0+ & \text{if } i, j \notin \mathcal{U} \\ \delta ||x_j||_2 + & \text{if } i \in \mathcal{U} \\ \delta ||x_i||_2 + & \text{if } j \in \mathcal{U} \\ \delta^2 & \text{if } i, j \in \mathcal{U} \end{cases}$$

□

### D.2 Bounding $E_{ij}$ and $\dot{E}_{ij}$ in the NTK

NTKs for GNNs with non-linear ReLU activation have $\mathbf{E}$ and $\dot{\mathbf{E}}$ with non-linear $\kappa_1(z)$ and $\kappa_0(z)$ functions in their definitions, respectively. In order to bound the NTK, we need a strategy to bound these quantities as well. In this section, we discuss our approach to bound $E_{ij}$ and $\dot{E}_{ij}$ through bounding the functions for any GNN with $L$ layers. For ease of exposition, we ignore the layer indexing for the terms of interest and it is understood from the context. Recollect that the definitions of $\mathbf{E}$ and $\dot{\mathbf{E}}$ are based on $\mathbf{\Sigma}$, which is a linear combination of $\mathbf{S}$ and the previous layer. So, we

consider that at this stage, we already have $\mathbf{\Sigma}$, $\mathbf{\Sigma}^L$ and $\mathbf{\Sigma}^U$. Now, we expand the functions in the definition and write $E_{ij}$ and $\dot{E}_{ij}$ using their corresponding $\mathbf{\Sigma}$ as follows:

$$E_{ij} = \frac{\sqrt{\Sigma_{ii}\Sigma_{jj}}}{\pi}\left(\frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}\left(\pi - \arccos\left(\frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}\right)\right) + \sqrt{1 - \frac{\Sigma_{ij}^2}{\Sigma_{ii}\Sigma_{jj}}}\right) \tag{23}$$

$$\dot{E}_{ij} = \frac{1}{\pi}\left(\pi - \arccos\left(\frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}\right)\right) \tag{24}$$

We derive the lower and upper bounds for $E_{ij}$ and $\dot{E}_{ij}$ in Algorithm 1.

**Algorithm 1** Procedure to compute $E_{ij}^L$, $E_{ij}^U$, $\dot{E}_{ij}^L$ and $\dot{E}_{ij}^U$

Given $\mathbf{\Sigma}$, $\mathbf{\Sigma}^L$ and $\mathbf{\Sigma}^U$
Let $s^l = \sqrt{\Sigma_{ii}^L \Sigma_{jj}^L}$, $s^u = \sqrt{\Sigma_{ii}^U \Sigma_{jj}^U}$
**if** $\Sigma_{ij}^L > 0$ **then**
  $a^l = \dfrac{\Sigma_{ij}^L}{s^u}$, $a^u = \dfrac{\Sigma_{ij}^U}{s^l}$
**else**
  $a^l = \dfrac{\Sigma_{ij}^L}{s^l}$, $a^u = \dfrac{\Sigma_{ij}^U}{s^u}$
**end if**
**if** $|\Sigma_{ij}^U| > |\Sigma_{ij}^L|$ **then**
  $b^l = \left(\dfrac{\Sigma_{ij}^L}{s^u}\right)^2$, $b^u = \left(\dfrac{\Sigma_{ij}^U}{s^l}\right)^2$
**else**
  $b^l = \left(\dfrac{\Sigma_{ij}^L}{s^l}\right)^2$, $b^u = \left(\dfrac{\Sigma_{ij}^U}{s^u}\right)^2$
**end if**
$E_{ij}^L = \frac{s^l}{\pi}\left(a^l\left(\pi - \arccos\left(a^l\right)\right) + \sqrt{1 - b^u}\right)$
$E_{ij}^U = \frac{s^u}{\pi}\left(a^u\left(\pi - \arccos\left(a^u\right)\right) + \sqrt{1 - b^l}\right)$
$\dot{E}_{ij}^L = \frac{1}{\pi}\left(\pi - \arccos\left(a^l\right)\right)$
$\dot{E}_{ij}^U = \frac{1}{\pi}\left(\pi - \arccos\left(a^u\right)\right)$

### D.3 Derivation of Theorem 3: NTK bounds are tight

We analyze the tightness of NTK bounds by deriving conditions on graph $\mathcal{G} = (\mathbf{S}, \mathbf{X})$ when $\Delta_{ij}^L$ and $\Delta_{ij}^U$ are attainable exactly. As our NTK bounding strategy is based on bounding the adversarial perturbation $\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T$ and the non-linear functions $\kappa_0(z)$ and $\kappa_1(z)$, it is easy to see that the bounds with non-linearities cannot be tight. So, we consider only linear GCN (=SGC), (A)PPNP and MLP with linear activations.

Now, we focus on deriving conditions for the given node features $\mathbf{X}$ using the classic result on the equality condition of Hölder's inequality [Steele, 2004], and then analyze the NTK bounds. Steele [2004, Fig. 9.1] shows that the bounds on $\langle \mathbf{a}, \mathbf{b} \rangle$ using the Höder's inequality is reached when $|\mathbf{a}_i|^p = |\mathbf{b}_i|^q \frac{\|\mathbf{a}\|_p^p}{\|\mathbf{b}\|_q^q}$. Using this, we analyze

$$\Delta_{ij} = \langle \mathbf{\Gamma}_i, \mathbf{X}_j \rangle + \langle \mathbf{\Gamma}_j, \mathbf{X}_i \rangle + \langle \mathbf{\Gamma}_i, \mathbf{\Gamma}_j \rangle \tag{25}$$

in which we call $\langle \mathbf{\Gamma}_i, \mathbf{\Gamma}_j \rangle$ as interaction term. Following this analysis, the tightness of NTK bounds is derived below for $p = \infty$ and $p = 2$.

**Case** $(i)$**:** $p = \infty$**.** In this case, the feature bounds in Eq. (19) are tight,

$$\forall j,\ \mathbf{X}_j \neq 0 \text{ and } \forall i, k\ \mathbf{\Gamma}_{ik} = c_i$$

where $c_i$ is some constant such that $\|\mathbf{\Gamma}_i\|_\infty \leq \delta$ so the perturbation budget is satisfied. As a result, the upper bound of $\Delta_{ij}$ in Lemma 1 is achieved exactly in the following cases,

**(a)** Number of adversarial nodes $= 1$: Here the interaction term in Eq. (25) is 0 for all $i$ and $j$. Then for the one adversarial node $i$, there exists $\mathbf{X}_j \in \mathbb{R}^d_+$, one can set $\mathbf{\Gamma}_i = +\delta \mathbf{1}_d$ to achieve the upper bound.

**(b)** Number of adversarial nodes $> 1$: Here the interaction term is $\neq 0$ for all the adversarial nodes $i$ and $j$. Then, for the adversarial nodes $i$ and $j$ if there exist $\mathbf{X}_i \in \mathbb{R}^d_+$ and $\mathbf{X}_j \in \mathbb{R}^d_+$ then for $\mathbf{\Gamma}_i = \mathbf{\Gamma}_j = +\delta \mathbf{1}_d$ upper bounds are achieved.

The NTKs with linear activations $Q_{ij}$ achieve the upper bound in these cases. Similarly, the lower bound in Lemma 1 is achieved exactly as discussed in the following,

**(a)** Number of adversarial nodes $= 1$: Here the interaction term in Eq. (25) is 0 for all $i$ and $j$. Then for the adversarial node $i$, there exists $\mathbf{X}_j \in \mathbb{R}^d_+$, one can set $\mathbf{\Gamma}_i = -\delta \mathbf{1}_d$ to achieve the lower bound.

**(b)** Number of adversarial nodes $> 1$: Here the interaction term is $\neq 0$ for all the adversarial nodes $i$ and $j$. Then, for the adversarial nodes $i$ and $j$ if there exist $\mathbf{X}_i \in \mathbb{R}^d_+$ and $\mathbf{X}_j \in \mathbb{R}^d_-$ then for $\mathbf{\Gamma}_i = -\delta \mathbf{1}_d$ and $\mathbf{\Gamma}_j = +\delta \mathbf{1}_d$,

leading to tight lower bounds of Lemma 1. The lower and upper tight bounds of $\Delta$ together leads to tight NTK bounds for linear activations. Note that there is no need to impose any structural restriction on the graph $\mathbf{S}$ to achieve the tight bounds for NTK.

**Case** $(ii)$**:** $p = 2$**.** In this case, the feature bounds in Eq. (22) are tight,

$$\forall i, j, \;\; \mathbf{X}_j \text{ and } \mathbf{\Gamma}_i \text{ are linearly dependent}$$

and $\|\mathbf{\Gamma}_i\|_2 \leq \delta$ so the perturbation budget is satisfied. As a result, the upper bound of $\Delta_{ij}$ in Lemma 2 is achieved exactly in the following,

**(a)** Number of adversarial nodes $= 1$: Here the interaction term in Eq. (25) is 0 for all $i$ and $j$. Then for the one adversarial node $i$, and any $\mathbf{X}_j \in \mathbb{R}^d$, one can set $\mathbf{\Gamma}_i = +\delta \frac{\mathbf{X}_j}{\|\mathbf{X}_j\|_2}$ to achieve the upper bound.

**(b)** Number of adversarial nodes $> 1$: Here the interaction term is $\neq 0$ for all the adversarial nodes $i$ and $j$. Then, for the adversarial nodes $i$ and $j$, if there exist $\mathbf{X}_i \in \mathbb{R}^d_+$ and $\mathbf{X}_j \in \mathbb{R}^d_+$ are linearly dependent, then for $\mathbf{\Gamma}_i = +\delta \frac{\mathbf{X}_j}{\|\mathbf{X}_j\|_2}$ and $\mathbf{\Gamma}_j = +\delta \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2}$ tight upper bound is achieved.

The NTKs with linear activations $Q_{ij}$ achieve the worst-case upper bound in these cases. Similarly, the lower bound in Lemma 2 is achieved exactly as discussed in the following,

**(a)** Number of adversarial nodes $= 1$: Here the interaction term in Eq. (25) is 0 for all $i$ and $j$. Then for the adversarial node $i$, and any $\mathbf{X}_j \in \mathbb{R}^d$, one can set $\mathbf{\Gamma}_i = -\delta \frac{\mathbf{X}_j}{\|\mathbf{X}_j\|_2}$ to achieve the lower bound.

**(b)** Number of adversarial nodes $> 1$: Here the interaction term is $\neq 0$ for all the adversarial nodes $i$ and $j$. Then, for the adversarial nodes $i$ and $j$, if there exist $\mathbf{X}_i \in \mathbb{R}^d_+$ and $\mathbf{X}_j \in \mathbb{R}^d_-$ are linearly dependent, then for $\mathbf{\Gamma}_i = -\delta \frac{\mathbf{X}_j}{\|\mathbf{X}_j\|_2}$ and $\mathbf{\Gamma}_i = +\delta \frac{\mathbf{X}_i}{\|\mathbf{X}_j\|_2}$,

leading to tight lower bounds of Lemma 1. The lower and upper tight bounds of $\Delta$ together leads to tight NTK bounds for linear activations. Note that there is no need to impose any structural restriction on the graph $\mathbf{S}$ to achieve the tight bounds for NTK, same as the $p = \infty$ case. We further note that only one instance of achieving the worst-case bound is stated, and one can construct similar cases, for example by considering opposite signs for the features and perturbations. □

# E Multi-class certification and extension to an inductive setting

In this section, we discuss first the certification for multi-class and afterwards, how our certificate extends from a transductive to an inductive setting.

**Multi-class classification.** To do multi-class classification using SVM with NTK, we choose the One-Vs-All strategy, where we learn $K$ classifiers. Formally, we learn $\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^K$ which has corresponding duals $\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^K$. In order to learn $\boldsymbol{\beta}^c$, all samples with class label $c$ are assumed to be positive and the rest negative. To not overload the notations, let's say for all $c$, $\boldsymbol{\beta}^c$ is the optimal

solution with the corresponding dual $\boldsymbol{\alpha}^c$. Then the prediction for a node $t$ is $c^* = \arg\max_c \hat{p}_t^c$ where $\hat{p}_t^c = \sum_{i=1}^m y_i \alpha_i^c Q_{ti}$ where $\boldsymbol{Q}$ is the NTK matrix.

Given this, we propose a simple extension of our binary certification where to certify a node $t$ as provably robust, we minimize the MILP objective in Theorem 1 for the predicted class $c^*$ and maximize the objective for the remaining $K-1$ classes. Finally, certify $t$ to be provably robust only if the objective for $c^*$ remains maximum. Formally, we state the objective below.

**Theorem 4.** *Node $t$ with original predicted class $c^*$ is certifiably robust against adversary $\mathcal{A}$ if $c' = c^*$ where $c'$ is defined in the following. Using the MILP $P(\boldsymbol{Q})$ in Theorem 1, we define*

$$P(\boldsymbol{Q})_c := P(\boldsymbol{Q}) \text{ using } \boldsymbol{\alpha}^c, \text{ with the only change in obj. to } (-1)^{\mathbb{1}[c \neq c^*]} \sum_{i=1}^m y_i Z_{ti}$$

$$c'_t = \arg\max_{c \in [K]} P(\boldsymbol{Q})_c. \tag{26}$$

**Inductive setting.** Our framework easily extends to inductive node classification as Theorem 1 is valid unchanged. The only change is that in inductive node classification, one has two graphs: $(i)$ a graph $G_{train}$ known at training time; and $(ii)$ a second graph $G_{test}$ with added test nodes available during testing. Thus, in the computation of the bounds $\tilde{Q}_{ij}^L$ and $\tilde{Q}_{ij}^U$, if both nodes $i$ and $j$ are known at training time, the corresponding NTK bounds should be computed only using $G_{train}$ excluding test nodes. Whereas, if either $i$ or $j$ are test nodes, the corresponding bounds need to be calculated using the expanded test graph $G_{test}$.

# F Proof of Proposition 1

We restate Proposition 1.

**Proposition 1.** *Problem $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ given by Eq. (2) is convex and satisfies strong Slater's constraint. Consequently, the single-level optimization problem $\mathrm{P}_3(\boldsymbol{Q})$ arising from $\mathrm{P}_2(\boldsymbol{Q})$ by replacing $\boldsymbol{\alpha} \in \mathcal{S}(\widetilde{\boldsymbol{Q}})$ with Eqs. (5) to (7) has the same globally optimal solutions as $\mathrm{P}_2(\boldsymbol{Q})$.*

Given any $\widetilde{\boldsymbol{Q}} \in \mathcal{A}(\boldsymbol{Q})$. We prove two lemmas, leading us towards proving Proposition 1.

**Lemma 3.** *Problem $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ is convex.*

*Proof.* The dual problem $\mathrm{P}_1^{\mathrm{b}}(\widetilde{\boldsymbol{Q}})$ associated do an SVM with bias term reads

$$\mathrm{P}_1^{\mathrm{b}}(\widetilde{\boldsymbol{Q}}): \min_{\boldsymbol{\alpha}} -\sum_{i=1}^m \alpha_i + \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m y_i y_j \alpha_i \alpha_j \widetilde{Q}_{ij} \text{ s.t.} \sum_{i=1}^m \alpha_i y_i = 0 \;\; 0 \leq \alpha_i \leq C \;\forall i \in [m] \tag{27}$$

It is a known textbook result that $\mathrm{P}_1^{\mathrm{b}}(\widetilde{\boldsymbol{Q}})$ is convex and we refer to Mohri et al. [2018] for a proof. A necessary and sufficient condition for an optimization problem to be convex is that the objective function as well as all inequality constraints are convex and the equality constraints affine functions. Furthermore, the domain of the variable over which is optimized must be a convex set. As removing the bias term of an SVM results in a dual problem $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ which is equivalent to $\mathrm{P}_1^{\mathrm{b}}(\widetilde{\boldsymbol{Q}})$ only with the constraint $\sum_{i=1}^m \alpha_i y_i = 0$ removed, the necessary and sufficient conditions for convexity stay fulfilled. □

Now, we define strong Slater's condition for $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ embedded in the upper-level problem $\mathrm{P}_2(\boldsymbol{Q})$ defined in Eq. (4), which we from here on will call strong Slater's constraint qualification [Dempe and Dutta, 2012].

**Def. 6** (Slater's CQ)**.** *The lower-level convex optimization problem $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ fulfills strong Slater's Constraint Qualification, if for any upper-level feasible $\widetilde{\boldsymbol{Q}} \in \mathcal{A}(\boldsymbol{Q})$, there exists a point $\boldsymbol{\alpha}(\widetilde{\boldsymbol{Q}})$ in the feasible set of $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ such that no constraint in $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ is active, i.e. $0 < \alpha(\widetilde{\boldsymbol{Q}})_i < C$ for all $i \in [m]$.*

**Lemma 4.** *Problem* $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ *fulfills strong Slater's constraint qualification.*

*Proof.* We prove Lemma 4 through a constructive proof. Given any upper-level feasible $\widetilde{\boldsymbol{Q}} \in \mathcal{A}(\boldsymbol{Q})$. Let $\boldsymbol{\alpha}$ be an optimal solution to $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$. We restrict ourselves to cases, where $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ is non-degenerate, i.e. the optimal solution to the SVM $f_\theta^{SVM}$ corresponds to a weight vector $\boldsymbol{\beta} \neq \mathbf{0}$. Then, at least for one index $i \in [m]$ it must hold that $\boldsymbol{\alpha}_i > 0$.

Assume that $j$ is the index in $[m]$ with the smallest $\boldsymbol{\alpha}_j > 0$. Let $\epsilon = \boldsymbol{\alpha}_j/m + 1 > 0$. Now, we construct a new $\boldsymbol{\alpha}'$ from $\boldsymbol{\alpha}$ by for each $i \in [m]$ setting:

- If $\alpha_i = 0$, set $\alpha'_i = \epsilon$.
- If $\alpha_i = C$, set $\alpha'_i = C - \epsilon$.

The new $\boldsymbol{\alpha}'$ fulfills $0 < \boldsymbol{\alpha}'(\widetilde{\boldsymbol{Q}})_i < C$ for all $i \in [m]$. If $\mathrm{P}_1(\widetilde{\boldsymbol{Q}})$ is degenerate, set $\boldsymbol{\alpha}'(\widetilde{\boldsymbol{Q}})_i = C/2$ for all $i \in [m]$. This concludes the proof. □

[Dempe and Dutta, 2012] establish that any bilevel optimization problem $U$ whose lower-level problem $L$ is convex and fulfills strong Slater's constraint qualification for any upper-level feasible point has the same global solutions as another problem defined by replacing the lower-level problem $L$ in $U$ with $L$'s Karash Kuhn Tucker conditions. This, together with Lemmas 3 and 4 concludes the proof for Proposition 1.

## G Setting big-$M$ constraints

**Proposition 2** (Big-$M$'s)**.** *Replacing the complementary slackness constraints Eq.* (7) *in* $\mathrm{P}_3(\boldsymbol{Q})$ *with the big-*$M$ *constraints given in Eq.* (8) *does not cut away solution values of* $\mathrm{P}_3(\boldsymbol{Q})$*, if for any* $i \in [m]$*, the big-*$M$ *values fulfill the following conditions. For notational simplicity* $j$ : *Condition*$(j)$ *denotes* $j \in \{j \in [m] : \textit{Condition}(j)\}$.

*If* $y_i = 1$ *then*

$$M_{u_i} \geq \sum_{j:y_j=1 \wedge \widetilde{Q}^U_{ij} \geq 0} C\widetilde{Q}^U_{ij} - \sum_{j:y_j=-1 \wedge \widetilde{Q}^L_{ij} \leq 0} C\widetilde{Q}^L_{ij} - 1 \tag{28}$$

$$M_{v_i} \geq \sum_{j:y_j=-1 \wedge \widetilde{Q}^U_{ij} \geq 0} C\widetilde{Q}^U_{ij} - \sum_{j:y_j=1 \wedge \widetilde{Q}^L_{ij} \leq 0} C\widetilde{Q}^L_{ij} + 1 \tag{29}$$

*If* $y_i = -1$ *then*

$$M_{u_i} \geq \sum_{j:y_j=-1 \wedge \widetilde{Q}^U_{ij} \geq 0} C\widetilde{Q}^U_{ij} - \sum_{j:y_j=1 \wedge \widetilde{Q}^L_{ij} \leq 0} C\widetilde{Q}^L_{ij} - 1 \tag{30}$$

$$M_{v_i} \geq \sum_{j:y_j=1 \wedge \widetilde{Q}^U_{ij} \geq 0} C\widetilde{Q}^U_{ij} - \sum_{j:y_j=-1 \wedge \widetilde{Q}^L_{ij} \leq 0} C\widetilde{Q}^L_{ij} + 1 \tag{31}$$

*To obtain the tightest formulation for* $P(\boldsymbol{Q})$ *from the above conditions, we set the big-*$M$*'s to equal the conditions.*

*Proof.* Denote by $UB$ an upper bound to $\sum_{j=1}^m y_i y_j Z_{ij}$ and by $LB$ a lower bound to $\sum_{j=1}^m y_i y_j Z_{ij}$. The existence of these bounds follows from $y_i$ and $y_j \in \{-1, 1\}$ and $Z_{ij} = \alpha_j \widetilde{Q}_{ij}$ with $0 \leq \alpha_j \leq C$ and $\widetilde{Q}^L_{ij} \leq \widetilde{Q}_{ij} \leq \widetilde{Q}^U_{ij}$, i.e. the boundedness of all variables.

$u_i$ and $v_i$ need to be able to be set such that $\sum_{j=1}^m y_i y_j Z_{ij} - u_i + v_i = 1$ (see Eq. (5)) can be satisfied given any $\boldsymbol{\alpha}^*$ and $\widetilde{\boldsymbol{Q}}^*$ part of an optimal solution to $\mathrm{P}_3(\boldsymbol{Q})$. By using $UB$ and $LB$ we get the following inequalities:

$$UB - u_i + v_i \geq 1 \tag{32}$$

and

$$LB - u_i + v_i \leq 1 \tag{33}$$

Denote $\sum_{j=1}^{m} y_i y_j Z_{ij}$ by $T$. Thus, if $T \geq 1$, setting $v_i = 0$ and $u_i \leq UB - 1 \wedge u_i \geq LB - 1$ allows to satisfy Eq. (5). If $T < 1$, setting $u_i = 0$ and $v_i \leq 1 - LB \wedge v_i \geq 1 - UB$ allows to satisfy Eq. (5). Note that for a given $i$, we are free to set $u_i$ and $v_i$ to arbitrary positive values, as long as they satisfy Eq. (5), as they don't affect the optimal solution value nor the values of other variables.

Thus, adding $u_i \leq UB - 1$ and $v_i \leq 1 - LB$ as constraints to $\mathrm{P}_3(\boldsymbol{Q})$ does not affect its optimal solution. Consequently, setting $M_{u_i} \geq UB - 1$ and $M_{v_i} \geq 1 - LB$, are valid big-$M$ constraints in the mixed-integer reformulation of the complementary slackness constraints $Eq.$ (7). The $UB$ and $LB$ values depend on the sign of $y_i$, $y_j$ and the bounds on $\alpha_j$ and $\widetilde{Q}_{ij}$ and the right terms in Eqs. (28) to (31) represent the respective $UB$ and $LB$ arising. This concludes the proof. □

# H Additional experimental details

## H.1 Datasets

The CSBM implementation is taken from [Gosch et al., 2023] publicly released under MIT license. Cora-ML taken from [Bojchevski and Günnemann, 2018] is also released under MIT license. Cora-ML has 2995 nodes with 8416 edges, and 7 classes. It traditionally comes with a 2879 dimensional discrete bag-of-words node feature embedding from the paper abstract. As we focus on continuous perturbation models, we use the abstracts provided by [Bojchevski and Günnemann, 2018] together with all-MiniLM-L6-v2, a modern sentence transformer from `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2` to generate 384-dimensional continuous node-feature embeddings. From Cora-ML, we extract the subgraph defined by the two most largest classes, remove singleton nodes, and call the resulting binary-classification dataset Cora-MLb. It has 1235 nodes and 2601 edges.

**Random Graph Model: CSBM.** A CSBM graph $\mathcal{G}$ with $n$ nodes is iteratively sampled as: (a) Sample label $y_i \sim Bernoulli(1/2)\ \forall i \in [n]$; (b) Sample feature vectors $\mathbf{X}_i | y_i \sim \mathcal{N}(y_i \boldsymbol{\mu}, \sigma^2 \mathbf{I}_d)$; (c) Sample adjacency $A_{ij} \sim Bernoulli(p)$ if $y_i = y_j$, $A_{ij} \sim Bernoulli(q)$ otherwise, and $A_{ji} = A_{ij}$. Following Gosch et al. [2023] we set $p, q$ through the maximum likelihood fit to Cora [Sen et al., 2008] ($p = 3.17\%$, $q = 0.74\%$), and $\boldsymbol{\mu}$ element-wise to $K\sigma/2\sqrt{d}$ with $d = \lfloor n/\ln^2(n) \rfloor$, $\sigma = 1$, and $K = 1.5$, resulting in an interesting classification scheme where both graph structure and features are necessary for good generalization. As mentioned in the main text, we sample $n = 200$ and choose 40 nodes per class for training, leaving 120 unlabeled nodes. Note that we do not need a separate validation set, as we perform 4-fold cross-validation (CV) for hyperparameter tuning.

## H.2 Architectures

We fix $\mathbf{S}$ to row normalized adjacency $\hat{\mathbf{D}}^{-1}\hat{\mathbf{A}}$ in GCN and SGC [Sabanayagam et al., 2023], and symmetric normalized adjacency $\hat{\mathbf{D}}^{\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{\frac{1}{2}}$ in APPNP as per the implementation in the respective work, where $\hat{\mathbf{D}}$ and $\hat{\mathbf{A}}$ are degree and adjacency matrices of the given graph $\mathcal{G}$ with an added self-loop. For CSBMs we fix $C = 0.01$ for comparability between experiments and models in the main section. We find that changing $C$ has little effect on the accuracy but can strongly affect the robustness of different architectures. Other parameters on CSBM and all parameters on real-world datasets are set using 4-fold cross-validation.

We outline the hyperparameters for Cora-MLb, for CSBM all parameters have been mentioned above except the Skip-$\alpha$ for GCN Skip-$\alpha$ and $\alpha$ for APPNP, which both have been set to 0.2.

- GCN (Row Norm.): $C = 0.75$
- GCN (Sym. Norm.): $C = 1$
- SGC (Row Norm.): $C = 0.75$

- SGC (Sym Norm.): $C = 0.75$
- APPNP (Sym. Norm.): $C = 1$, $\alpha = 0.1$
- MLP: $C = 0.5$
- GCN Skip-$\alpha$: $C = 1$, $\alpha = 0.1$
- GCN Skippc: $C = 0.5$

For Cora-ML, the following hyperparameters were set:

- GCN (Row Norm.): $C = 0.05$
- SGC (Row Norm.): $C = 0.0575$
- MLP: $C = 0.004$

again chosen by 4-fold cross-validation, but choosing the result with lowest $C$ in the standard deviation of the best validation accuracy, to reduce runtime of the MILP certification process.

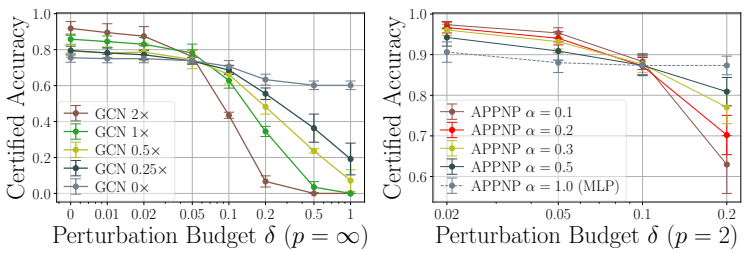
### H.3 Hardware

Experiments are run on CPU using Gurobi on an internal cluster. Experiments for CSBM, Cora-MLb do not require more than 15GB of RAM. Cora-ML experiments do not require more than 20GB of RAM. The time to certify a node depends on the size of MILP as well as the structure of the concrete problem. On our hardware, for CSBM and Cora-MLb certifying one node typically takes several seconds up to one minute on a single CPU. For Cora-ML, certifying a node can take between one minute and several hours ($\leq 10$) using two CPUs, depending on the difficulty of the associated MILP.

## I Insights on the role of graph structure and architectures

We exemplify study directions enabled through our certification framework. By leveraging CSBMs, we study the effect of graph connectivity in the poisoning unlabeled setting in Fig. 4a for GCN. Interestingly, we observe an inflection point at perturbation strength $\delta = 0.05$, where higher connectivity leads to higher certified accuracy against small perturbations, whereas higher connectivity significantly worsens certified accuracy for strong perturbations. These trends are consistent across various architectures and attack settings as presented in App. K.2.

Secondly, we study the effect of different $\alpha$ choices in APPNP on its certified accuracy in poison labeled setting in Fig. 4b. Interestingly, it also shows an inflection point in the perturbation strength ($\delta = 0.1$), where higher $\alpha$ increases the provable robustness for larger $\delta$, whereas worsens the provable robustness for smaller $\delta$ in Cora-MLb. Notably, this phenomenon is unique to the $PU$ setting (see App. L.2) and is similarly observed in CSBM as shown in App. K.2. Although this setup seems to be similar to the connectivity analysis, it is different as the $\alpha$ in APPNP realizes weighted adjacency rather than changing the connectivity of the graph, that is, increasing or decreasing the number of edges in the graph. We compare different normalization choices for $\mathbf{S}$ in GCN and SGC in App. L.3. Through these analyses, it is significant to note that our certification framework enables informed architectural choices from the perspective of robustness.

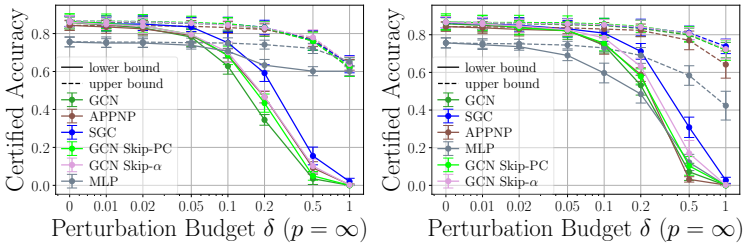



(a) CSBM: $PU$ (b) Cora-MLb: $PU$

Figure 4: (a) Graph connectivity analysis where $c\times$ is $cp$ and $cq$ in CSBM model. (b) APPNP analysis based on $\alpha$.

# J Tightness of QPCert

The SVM quadratic dual problem is solved using QPLayer [Bambade et al., 2023], a differentiable quadratic programming solver. Thus, for evaluating tightness, we use APGD [Croce and Hein, 2020] with their reported hyperparameters as attack to compute an upper bound on the provable robustness by differentiate through the learning process (QPLayer). The results in Fig. 5 show that the provable robustness bounds are tight for small pertubation budgets $\delta$ but less tight for larger $\delta$, demonstrating one limitation. While theoretically, the NTK bounds are tight (Theorem 3), the approach of deriving element-wise bounds on $\boldsymbol{Q}$ to model $\mathcal{A}$ leading to a relaxation of $\mathrm{P}_3(\boldsymbol{Q})$ can explain the gap between provable robustness and empirical attack. Thus, we are excited about opportunities for future work to improve our approach for modeling $\mathcal{A}$ in the MILP $P(\boldsymbol{Q})$.



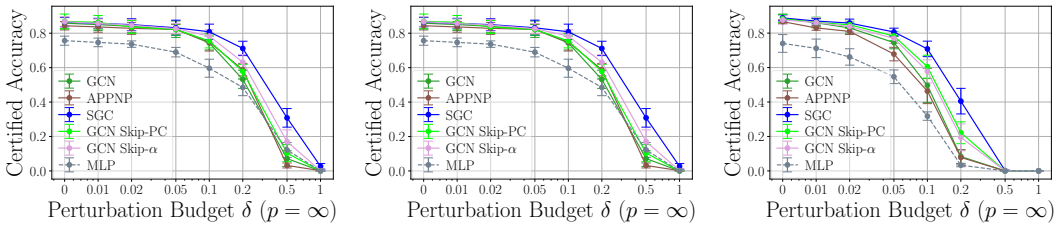

(a) CSBM: $PU$ (b) CSBM: $PL$

Figure 5: Tightness of our certificate. $PU$ and $PL$ with $p_{adv} = 0.2$.

# K Additional Results: CSBM

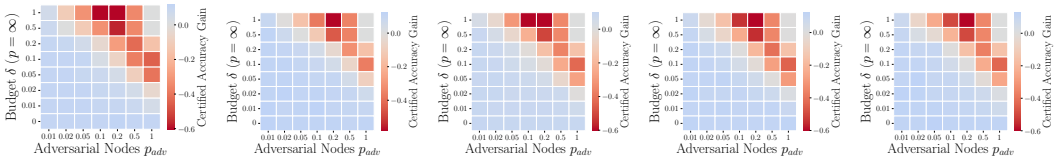
## K.1 Evaluating QPCert and importance of graph information

Fig. 6a shows the same result as Fig. 2a from Sec. 4 establishing that including graph information boosts worst-case robustness in CSBM too. This also shows that the result is not dataset-specific. In Fig. 6, we provide the remaining settings in correspondence to Fig. 3, Poison Labeled $PL$ and Backdoor Labeled $BL$ for CSBM. Similarly, the heatmaps showing the certified accuracy gain with respect to MLP is presented in Fig. 7.




(a) CSBM: $PL$, $p_{adv} = 1$ (b) CSBM: $PL$, $p_{adv} = 0.2$ (c) CSBM: $BL$, $p_{adv} = 0.2$

Figure 6: Certifiable robustness for different (G)NNs in Poisoning Labeled (PL) and Backdoor Labeled (BL) setting.

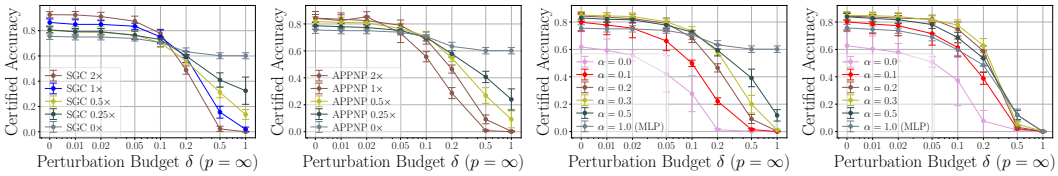



(a) GCN (b) SGC (c) APPNP (d) GCN Skip-PC (e) GCN Skip-$\alpha$

Figure 7: Heatmaps showing certified accuracy gain that is the difference of certified accuracy between GCN and MLP of different GNNs for Poison Unlabeled ($PU$) setting.

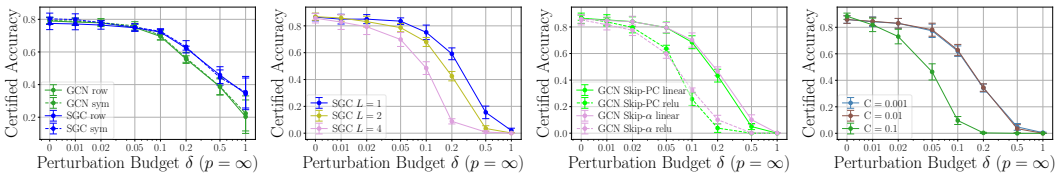
### K.2 On graph connectivity and architectural insights

We present the sparsity analysis for SGC and APPNP in (a) and (b) of Fig. 8, showing a similar observation to GCN in App. I. The APPNP $\alpha$ analysis for $PU$ and $PL$ are provided in (c) and (d) of Fig. 8, showing the inflection point in $PU$ but not in $PL$. Additionally, we show the influence of depth, linear vs ReLU, regularization $C$ and row vs symmetric normalized adjacency in Fig. 9.




(a) SGC, $PU$ (b) APPNP, $PU$ (c) APPNP, $PU$ (d) APPNP, $PL$

Figure 8: (a)-(b): Graph connectivity analysis where $c\times$ is $cp$ and $cq$ in CSBM model. GCN is provided in Fig. 4a. (c)-(d): APPNP analysis based on $\alpha$.

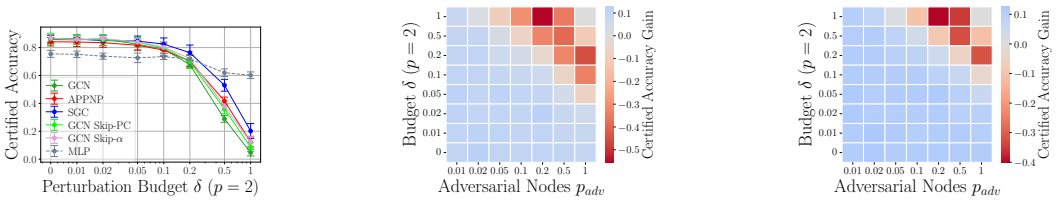



(a) $\mathbf{S}$ in GCN, SGC (b) Hidden layers $L$ (c) Skip connections (d) Regularization

Figure 9: (a): Symmetric and row normalized adjacencies as the choice for $\mathbf{S}$ in GCN and SGC. (b): Effect of number of hidden layers $L$. (c): Linear and relu for the Skip-PC and Skip-$\alpha$. (d): Regularization $C$ in GCN. All experiments in $PU$ setting and $p_{adv} = 0.2$.

### K.3 Results for $p = 2$ perturbation budget

We present the results for $p = 2$ perturbation budget evaluated on CSBM and all the GNNs considered. We focus on Poison Unlabeled setting. Fig. 10 and Fig. 11 show the results for this case. Certifiable robustness for all GNNs and accuracy gain with respect to MLP for GCN and SGC are provided in Fig. 10. Other heatmaps showing the accuracy gain with respect to MLP is in Fig. 11. All the results are in identical to $p = \infty$ setting and we do not see any discrepancy.

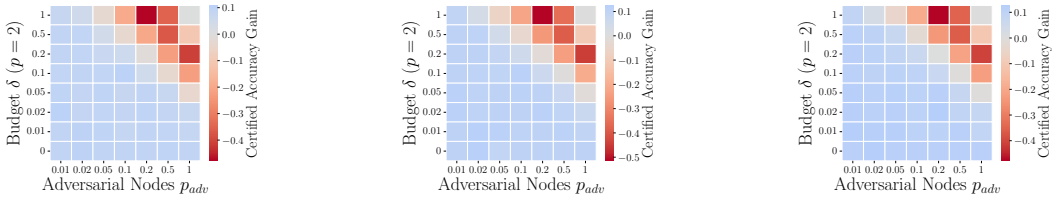



(a) CSBM: $PU$ (b) GCN (c) SGC

Figure 10: (a): Certifiable robustness for different (G)NNs in Poisoning Unlabeled (PU). (b)-(c): Certified accuracy gain for GCN and SGC. All experiments with Poisoning Unlabeled (PU) and $p_{adv} = 0.2$

### K.4 Comparison between $p = \infty$ and $p = 2$

We provide a comparison between $p = \infty$ and $p = 2$ perturbation budget, showing that $p = 2$ is tighter than $p = \infty$ for the same budget as expected.




(a) APPNP (b) GCN Skip-PC (c) GCN Skip-$\alpha$

Figure 11: Heatmaps of different GNNs for Poison Unlabeled $(PU)$ setting.

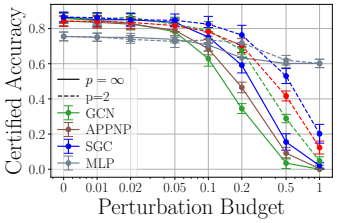
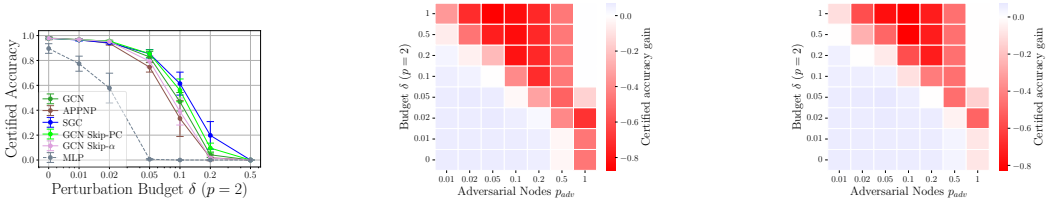



Figure 12: Comparison between $p = \infty$ and $p = 2$ for Poison Unlabeled setting. $p_{adv} = 0.2$.

# L Additional results: Cora-MLb

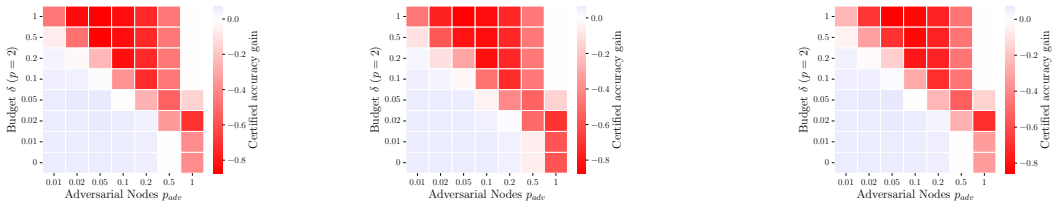
## L.1 Evaluating QPCert

Fig. 13a shows the certified accuracy on Cora-MLb for the $BL$ settings for $p_{cert} = 0.1$. Figs. 13b, 13c and 14 show a detailed analysis into the certified accuracy difference of different GNN architectures for PU setting for $p_{cert} = 0.1$.




(a) BL, $p_{adv} = 0.1$ (b) GCN (c) SGC

Figure 13: (a) Backdoor Labeled $(BL)$ Setting. (b)-(c) Heatmaps of GCN and SGC for Poison Unlabeled $(PU)$ setting on Cora-MLb with $p_{adv} = 0.1$.

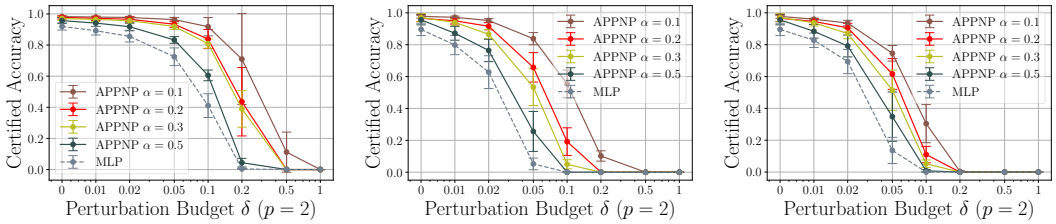



(a) APPNP (b) GCN Skip-$\alpha$ (c) GCN Skippc

Figure 14: Heatmaps of APPNP, GCN Skip-$\alpha$ and GCN Skippc for Poison Unlabeled $(PU)$ setting on Cora-MLb with $p_{adv} = 0.1$.

## L.2 APPNP

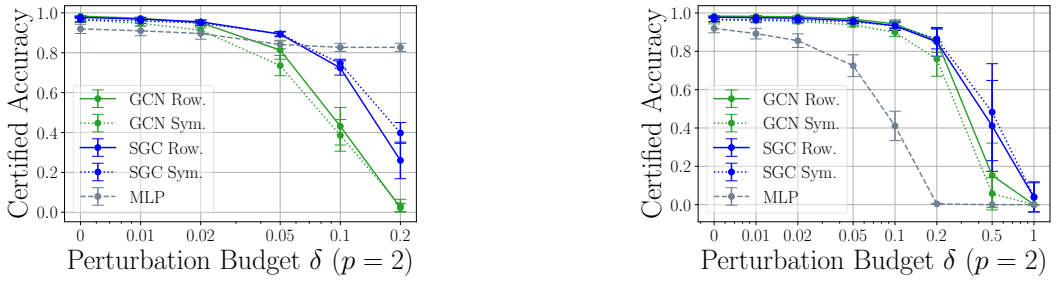
Fig. 15 shows that the inflection point observed in Fig. 4b is not observed in the other settings.




(a) PL (b) BL (c) BU

Figure 15: Cora-MLb, all settings with $p_{adv} = 0.05$.

### L.3 Symmetric vs. row normalization of the adjacency matrix

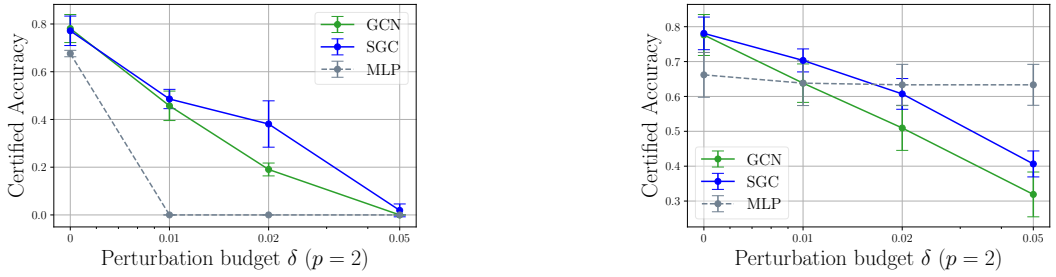



(a) $\mathbf{S}$ in GCN, SGC, $PU$ (b) $\mathbf{S}$ in GCN, SGC, $PL$

Figure 16: Influence of symmetric and row normalized adjacency in GCN and SGC for poison unlabeled and poison labeled settings.

## M Additional results: Cora-ML

For Cora-ML we choose 100 test nodes at random and investigate in Fig. 17a the poison labeled $(PL)$ setting with a strong adversary $p_{adv} = 1.0$ for GCN, SGC and MLP. It shows that QPCert can provide non-trivial robustness guarantees even in multiclass settings. Fig. 17b shows the results for poison unlabeled $(PU)$ and $p_{adv} = 0.05$. Only SGC shows better worst-case robustness than MLP. This, together with both plots showing that the certified radii are lower compared to the binary-case, highlights that white-box certification of (G)NNs for the multiclass case is more challenging for QPCert and an interesting direction for future research.




(a) Cora-ML PL $p_{adv} = 1.0$ (b) Cora-ML PU $p_{adv} = 0.05$

Figure 17: Cora-ML results for PL and PU.

## N Related works

**Poisoning certificates.** The literature on poisoning certificates is significantly less developed than certifying against test-time (evasion) attacks [Li et al., 2023] and we provide an overview in Table 2.

Table 2: Representative selection of data poisoning and backdoor attack certificates. To the best of our knowledge, it contains all white-box works. Our work presents the ***first*** white-box certificate applicable to *(graph) neural networks* and *Support Vector Machines* (SVMs). Poisoning refers to (purely) training-time attacks. A backdoor attack refers to joint training-time and test-time perturbations. Certificates apply to different attack types: $(i)$ Clean-label: modifies the features of the training data; $(ii)$ Label-flipping: modifies the labels of the training data; $(iii)$ Joint: modifies both features and labels; $(iv)$ General attack: allows (arbitrary) insertion/deletion, i.e., like $(iii)$ but dataset size doesn't need to be constant; $(v)$ Node injection: particular to graph learning, refers to adding nodes with arbitrary features and malicious edges into the graph. It is most related to $(iv)$ but does not allow deletion and can't be compared with $(i)$ and $(ii)$. Note that certificates that only certify against $(iii) - (v)$ cannot certify against clean-label or label-flipping attacks individually.

| | | **Deterministic** | **Certified Models** | **Perturbation Model** | | | **Applies to Node Cls.** | **Approach** |
|---|---|---|---|---|---|---|---|---|
| | | | | Pois. | Backd. | Attack Type | | |
| [Ma et al., 2019] | Black Box | ✗ | Diff. Private Learners | ✓ | ✗ | Joint | ✗ | Differential Privacy |
| [Liu et al., 2023] | | ✗ | Diff. Private Learners | ✓ | ✗ | General | ✗ | Differential Privacy |
| [Wang et al., 2020] | | ✗ | Smoothed Classifier | ✗ | ✓ | Joint | ✗ | Randomized Smoothing |
| [Weber et al., 2023] | | ✗ | Smoothed Classifier | ✗ | ✓ | Clean-label | ✗ | Randomized Smoothing |
| [Zhang et al., 2022] | | ✗ | Smoothed Classifier | ✓ | ✓ | Joint | ✗ | Randomized Smoothing |
| [Lai et al., 2024] | | ✗ | Smoothed Classifier | ✓ | ✗ | Node Injection | ✓ | Randomized Smoothing |
| [Jia et al., 2021] | | ✗ | Ensemble Classifier | ✓ | ✗ | General | ✗ | Ensemble (Majority Vote) |
| [Rosenfeld et al., 2020] | | ✓ | Smoothed Classifier | ✓ | ✗ | Label Flip. | ✗ | Randomized Smoothing |
| [Levine and Feizi, 2021] | | ✓ | Ensemble Classifier | ✓ | ✗ | Label Flip./General | ✗ | Ensemble (Majority Vote) |
| [Wang et al., 2022] | | ✓ | Ensemble Classifier | ✓ | ✗ | General | ✗ | Ensemble (Majority Vote) |
| [Rezaei et al., 2023] | | ✓ | Ensemble Classifier | ✓ | ✗ | General | ✗ | Ensemble (Run-Off Election) |
| [Drews et al., 2020] | White Box | ✓ | Decision Trees | ✓ | ✗ | General | ✗ | Abstract Interpretation |
| [Meyer et al., 2021] | | ✓ | Decision Trees | ✓ | ✗ | General | ✗ | Abstract Interpretation |
| [Jia et al., 2022] | | ✓ | k-Nearest Neighbors | ✓ | ✗ | General | ✗ | Majority Vote |
| [Bian et al., 2024] | | ✓ | Naive Bayes Classifier | ✓ | ✗ | Clean-label | ✗ | Algorithmic |
| **Ours** | | ✓ | **NNs & SVMs** | ✓ | ✓ | Clean-label | ✓ | NTK & Linear Programming |

Black-box certificates for poisoning are derived following three different approaches: $(i)$ Randomized smoothing, a popular probabilistic test-time certificate strategy [Cohen et al., 2019], in which randomization performed over the training dataset [Rosenfeld et al., 2020, Weber et al., 2023, Zhang et al., 2022]. Other than data partitioning, a common defense is to sanitize the data, and Hong et al. [2024] certifies diffusion-based data sanitation via randomized smoothing. $(ii)$ Ensembles: Creating separate partitions of the training data, training individual base classifiers on top of them and certifying a constructed ensemble classifier [Levine and Feizi, 2021, Jia et al., 2021, Wang et al., 2022, Rezaei et al., 2023]; Jia et al. [2021] and Chen et al. [2022] offer certificates and collective certificates, respectively, for bagging, while Levine and Feizi [2021] and Wang et al. [2022] derive certificates for aggregation-based methods tailored for black-box classifiers. $(iii)$ Differential Privacy[3] (DP): Ma et al. [2019] show that any $(\epsilon, \delta)$-DP learner enjoys a certain provable poisoning robustness. Liu et al. [2023] extend this result to more general notions of DP. Xie et al. [2023] derives guarantees against arbitrary data poisoning in DP federated learning setup. However, white-box deterministic poisoning certificates remain sparse. Drews et al. [2020] and Meyer et al. [2021] derive poisoning certificates for decision trees using abstract interpretations, while Jia et al. [2022] provides a poisoning certificate for nearest neighbor algorithms based on their inherent majority voting principle. Recently, Bian et al. [2024] derives a poisoning certificate for naive Bayes classification. However, none of these approaches can be extended to NNs.

**Poisoning attacks and defense using the bilevel problem.** The bilevel problem Eq. (3) is investigated by several works in the context of developing a poisoning attack or empirical defense, including for SVMs [Biggio et al., 2012, Xiao et al., 2015, Koh and Liang, 2017, Jagielski et al., 2018]. Notably Mei and Zhu [2015] reformulate the bilevel problem $\mathrm{P}_2(\boldsymbol{Q})$ for SVMs to a bilinear single-level problem similar to $\mathrm{P}_3(\boldsymbol{Q})$ but only solve it heuristically for attack generation and do not realize the possibility of a MILP reformulation and certification. Koh and Liang [2017] also considers the bilevel problem to detect and also generate poisoned samples using influence functions (gradient and Hessian vector product). Other works [Biggio et al., 2012, Xiao et al., 2015] use the bilevel problem with SVM hinge loss and regularized ERM to generate poison samples, and solve it using iterative gradient ascent.

[3]The mechanism to derive a poisoning certificate from a certain privacy guarantee is model agnostic, thus we count it as black-box. However, the calculated privacy guarantees may depend on white-box knowledge.

**Graphs.** Currently, there are no poisoning certificates for clean-label attacks specifically developed for GNNs or the task of node classification. [Lai et al., 2024] is the only work on poisoning certification of GNNs, but differ incomparably in their threat model and are black-box as well as not applicable to backdoors. Lingam et al. [2024] develops a label poisoning attack for GNNs using the bilevel problem with a regression objective and including NTKs as surrogate models. However, there are many works on certifying against test-time attacks on graphs and Günnemann [2022] provides an overview.

We note that [Steinhardt et al., 2017] develops statistical bounds on the loss that are not applicable to certify classification. Robustness to data poisoning can be related to protection of privacy [Ma et al., 2019], but may affect fairness [Ma et al., 2022]. We discuss the broader impact in App. O.

# O Broader impact

Our method represents a robustness certificate for white-box models. This allows a more informed decision when it comes to safety aspects of currently used models. However, insights into worst-case robustness can be used for good but potentially also by malicious actors. We strongly believe that research about the limitations of existing models is crucial in making models safer and thus, outweighs potential risks. We are not aware of any direct risks coming from our work.