Robustness analysis of graph-based machine learning

Henry Kenlay
Worcester College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

7th October, 2022

Abstract

Graph-based machine learning is an emerging approach to analysing data that is or can be well-modelled by pairwise relationships between entities. This includes examples such as social networks, road networks, protein-protein interaction networks and molecules. Despite the plethora of research dedicated to designing novel machine learning models, less attention has been paid to the theoretical properties of our existing tools. In this thesis, we focus on the robustness properties of graph-based machine learning models, in particular spectral graph filters and graph neural networks. Robustness is an essential property for dealing with noisy data, protecting a system against security vulnerabilities and, in some cases, necessary for transferability, amongst other things. We focus specifically on the challenging and combinatorial problem of robustness with respect to the topology of the underlying graph. The first part of this thesis proposes stability bounds to help understand to which topological changes graph-based models are robust. Beyond theoretical results, we conduct experiments to verify the intuition this theory provides. In the second part, we propose a flexible and query-efficient method to perform black-box adversarial attacks on graph classifiers. Adversarial attacks can be considered a search for model instability and provide an upper bound between an input and the decision boundary. In the third and final part of the thesis, we propose a novel robustness certificate for graph classifiers. Using a technique that can certify individual parts of the graph at varying levels of perturbation, we provide a refined understanding of the perturbations to which a given model is robust. We believe the findings in this thesis provide novel insight and motivate further research into both understanding stability and instability of graph-based machine learning models.

Dedication

To Grandma, who always wanted a Dr. in the family.

Declaration

I declare that this thesis is entirely my own work, and except where otherwise stated, describes my own research.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Xiaowen Dong. Your unwavering support, guidance, teachings, advice and encouragement have been instrumental in my journey. I feel privileged to have been among your first cohort of DPhil students. Whenever I visited your office feeling uncertain about my research capabilities, you consistently uplifted me and reinforced my confidence, convincing me that I could overcome any challenge. Your insights and expertise have been invaluable. I could not have completed this journey without you. Furthermore, I would like to thank Stephen Roberts, my co-supervisor, whose insightful discussions and encouragement, particularly in my first year of research, helped shape the direction of my work and inspired me.

Thank you to all my collaborators and colleagues who played an essential part in the research that formed this thesis: Xiaowen Dong, my primary collaborator and source of inspiration; Dorina Thanou, whose expertise and guidance were invaluable; Pierre Osselin, Xingchen Wan, Robin Ru, Arno Blaas, and Michael Osborne, Deborah Sulem, and Mihai Cucuringu who provided critical insights and feedback.

Thank you to the EPSRC and the AIMS CDT for the invaluable opportunity and funding of my studies. The EPSRC's financial support allowed me to focus on my research without the burden of financial constraints. At the same time, the AIMS CDT's interdisciplinary training program broadened my knowledge and skills, enhancing the quality of my research. In particular, I want to express my gratitude to Wendy Poole, the AIMS program administrator, for her unwavering support and help. You made the university bureaucracy much less painful and organised outstanding events for the program. Above all, you have been an incredible and supportive friend along the way.

I met many lifelong friends in the AIMS program, and I'm pleased to have the opportunity to share this journey alongside them. Bernardo Pérez, Tom Pretty, Kyriakos Polymenakos, Stacy Pu, and Pierre Osselin, it was great having you around the lab; you made it a more lively and enjoyable place to work, study, and gossip. I want to thank my friend Vitaly Kurin, with whom I shared many thoughtful and insightful discussions. I will cherish the memories of our many walks around Oxford together. I'll be forever grateful for supporting me at my lowest and help-

ing me immensely during my job search. Tim Rudner, thank you for your advice and support in the first year and your friendship throughout. Rhydian Windsor, thank you for being a great friend and source of inspiration, knowledge and support. And to the many other friends I met during my time in Oxford, including Rebeca Gutiérrez, Hannah Senior, Edd Hornsby, Maddy Malhotra, Siddhant Gangapurwala, Mark Finean, Robert McCraith, Yuki Asano, Matthew Newton, Anna Gautier, Tom Steeples, Prannay Kaul, Saad Hamid, Ivan Kiskin, Baskaran Sripathmanathan, and many others.

I also want to acknowledge friends I met and kept in contact with in previous programs at the University of Cambridge and the University of Warwick and friends I have had the pleasure of keeping in contact with from school. My journey would not have been the same without them.

I was lucky to be accepted to an internship program at Twitter in 2021. I am grateful to Twitter for this opportunity and those who collaborated with me, helped me understand graph machine learning at a deeper level and became my friends during this time: Federico Monti, Nils Hammerla, Emanuele Rossi, Michael Bronstein, Ben Chamberlain, Maria Gorinova, Katarzyna Janocha and Davide Eynard.

During my DPhil, a select group gave me immense support, love and protection. Thank you to my partner Laura Barrera and my closest friends Andy Conboy[1], Bruno Contrino and Frazer I. B. D. Clark.

Finally, I want to thank my family for their persistent encouragement throughout my academic journey despite their unfamiliarity with the path I took. Thank you, Mum, Dad and my stepdad Andy, for raising me and instilling in me the values of perseverance and determination. Any success I have would not have been possible without you. Thank you to all my siblings for supporting my ambitions and ensuring there is never a dull moment in your company. I want to share my appreciation to Sara and Dama, in particular, for opening your home to me during the COVID-19 pandemic, providing a safe and supportive environment for my studies, and Jack for our daily exchanges, which constantly make me laugh, smile, and inspire me.

My DPhil journey would not have been possible without the above-mentioned individuals and many others whose names I may not have mentioned. Each of you has played a significant role in shaping my academic path, and for that, I am truly grateful.

[1]Love you m9.

Contents

Chapter 1

Introduction

A graph is a general-purpose data structure that uses edges to model dyadic interactions between entities, which are modelled as the nodes (also called vertices) of the graph. Many types of data in the real world reside or can be modelled as residing on graph domains, such as those collected in sensor, biological, and social networks.

In the last decade, the signal processing and machine learning community have developed a plethora of tools to analyse and learn from data that resides on graph-structure domains. The graph signal processing (GSP) community consider graphs as irregular domains on which signals live and have analysed this data by generalising and adapting signal processing ideas to develop tools such as spectral graph filters and spectral graph neural networks [36, 111, 101, 50, 121, 43, 100]. On the other hand, the deep learning community has approached learning on graphs by designing neural networks such as those based on message passing which are suitable to the graph domain, in particular by considering desirable invariance and equivariance properties [18, 57, 7, 62, 19].

Despite the ever-increasing modelling approaches and architectures designed to operate over graph-structured data, less attention has been paid to their properties and characteristics. Few papers are dedicated to the theoretical analysis of these models. Even from an empirical perspective, most models are assessed on just their predictive accuracy.

In this thesis, we focus on the robustness properties of graph-based machine learning models, including spectral graph filters, which are the dominant tool from the graph signal processing community and graph neural networks, which are the most common tool within the graph machine learning community.

1.1 Motivation

Robustness can take on many meanings, so we begin by providing a high-level definition adopted in this thesis. Specifically, we aim to understand the effect of small

perturbations on the input graph at inference time. We call a model robust to said perturbations if the perturbations do not create large changes in the output of the model. We will consider and discuss other works that take on different definitions of robustness and outline their meaning in the context. More broadly, robustness covering our definition and other contexts means small changes to the machine learning pipeline give small changes in other parts of the machine learning pipeline.

There are many motivations for studying robustness with respect to the underlying graph. We provide a non-exhaustive list providing a practical real-world example for each.

1. **Graph inferred from noisy data.** Activity in the brain can be modelled by a functional brain network, whereby nodes represent brain regions of interest and edges are inferred from time series data generated by functional magnetic resonance imaging (fMRI). In this case, we expect the time series data to contain aleatoric uncertainty, which in turn causes the inferred graph to be noisy and potentially unreliable[1]. Because this topological noise is due to unwanted noise in the original data, robustness to the noise in the graph would be desirable.

2. **Robustness to adversaries.** Graph-based models may be deployed in environments where bad actors are present. For example, online social networks can be modelled as a graph, and the presence of bad actors is well established in this domain. Bad actors may attempt to manipulate the model with purposefully crafted perturbations by attempting to add edges (by sending connection requests to other accounts) or adding nodes (by creating fake profiles). Here, we would desire our models to perform well even in the worst case (i.e. when bad actors abuse the model).

3. **Transferability.** Consider a physical object digitised using a 3D scanner in the form of a point cloud. This point cloud can be turned into a graph using a triangulation algorithm to generate a mesh graph. In this case, the graph conveniently represents the geometric object. However, two scans of the same object can give different mesh graphs due to slight variations in the environment, such as lighting or the precise position of the camera or object. In these cases, the graph representation of the underlying entity is not unique. However, it would be desirable for a model to produce the same or similar representations for graphs representing the same underlying entity. This property is more broadly known as transferability. Robustness is necessary for transfer-

[1] Aleatoric uncertainty is uncertainty due to noise inherent to the observations as opposed to uncertainty in the model.

ability because failures of a model to adapt to even small changes in the input mean we cannot expect it to adapt to large changes.

4. **Evolving graphs.** Temporal graphs are graphs where the topology gradually changes through time. For example, a financial network can be modelled through correlations of the underlying asset prices. Because assets may become more or less correlated over long periods, the graph topology will evolve over time. In this case, it is usually desirable for representations given by the model to also change gradually through time.

5. **Partial observability.** Sometimes the graph data we have only partially represents the underlying relationship between the nodes in the graph. For example, an online social network gives us an approximation of real-world friendships. It is an approximation because people who are not friends in real life may connect on the social network, and friends in real life may not yet have connected on the online platform. In many cases, we desire learned representations to capture people's real-world friendships despite only having partial knowledge of friendships through the online friendship graph.

Despite the many motivations to study robustness to changes in the underlying topology, there is scarce research dedicated to understanding these robustness properties of graph-based machine learning and graph signal processing models. In this thesis, we present five technical chapters which address this limitation of understanding. We will consider the robustness of topological perturbation of many commonly used models through various perspectives. We will address the mostly unexplored yet important challenge of linking topological properties to stability, i.e. how changes in specific characteristics of the graph can influence robustness.

1.2 Outline and Contributions

This thesis consists of three distinct approaches to analysing robustness. In the first part (Chapters 3, 4 and 5), we construct and analyse Lipschitz-like bounds, which we call stability bounds, for spectral graph filters and graph neural networks. In these chapters, we primarily consider the stability of the overall node embeddings generated by these models, which are an intermediary representation for many downstream graph learning tasks. We will hone our focus on graph classification tasks for the second and third parts of the thesis.

In the second part of the thesis (Chapter 6) we consider a data driven approach to generate small perturbations that change the predicted label of a trained graph classifier. In the machine learning community, these perturbed inputs are referred to

as adversarial attacks in the evasion setting. A successful attack provides an upper bound on the minimum distance between a sample and the decision boundary.

In the third and final part of the thesis (Chapter 7), we consider robustness certificates, a verification technique which gives a lower bound of the distance between a sample and the decision boundary. To circumvent the difficulty of computing such a certificate, we consider the framework of randomised smoothing. In this framework, a smoothed classifier is built by applying a noise distribution over the input and computing probabilistic certificates for the smoothed classifier.

The specific contributions of each chapter are as follows.

- In Chapter 3, we prove that polynomial spectral graph filters are stable to perturbations in the underlying graph by considering them as functions as matrices and proving Lipschitz-like bounds. The Lipschitz constant is in terms of the model parameters and characterises the flexibility of the function, and the maximum level of instability one may observe through perturbation of the input graph. We experimentally highlight how topological properties can influence stability. This chapter is based on ***Kenlay, H.**, Thanou, D., & Dong, X. (2020). On the stability of polynomial spectral graph filters. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

- In Chapter 4, we give a tighter bound for a broader class of spectral graph filters. Furthermore, we develop an interpretable bound which gives a theoretical motivated interpretation of when spectral graph filters are stable. We conduct extensive experiments to validate intuition gained from the bound. This chapter is based on ***Kenlay, H.**, Thanou, D., & Dong, X. (2021). Interpretable stability bounds for spectral graph filters. International Conference on Machine Learning (ICML).*

- In Chapter 5, we prove stability bounds for graph neural networks when edge rewiring perturbations are applied to the underlying graph. Then, similar to Chapter 4, we extend these bounds to give an intuitive interpretation for when graph neural networks are stable to perturbations due to rewiring. This chapter is based on ***Kenlay, H.**, Thanou, D., & Dong, X. (2021). On the stability of graph convolutional neural networks under edge rewiring. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

- In Chapter 6 we propose a black-box algorithm to generate adversarial examples in the context of graph classifciation. Although often motivated through the lens of security, we believe adversarial examples are of interest to researchers interested in stability as a method to search for instabilities. Our

approach is based on Bayesian Optimisation. We provide a quantitative analysis of the adversarial examples generated by our proposed technique. This part of thesis is presented in Chapter 6 and is based on *Wan, X., **Kenlay, H.**, Ru, B., Blaas, A., Osborne, M. A., & Dong, X. (2021). Adversarial Attacks on Graph Classification via Bayesian Optimisation. Advances in Neural Information Processing Systems (NeurIPS).*

- In Chapter 7 we propose a structure-aware robustness certificate based on randomised smoothing. Existing approaches give pessimistic certificates by considering the distance of the graph to the decision boundary in all directions equally. We build upon existing approaches by considering certificates that certify collections of node pairs in the graph to potentially different magnitudes, which leads to more refined certificates and better understanding of the data point and its relation to the decision boundary. This part of thesis is presented in Chapter 7 and is based on *Osselin, P.*, **Kenlay, H.***, & Dong, X. (2023). Structure-aware robustness certificates for graph classification. Uncertainty in Artificial Intelligence (UAI)*

The ideas, figures and content of the technical chapters in the first part (Chapters 3, 4 and 5) originate predominantly from me. In Chapter 6, I contributed to the data processing, implementation and training of graph machine learning models and assisted with experimental setup and writing of the manuscript. In Chapter 7, I contributed to developing the theoretical framework, assisted in proving and refining the theoretical results, writing the code and conducting experiments, generating the figures and assisted in the writing. Furthermore, I contributed to the following works not presented in this thesis over the duration of the DPhil.

- Rossi, E., **Kenlay, H.**, Gorinova, M. I., Chamberlain, B. P., Dong, X., & Bronstein, M. (2022). On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. Learning on Graphs Conference (LoG).

- Sulem, D., **Kenlay, H.**, Cucuringu, M., & Dong, X. (2024). Graph similarity learning for change-point detection in dynamic networks. Machine Learning.

Before the technical chapters, we provide preliminary material and survey relevant literature in Chapter 2. Finally, we conclude the thesis in Chapter 8.

*Both authors contributed equally to this work.

Chapter 2

Background

In this chapter, we briefly introduce some background topics, including standard definitions and notation that will frequently occur throughout the rest of the thesis. We also provide a review of the literature related to the technical chapters of the thesis.

2.1 Preliminaries

2.1.1 Graphs

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a finite set of vertices (also called nodes) $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V}^2$ which represent a general pairwise relationship between the vertices. We will write $\mathcal{V}(\mathcal{G})$ and $\mathcal{E}(\mathcal{G})$ to denote the vertices and edges of a specific graph $\mathcal{G}$. The order of the graph $n = |\mathcal{V}|$ is the number of vertices. We will say a graph is undirected if $(u, v) \in \mathcal{E} \iff (v, u) \in \mathcal{E}$ and write $u \sim v$ to denote that an edge exists between vertices u and v. A self-loop is an edge from a node to itself. A weight can be assigned to edges via a weight function $w : \mathcal{E} \to \mathbb{R}_{\geq 0}$ to give a weighted graph. An unweighted or binary graph is a graph without edge weights, which can also be thought of as assigning an edge weight of 1 to all edges. Unless specified otherwise, we will be concerned with undirected and unweighted graphs with no self-loops. A graph with these properties is called a simple graph. By fixing an ordering of the vertices, we can represent a graph with an $n \times n$ adjacency matrix $\mathbf{A}$ such that $\mathbf{A}_{u,v} = w((u, v))$ if an edge exists between nodes u and v[1].

Two graphs $\mathcal{G}$ and $\mathcal{G}'$ are isomorphic if there exists an bijective edge-preserving map $\phi : \mathcal{V}(\mathcal{G}) \to \mathcal{V}(\mathcal{G}')$. An edge-preserving map is a map such that $(u, v) \in \mathcal{E}(\mathcal{G})$ if and only if $(\phi(u), \phi(v)) \in \mathcal{E}(\mathcal{G}')$. The open neighbourhood, or just neighbourhood, of a vertex u in an undirected graph $\mathcal{G}$ is given by vertices that are adjacent $\mathcal{N}(u) = \{v \in \mathcal{V} : u \sim v\}$. The closed neighbourhood of a node u is the set containing the

[1] We have, without loss of generality, assumed the vertices are given by $1, \ldots, n$.

neighbourhood nodes and the central node $\mathcal{N}[u] = \mathcal{N}(u) \cup \{u\}$. The degree d_u of a node u is the sum of weights assigned to edges between the node u and adjacent nodes. For unweighted graphs this is equivalently the size of its neighbourhood $d_u = |\mathcal{N}(u)|$. The degree matrix $\mathbf{D}$ of a graph is defined as $\mathbf{D} = \text{diag}((d_1, \ldots, d_n))$ where the diag operator maps a vector of size n to an $n \times n$ matrix with zeros on the off-diagonal and the vector along the diagonal. A node with degree zero is called an isolated node.

2.1.2 Graph signal processing

Recently, graph signal processing (GSP) has emerged as a field which extends high dimensional data analysis to graph-structured data [36, 117, 101, 121, 43, 100]. One of the challenges in graph signal processing is adapting ideas from the signal processing literature and generalising them to signals that reside in a graph-structured domain. A (scalar) graph signal is a mapping $x : \mathcal{V} \to \mathbb{R}$ from nodes to some scalar value which represents the signal value on that node[2]. We will refer to this node-wise representation as the spatial representation. When the vertex order is fixed, we can compactly write the signal as $\mathbf{x} \in \mathbb{R}^n$ where $\mathbf{x}_u$ is the signal value on node u. Vector-valued signals can be represented by a signal (or feature) matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. Each column is a scalar graph signal, also called a feature map in the machine learning literature. Each row $\mathbf{X}_u$ represents a d-dimensional vector signal for node u.

Graph shift operator

The shift operator is a fundamental operation in discrete-time signal processing. Given some discrete-time signal $x : \mathbb{Z} \to \mathbb{R}$ the shift operator maps $x(t)$ to $x(t+1)$. Because there is no natural ordering to vertices in a graph, it is not straightforward to generalise the shift operator. A graph shift operator $\boldsymbol{\Delta}$, which maps graph signals to graph signals, is defined as an operator such that the output at node u is a function of the graph signal values in the closed neighbourhood of the node.

There are many choices of graph shift operator, with perhaps the simplest being the adjacency matrix. Here, we will consider two graph shift operators based on discrete counterparts to the Laplacian operator. Analogous to the continuous case, this leads to graph Fourier analysis based on expanding a signal in terms of the eigenbasis of the Laplacian operator [117]. The combinatorial Laplacian operator is defined to be

$$\mathbf{L}_{\text{comb}} = \mathbf{D} - \mathbf{A}, \tag{2.1}$$

[2]We use $\mathbb{R}$ for simplicity, but the domain could also be some other field such as the complex numbers.

whereas the normalised Laplacian matrix is defined to be

$$\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{L}_{\text{comb}}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}. \tag{2.2}$$

Throughout this section, we will refer to the normalised Laplacian as the Laplacian matrix. Similar to the continuous case, the Laplacian matrix corresponds to a linear transform which gives us the difference between the value of a signal at a point and the average of its neighbours (after normalising nodes by their degree):

$$(\mathbf{L}\mathbf{x})_u = \frac{1}{\sqrt{d_u}} \sum_{v \in N_u} w((u,v)) \left(\frac{\mathbf{x}_u}{\sqrt{d_u}} - \frac{\mathbf{x}_v}{\sqrt{d_v}} \right). \tag{2.3}$$

Because the graph Laplacian $\mathbf{L}$ is a real symmetric matrix, it is diagonalisable, i.e. it omits a complete set of orthonormal eigenvectors. Furthermore, these eigenvectors and the corresponding eigenvalues are real-valued. Denote the matrix eigendecomposition as $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T$ where $\mathbf{U} = [\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}]$ is a matrix with eigenvectors as columns and Λ is a diagonal matrix of corresponding eigenvalues. The eigenvalues $\{\lambda_i\}_{i=0,\ldots,n-1}$ satisfy $0 = \lambda_0 \leq \lambda_1, \ldots \lambda_n \leq 2$. The multiplicity of the zero eigenvalue is equal to the number of connected components, and $\lambda_{\max} = 2$ if and only if the graph is a bipartite graph [33]. Unlike the normalised Laplacian, the combinatorial Laplacian does not have a bounded spectrum, making the former preferable for certain applications, such as constructing spectral graph filters, which we explore later in this section.

Finally, we will also consider the augmented adjacency matrix defined to be

$$\mathbf{\Delta}_\gamma = \mathbf{D}_\gamma^{-1/2}\mathbf{A}_\gamma\mathbf{D}_\gamma^{-1/2}, \tag{2.4}$$

where $\mathbf{A}_\gamma = \mathbf{A} + \gamma\mathbf{I}$ and $\mathbf{D}_\gamma = \mathbf{D} + \gamma\mathbf{I}$ with $\gamma \geq 0$. When $\gamma = 0$ this matrix is known as the the normalised adjacency matrix. We will often consider the case when $\gamma = 1$ and in this case will write $\mathbf{\Delta}$.

Graph Fourier transform

The combinatorial Laplacian can be used to define the following quadratic form

$$\mathbf{x}^\intercal\mathbf{L}_{\text{comb}}\mathbf{x} = \sum_{u \sim v} w((u,v))(\mathbf{x}_u - \mathbf{x}_v)^2, \tag{2.5}$$

and the normalised Laplacian gives a similar quantity where signal values are normalised the by the node degree

$$\mathbf{x}^\intercal \mathbf{L}\mathbf{x} = \sum_{u \sim v} w((u,v)) \left(\frac{\mathbf{x}_u}{\sqrt{d_u}} - \frac{\mathbf{x}_v}{\sqrt{d_v}} \right)^2. \tag{2.6}$$

By inspecting the right-hand side of these equations, it becomes evident they are measures of signal variation across edges. These quadratic forms can be considered measures of signal smoothness over the underlying graph.

We can use this idea of smoothness to define a Fourier-like basis of signals in the graph domain. We will use the Laplacian matrix to construct a graph Fourier basis, but other graph shift operators can be used [100, Section 3.1.4]. Similar to the classical Fourier basis, the basis vectors become increasingly non-smooth. We can construct a basis that becomes increasingly non-smooth by considering the solutions to the sequential set of optimisation problems

$$\mathbf{u}_k = \underset{\substack{\mathbf{x} \in \mathbb{R}^n \\ \|\mathbf{x}\|_2 = 1 \\ \mathbf{x}^\intercal \mathbf{u}_{k'} = 0,\ k' \in \{0,\dots,k-1\}}}{\operatorname{argmin}} \mathbf{x}^\intercal \mathbf{L}\mathbf{x}. \tag{2.7}$$

Intuitively, the k-th solutions is the smoothest signal (as measured by the Laplacian quadratic form) that is normalised and orthogonal to the $k-1$ solutions already constructed. The Courant–Fischer min-max theorem tells us that these solutions are exactly the Eigenvectors of the Laplacian matrix [66, Theorem 4.2.6]. See [36, 18, 100] for similar derivations and discussions around this interpretation.

Now we have a notion of a Fourier basis over graphs, we can define the graph Fourier transform as

$$\hat{\mathbf{x}} = \mathbf{U}^\intercal \mathbf{x}. \tag{2.8}$$

Similar to classical signal processing, we refer to $\hat{\mathbf{x}}$ as the spectrum of a signal and call this representation of the signal the spectral representation. We can recover a signal from its spectrum using the inverse graph Fourier transform

$$\mathbf{x} = \mathbf{U}\hat{\mathbf{x}} \tag{2.9}$$

By considering a discrete-time signal with periodic boundaries as a signal on a cycle graph, the graph Fourier basis can be seen as a generalisation of the discrete Fourier basis. Indeed, in this case, the graph Fourier basis is equivalent to the discrete Fourier basis [111, Section V.D].

Spectral graph filters

A fundamental operation in the analysis and transformation of signals is filtering. A filter amplifies or attenuates frequencies of the input signal by performing pointwise multiplication of the signal in the frequency domain with a so-called transfer function evaluated at the frequencies.

The convolution theorem from signal processing states that multiplication in the spectral domain is equivalent to convolution in the spatial domain. Having derived an analogue to the Fourier transform on the graph domain, we can use this idea to generalise filtering of signals in a graph-structured domain.

A filter $h : \mathbb{R} \to \mathbb{R}$ is applied by first computing the spectrum of the signal using the graph Fourier transform, followed by piece-wise multiplication of the spectrum and the filter evaluated at the Laplacian eigenvalues, and finally using the inverse graph Fourier transform we recover the filtered signal in the spatial domain[3]. In equation form, the filtered signal $\mathbf{y}$ is computed as

$$\mathbf{y} = \mathbf{U}\,\mathrm{diag}(h(\lambda_1), \ldots, h(\lambda_n))\mathbf{U}^\intercal\mathbf{x} \tag{2.10}$$

where h is applied element-wise to the diagonal entries of Λ. If h can be realised as a matrix-valued function, then by applying the filter directly to the Laplacian, one can bypass the computationally expensive and possibly numerically unstable need to compute the eigendecomposition and compute the filter directly

$$\mathbf{y} = h(\mathbf{L})\mathbf{x}. \tag{2.11}$$

For diagonalisable matrices, such as the Laplacian matrix of a simple graph, this is one of the multiple equivalent definitions of matrix-valued functions [63].

By choosing filters from a suitable functional family, spectral graph filters can be computed efficiently. For example, polynomial filters can be computed via repeated matrix-matrix multiplications. Another example is by [42], who use Chebyshev polynomials whereby the filter can be computed with matrix-vector multiplications. Many real-world graphs are sparse, meaning these linear algebra operations can be computed more efficiently using sparse matrix representations.

2.1.3 Graph neural networks

Graph neural networks (GNNs) are architectures based on neural networks which can be applied to graph-structured data. Most architectures consist of rounds of graph convolutional layers, which transform the node representations. For some tasks, such

[3] we use $\mathbb{R}$ as the domain of the filter for simplicity but the filter only needs to be defined on the eigenvalues of the Laplacian.

as graph classification, a pooling layer is used to turn node embeddings into a graph embedding. Graph neural networks are often designed from a spatial or spectral perspective. In the spatial perspective, data is aggregated across the graph via message passing [19, Section 5.3]. On the other hand, the spectral perspective uses spectral graph filters as building blocks [50]. There is an equivalence between graph convolutions regardless of if they are designed in the spatial or spectral domain [4]. We cover some of the most influential architectural designs appearing in this thesis's subsequent technical chapters. An overview of the many proposed architectures can be found in the following surveys [147, 162]. General graph neural networks frameworks are considered in [57, 7, 19].

ChebNet

A graph convolution layer based on Chebyshev polynomials was proposed by Defferrard, Bresson, and Vandergheynst [42]. In many follow up works architectures based on this design are referred to as ChebNet or ChebyNet. The graph convolution is defined by applying a parametric polynomial spectral graph filter to the signal feature maps. The authors use Chebyshev polynomials to reduce the time complexity of applying the filtering operation. A scaled version of the normalised Laplacian is used $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_n - \mathbf{I}_n$. By using the recurrence relation of Chebyshev polynomials we can compute the following terms

$$\mathbf{y}_k = \begin{cases} 2\tilde{\mathbf{L}}\mathbf{y}_{k-1} - \mathbf{y}_{k-2} & \text{if } k \geq 2 \\ \tilde{\mathbf{L}}\mathbf{x} & \text{if } k = 1 \\ \mathbf{x} & \text{if } k = 0 \end{cases}. \tag{2.12}$$

The filter output for a K order polynomial is given by $\mathbf{y} = g_\theta(\mathbf{x}) = [\mathbf{y}_0, \ldots, \mathbf{y}_{K-1}]$ where $\theta \in \mathbb{R}^K$ are learnable filter parameters. The graph convolution takes as input F_{in} feature maps $\mathbf{X}^{(l-1)} \in \mathbb{R}^{n \times F_{\text{in}}}$ and outputs F_{out} filter maps $\mathbf{X}^{(l)} \in \mathbb{R}^{n \times F_{\text{out}}}$ according to the equation

$$\mathbf{X}^{(l)}_{s,j} = \sigma\left(\sum_{i=1}^{F_{\text{in}}} g_{\theta_{i,j}}(\tilde{\mathbf{L}})\mathbf{X}^{(l-1)}_{s,i}\right), \tag{2.13}$$

where $\sigma(\cdot)$ is an activation function.

Graph Convolution Network

By considering a first order (affine) Chebyshev polynomial, followed by a reparameterisation trick, the graph convolution network (GCN) layer is given by

$$\mathbf{X}^{(l)} = \sigma\left(\mathbf{\Delta}\mathbf{X}^{(l-1)}\mathbf{\Theta}^{(l)}\right) \tag{2.14}$$

where $\mathbf{\Delta}$ is the augmented adjacency matrix and $\mathbf{\Theta}^{(l)} \in \mathbb{R}^{F_{\text{in}},F_{\text{out}}}$ are the parameters of the layer. The GCN layer can be interpreted as a fixed low-pass filtering operation followed by a learnable linear node feature update [144, 98].

Graph Isomorphism Network

Xu et al. [151] consider the expressivity of GCN by considering how well a GCN model could distinguish (output unique embeddings) two non-isomorphic graphs. The authors proved GCN is less powerful than the Weisfeiler Lehman (WL) test in the sense that if the GCN can distinguish two graphs then so can the WL-test, but there exist non-isomorphic graphs that the WL-test can distinguish but that GCN cannot. The authors propose the Graph Isomorphism Network (GIN) which has the same expressive power as the WL-test. The GIN layer is defined as

$$\mathbf{X}^{(l)} = \text{MLP}^{(l)}\left((1+\epsilon^{(l)})\mathbf{X}^{(l-1)} + \mathbf{A}\mathbf{X}^{(l-1)}\right), \tag{2.15}$$

where $\epsilon^{(l)}$ is a learnable scalar parameter and $\text{MLP}^{(l)}$ is a multi-layered perceptron.

2.2 Literature review

In this section, we provide a high-level and broad literature review covering robustness of graph-based machine learning. More specific and detailed discussions are presented in individual chapters.

2.2.1 Stability

Some of the earlier works considering stability analysis of spectral graph filters and graph analysis tools with respect to edge additions and deletions include stability analysis of centrality measures [112], filtering over random time varying graphs [69], statistical approximation of the characterisation of the network topology changes under small changes [25] and graph signal processing in the presence of uncertainty of the graph topology [24]. These works mostly consider stability to random noise and provide probabilistic bounds. We will consider stability bounds which include worst case perturbations.

One of the earlier works on stability bounds was by Levie et al [86]. In this work, the authors give an upper bound of change which they claim grows linearly in the distance between the graph shift operators under the operator norm before and after perturbation for a large class of spectral graph filters. The bound is based on analysis in the Cayley smoothness space (which includes polynomial filters). The main limitations of this result is that the constant which depends on the filter is not easily interpretable and the bound is only valid for sufficiently small perturbation. In Chapter 3 we overcome the limitation of requiring small perturbation and in Chapter 4 we extend this work further to give an structurally interpretable elucidating the kinds of topological changes spectral graph filters are stable to.

Another approach to stability is to consider a graph as a discretisation of some underlying continuous space. Levie et al. [88] consider weighted graphs to be discretisation of an underlying topological space. Under the assumption that a graph and a perturbed graph are discretised from the same underlying space, they prove that graph neural networks are transferable. Using this result, they prove robustness to small perturbations of the graph.

Gama, Bruna and Ribeiro [47, 48] prove that a class of spectral graph filters are stable to changes in the graph topology. Furthermore, they prove spectral graph neural networks using filters in this class as filter banks and ReLU non-linearities [50] are also stable. This line of work builds on previous stability analysis of graph scattering transforms [49, 163], a multi-scale deterministic feature representation of graph signals. The main difference between the aforementioned line of work and the work presented in this thesis is how the magnitude of perturbation is measured. In [47] the authors posit that simple additive error as a measure does not reflect the fact that the distance between isomorphic graphs can be non-zero. To address this concern they propose Relative Perturbation Modulo Permutation to consider a relative measure of perturbation considering all node permutations for the perturbed graph. This measure is at least as hard to compute as the graph isomorphism test for which no polynomial time algorithm is known [3]. We believe that the additive error approach of Chapters 3, 4 and 5, which does not consider node permutations, and the approach of [47] are both useful, depending on the specific use case. For example, if the graph represents a polygon mesh, then the node labelling in the perturbed graph does not have any meaningful interpretation, as they are just used to construct matrix representations. In this case, considering permutations is appropriate. However, in a social network, the node labelling will typically correspond to the identification of a user, in which case it makes sense to consider the labelling as fixed between the original and perturbed graph.

Graphon neural networks are introduced by Ruiz, Chamon, and Ribeiro [110] building on the foundations of graphon signal processing [110, 109]. A graphon

is a symmetric function $W : [0,1] \times [0,1] \to [0,1]$ which can be thought of as a weighted undirected graph on uncountable nodes. A graphon is the limit object of a sequence of graphs, and the graphon has the same limiting homomorphism density and thus captures the structural properties of the graph sequence. The authors prove that under mild assumptions graphon neural networks approximate graph neural networks in the sense that the output of the graphon neural network on a graphon and continous signal approximates the output of the graph neural networks with the same filters applied to a graph and signal sampled from their continous counterparts. Furthermore, under the same mild assumptions, if two distinct graphs and corresponding signals are sampled then the representations produced by the graph neural network converge as the sampled graphs increase in size.

In a follow up work Ruiz, Chamon, and Ribeiro [109] prove that graphon neural networks are stable to perturbations to the graphon. By combining this with the results of [110], they show that sampled graph neural networks are stable to the corresponding discrete sampling of the graphon and perturbed graphon. Maskey, Levie, and Kutyniok [93] take a similar approach to [109] and prove a similar result using more relaxed assumptions.

Related to this, Keriven, Bietti, and Vaiter [80] consider the behaviour of graph neural networks on large random graphs. Random graphs in this work are given by nodes sampled from an underlying latent metric space according to some distribution and a kernel which controls the probability of an edge between nodes. Unlike the graphon model, the random graph model considered in [80] can model sparse graphs. The graph signal on a node is a function of the latent variable that generated the node. The authors show convergence between a graph convolutional neural network applied to a random graph and a continous graph convolutional neural network applied to the corresponding latent space. The authors study the stability of GCN with respect the Wasserstein metric, which considers a smallest distance over all node permutations similar to [47]. Furthermore, the authors study robustness of continuous graph neural networks under deformations of the underlying metric space. Numerical experiments backing up the theory of convergence and stability are demonstrated.

In Chapters 3, and 4 we will consider stability bounds for spectral graph filters, whereby we give upper bounds on the output change with respect to a change in the topology. We will consider a similar bound in 5 for graph neural networks. Our approach and theoretical result differ from those outlined in a few aspects. First, our bounds considers general perturbations consisting of edge deletions and additions with no assumptions on how the graphs are generated (e.g. from a discrete sampling of the same continuous space). Furthermore, we provide bounds that have structurally related terms allowing interpretations on the types of perturbation

models are robust to.

2.2.2 Adversarial attack and defence

An adversarial attack is an attempt to fool a machine learning algorithm using deceptive data. Early works in adversarial machine learning consider adversarial attacks against naive Bayes classifiers [39], support vector machines [12] and PCA subspace methods [107].

The works of Szegedy et al. [128] and Biggio et al. [13] highlighted the susceptibility of deep neural networks to adversarial attack. The type of adversarial attack presented by these works are known as an evasion attack. In the evasion attack setting, a small perturbation is made to a data point that causes a pre-trained classifier to misclassify the sample. The perturbed data point is referred to as an adversarial example and the pre-trained classifier is called the victim model. The existence of adversarial examples has implications for security and privacy, amongst other things.

Adversarial attack is a broad term covering a range of specific settings and assumptions. Therefore, we begin by defining some terminology to help categorise approaches, following the taxonomy of [74, Section 3]. Beyond the evation setting described, another type of attack that happens at test time are membership inference attack where the aim is to discover if a specific sample was used in the training set of the model. Alternative to attacks at inference time, there are settings where the attack happens at training time. Poison attacks change the training dataset to hinder the ability for a model to learn parameters with good predictive power. In a backdoor attack the training dataset is modified in a way to trigger a specific unexpected behaviour on some specific types of test point.

Another consideration is how much knowledge we assume the attacking agent has access to. In the most generous case, the attacker has access to the model and the data the model was trained on. In the case of neural networks (or other differentiable models) an attacker in this setting can also exploit gradients in the attack. This is called the white-box setting. On the other hand, the black-box setting is a more restrictive and realistic scenario whereby the attacker can only query the victim model. The grey-box setting lies between these scenarios whereby the attacker has access to some of the information described. For example, the attacker is allowed access to the model but not the training data.

Some final considerations are the type of victim model, which may be designed or trained to be robust. The type of model may also determine which attacks are actually possible (i.e. a gradient based attack cannot be applied to a non-differentiable model). The goal of the attacker, which is also related to the type of attack, may

need to be specified. For example in the evasion setting of a graph classifier, the goal might be to fool the model to predict an incorrect class (untargeted), or it may be to fool the model to predict a specific incorrect class (targeted). Usually some constraint is applied on the type of perturbations an attacker can make. Taking again the evasion setting example in the context of graph classification, there may limitations on the number of edges the attacker can change, what edges and nodes may be modified, constraints on changes to the topological properties and maybe even domain specific limitations. The final consideration is the specific machine learning task (e.g. graph or node classification). We consider common graph learning tasks in the following subsection.

To combat adversarial attacks, adversarial defence research is focused on strategies and algorithms that result in machine learning models which are robust to such attacks. One of the most successful adversarial defence strategies is adversarial training [92]. In adversarial training a network is trained by alternating between minimising the loss function on a training dataset, and generating adversarial examples on the trained classifier and adding these into the training dataset. For further information, we refer the reader to the following survey papers which consider adversarial attack and defence in a general setting [157, 148].

Adversarial attack on graph-based models

One of the earlier works in attacking graph-based models considers two attack methods to fool graph clustering algorithms [32]. The authors consider attacking three graph clustering algorithms (spectral clustering, the Louvain algorithm and node2vec). Waniek et al. [140] consider social networks and how individual nodes lower their network centrality without comprising their influence in the network. A simple heuristic DICE[4] is proposed whereby a small number of edges within communities are removed and a small number of edges between communities are added.

The first adversarial attack designed for graph neural networks in the context of semi-supervised node classification is proposed by Zügner, A, and Günnemann [164]. The authors propose Nettack, an adversarial attack that handles both the evation and poisoning setting and also considers (binary) node features. The authors linearise the victim model and propose a greedy approach to solving a bilevel optimisation problem corresponding to a poison attack. Nettack can fool graph convolutional neural networks with few perturbations whilst preserving feature co-occurrence and degree distribution statistics.

One of the earlier works that propose adversarial attacks to graph classification is by Dai et al. [37]. The authors propose a number of techniques, including RL-

[4] Disconnect Internally Connect Externally

S2V, which uses reinforcement learning to attack both node and graph classifiers in a black-box manner. Furthermore, the authors propose a gradient based attack and a genetic algorithm based attack. Another reinforcement learning adversarial attack ReWatt perturbs graph through a rewiring operation [91]. A white-box optimisation strategy (alternating direction method of multipliers) is proposed in [74].

A number of works use gradients to attack graph based models. Chen et al. [29] use fast gradient attacks on the graph structure. Wu et al. [145] propose an adversarial attack on graphs based on integrated gradients and statistically analyse the topological properties of the resulting adversarial examples. Similarly, Xu et al. [150] use projected gradient descent as a search method for adversarial examples. Zügner and Günnemann [166] propose using meta gradients to generate graph structure poisoning attacks.

Some works have studied the commonalities and patterns in generated adversarial examples. Entezari et al. [46] study adversarial examples generated by Nettack and find only low-valued singular components of the graph adjacency matrix are perturbed. Zügner et al. [167] demonstrated that Nettack tends to insert edges to nodes with low two-hop degree where two-hop degree of a node is defined to be the sum of node degrees in the two hop neighbourhood of the node. Furthermore, edges tend me be added between nodes with different labels, and removed between nodes with same label, mimicking the heuristic of DICE. By considering local (neighbourhood) entropy of labels and features [145] find a variety of adversarial attacks increase entropy. The authors also find the rank of the adjacency increases. In Chapter 6 we propose an adversarial attack for graph classifiers and observe perturbed edges tend to be clustered, modify community structure and perturb edges around low-degree nodes. Jin et al. [74] experiment with a variety of attack methods, and find the attacks tend to add rather than delete edges, increase the rank of the adjacency, and reduce the connectivity of the graph as measured by the clustering coefficient.

Due to the threat of adversarial attacks, methods have been developed to protect graph-based models against attack and architectures have been designed to be inherently robust against attack. Based on the observation that Nettack perturbed low-valued singular components in the adjacency, pre-processing an input graph by taking a low-rank approximation of the adjacency matrix can help protect the model from adversarial attacks [46]. Similarly, GNNGuard modifies the input graph by assigning high weights to edges connecting similar nodes whilst pruning weights between unrelated nodes. Based on the theory of robust statistics, Geisler, Zügner, and Günnemann [54] proposes using a soft median as an aggregation function in graph neural networks and demonstrate superior robustness to adversarial attack.

For a more indepth discussion, there are a number of surveys, reviews and book chapters covering adversarial attack and defence on graphs [126, 31, 148, 74, 61].

2.2.3 Certified robustness

Adversarial attacks motivate research into adversarial defence methods. However, an adversarial defence that is effective to existing adversarial attacks may become vulnerable to future innovations in adversarial attack research. Certified robustness can be motivated as a way of guaranteeing defence to any adversarial attack. The goal of a robustness certificate is to provably guarantee invariance of the model output with respect to any perturbations under some conditions (for example the perturbation is bounded by some specified magnitude).

Early approaches to certified robustness of neural networks involve exact verification solvers. Examples include using Satisfiability Modulo Theories (SMT) solvers [76] or integer programming [130]. These works consider the adversarial polytope, which is the image of the model on all perturbed inputs permitted under the attack model (usually all samples within some small distance of the input). Although able to reason about the adversarial polytope exactly, allowing one to compute optimal certificates, the combinatorial nature of these approaches mean they cannot be scaled to deeper models which can give rise to adversarial polytopes with more linear regions.

Beyond this approach, one can instead reason over convex outer bounds on the adversarial polytope. Raghunathan, Steinhardt, and Liang [102] use semidefinite relaxation to generate a relaxation of the adversarial polytope to train certifiably robust networks. On the other hand Wong and Kolter [143] exploited the theory of duality to train certifiably robust networks. Building on this Gowal et al. [60] use interval bound propagation and showed it to be more scalable and give higher certified accuracy. Unfortunately, these approaches do not scale to deep networks used to solve problems such as ImageNet.

An alternative approach to these methods is known as randomised smoothing, whereby noise is added to the input of a base classifier and the expected output is the output of the smoothed classifier. This technique was proposed by Lecuyer et al. [84] and Singla and Feizi [119], albeit with loose guarantees. Cohen, Rosenfeld, and Kolter [34] formalised the approach further giving a tight robustness certificate for perturbations bounded by an ℓ_2 norm when smoothing is from Gaussian noise. This approach was the first to be scalable to ImageNet scale models. Because smoothed classifiers can be not be evaluated exactly, the output is estimated via sampling combined with lower confidence bounds on the leading class's probability. The certificates are therefore probabilistic. This lower bound gives rise to a theoretical limit on the level of certification, leading to 'certified accuracy waterfalls' [125]. Another consideration of randomised smoothing is that they involve a accuracy-robustness trade off, controlled by the level of noise.

Certified robustness for graph-based models

The first robustness certificate proposed for graph-based models was in the work of Zügner and Günnemann [165]. In this work, the authors consider semi-supervised node prediction over graphs with binary attributes using graph neural networks. The admissible perturbations are those bounded under the ℓ_0 seminorm. Following the approach of Wong and Kolter [143], a convex relaxation is considered and a dual problem is solved via linear programming. Based on this, a robust training procedure is proposed.

Early works that consider certificates under topological change include Bojchevski and Günnemann [15] who proposed a certificate for a class of linear graph neural networks based on PageRank diffusion. Jin et al. [73] propose the first certificate for graph classifiers which consist of a single GCN layers following by pooling and a final linear layer. A convex relaxation of the adversarial polytope based on Lagrange Duality is considered. Although the computed certificates are exact, the framework is specific to this simple graph classifier model and is expensive to compute.

Existing randomised smoothing approaches for certifying graph based classifications models consider certificates for the number of edges flips. This is achieved by flipping an edge between each node pair independently with the same parameter p. Jia et al. [70] applied this to community detection algorithms, whereas Gao, Hu, and Gong [53] and Wang et al. [137] applied the same principles to graph classification. Bojchevski, Klicpera, and Günnemann [16] noted that due to the sparse nature of many real-world graphs, adding and deleting edges between node pairs with the same probability leads to many more edges being added than deleted. To account for this, the authors propose using different probabilities in the smoothing for adding and deleting edges. This idea is generalised further in Chapter 7 by allowing for node pairs to belong to disjoint communities, each with independent flip probabilities. Recently, Jin, Yu, and Zhang [71] consider certifying graph classifiers under a different admissible perturbation measure by considering the Orthogonal Gromov-Wasserstein discrepency.

Chapter 3

On the Stability of Polynomial Spectral Graph Filters

In this Chapter we introduce a notion of robustness for spectral graph filters based on changes to the graph topology. We prove that polynomial graph filters satisfy this notion of robustness. More specifically, we prove that the change in filter output is bounded linearly by the change in the normalised graph Laplacian matrix. In Chapter 5 we will show the this property also holds for graph neural networks, which can be thought of as neural networks which use spectral graph filter layers. In this chapter we empirically demonstrate that properties of a structural perturbation, specifically the relative locality of the edges removed in a binary graph, effect the change in the normalised graph Laplacian. This is a theme we will explore further in Chapter 4.

3.1 Introduction

A fundamental operation in the analysis and transformation of signals is filtering. Filtering amplifies or attenuates frequencies of the input signal by performing pointwise multiplication of the signal in the frequency domain with a so-called transfer function evaluated at the frequencies. This idea can be generalised to filtering of signals in a graph-structured domain. The challenge then is to define a suitable frequency domain for signals defined on graphs. Recent advances in spectral graph theory and graph signal processing provide us with a notion of a frequency domain on graphs via the graph Fourier transform. The graph Fourier transform helps define the concept of a filtering operation as well as convolution for signals defined on graphs [36]. This enables the design of efficient graph representation learning models, such as graph neural networks [147].

When utilising spectral graph filters for learning representations a necessary

condition for transferability in certain tasks is stability. Stability can be loosely defined to be the property such that if we add a small perturbation to the input graph, the output of the filter is also perturbed by a small amount. In the given semi-supervised learning example, small changes to people's social circles is unlikely to drastically change the way they vote. Based on this reasoning, if we use graph filters as part of an end-to-end graph-based classification system we would desire the property of stability. However, bounding the change in filter output by the absolute change in a graph matrix, e.g., the normalised graph Laplacian, may not be desirable as the latter is not always a natural or interpretable metric of change. In the social network example, a more natural unit of change is the addition or deletion of an edge (a friendship). Given a structural perturbation to a graph under a fixed budget, e.g., number of edges allowed to be changed, it is possible to observe different magnitudes of change in the normalised Laplacian matrix of the graph.

Our first contribution in this work is to show that polynomial spectral graph filters are stable, by proving that the change in the output of the filters is linearly bounded by that in the normalised graph Laplacian matrix. This result is similar to a recent work of Levie et al. [86] which proves that spectral graph filters in the Cayley smoothness space (which includes polynomial filters) are linearly stable, but does not require the perturbations to be sufficiently small as in their case. Gama et al. [47] study as well the stability of convolutional graph neural networks using spectral graph filters for convolutional layers. However, compared to our work, they consider a different metric to measure the distance between two graphs. Our work constitutes another approach to stability analysis of spectral graph filters, and contributes more generally to the existing literature on robustness of graph signal processing and graph analysis tools [69], [112], [25], [24]. Our second contribution is to make the first step towards demonstrating the importance of understanding the structure of perturbation in the graph, by showing empirically that the change in the normalised graph Laplacian, given a fixed budget of edge removals, is affected by the relative locality of these edges in the graph. We believe that the combination of these two lines of investigation will pave the way to designing more stable graph filters in light of the nature of the structural perturbation in the graph, with implications in scenarios such as adversarial attacks and defence [164].

3.2 Preliminaries

We consider an undirected graph without self loops $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with $n = |\mathcal{V}|$ vertices and $m = |\mathcal{E}|$ edges and an adjacency matrix $\mathbf{W}$. The non-zero entry of the adjacency matrix $\mathbf{W}_{ij}$ denotes the weight of an edge $\{i, j\} \in \mathcal{E}$. For binary unweighted graphs the degree d_u of the node u is the number of nodes adjacent to

u. A node with degree 0 is called an isolated node. The k-hop neighbourhood of a node u in a graph is the set of all nodes which can be reached from u via a path of at most k edges. The normalised Laplacian matrix of a graph $\mathcal{G}$ is defined to be $\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ where $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the $n \times n$ identity matrix and $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = d_i$. As the normalised Laplacian is a real symmetric matrix, it admits an eigendecomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. Here $\mathbf{\Lambda} = \operatorname{diag}(\lambda_0, \lambda_1, \dots \lambda_{n-1})$ is the diagonal matrix of real eigenvalues of $\mathbf{L}$ in increasing order and $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1 \dots, \mathbf{u}_{n-1}]$ is the orthonormal matrix of corresponding eigenvectors as columns. It is a well known result that the eigenvalues are contained in the interval $[0, 2]$ with $\lambda_0 = 0$, and $\lambda_{n-1} = 2$ if and only if $\mathcal{G}$ is bipartite [33].

A signal on a graph is a function $x : \mathcal{V} \to \mathbb{R}$ that can be compactly represented as a vector $\mathbf{x} \in \mathbb{R}^n$ with $\mathbf{x}_i$ representing the signal value of node i. The graph Fourier transform of a graph signal $\mathbf{x}$ is defined as $\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$ and the inverse graph Fourier transform is given by $\mathbf{U}\hat{\mathbf{x}}$. Filtering in graph signal processing can be defined as operating in the frequency domain using a smooth transfer function $g(\lambda_i)$ which amplifies or attenuates each of the frequency components $\mathbf{u}_i$ of the graph signal. Computing the eigendecomposition explicitly is prohibitively expensive for large graphs; however, we can instead operate directly on the normalised Laplacian matrix:

$$\mathbf{U}\operatorname{diag}(g(\lambda_0), \dots, g(\lambda_{n-1}))\mathbf{U}^\top\mathbf{x} = g(\mathbf{L})\mathbf{x},$$

to obtain an equivalent result. For diagonalisable matrices this is one of the multiple equivalent definitions of matrix functions [63].

An order K polynomial graph filter applied to the normalised graph Laplacian matrix $\mathbf{L}$ and graph signal $\mathbf{x}$ is computed as so:

$$g_\theta(\mathbf{L})\mathbf{x} = \sum_{k=0}^{K} \theta_k \mathbf{L}^k\mathbf{x},$$

where $\theta = (\theta_0, \dots, \theta_K) \in \mathbb{R}^{K+1}$ are the parameters of the filter. The graph filter $g_\theta : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is a matrix-valued function. A scaling of the normalised Laplacian matrix can be used to ensure that the eigenvalues lie in the range $[-1, 1]$ to improve numerical stability of the filtering operation. In [42] the scaled normalised Laplacian is defined to be $2\mathbf{L}/\lambda_{n-1} - \mathbf{I}_n$. In [81] the simpler scaling of $\tilde{\mathbf{L}} = \mathbf{L} - \mathbf{I}_n$ was used which does not require calculating the largest eigenvalue of the normalised Laplacian matrix. We will adopt the latter of these two scalings and refer to this matrix as the scaled normalised Laplacian matrix. From here on in $\mathbf{L}$ will refer to the scaled normalised Laplacian matrix and any mention of normalised Laplacian matrix is assumed to be scaled. Notice that the scaled normalised Laplacian is the negative normalised adjacency matrix.

3.3 Stability of polynomial filters

In this section, we prove that polynomial graph filters are stable. In particular, we state that a spectral graph filter g is (linearly) stable according to the following definition

Definition 1. *A spectral graph filter $g : \mathbb{R} \to \mathbb{R}$ is said to be linearly stable with respect to the scaled normalised Laplacian matrix, if for any scaled normalised Laplacian matrices $\mathbf{L}$ and $\mathbf{L}_p$ of equal size, we have that*

$$\|g(\mathbf{L}) - g(\mathbf{L}_p)\|_2 \leq C\|\mathbf{L} - \mathbf{L}_p\|_2 \tag{3.1}$$

for some positive constant $C \in \mathbb{R}$. The positive constant C is referred to as a stability constant.

where $\|\mathbf{A}\|_2$ is the operator norm of a matrix $\mathbf{A}$. We call the left hand side of Eq. (3.1) the filter distance, and $\|\mathbf{L} - \mathbf{L}_p\|_2$ the error norm. Notice that this definition is that of a Lipschitz continuous matrix-valued function [9, Section X.6], but restricted to the domain of valid scaled normalised Laplacian matrices.

We begin by motivating why we wish to bound the filter distance. Assume the signal is non-zero. Consider the relative output distance of a polynomial graph filter g_θ relative to some perturbation

$$\frac{\|g_\theta(\mathbf{L})\mathbf{x} - g_\theta(\mathbf{L}_p)\mathbf{x}\|_2}{\|\mathbf{x}\|_2},$$

where $\mathbf{L}$ is the input normalised graph Laplacian and $\mathbf{L}_p$ is the normalised graph Laplacian of the perturbed graph. In our setup, we assume that the signal parameters and the input signal are fixed.

We can see that by definition the relative output distance is bounded by the filter distance

$$\frac{\|g_\theta(\mathbf{L})\mathbf{x} - g_\theta(\mathbf{L}_p)\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \leq \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|g_\theta(\mathbf{L})\mathbf{x} - g_\theta(\mathbf{L}_p)\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \stackrel{\text{def}}{=} \|g_\theta(\mathbf{L}) - g_\theta(\mathbf{L}_p)\|_2.$$

The looseness of this bound depends on the signal in relation to where the perturbation is taking place in the graph.

The simplest case to consider is the first order polynomial graph filter (i.e., $K = 1$) where $\|g_\theta(\mathbf{L}) - g_\theta(\mathbf{L}_p)\|_2 = |\theta_1|\|\mathbf{L} - \mathbf{L}_p\|_2$. Thus, the output distance is proportional to the error norm.

3.3.1 Higher-order filters

We now consider higher-order polynomial filters. We prove that the filter distance is bounded above by some constant times the error norm, where the constant depends on the filter parameters. To do so, we will use Taylor's Theorem for matrix-valued functions where the magnitude of the remainder term under the operater norm gives us exactly the filter distance. We will then bound the remainder term to show our main result. We begin by stating Taylor's theorem for matrix-valued functions.

Theorem 1 (Theorem 2.2, [40]). *Let f have a power series expansion about the origin with radius of convergence r and let $\mathcal{D} \subset \mathbb{C}$ be a simply connected open set within the circle of radius r centred at 0. Let $\mathbf{A}, \mathbf{E} \in \mathbb{C}^{n\times n}$ be such that $\Lambda(\mathbf{A}), \Lambda(\mathbf{A}+\mathbf{E}) \subset \mathcal{D}$. Then for any $k \in \mathbb{N}$*

$$f(\mathbf{A}+\mathbf{E}) = T_k(\mathbf{A},\mathbf{E}) + R_k(\mathbf{A},\mathbf{E}), \tag{3.2}$$

where

$$T_k(\mathbf{A},\mathbf{E}) = \sum_{j=0}^{k} \frac{1}{j!} D_f^{[j]}(\mathbf{A},\mathbf{E}),$$
$$R_k(\mathbf{A},\mathbf{E}) = \frac{1}{2\pi i} \int_\Gamma f(z)(z\mathbf{I}_n - \mathbf{A} - \mathbf{E})^{-1} \left[\mathbf{E}(z\mathbf{I}_n - \mathbf{A})^{-1}\right]^{k+1} dz,$$

and Γ is a closed contour in $\mathcal{D}$ enclosing $\Lambda(\mathbf{A})$ and $\Lambda(\mathbf{A}+\mathbf{E})$ where $\Lambda(\mathbf{A})$ is the spectrum of a matrix $\mathbf{A}$.

In the theorem, the terms

$$D_f^{[j]}(\mathbf{A},\mathbf{E}) = \left.\frac{d^j}{dt^j}\right|_{t=0} f(\mathbf{A}+t\mathbf{E})$$

denote the jth order Fréchet derivatives [64]. The term $R_k(\mathbf{A},\mathbf{E})$ in Eq. (3.2) is the remainder term in the Taylor expansion which the following Lemma provides an analytic bound for.

Lemma 1 (Lemma 3.1, [40]). *Let f and $\mathcal{D}$ satisfy the criteria of Theorem 1. Furthermore, let $\epsilon > 0$ be such that $\Lambda_\epsilon(\mathbf{A}) \subset \mathcal{D}$ and $\Lambda_\epsilon(\mathbf{A}+\mathbf{E}) \subset D$, and take $\tilde{\Gamma}_\epsilon \subset \mathcal{D}$ to be a closed contour that encloses both $\Lambda_\epsilon(\mathbf{A})$ and $\Lambda_\epsilon(\mathbf{A}+\mathbf{E})$. Then the remainder term $R_k(\mathbf{A},\mathbf{E})$ is bounded by*

$$\|R_k(\mathbf{A},\mathbf{E})\| \leq \frac{\|\mathbf{E}\|^{k+1}\tilde{L}_\epsilon}{2\pi\epsilon^{k+2}} \max_{z\in\tilde{\Gamma}_\epsilon} |f(z)|,$$

where $\tilde{L}_\epsilon$ is the length of $\tilde{\Gamma}_\epsilon$. In particular, when a circular contour centred at 0 is

used,

$$\|R_k(\mathbf{A},\mathbf{E})\| \leq \frac{\|\mathbf{E}\|^{k+1}\tilde{\rho}_\epsilon}{\epsilon^{k+2}} \max_{\phi\in[0,2\pi]} \left|f\left(\tilde{\rho}_\epsilon e^{i\phi}\right)\right|, \tag{3.3}$$

where $\tilde{\rho}_\epsilon = \max\{|z| : z \in \Lambda_\epsilon(\mathbf{A}+\mathbf{E}) \cap \Lambda_\epsilon(\mathbf{A})\}$ *is the radius of the circle.*

In the above lemma, $\Lambda_\epsilon(\mathbf{X}) = \{z \in \mathbb{C} : \|(z\mathbf{I}_n - \mathbf{A})^{-1}\| \geq \epsilon^{-1}\}$ is the ϵ-pseudospectrum of a matrix $\mathbf{X} \in \mathbb{C}^{n\times n}$ [132].

The following lemmas will be used to further bound the remainder term given in Eq. (3.3). The proofs are given in Appendix A.1.

Lemma 2. *Let $f(z)$ be a complex polynomial of degree K, if $|z| \geq 1$ then*

$$|f(z)| \leq |z|^K \max_{\phi\in[0,2\pi]} |f(e^{i\phi})|.$$

Lemma 3. *Let $K \geq 2$ then*

$$\inf_{\epsilon>0} \frac{(1+\epsilon)^{K+1}}{\epsilon^2} = \frac{1}{4}(K^2-1)\left(\frac{K+1}{K-1}\right)^K.$$

We now use the above developments to state and prove the main result of this section.

Theorem 2. *Consider a polynomial graph filter of order K:*

$$g_\theta(\mathbf{L}) = \sum_{k=0}^{K} \theta_k \tilde{\mathbf{L}}^k,$$

where $\theta \in \mathbb{R}^{K+1}$ are the polynomial coefficients and $\tilde{\mathbf{L}}$ is the scaled normalised Laplacian of an input graph. Consider perturbing the graph and let the Laplacian of the perturbed graph be $\mathbf{L}_p$. Then the following holds:

$$\|g_\theta(\mathbf{L}) - g_\theta(\mathbf{L}_p)\|_2 \leq \frac{1}{4}\|\theta_{-0}\|_1(K^2-1)\left(\frac{K+1}{K-1}\right)^K \|\mathbf{L}-\mathbf{L}_p\|_2,$$

where $\theta_{-0} = (\theta_1,\ldots,\theta_K)$ is the vector of polynomial coefficients for all terms apart from the constant term, with the 1-norm of a vector is defined as $\|\mathbf{x}\|_1 = \sum_i |\mathbf{x}_i|$.

Proof. We proceed by applying Theorem 1 to our polynomial graph filter. Polynomial functions have an infinite radius of convergence so we may take $\mathcal{D} = \mathbb{C}$. According to Eq. (3.2) with $k = 0$ we get

$$g_\theta(\mathbf{L}_p) - g_\theta(\mathbf{L}) = R_0(\mathbf{L}, \mathbf{L}_p - \mathbf{L}).$$

Under the operator norm, the ϵ-pseudospectrum $\Lambda_\epsilon(A)$ of a normal matrix A is the union of open balls of radius ϵ around the eigenvalues of A [132, Theorem 2.2]. In

particular, if a point lies in the ϵ-pseudospectrum of a scaled normalised Laplacian matrix, it is at most distance ϵ from some point in $[-1, 1]$. Thus, by the triangle inequality all points in the ϵ-pseudospectrum are at most distance $1 + \epsilon$ from the origin. Using this, we can bound the remainder term by Lemma 1 using Eq. (3.3) using a circle contour of radius $1 + \epsilon$ centred at 0. This gives us that

$$\|g_\theta(\mathbf{L}) - g_\theta(\mathbf{L}_p)\|_2 \leq \frac{\|\mathbf{L} - \mathbf{L}_p\|_2(1+\epsilon)}{\epsilon^2} \max_{\phi\in[0,2\pi]} |g_\theta((1+\epsilon)e^{i\phi})|.$$

By taking $z = (1 + \epsilon)e^{i\theta}$ in Lemma 2 we get

$$\|g_\theta(\mathbf{L}) - g_\theta(\mathbf{L}_p)\|_2 \leq \frac{\|\mathbf{L} - \mathbf{L}_p\|_2(1+\epsilon)^{K+1}}{\epsilon^2} \max_{\phi\in[0,2\pi]} |g_\theta(e^{i\phi})|.$$

Since ϵ is an arbitrary non-negative scalar value, we may minimise the right hand side with respect to ϵ. By Lemma 3 we get

$$\|g_\theta(\mathbf{L}) - g_\theta(\mathbf{L}_p)\|_2 \leq \frac{1}{4}(K^2 - 1)\left(\frac{K+1}{K-1}\right)^K \|\mathbf{L} - \mathbf{L}_p\|_2 \max_{\phi\in[0,2\pi]} |g_\theta(e^{i\phi})|.$$

We give a naive upper bound to the magnitude of the polynomial over the unit circle by noting that for all $\phi \in [0, 2\pi]$ we have that

$$|g_\theta(e^{i\phi})| = |\sum_{k=0}^{K} \theta_k e^{i\phi k}| \leq \sum_{k=0}^{K} |\theta_k||e^{i\phi k}| = \|\theta\|_1. \tag{3.4}$$

Finally, note that the $\theta_0\mathbf{I}_n$ terms cancel out in the calculation of $g_\theta(\mathbf{L}) - g_\theta(\mathbf{L}_p)$ so we may, without loss of generality, assume θ_0 is zero. Using this observation, combined with the bound given in Eq. (3.4), we obtain the result. □

Theorem 2 shows that the difference between the outputs of two polynomial filters is bounded by a quantity which scales linearly with the error norm, hence proves the stability of polynomial spectral filters. Empirically, however, we found this bound to be quite loose and the development of practical bounds is an open research direction.

3.4 Importance of structural perturbation

Theorem 2 bounds the filter distance by the error norm. However, absolute change in the graph Laplacian is not a quantity that is easy to interpret, in the sense that it does not provide any insight into the actual structural perturbation of the graph. Furthermore, different structural perturbation under a fixed budget can lead to different magnitudes of change in the Laplacian. For example, considering perturbing

undirected binary graphs by removing a fixed number of edges, the locality of these edges in the graph will lead to different error norm.

Our intuition for this is guided by the way the entries of the Laplacian matrix change under perturbation of the graph topology. Note that if the edges are sufficiently far apart, the changes in the matrix incurred by each edge removal is independent. However, if two edges are close then an entry may change due to both edge removals. Based on the same intuition of considering the change of entries of the Laplacian matrix, we expect the degree of the endpoints of the edges we remove to also play a role. We leave further investigation of this as future work.

To quantify the relative locality of a set of edges, we propose the following definition.

Definition 2. *A set of edges $\mathcal{R} \subseteq \mathcal{E}$ is k-hop localised in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with respect to a node $v \in \mathcal{V}$ if the edge set of the graph induced by the k-hop neighbourhood of v contains all edges in $\mathcal{R}$.*

In this paper, we empirically test the effect of relative locality of edge removals in affecting the change in the normalised graph Laplacian, under different types of random graph models. We present these results in the following section.

3.5 Experiments

We begin this section by showing that the filter distance empirically scales linearly with the error norm in line with the result of Theorem 2. We then test our hypothesis that the locality of edges being removed affects the magnitude of the error norm as discussed in Section 3.4. Code for reproducing the results in this section is available online[1].

To demonstrate the linear scaling property of the result of Theorem 2 we consider the following experimental setup. We generate Barabási-Albert graphs with $n = 200$ nodes and randomly remove each edge with independent probability of 0.5 to give a perturbed graph [5]. We only consider perturbations which do not disconnect the graph. We then look at the filter distance for low pass polynomial filter of order $K \in \{1, 2, 3\}$. The results are shown in Fig 3.1. The filter distance can be seen to scale linearly with the error norm consistent with Theorem 2 which states that polynomial filters are linearly stable.

To test the effect that the locality of edge removals has on the error norm we conducted repeats of the following experiment. We perturbed a graph by randomly selecting a node and remove a fixed number of edges in the k-hop subgraph around that node. We also considered removing edges uniformly at random (we say in this

[1] `https://github.com/henrykenlay/spgf`.

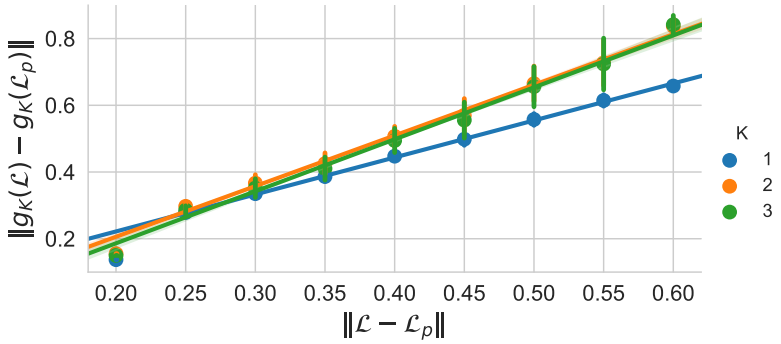
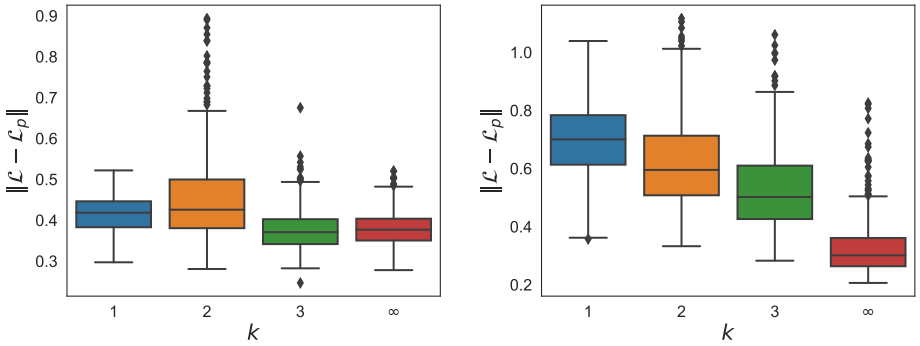


Figure 3.1: A plot of the error norm and the filter distance for different order polynomial filters. The bars indicate the standard deviation of the filter distance.



(a) Barabási-Albert graph (b) Sensor network graph

Figure 3.2: Varying effects of removing a fixed percentage of edges with different localisation.

case the edges are ∞-hop localised). We only consider an experiment valid if the graph remains connected. We consider two random graph models. The first random graph model we consider is the Barabási-Albert random graph model with $n = 500$ where we connect a node with 3 edges at each step of the network generation. The second random graph model we consider is a sensor network model. Synthetic sensor networks are generated by uniformly sampling $n = 500$ points in the unit square and connecting nodes which are under some threshold distance. The settings of these experiments are summarised in Table 3.1; obtained results are illustrated in Fig. 3.2.

In the experiments, we see that as we increase the size of the neighbourhood, the median error norm is reduced. The effect is more pronounced for the sensor network graph. We believe this is due to the sensor networks greater diameter, leading the perturbations under different k-hop neighbourhoods to be more distinct.

Table 3.1: Summary statistics of the graphs used in experiments

	Barabási-Albert	Sensor network
Number of edges m	1491	2724-3130
Diameter	5-6	18-24
Edges removed	74 (5%)	136-156 (5%)

3.6 Conclusion

In this chapter, we introduced a notion of stability for a polynomial spectral graph filter. Furthermore, we demonstrated empirically how the distribution of the observed error norm can change under a fixed structural perturbation budget. Despite making an empirical link between the localisation of the perturbation in the spatial domain and the magnitude $\|\mathbf{L}-\mathbf{L}_p\|_2$, there is no spatial interpretation of $\|\mathbf{L}-\mathbf{L}_p\|_2$. Inspired by this, the next chapter makes a theoretical link between $\|\mathbf{L}-\mathbf{L}_p\|_2$, and the localisation of the perturbation, as well as the degree of the nodes which are adjacent to perturbed edges. We also provide a more extensive experimental analysis. In Chapter 5 we provide similar bounds to graph neural networks, which make use of spectral graph filters to propagate information across the graph.

Chapter 4

Interpretable Stability Bounds for Spectral Graph Filters

In this Chapter, we study filter stability and provide a novel and interpretable upper bound on the change of filter output, where the bound is expressed in terms of the endpoint degrees of the deleted and newly added edges, as well as the spatial proximity of those edges. This is in contrast to the bound introduced in the previous Chapter, which was in terms of the error norm. This upper bound allows us to reason, in terms of structural properties of the graph, when a spectral graph filter will be stable. We further perform extensive experiments to verify intuition that can be gained from the bound.

4.1 Introduction

A graph is a general-purpose data structure that uses edges to model pairwise interactions between entities, which are modelled as the nodes of the graph. Many types of data in the real-world reside on graph domains, such as those collected in sensor, biological, and social networks. This has sparked a major interest in recent years in developing machine learning models for graph-structured data [26], leading to the fast-growing fields of graph signal processing [36, 101] and geometric deep learning [18].

Spectral graph filters, a generalisation of classical filters to the graph domain via spectral graph theory [33], are a ubiquitous tool designed to process graph-structured data. In addition to various signal processing tasks [36, 101], graph filters are becoming an important tool for machine learning tasks defined on graphs [43]. For example, they have been used to define convolution on graphs and design graph neural networks, which lead to state-of-the-art performance in both node and graph classification [20, 42, 81, 87, 144, 104, 4].

Despite the surge of research proposing new graph-based machine learning models, significantly less attention has been paid to the understanding of theoretical properties, such as stability, of existing models, in particular graph filters. Informally, a filter is considered to be stable against a perturbation if, after being applied to a signal, it does not lead to large changes in the filter output. In the context of graph-structured data, stability can be defined with respect to perturbation to the signal or the underlying topology. We focus on the latter in this work as graph filters are typically through a function of the graph topology.

Stability is important mainly for two reasons. First, real-world graph-structured data often come with perturbations, either due to measurement error or inaccuracy in the graph construction. Second, when these data are used in machine learning tasks, stability of the graph filters is important to designing learning algorithms that are robust to small perturbations. As a practical example, graph filters are often used to extract spatio-temporal predictive features from fMRI signals realised as signals on a structural brain network. The underlying structural brain network is typically an approximation of the true brain connectivity and therefore the topology will inherently be noisy. Nevertheless, we would desire the predictions, and thus the filtering process, to be robust to the noise inherent in this data.

There has only been a handful of papers considering the stability of spectral graph filters. Among the existing works [86, 79, 48] which address this, most provide an upper bound on the change of the filter output. These upper bounds are in terms of the magnitude of perturbation and lack interpretation in terms of how the structure of the graph has changed, for example the degree of the nodes. This limitation hampers the design of strategies that could defend efficiently against potential adversaries. A notable exception in the literature is the recent work of Kenlay, Thanou, and Dong [78]; however, this work only considers degree preserving edge rewiring which is a stringent assumption that does not cover many perturbations observed in practical scenarios.

In this work, we provide a novel upper bound for the output change of spectral graph filters under topological perturbation, i.e., edge deletions and additions, of an unweighted and undirected graph. Unlike previous works, our bound is interpretable in the sense that it is expressed in terms of the structural properties of the graph and the perturbation. The bound helps us understand sufficient conditions under which a spectral graph filter will be stable. Specifically, we show that, when edges are deleted and added to a graph to obtain the perturbed graph, the filter will be stable if 1) the endpoints of these edges have sufficiently high degree, and 2) the perturbation is not concentrated spatially around any one node. We further verify the intuition behind our theoretical results using synthetic experiments.

Our study has two main contributions. First, to the best of our knowledge, our

theoretical analysis is one of the first that provides sufficient conditions for a graph filter to be robust to the perturbation, where the conditions are in terms of the structural properties rather than the magnitude of change. Second, unlike previous theoretical studies, we perform extensive experiments to validate the intuition gained from the bound. In particular, we examine how the filter output changes for a range of perturbation strategies including random strategies, an adversarial attack strategy and a robust strategy which is derived from insight that the bound provides. Furthermore, we experiment with a range of random graph models as well as real-world data sets, and examine how different properties of these graphs, for example the degree distribution, have an effect on the filter stability. Overall, we believe this study fills an important gap in our understanding of spectral graph filters, and future work based on these ideas can have broad implications for understanding and designing robust graph-based machine learning models that utilise graph filters, most notably a wide range of designs of graph neural networks [4].

4.2 Related work

4.2.1 Stability of graph filters

Stability of graph filters has been mainly studied by characterising the magnitude of perturbation caused by changes to a graph shift operator (GSO) under the operator norm. One such effort is the work of Levie, Isufi, and Kutyniok [86], where filters are shown to be stable in the Cayley smoothness space, with the output change being linearly bounded. The main limitations of this result is that the constant which depends on the filter is not easily interpretable and the bound is only valid for sufficiently small perturbation. In a similar vein, Kenlay, Thanou, and Dong [79] proves that polynomial graph filters are linearly bounded under changes to the shifted normalised Laplacian matrix by applying Taylor's theorem for matrix functions [40]. We build upon this work by giving a tighter bound for a larger class of filters, and providing theoretical basis for how the magnitude of change in the Laplacian matrix relates to change in the structural properties of the graph such as the degree of the nodes and the distribution of the perturbed edges.

4.2.2 Adversarial attack and defence

Adversarial attacks are an optimisation based data-driven approach to finding perturbations for which graph-based models are not robust. In particular, it has been shown that the output of graph neural networks can change drastically even under small changes to the input generated from adversarial examples [164, 126]. As our

bound necessarily covers worst-case scenarios, adversarial attacks provide insight into the tightness of our bound.

4.3 Preliminaries and problem formulation

We define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of n vertices and $\mathcal{E}$ a set of edges. We write $u \sim v$ if node u is connected to v and $u \not\sim v$ otherwise. By fixing a labelling on the nodes we can encode $\mathcal{G}$ into a binary adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$. The degree d_u of a node u indicates the number of nodes connected to u and we define the degree matrix as $\mathbf{D} = diag(d_1, \ldots, d_n)$. A node u is said to be isolated if it has degree zero. The normalised Laplacian matrix is defined as $\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ where $\mathbf{I}_n$ is the identity matrix of dimension n and conventionally the entry $\mathbf{D}_{uu}^{-1/2}$ is set to zero if the node u is isolated. The entries of $\mathbf{L}$ can be explicitly written as

$$\mathbf{L}_{uv} = \begin{cases} 1 & \text{if } u = v \\ \frac{-1}{\sqrt{d_u d_v}} & \text{if } u \sim v \text{ and } u \neq v \\ 0 & \text{otherwise} \end{cases}. \tag{4.1}$$

The normalised Laplacian matrix is an example of a GSO, a generalisation of the shift operator from classical signal processing which can be used as a building block to construct a graph signal processing framework [101]. The matrix $\mathbf{L}$ is real and symmetric, and thus has an eigendecomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where $\mathbf{\Lambda} = \text{diag}(\lambda_1 \ldots \lambda_n)$ are the eigenvalues such that $0 = \lambda_1 \leq \ldots \leq \lambda_n \leq 2$ and $\mathbf{U}$ is the matrix where the columns are the corresponding unit norm eigenvectors.

We can define a graph signal $x : \mathcal{V} \to \mathbb{R}$ as an assignment of each node to a scalar value; this can be compactly represented by a vector $\mathbf{x}$ such that $\mathbf{x}_i = x(i)$. The graph Fourier transform can be defined as $\hat{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$ and the inverse graph Fourier transform is then given by $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. With a notion of frequency, filtering signals on graphs amounts to amplifying and attenuating the frequency components in the graph Fourier domain, i.e., $y = \mathbf{U}\,\text{diag}(g(\lambda_1), \ldots, g(\lambda_n))\mathbf{U}^T x = \mathbf{U}g(\Lambda)\mathbf{U}^T\mathbf{x} = g(\mathbf{L})\mathbf{x}$, where $g(\cdot)$ is a function over the range of eigenvalues that corresponds to the characteristics of the filter. We will abuse notation by evaluating $g : \mathbb{R} \to \mathbb{R}$ on the domain $\mathbb{R}^{n\times n}$ using this definition of matrix-valued functions[1].

We are primarily concerned with the stability of spectral graph filters where the filter parameters are fixed. This scenario is relevant to hand-tuned filters or during inference of a pre-trained model. An adversarial attack in this setting is known as an evasion attack. Specifically, we consider edge deletions and additions to the graph to give a perturbed graph $\mathcal{G}_p$ and use $\mathbf{L}_p$ to denote the normalised Laplacian of $\mathcal{G}_p$.

[1]This is one of a few equivalent ways used to define matrix-valued functions [63]

We consider the magnitude of the error matrix, i.e., $\|\mathbf{E}\|_2 = \|\mathbf{L}_p - \mathbf{L}\|_2$, which we call the error norm where $\|\cdot\|_2$ is the operator norm when applied to matrices and the ℓ_2-norm when applied to vectors. We will also consider the matrix one norm $\|\mathbf{E}\|_1 = \max_i \sum_j |\mathbf{E}_{ij}|$ and the matrix infinity norm $\|\mathbf{E}\|_\infty = \max_j \sum_i |\mathbf{E}_{ij}|$. If we denote $\mathbf{E}_u$ as the row corresponding to node u of $\mathbf{E}$ we can write the matrix one norm as $\|\mathbf{E}\|_1 = \max_u \|\mathbf{E}_u\|_1$ where $\|\cdot\|_1$ is the Manhattan or ℓ_1-norm when applied to vectors. The goal of this study is two-fold: 1) understand how the relative output of a filter changes when we perturb the topology of the underlying graph; 2) what is the impact of the structural properties of the perturbation on filter stability. In particular, the structural properties we consider are the degree of the nodes before and after perturbation and how much the perturbation is concentrated around each node. We address the first goal in Section 4.4 and the second in Section 4.5. We experimentally validate the insights gained by our bound in Section 4.6.

4.4 Linearly stable filters

Our notion of stability is based on relative output distance defined as

$$\frac{\|g(\mathbf{\Delta})\mathbf{x} - g(\mathbf{\Delta}_p)\mathbf{x}\|_2}{\|\mathbf{x}\|_2}, \tag{4.2}$$

where g is a spectral graph filter, $\mathbf{x}$ is an input graph signal and $\mathbf{\Delta}$ is the GSO of the graph $\mathcal{G}$ (similarly $\mathbf{\Delta}_p$ is the GSO of $\mathcal{G}_p$). If we assume $\mathbf{x}$ has unit norm then the above is equivalent to absolute output distance. We can bound this quantity by what we call the filter distance which measures the largest possible relative output change of the filter over non-zero signals:

$$\begin{aligned}\frac{\|g(\mathbf{\Delta})\mathbf{x} - g(\mathbf{\Delta}_p)\mathbf{x}\|_2}{\|\mathbf{x}\|_2} &\leq \max_{\mathbf{x}\neq 0} \frac{\|g(\mathbf{\Delta})\mathbf{x} - g(\mathbf{\Delta}_p)\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \\ &\stackrel{\text{def}}{=} \|g(\mathbf{\Delta}) - g(\mathbf{\Delta}_p)\|_2. \end{aligned} \tag{4.3}$$

In Kenlay, Thanou, and Dong [79], the authors bound the filter distance of a graph filter g where g is a polynomial. The bound is given by some constant times the error norm $\|\mathbf{E}\|_2$, where the constant depends on the filter. When a filter satisfies this property we say it is linearly stable which we define as follows.

Definition 3. *A spectral graph filter $g : \mathbb{R} \to \mathbb{R}$ is said to be linearly stable with respect to a type of graph shift operators, if for any graph shift operators $\mathbf{\Delta}$ and $\mathbf{\Delta}_p$ of this type, we have that*

$$\|g(\mathbf{\Delta}) - g(\mathbf{\Delta}_p)\|_2 \leq C\|\mathbf{E}\|_2 \tag{4.4}$$

Table 4.1: Examples of linearly stable graph filters used for machine learning.

Filter	Functional form	GSO	Stability constant C	Use
Polynomial filter	$\sum_{k=0}^{K} \theta_k \lambda^k$	$\frac{2\mathbf{L}}{\lambda_{\max}} - \mathbf{I}_n$	$\sum_{k=1}^{K} k\lvert\theta_k\rvert$	Chebynet [42]
Low-pass filter	$(1+\alpha\lambda)^{-1}$	$\mathbf{L}$	α	Low-pass filtering [103]
Monomial	λ^K	$\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$	K	Simple GCN [144]
Identity	λ	$\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$	1	GCN [81]

for some positive constant $C \in \mathbb{R}$. The positive constant C is referred to as a stability constant.

This definition is a more general definition compared to Equation 3.1, which is a special case of when the graph shift operator is the scaled normalised Laplacian matrix. Two types of filters of particular interest are the polynomial filters, i.e., $g(\lambda) = \sum_{k=0}^{K} \theta_k \lambda^k$, where $\{\theta_k\}_{k=0}^{K}$ are the polynomial coefficients, and the low-pass filters, i.e., $g(\lambda) = (1+\alpha\lambda)^{-1}$, where $\alpha > 0$ is some constant. Polynomial filters are used in a variety of graph-based machine learning. We list some of them in Table 4.1. It was recently proved that polynomial filters are linearly stable with respect to the shifted normalised Laplacian matrix $\mathbf{L} - \mathbf{I}_n$ [79]. A simpler proof with a tighter bound (smaller stability constant) was given to show linear stability with respect to the augmented adjacency matrix $\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_n}$ and $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I_n}$ [78]. In addition, the following more general result holds.

Proposition 1. *Polynomial filters $g(\lambda) = \sum_{k=0}^{K} \theta_k \lambda^k$ are linearly stable with respect to any GSO where the spectrum lies in $[-1, 1]$. The stability constant is given by $C = \sum_{k=1}^{K} k\lvert\theta_k\rvert$.*

Proof: See Supplementary Material. Another important class of filters are low-pass filters, which are linearly stable with respect to the normalised Laplacian matrix.

Proposition 2. *The low-pass filter $g(\lambda) = (1+\alpha\lambda)^{-1}$ is linearly stable with respect to the normalised Laplacian matrix. The constant is given by $C = \alpha$.*

Proof: See Supplementary Material. A thorough characterisation of linearly stable filters is beyond the scope of this work. Instead, this section serves to motivate why we are interested in analysing the magnitude of $\|\mathbf{E}\|_2$: some common types of filters are stable to small perturbation when the perturbation is measured by the error norm $\|\mathbf{E}\|_2$. Although this is an intuitive choice, it is not immediately clear how $\|\mathbf{E}\|_2$ is related to the characteristics of the structural properties of the perturbation. This motivates us to provide an upper bound on $\|\mathbf{E}\|_2$ in Section 4.5 in terms of interpretable characteristics in the structural domain.

4.5 Interpretable bound on filter output change

In this section we bound $\|\mathbf{E}\|_2$ by interpretable properties relating to the structural change. Given a perturbation and a node u we denote $\mathcal{A}_u$, $\mathcal{D}_u$, and $\mathcal{R}_u$ as the set of adjacent nodes for newly added edges, deleted edges, and remaining edges around u, respectively. We denote $\Delta_u^+ = |\mathcal{A}_u|$ and $\Delta_u^- = |\mathcal{D}_u| < d_u$ the number of edges added and deleted around u, respectively, and $\Delta_u = \Delta_u^+ - \Delta_u^-$ as the change of degree. We denote $d'_u = d_u + \Delta_u$ as the degree of node u in $\mathcal{G}_p$. We define $\alpha_u = \max_{v \in \mathcal{N}_u \cup \{u\}} |\Delta_v|/d_v$, where $\mathcal{N}_u$ is the 1-hop neighbourhood of node u, as the maximum relative change in degree among a node u and its neighbours. In addition, we define $\delta_u = \min_{v \in \mathcal{N}_u} d_v$ as the smallest degree of the nodes neighbouring node u, and δ'_u as the same quantity in the perturbed graph. We assume that both the graph $\mathcal{G}$ and the perturbed graph $\mathcal{G}_p$ do not contain isolated nodes.

Our approach to upper bounding $\|\mathbf{E}\|_2$ relies on the inequality $\|\mathbf{E}\|_2^2 \leq \|\mathbf{E}\|_1 \|\mathbf{E}\|_\infty$ [65, Section 6.3]. As $\mathbf{E}$ is Hermitian, $\|\mathbf{E}\|_1 = \|\mathbf{E}\|_\infty$ thus simplifying this inequality to become $\|\mathbf{E}\|_2 \leq \|\mathbf{E}\|_1$. There may exist strategies which give tighter bounds, but the benefit of this approach is that $\|\mathbf{E}\|_1$ leads to an interpretation in the structural domain. Thus, we are making use of the following inequality

$$\|\mathbf{E}\|_2 \leq \|\mathbf{E}\|_1 = \max_{u \in \mathcal{V}} \|\mathbf{E}_u\|_1. \tag{4.5}$$

By considering how the entries of $\mathbf{L}$ in Eq. (4.1) change, we have the following closed-form expression for $\|\mathbf{E}_u\|_1$:

$$\|\mathbf{E}_u\|_1 = \sum_{v \in \mathcal{D}_u} \frac{1}{\sqrt{d_u d_v}} + \sum_{v \in \mathcal{A}_u} \frac{1}{\sqrt{d'_u d'_v}} + \sum_{v \in \mathcal{R}_u} \left| \frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{d'_u d'_v}} \right|.$$

The results of this section bound the three terms in this expression, leading to an overall bound to $\|\mathbf{E}_u\|_1$ hence to $\|\mathbf{E}\|_1$ and $\|\mathbf{E}\|_2$. We proceed by bounding each of the terms in Eq. (5.2).

4.5.1 Bounding the error norm

Recall that δ_u is the smallest degree in the neighbourhood of a node u, allowing us to bound the first term in Eq. (5.2) by replacing d_v with $\delta_u \leq d_v$ in the denominator to give:

$$\sum_{v \in \mathcal{D}_u} \frac{1}{\sqrt{d_u d_v}} \leq \sum_{v \in \mathcal{D}_u} \frac{1}{\sqrt{d_u \delta_u}} = \frac{\Delta_u^-}{\sqrt{d_u \delta_u}}. \tag{4.6}$$

Similarly, we can bound the second term in Eq. (5.2) as:

$$\sum_{v\in\mathcal{A}_u}\frac{1}{\sqrt{d'_u d'_v}} \leq \sum_{v\in\mathcal{A}_u}\frac{1}{\sqrt{d'_u \delta_u}} = \frac{\Delta_u^+}{\sqrt{d'_u \delta'_u}}. \tag{4.7}$$

To bound the third term in Eq. (5.2), we first introduce the following lemma.

Lemma 4. *Let $\alpha_u \in [0,1)$. Then the following holds:*

$$\begin{aligned}\sum_{v\in\mathcal{R}_u}\left|\frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{d'_u d'_v}}\right| &\leq \sum_{v\in\mathcal{R}_u}\left(\frac{\alpha_u}{1-\alpha_u}\right)\frac{1}{\sqrt{d_u d_v}} \\ &\leq \left(\frac{\alpha_u}{1-\alpha_u}\right)\frac{d_u - \Delta_u^-}{\sqrt{d_u \delta_u}}.\end{aligned} \tag{4.8}$$

Proof: See Appendix B.1. The assumption on α_u can be interpreted as follows. If $\alpha_u = 0$ then the degree of u and that of all nodes in the neighbourhood of u are unchanged, so the third term in Eq. (5.2) becomes zero. If $\alpha_u \geq 1$, then for some node v in $\mathcal{N}_u \cup \{u\}$ we have $|\Delta_v|/d_v \geq 1$. Notice that for all nodes we have $\Delta_v > -d_v$, since the degree after perturbation δ'_v is strictly positive (recall that we do not allow isolated nodes). Therefore, we must instead have $\Delta_v \geq d_v$ which implies $d'_v \geq 2d_v$. In other words, the assumption $\alpha_u < 1$ means that for all nodes v in $\mathcal{N}_u \cup \{u\}$ we have $d'_v < 2d_v$. This limits large amount of change around low degree nodes. It can be noted that if a perturbation does not alter the degree distribution then $\alpha_u = 0$ for all nodes and the third term in Eq. (5.2) vanishes. We will consider a particular case of degree preserving perturbation in Section 5.5.

By combining the bounds in Eq. (4.6) and Eq. (4.7) with Lemma 4, we can further bound Eq. (5.2):

$$\|\mathbf{E}_u\|_1 \leq \frac{\Delta_u^-}{\sqrt{d_u \delta_u}} + \frac{\Delta_u^+}{\sqrt{d'_u \delta'_u}} + \left(\frac{\alpha_u}{1-\alpha_u}\right)\frac{d_u - \Delta_u^-}{\sqrt{d_u \delta_u}}. \tag{4.9}$$

By further combining this bound with Eq. (4.5), we arrive at our main result.

Theorem 3. *Assume that $\alpha_u \in [0,1)$ holds for all nodes $u \in \mathcal{V}$. Then the following holds:*

$$\|\mathbf{E}\|_2 \leq \max_{u\in\mathcal{V}}\left\{\frac{\Delta_u^-}{\sqrt{d_u \delta_u}} + \frac{\Delta_u^+}{\sqrt{d'_u \delta'_u}} + \left(\frac{\alpha_u}{1-\alpha_u}\right)\frac{d_u - \Delta_u^-}{\sqrt{d_u \delta_u}}\right\} \tag{4.10}$$

Proof: See Supplementary Material. We will explore the looseness of the bound given in Eq. (4.9) and Eq. (4.10) in Section 4.6. In practice, the bound might be loose but it provides insight into when we expect filters to be stable. We will discuss this insight in Section 4.5.4. In the following subsection, we will consider a special case where we can produce a bound that is tighter in practice.

4.5.2 Bounding the error norm under edge rewiring

Degree preserving rewiring is a type of edge rewiring such that the perturbation does not change the original degree distribution [78]. Given two edges $u \sim v$ and $u' \sim v'$ such that $u \not\sim v', u \not\sim u', v \not\sim v'$ and $v \not\sim u'$, the double edge rewiring operation deletes the two edges and introduces the edges $u \sim u'$ and $v \sim v'$ (see Fig. B.2 for an illustration). The perturbation consists of two edge deletions and two edge additions and does not change the degree of any nodes involved. This model of perturbation approximately arises in practical applications, where the capacity of a node is fixed and remains at full load such as in communication networks [11]. In this specific scenario, $\alpha_u = 0, \delta_u = \delta'_u$ and $d_u = d'_u$. Furthermore, we know that $\Delta_u^- = \Delta_u^+ = r_u$ where we define r_u as the number of rewiring operations involved around a node u. Using Theorem 3 we get the following corollary.

Corollary 1. *If the perturbation consists of only double edge rewiring operations then:*

$$\|\mathbf{E}\|_2 \leq \max_{u \in \mathcal{V}} \frac{2r_u}{\sqrt{d_u \delta_u}}. \tag{4.11}$$

A similar bound has been recently derived to bound the change in feature representations of certain graph neural network architectures [78].

4.5.3 Bounding filter output change

To obtain a full bound on filter output change, we combine together bounds that are developed in previous sections. Consider a spectral graph filter g which is linearly stable with respect to the normalised Laplacian matrix. We then have the following bound for the filter output change:

$$\begin{aligned} \frac{\|g_\theta(\mathbf{L})\mathbf{x} - g_\theta(\mathbf{L}_p)\mathbf{x}\|_2}{\|\mathbf{x}\|_2} &\leq \|g_\theta(\mathbf{L}) - g_\theta(\mathbf{L}_p)\|_2 \leq C\|\mathbf{E}\|_2 \\ &\leq C \max_{u \in \mathcal{V}} \left\{ \frac{\Delta_u^-}{\sqrt{d_u \delta_u}} + \frac{\Delta_u^+}{\sqrt{d'_u \delta'_u}} + \left(\frac{\alpha_u}{1-\alpha_u}\right) \frac{d_u - \Delta_u^-}{\sqrt{d_u \delta_u}} \right\}. \end{aligned} \tag{4.12}$$

The first inequality is introduced in Eq. (4.3) which relates relative output distance and filter distance. The second inequality comes from our assumption that the graph filter is linearly stable (Eq. (4.4)) with a stability constant C. Finally, we can make use of Eq. (4.10) to establish the third inequality, which provides a structurally interpretable bound on the relative output distance. We discuss interpretations of this result in the following subsection.

4.5.4 Interpretation of the bound

The bounds given in this section let us reason about sufficient conditions under which a perturbation leads to small change in graph filter output. We can conclude from Eq. (4.5) that perturbations which cause small changes to $\|\mathbf{E}_u\|_1$ over all nodes u guarantee small change in terms of $\|\mathbf{E}\|_2$. When would $\|\mathbf{E}_u\|_1$ be small for a particular node? If α_u is small ($\alpha_u \approx 0$), then $\alpha_u/(1-\alpha_u) \approx 0$ and $1-\alpha_u/(1-\alpha_u) \approx 1$. Therefore the right hand side of Eq. (4.9) becomes approximately:

$$\|\mathbf{E}_u\|_1 \approx \frac{\Delta_u^+}{\sqrt{d'_u \delta'_u}} + \frac{\Delta_u^-}{\sqrt{d_u \delta_u}}. \tag{4.13}$$

This approximation holds, which in turn leads to a small $\|\mathbf{E}_u\|_1$, if we add and delete edges only between nodes with large degrees. The approximation becomes equality in the case when the degree distribution is preserved such as in Section 5.5. When would $\|\mathbf{E}_u\|_1$ be small for all nodes? Intuitively, this requires perturbations to be distributed across the graph, i.e., not concentrated around any one node. Therefore, spectral graph filters are most robust when we a) add or delete edges between high degree nodes, and b) do not perturb too much around any one node. In the next section, we empirically verify the looseness of each of the bounds and the intuition it provides.

4.6 Experiments

We empirically verify the looseness of the bounds derived in the previous section. We perform an extensive study of the looseness of these bounds by considering a variety of experimental conditions in terms of different graph types (both random graph models and real-world graph data) and perturbation strategies. Clearly, as the overall bound in Eq. (4.12) is obtained from a chain of inequalities, its looseness is affected by the looseness of each individual bound (Eqs. (4.3), (4.4), (4.5), (4.9), (4.10)). For completeness, the looseness of the inequalities relating the relative output distance and the filter distance (Eq. (4.3)) is illustrated in Fig. B.3, and that of the inequality relating the filter distance and the constant times the error norm (Eq. (4.4)) in Fig. B.4.

4.6.1 Experimental setup

We generate synthetic graphs on 100 nodes, using different random graph models. With the exception of the stochastic block model and the real-world data, we generate features on the nodes of the graph by taking a random convex combination of the first 10 eigenvectors of the normalised graph Laplacian. The latter results in

relatively smooth signals on the graph. For the stochastic block model, we generate graphs with three equal-size communities generating each community's features using a Gaussian with means 2, 0, and -2 respectively and unit variance. For the real-world data we select a continuous feature channel and normalise by subtracting the mean and dividing by the standard deviation. Gaussian noise is then added to generate noisy signals at a signal-to-noise ratio of 0 dB (equal levels of signal and noise).

For the sake of simplicity, we focus in this paper on a fixed low-pass filter $g(\lambda) = (1 + \lambda)^{-1}$, which has been widely used for signal denoising [103] and semi-supervised learning [8] tasks. This filter has stability constant $C = 1$ and thus satisfies the inequality $\|g(\mathbf{L}) - g(\mathbf{L}_p)\|_2 \leq \|\mathbf{E}\|_2$, due to Proposition 2. We note though that the experiments may be extended to any type of linearly stable graph filter. We compare the filtering outcome before and after perturbation to the graph topology in a signal denoising task. To this end, we are interested in bounding the relative output distance between the denoised signal before and after perturbation, for different random graph models and perturbation strategies. The magnitude of the perturbation is set at $\lfloor 10\% \cdot |\mathcal{E}| \rfloor$ edge edits. Each experiment is repeated 100 times using different random seeds. We consider varying the experimental settings such as the size of the graphs, amount of the perturbation and level of noise in Appendix B.4.

We note that in some experiments the assumption of Lemma 4 is not satisfied, i.e., for some node u it holds that $\alpha_u \geq 1$. We call an experiment valid if the assumption holds for all nodes. The validity of an experiment depends on both the strategy of perturbation and the type of graph used (Table B.2). We only report results from valid experiments for plots directly related to the bounds given in Lemma 4 or Theorem 3. We discuss this further in Supplementary Material.

We use a variety of random graph models including the Erdős-Rényi model (ER), Barabási–Albert model (BA), Watts–Strogatz model (WS), K-regular graphs (K-Reg), Stochastic Block Model (SBM), and assortative graphs (Assortative) [6, Chapter 7]. The random graph models include graphs with low variance (WS and K-Reg) and high variance (SBM, ER, Assortative and BA) in degree distribution. We also consider two real-world data sets, namely PROTEINS_full and ENZYMES. To control for graph size we used 100 graphs with n from 40 to 50 for ENZYMES and 50 to 75 for PROTEINS_full. The standard deviation of the node degrees averaged across the 100 graphs were 1.01 for ENZYMES and 1.03 for PROTEINS_full, similar to synthetic graphs with low degree variance. The standard deviation of the degree distribution averaged over each graph type is given in Table B.1. Further details of the random graph models are found in Supplementary Material.

The perturbation strategies under consideration, for a fixed budget, are as fol-

lows: 1) randomly selecting edges to delete (Delete); 2) randomly selecting edges to add (Add); 3) using half the budget to randomly add and half to randomly delete (Add/Delete); 4) using degree preserving double edge rewiring as described in Section 5.5 (Rewire), which is a special case of Add/Delete (we consider a single double edge rewiring to be four edits, i.e., two deletions and two additions); 5) projected gradient descent (PGD), which is used to find adversarial examples by perturbing the graphs similarly to that described in Xu et al. [150]; and 6) sequentially deleting or adding edges in a greedy manner to minimise $\|\mathbf{E}\|_1$ (Robust). Further details of the perturbation strategies are described in Supplementary Material.

4.6.2 How tight is the bound $||\mathbf{E}||_2 \leq ||\mathbf{E}||_1$?

We upper bound $\|\mathbf{E}\|_2$ using the inequality given in Eq. (4.5). In order to quantify the tightness of the bound, we compare in Fig. B.5 the values of $\|\mathbf{E}\|_1$ and $\|\mathbf{E}\|_2$ for different perturbation strategies, by illustrating their correlation. We note that Robust leads to the smallest values of $\|\mathbf{E}\|_2$ among all perturbation strategies. This is expected as Eq. (4.5) tells us that small values of $\|\mathbf{E}\|_1$, which are achieved with the Robust perturbation strategy, guarantee small values of $\|\mathbf{E}\|_2$. As a matter of fact, we observe experimentally that the two norms are correlated ($r = 0.90$), confirming that in practice if we observe $\|\mathbf{E}\|_2$ to be small then $\|\mathbf{E}\|_1$ is likely to be small too.

We show in Fig. B.6 the looseness of the bound given in Eq. (4.5) among the different perturbation strategies and graph models[2]. We see that the bound is tightest for Robust and Rewire. It is interesting to observe that Rewire sometimes gives a tight bound for the graphs with low variance in degree distribution.

4.6.3 How tight are the bounds on $||\mathbf{E}||_1$ and $||\mathbf{E}||_2$?

We now turn our attention to the bound given in Section 4.5. As well as calculating the overall value of $||\mathbf{E}_u||_1$ we can also compute the contributions of the three terms in Eq. (5.2). This allows us to evaluate the looseness of each term as well as the overall looseness of Eq. (4.9). For each experiment we selected the node u such that $u = \arg\max\|\mathbf{E}_u\|_1$ and calculated the terms and bounds for this node. The results for each term are shown in Fig. B.7 and that for the overall looseness in Fig. 4.1.

As one can see from Fig. B.7, the bounds for particular terms are tight in certain scenarios. For example, the inequality in Eq. (4.6) becomes equality when $d_v = \delta_u$ for all neighbouring nodes v of u, which is the case for 3-Reg graphs. The inequality in Eq. (4.6) and Eq. (4.7) is loosest when $\delta_u \ll d_v$ for many neighbouring nodes v.

[2]We do not display outliers for figures that appear in main text to make the rest of the data easier to visualise.

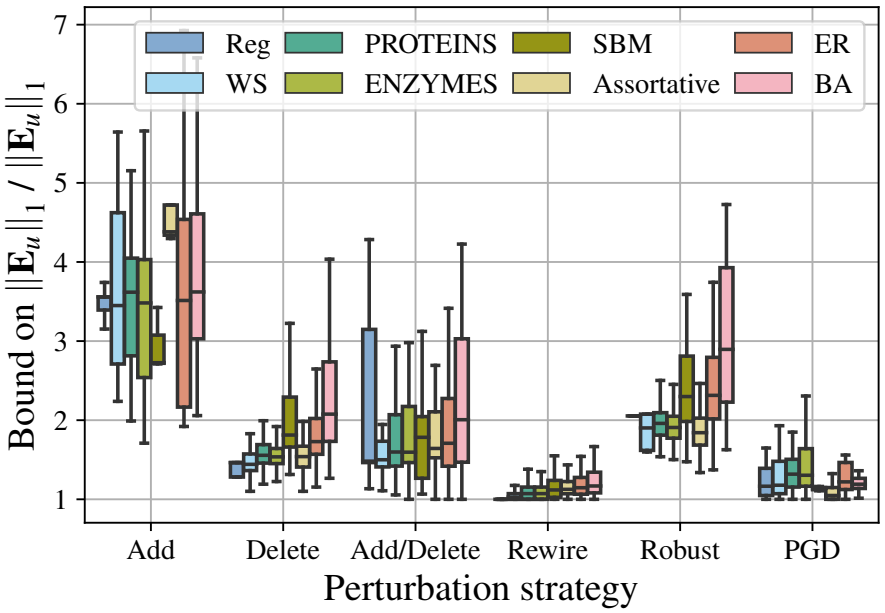


Figure 4.1: Looseness of the bound given in Eq. (4.9).

This is likely to occur when the degree distribution has high variance, explaining why the bound for the first and second term are looser for graphs with higher variance in degree distribution. The bound on the third term is the loosest in practice. From Fig. 4.1, the overall bound is tightest for the Rewire strategy, and this is because both the third term and the bound for it are zero in this case.

Fig. 4.2 shows the looseness of the bound on $||\mathbf{E}||_2$ in Eq. (4.10). The overall pattern is similar to that in Fig. 4.1, where the bound is tight in some rewiring experiments. In general the bound performs poorly on BA graphs, likely due to the skewed degree distribution.

4.6.4 When are filters robust?

In this section we take a holistic view of the bound in Eq. (4.12), considering how the relative output distance is affected by the perturbation strategy, graph model, and the statistics that appear throughout our chain of inequalities. We use insight from our bounds to demonstrate scenarios where a filter is robust. Fig. 4.3 shows how the different graphs and strategies effect each of the quantities that appear in our bounds. In many cases, Robust gives overall the smallest values where PGD gives the largest, which is expected due to the nature of these strategies. It is interesting to note that the Robust strategy gives smallest relative output distance for synthetic graphs with high degree variance (ER, Assortative, BA and SBM), as well as the real-world data sets, but not for those with low degree variance (WS, 3-Reg). When the degree distribution is flat all strategies perturb edges with similar endpoint degrees due to the lack of choice. On the other hand, the PGD strategy

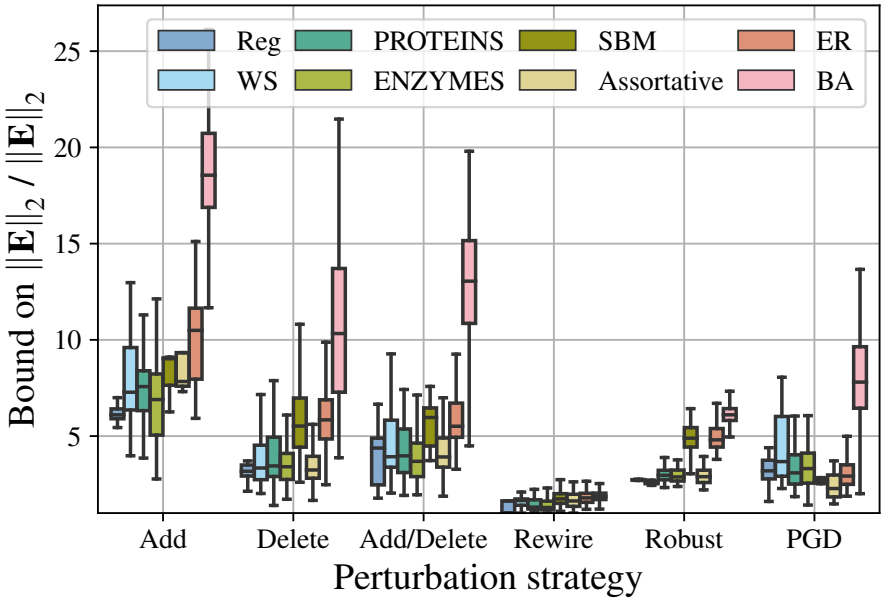


Figure 4.2: Looseness of the bound given in Eq. (4.10).

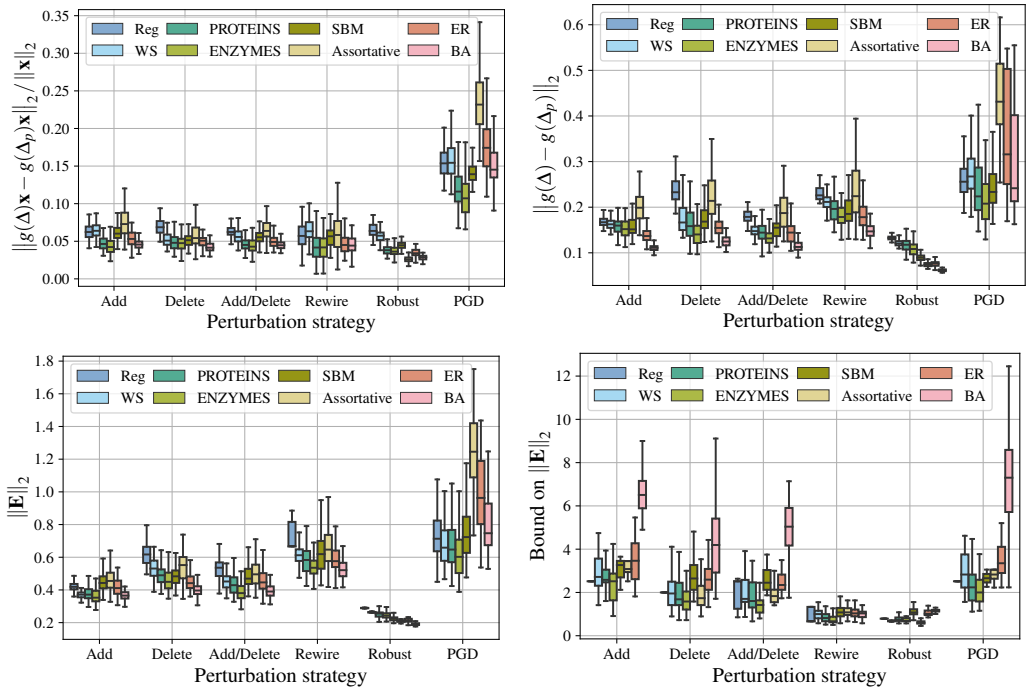


Figure 4.3: How different statistics vary across experimental setups.

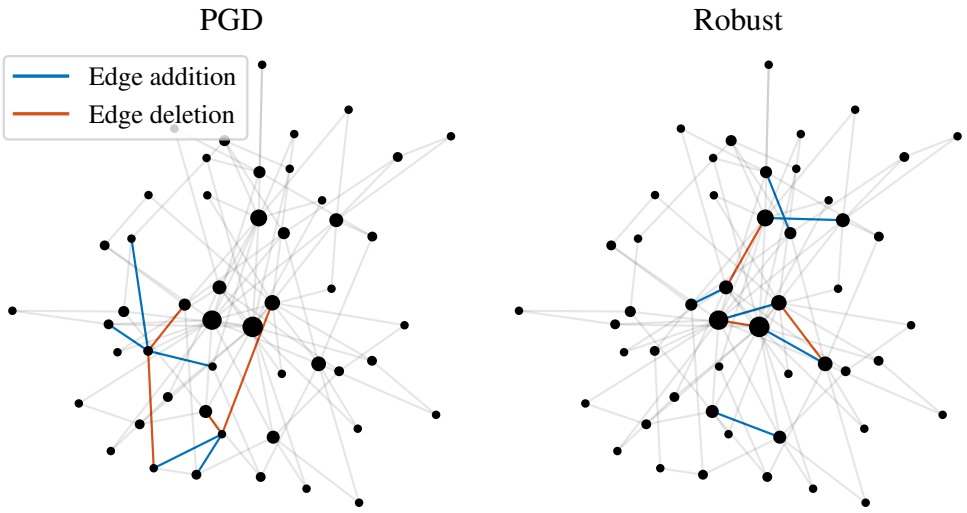


Figure 4.4: Perturbations of BA graphs ($n = 50$). The original and both perturbed graphs have a diameter of 5. The size of the node is proportional to the node degree.

gives larger changes to graphs with higher variance in degree distribution, suggesting the existence of low degree nodes or non-uniform degree distributions that are more vulnerable to adversarial attacks.

In summary, our bounds suggest that filters are robust when we modify edges where the endpoint degrees are high and that the perturbation is distributed across the graph. BA graphs of n nodes have a small diameter that grows asymptotically $\mathcal{O}(\log n / \log \log n)$ [17]. Consequentially, in our experiments on small BA graphs, most edges that are added and deleted are in close proximity. Furthermore, BA graphs have a power-law degree distributions. This type of graph model allows us to control for the distribution of the perturbation and observe instead how the endpoint degrees change across strategies. One can see how PGD tends to target small-degree nodes whereas Robust targets edges connected to the large-degree hubs (Fig. 4.4).

We finally control for the degree distribution by considering K-regular graphs. In Fig. 4.5 we can see that Robust deletes and adds edges between nodes in a distributed manner, whereas PGD tends to add edges that are adjacent to each other. This verifies the insight from the bound in terms of robustness with respect to spatial distribution of perturbation.

4.7 Discussion

In this chapter, we develop novel interpretable bounds to help elucidate certain types of perturbations against which spectral graph filters are robust. The bound introduced in Chapter 3 were tightened. We show that filters are robust when we modify edges where the endpoint degrees are high, and the perturbation is distributed across

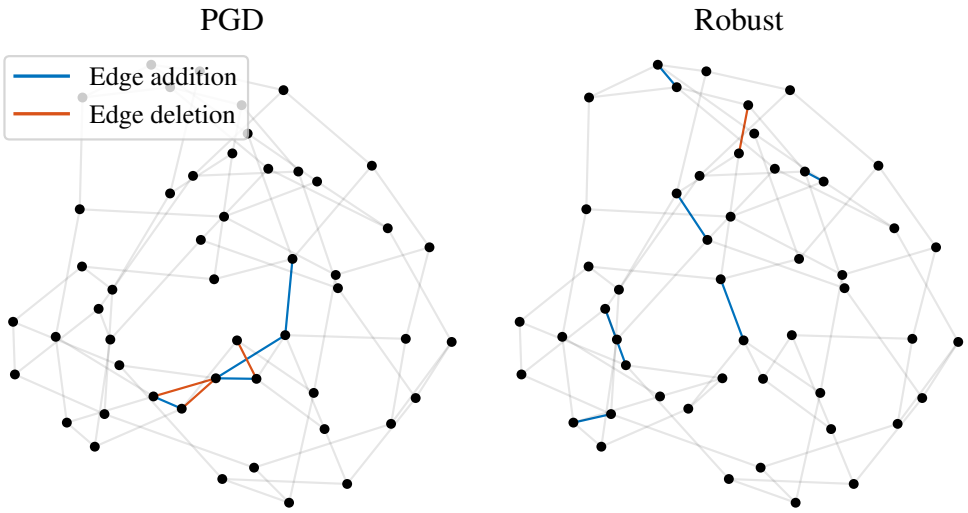


Figure 4.5: Perturbations of 3-regular graphs ($n = 50$).

the graph. Although these bounds are likely to be loose in practice, they provide qualitative insight which we validate through extensive experiments.

We believe that our work can be used in future research to investigate further the stability of graph-based machine learning algorithms. Studying additional perturbation models beyond edge deletion/addition, relaxing the assumption on α_u, allowing nodes to become isolated, and considering perturbation to node features, may all increase the applicability of the framework to practical scenarios. Further statistical investigation into how much the role of degree and edge locality effect stability is an important future direction. Considering weighted graphs is another natural extension of the proposed bounds. Our study is limited to a single fixed filter operating on a particular class of graph shift operator.

In the next chapter we prove that stability bounds extend to some graph neural network architectures. Graph neural networks are multilayered neural networks which use specific spectral graph filters as fixed non-learnable layers, so they are a natural direction to extend the bounds covered in this chapter and the chapter prior.

In this chapter we made use of an adversarial attack (PGD) to measure how much a filter output can be perturbed. Adversarial attacks can be used as a tool for measuring the maximum instability of a filter. It is complementary way of measuring robustness of machine learning models compared to the approach of using upper bounds on the output that we have taken thus far. We will introduce a state-of-the-art adversarial attack for graph classifiers in Chapter 6.

Chapter 5

On the Stability of Graph Convolutional Neural Networks under Edge Rewiring

Graph neural networks are experiencing a surge of popularity within the machine learning community due to their ability to adapt to non-Euclidean domains and instil inductive biases. Despite this, their stability, i.e., their robustness to small perturbations in the input, is not yet well understood. Although there exists some results showing the stability of graph neural networks, most take the form of an upper bound on the magnitude of change due to a perturbation in the graph topology. However, the change in the graph topology captured in existing bounds tend not to be expressed in terms of structural properties, limiting our understanding of the model robustness properties. In this chapter, similar to the previous chapter, we develop an interpretable upper bound elucidating that graph neural networks are stable to rewiring between high degree nodes. We focus primarily on edge rewiring to make the interpretation simpler. This bound and further research in bounds of similar type provide further understanding of the stability properties of graph neural networks.

5.1 Introduction

Recently, there has been an increasing amount of research on the use of machine learning models which operate on graph-structured data [18, 147]. Graphs encode pairwise interactions and can help model non-Euclidean data such as social networks and molecules as well as impart inductive biases [7]. Graph signal processing (GSP), for example, generalises traditional signal processing to network domains allowing us to transfer many of the existing tools such as filtering and sampling [36, 43]. GSP

also allows us to generalise convolutions providing a framework to design graph convolutional neural networks (GCNNs) where the convolutional layers are filter banks of spectral graph filters [42, 87, 10]. To date, much of the research has focused on the predictive performance of these models with far fewer studies looking at their theoretical properties.

Like their Euclidean counterparts, GCNNs are susceptible to adversarial attacks [164, 37, 74]. An adversarial attack is a small but targeted perturbation of the input which causes large changes in the output [128]. In the case of GCNNs, the modification of a few edges in the input graph can change significantly the learned node representations so that the prediction in the downstream task, typically node or graph classification, switches from a correct to incorrect prediction. One approach to understanding the vulnerability of these models to adversarial attacks is to consider their robustness against perturbation. Robustness is an important property for reliable deployment of machine learning models in the real world, especially in domains where adversaries are common (e.g., on the web).

Existing literature has shown that large functional classes of spectral graph filters, a key component of GCNNs, are bounded by the magnitude of the change in the input graph. The main drawback of these approaches is that the bounds involved in quantifying stability to specific changes in the topology do not have natural structural interpretations. For example, if we delete just a single edge, it is unclear how loose the bound on the output change will be even if we know the statistical properties about the edge and endpoint nodes.

Our main goal is to extend bounds found in the existing literature so that they have an interpretation in terms of the structural properties of the existing and perturbed graphs. To do this, we focus on polynomial graph filters, and build on our previous work [79], by providing a new bound that is tighter and generalises to the family of normalised augmented adjacency matrices. We then bound the change in normalised augmented adjacency matrix by considering the largest change around each node where the change admits a structural interpretation. As a specific example, we consider perturbations that do not modify the degree distribution of the graph (i.e., double edge rewiring, illustrated in Fig. 5.1). We then discuss under which scenarios, deleting and adding edges between nodes, guarantee the filter to be robust to perturbations. To demonstrate how these theoretical results can be combined, we bound the change in the representations learned by two well-known GCNN architectures. Specifically, we consider simple graph convolutional networks [144] and multilayered graph convolutional network [81] for scenarios where the graph topology is perturbed using edge rewiring.

5.2 Related work

Gama et al. prove that a class of spectral graph filters are stable to changes in the graph topology [47]. Furthermore, GCNNs using filters in this class as filter banks and ReLU nonlinearities are also stable. The main difference between this work and the work presented in this paper is on how the magnitude of perturbation is measured. In [47] the authors posit that simple additive error as a measure does not reflect the fact that the distance between isomorphic graphs can be non-zero. To address this concern they consider a relative measure of perturbation and consider all node permutations for the perturbed graph. We believe that the additive error approach of [79, 86] and this work, which does not consider node permutations, and the approach of [47] are both useful. For example, if the graph represents a polygon mesh, then the node labelling in the perturbed graph does not have any meaningful interpretation, as they are just used to construct matrix representations. In this case, considering permutations is appropriate. However, in a social network, the node labelling will typically correspond to the identification of a user, in which case it makes sense to consider the labelling as fixed between the original and perturbed graph.

5.3 Problem formulation

We consider unweighted and undirected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a finite set of nodes and $\mathcal{E}$ is the edge set. The adjacency matrix $\mathbf{A}$ encodes connections between nodes with $\mathbf{A}_{uv} = 1$ if there is an edge between nodes u and v and zero otherwise. The degree d_u of a node u is the number of nodes that u is connected to. A graph signal is a function $x : \mathcal{V} \to \mathbb{R}$ that assigns a scalar value to each node. By fixing a labelling of the nodes we can represent this as a vector $\mathbf{x} \in \mathbb{R}^n$ where $\mathbf{x}_i$ is the function value of node i and $n = |\mathcal{V}|$ is the number of nodes in the graph. We will be concerned with the normalised augmented adjacency matrix $\mathbf{\Delta}_\gamma = \mathbf{D}_\gamma^{-1/2}\mathbf{A}_\gamma\mathbf{D}_\gamma^{-1/2}$ where $\mathbf{A}_\gamma = \mathbf{A} + \gamma\mathbf{I}$ and $\mathbf{D}_\gamma = \mathbf{D} + \gamma\mathbf{I}$ with $\gamma \geq 0$. When $\gamma = 0$ this matrix is the normalised adjacency matrix. For simplicity, we drop the γ subscript when the context is clear or the specific value of γ is unimportant. We can write $\mathbf{\Delta}$ in terms of its entries as

$$\mathbf{\Delta}_{uv} = \begin{cases} \frac{\gamma}{d_u+\gamma} & \text{if } u = v \\ \frac{1}{\sqrt{(d_u+\gamma)(d_v+\gamma)}} & \text{if } \mathbf{A}_{uv} = 1 \text{ and } u \neq v\,. \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Normalised augmented adjacency matrices admit an eigendecomposition $\mathbf{\Delta} =$

$\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$ where the columns of $\mathbf{U}$ are the orthonormal eigenvectors, and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots \lambda_n)$ where $\lambda_1 \leq, \dots, \leq \lambda_n$ is the diagonal matrix of eigenvalues. The graph Fourier transform of a signal $\mathbf{x}$ is given by $\hat{\mathbf{x}} = \mathbf{U}^\intercal\mathbf{x}$, with the inverse graph Fourier transform defined as $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. With a notion of Fourier basis, a spectral graph filter is defined as a function $g : \mathbb{R} \to \mathbb{R}$ which amplifies and attenuates specific frequencies. We can filter a signal by applying the filter to the graph shift operator (in this case $\mathbf{\Delta}$) directly

$$\mathbf{U}\,\text{diag}(g(\lambda_1), \dots g(\lambda_n))\mathbf{U}^\intercal\mathbf{x} = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^\intercal\mathbf{x} = g(\mathbf{\Delta})\mathbf{x}.$$

In this work $\|\cdot\|_2$ represents the Euclidean norm when applied to vectors and the operator norm when applied to matrices. We will also consider the Frobenius norm $\|\mathbf{A}\|_F^2 = \sum_{i,j}\mathbf{A}_{i,j}^2$, the matrix one norm $\|\mathbf{A}\|_1 = \max_i \sum_j |\mathbf{A}_{ij}|$ and the matrix infinity norm $\|\mathbf{A}\|_\infty = \max_j \sum_i |\mathbf{A}_{ij}|$. The eigenvalues of the normalised augmented adjacency matrix lie in the interval $[-1, 1]$. Furthermore, 1 is always an eigenvalue so $\|\mathbf{\Delta}\|_2 = 1$.

Given a graph $\mathcal{G}$ and a perturbed graph $\mathcal{G}_p$ with normalised augmented adjacency matrices $\mathbf{\Delta}$ and $\mathbf{\Delta}_p$ respectively, our goal is twofold. First, we want to understand the stability of spectral graph filters in terms of structural perturbation, by providing bounds that quantify the change in the output. Second, we want to find sufficient conditions for this change to be small, by using interpretable structural properties of the graphs and the perturbation. We address the first goal in Sec. 5.4 and the second in Sec. 5.5. Furthermore, we demonstrate how these theoretical contributions can be combined to give insight into the stability of GCNNs. In Sec. 5.6, we combine the results of Sec. 5.4 and Sec. 5.5 to show how we can provide sufficient conditions for the stability of two popular GCNN architectures.

5.4 Stability of polynomial filters

Our notion of stability is based on relative output distance which is bounded by the filter distance

$$\frac{\|g(\mathbf{\Delta})\mathbf{x} - g(\mathbf{\Delta}_p)\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \leq \max_{\mathbf{x}\neq 0} \frac{\|g(\mathbf{\Delta})\mathbf{x} - g(\mathbf{\Delta}_p)\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \stackrel{\text{def}}{=} \|g(\mathbf{\Delta}) - g(\mathbf{\Delta}_p)\|_2.$$

In [79] we bounded the filter distance of polynomial filters by some constant times the error $\|\mathbf{E}\|_2$, where the constant depends on the filter and the error is the magnitude of the difference between normalised Laplacian matrices. We say that by satisfying this condition the filters are stable. Although the focus of this paper is not on tight bounds, we improve this bound by finding a smaller constant for polynomial filters. To do so we will use the following Lemma.

Lemma 5 (Lemma 3, [86]). *Suppose* $\mathbf{B}, \mathbf{D}, \mathbf{E} \in \mathbb{C}^{N\times N}$ *are Hermitian matrices satisfying* $\mathbf{B} = \mathbf{D} + \mathbf{E}$*, and* $\|\mathbf{B}\|_2, \|\mathbf{D}\|_2 \leq C$ *for some* $C > 0$*. Then for every* $l \geq 0$

$$\|\mathbf{B}^l - \mathbf{D}^l\|_2 \leq lC^{l-1}\|\mathbf{E}\|_2.$$

From here on in we will write $\mathbf{E} = \mathbf{\Delta_p} - \mathbf{\Delta}$ to be the difference of normalised augmented adjacency matrices between a graph $\mathcal{G}$ and a perturbed version of the graph $\mathcal{G}_p$. The smaller constant is given by the following proposition.

Proposition 3. *Let* $\mathbf{\Delta}$ *and* $\mathbf{\Delta}_p$ *be the normalised augmented adjacency matrix for* $\mathcal{G}$ *and* $\mathcal{G}_p$*. Consider a polynomial graph filter* $g_\theta(\lambda) = \sum_{k=0}^{K} \theta_k \lambda^k$*. Then*

$$\|g_\theta(\mathbf{\Delta}) - g_\theta(\mathbf{\Delta}_p)\|_2 \leq \sum_{k=1}^{K} k|\theta_k|\|\mathbf{E}\|_2.$$

Proof. Using the triangle inequality followed by an application of Lemma 5 with constant $C = 1$ we get

$$\|g_\theta(\mathbf{\Delta}) - g_\theta(\mathbf{\Delta}_p)\|_2 = \|\sum_{k=1}^{K} \theta_k(\mathbf{\Delta}^k - \mathbf{\Delta}_p^k)\|_2 \leq \sum_{k=1}^{K} |\theta_k|\|\mathbf{\Delta}^k - \mathbf{\Delta}_p^k\|_2 \leq \sum_{k=1}^{K} k|\theta_k|\|\mathbf{E}\|_2. \quad \square$$

In this work, we assume that the parameters of the model are fixed before and after perturbation. In the adversarial learning literature, an attack that modifies the input to cause large changes in the output whilst the model parameters are fixed is known as an evasion attack [74]. Robust models in the context of our work are those that are robust to evasion attacks with respect to the graph structure.

5.5 Robustness to edge rewiring perturbations

In this section we bound the error term $\|\mathbf{E}\|_2$ by interpretable properties relating to the structural change. Consider a graph $\mathcal{G}$ which we perturb to arrive at $\mathcal{G}_p$. Our approach to upper bounding $\|\mathbf{E}\|_2$ relies on the inequality $\|\mathbf{E}\|_2^2 \leq \|\mathbf{E}\|_1\|\mathbf{E}\|_\infty$ [65, Section 6.3]. As $\mathbf{E}$ is symmetric $\|\mathbf{E}\|_1 = \|\mathbf{E}\|_\infty$ giving $\|\mathbf{E}\|_2 \leq \|\mathbf{E}\|_1$. There may exist strategies which give tighter bounds, but the benefit of this approach to bounding the error term is that $\|\mathbf{E}\|_1$ leads to an interpretation in the structural domain. For a matrix $\mathbf{E}$ we write $\mathbf{E}_u$ as the uth column of $\mathbf{E}$ so $\mathbf{E}_u^\intercal$ is the uth row. The row $\mathbf{E}_u^\intercal$ corresponds to the node u in the graph. By definition $\|\mathbf{E}\|_1 = \max_{u\in\mathcal{V}}\|\mathbf{E}_u^\intercal\|_1$ where $\|\mathbf{E}_u^\intercal\|_1 = \sum_v |\mathbf{E}_{uv}|$ is the Manhattan norm of the row. Perturbations which cause small changes to $\|\mathbf{E}_u^\intercal\|_1$ over all nodes u guarantee small change in terms of $\|\mathbf{E}\|_2$.

The focus of this section is on developing an interpretable upper bound on $\|\mathbf{E}\|_2$. Before establishing an upper bound, we briefly mention the following lower bound

to the error term

$$\max_{i,j}|\mathbf{E}_{ij}| \stackrel{\text{def}}{=} \|\mathbf{E}\|_{\max} \leq \|\mathbf{E}\|_2.$$

This lower bound gives us sufficient conditions for large values of the error term. For example, deleting or adding an edge such that $\sqrt{(d_u+\gamma)(d_v+\gamma)}$ is small (e.g., if both degrees are small) will cause $\|\mathbf{E}\|_2$ to be large. In these cases, our bound will be loose and cannot provide guarantees about the filter robustness.

We aim to understand what type of perturbations spectral graph filters may be robust to. As a specific example consider perturbations that preserve the degree of the nodes. For this scenario we can consider how the entries change from $\boldsymbol{\Delta}$ to $\boldsymbol{\Delta}_p$ by considering Eq. (5.1) to write a closed-form for $\|\mathbf{E}_u^{\intercal}\|_1$. We will write $\mathcal{A}_u$ to be the set of edges added around a node u and $\mathcal{D}_u$ to be the set of edges deleted around a node u. The diagonal entries of $\boldsymbol{\Delta}$ and $\boldsymbol{\Delta}_p$ remain unchanged, and each edge deletion flips the entry from zero to the second case of Eq. (5.1), whilst edge addition flips the entry the other way. This insight lets us write $\|\mathbf{E}_u^{\intercal}\|_1$ in closed-form as

$$\begin{aligned}\|\mathbf{E}_u^{\intercal}\|_1 &= \sum_{v\in\mathcal{D}_u}\frac{1}{\sqrt{(d_u+\gamma)(d_v+\gamma)}} + \sum_{v\in\mathcal{A}_u}\frac{1}{\sqrt{(d_u+\gamma)(d_v+\gamma)}} \\ &= \frac{1}{\sqrt{d_u+\gamma}}\left(\sum_{v\in\mathcal{D}_u}\frac{1}{\sqrt{d_v+\gamma}} + \sum_{v\in\mathcal{A}_u}\frac{1}{\sqrt{d_v+\gamma}}\right) \end{aligned} \tag{5.2}$$

One such perturbation is edge rewiring that preserves degree distribution. We define double edge rewiring as a function of two edges (u,v) and (u',v') such that u is not connected to u' or v' and similarly v is not connected to u' and v'. The operation consists of deleting edges (u,v) and (u',v') and adding edges (u,u') and (v,v'). This operation is depicted graphically in Fig. 5.1. Although the precise definition of rewiring in this work is slightly different, the idea of rewiring has been proposed as a strategy to make modifications imperceptible in the context of topological adversarial attacks [91]. Approximately preserving the degree distribution has also been a criterion used to define imperceptibility [164]. Beyond the adversarial attack literature, double edge rewiring has been used to model changes in a network where the capacity of a node is fixed and remains at full load such as in communication networks [11].

We will write R_u to be the number of rewiring operations involving u and write δ_u to be the smallest degree amongst either the nodes u disconnects with or is now connected to. Each rewiring causes a single edge deletion and edge addition for each node involved so that the number of terms in each sum, the cardinality of sets $\mathcal{D}_u$

(a) Before rewiring (b) After rewiring

Figure 5.1: In the rewiring operation the red edges are deleted and the blue edges are added. The degree of each node remains the same.

and $\mathcal{A}_u$, is R_u. Using this we can bound Eq. (5.2) to get that

$$\|\mathbf{E}_u^\intercal\|_1 \leq \frac{1}{\sqrt{d_u+\gamma}}\left(\sum_{v\in\mathcal{D}_u}\frac{1}{\sqrt{\delta_u+\gamma}}+\sum_{v\in\mathcal{A}_u}\frac{1}{\sqrt{\delta_u+\gamma}}\right)=\frac{2R_u}{\sqrt{(d_u+\gamma)(\delta_u+\gamma)}}.$$

The largest possible value for the right hand side of the above equation over all nodes u provides an upper bound for $\|\mathbf{E}\|_2$. From this, we can draw some conclusions as to when the filter will be robust to rewiring operations. The first is that one should not rewire around one node significantly as this will increase R_u. To keep $\|\mathbf{E}\|_1$ small, and thus $\|\mathbf{E}\|_2$ small, we must keep $\|\mathbf{E}_u^\intercal\|_1$ small for all nodes u suggesting the perturbation should be distributed across the graph. Related to this observation, in [79] it was numerically demonstrated that the locality of perturbations can play a role in the magnitude of the error. The second strategy to ensure robustness is to rewire between high degree nodes. This will cause d_u to be large and therefore $\|\mathbf{E}_u^\intercal\|_1$ will be small. Finally, using a normalised augmented adjacency matrix with larger values of γ can cause smaller changes.

5.6 Stability of GCNN Models

We make use of results in previous sections to analyse the stability of two popular GCNN models, i.e., the simple graph convolutional networks (SGCN) [144] and the multilayered graph convolutional network (GCN) [81].

5.6.1 SGCN

The SGCN model is motivated by considering a multilayered GCN with the activation functions removed. By removing the activation functions the model boils down to a fixed monomial filter of order K followed by applying a fully connected layer and a softmax layer to the node features.

Let the input data be given by the matrix $\mathbf{X}\in\mathbb{R}^{n\times d}$ where d is the dimension of the features associated with each node. We can consider $\mathbf{X}$ as d stacked graph signals which we will call feature maps. We will assume that each column (feature

map) of $\mathbf{X}$ has unit norm. The output is a matrix $\mathbf{Y} \in \mathbb{R}^{n\times c}$, representing class probabilities for each node. The SGCN model is defined as

$$\mathbf{Y} = \text{softmax}(\tilde{\mathbf{\Delta}}^K \mathbf{X}\mathbf{\Theta})$$

where $\text{softmax}(\mathbf{x})_i = \exp(\mathbf{x}_i)/\sum_i \exp(\mathbf{x}_i)$ normalises the rows to be probability distributions. We write $\tilde{\mathbf{\Delta}} \stackrel{\text{def}}{=} \mathbf{\Delta}_1$ as the normalised augmented adjacency matrix with $\gamma = 1$. In this subsection, we will analyse how the logits, the node representations before the softmax is applied, change. The softmax function has a Lipschitz constant of 1 so any bound on the logits can trivially be applied to the model outputs [51][Proposition 4]. We begin by stating the following Lemma.

Lemma 6. *Let* $\mathbf{B} \in \mathbb{R}^{m\times r}$ *and* $\mathbf{C} \in \mathbb{R}^{r\times n}$ *then*

$$\|\mathbf{BC}\|_F \leq \|\mathbf{B}\|_F \|\mathbf{C}\|_2.$$

Proof. By decomposing $\mathbf{BC}$ in terms of the rows $\mathbf{b}_k^\intercal$ of $\mathbf{B}$ we get

$$\|\mathbf{BC}\|_F^2 = \sum_{k=1}^{m} \|\mathbf{b}_k^\intercal \mathbf{C}\|_2^2 \leq \sum_{k=1}^{m} \|\mathbf{b}_k^\intercal\|_2^2 \|\mathbf{C}\|_2^2 = \|\mathbf{B}\|_F^2 \|\mathbf{C}\|_2^2. \tag{5.3}$$

The inequality follows from the observation that

$$\|\mathbf{b}^\intercal \mathbf{C}\|_2 = \|(\mathbf{b}^\intercal \mathbf{C})^\intercal\|_2 = \|\mathbf{C}^\intercal \mathbf{b}\|_2 \leq \|\mathbf{C}^\intercal\|_2 \|\mathbf{b}\|_2 = \|\mathbf{b}\|_2 \|\mathbf{C}\|_2.$$

Taking the square root of both sides of Eq. (5.3) gives the result. □

Using this we can provide a bound on how much the logits can change under the Frobenius norm. To motivate why we are interested in the Frobenius norm here, consider taking the Euclidean norm of each node output to measure the amount of change in node representation. Taking the mean squared error of these distances amounts to taking the Frobenius norm of the logits matrix.

Proposition 4. *The distance of the logits is bounded like so*

$$\|\tilde{\mathbf{\Delta}}^K \mathbf{X}\mathbf{\Theta} - \tilde{\mathbf{\Delta}}_p^K \mathbf{X}\mathbf{\Theta}\|_F \leq \sqrt{d} K \|\mathbf{E}\|_2 \|\mathbf{\Theta}\|_2$$

Proof. We first note that $\|\mathbf{X}\|_F = \sqrt{d}$. Using this, two applications of Lemma 6, and an application of Proposition 3 we get

$$\begin{aligned}
\|\tilde{\mathbf{\Delta}}^K \mathbf{X}\mathbf{\Theta} - \tilde{\mathbf{\Delta}}_p^K \mathbf{X}\mathbf{\Theta}\|_F &\leq \|\tilde{\mathbf{\Delta}}^K \mathbf{X} - \tilde{\mathbf{\Delta}}_p^K \mathbf{X}\|_F \|\mathbf{\Theta}\|_2 \\
&\leq \|\tilde{\mathbf{\Delta}}^K - \tilde{\mathbf{\Delta}}_p^K\|_2 \|\mathbf{X}\|_F \|\mathbf{\Theta}\|_2. \\
&\leq \sqrt{d} K \|\mathbf{E}\|_2 \|\mathbf{\Theta}\|_2.
\end{aligned}$$

□

5.6.2 Multilayer GCN

We now consider a multilayered GCN model. We again consider the logits of a model which this time consists of multiple GCN layers with pointwise non-linearities giving the lth layer representation as

$$\mathbf{X}^{(l)} = \sigma(\tilde{\mathbf{\Delta}}\mathbf{X}^{(l-1)}\mathbf{\Theta}^{(l)}),$$

where σ is the ReLU activation function and $\mathbf{\Theta}^{(l)}$ are the layer parameters. We will consider the number of feature maps to be the same throughout the model.

Proposition 5. *Let $\mathbf{X}^{(l)} = \sigma(\tilde{\mathbf{\Delta}}\mathbf{X}^{(l-1)}\mathbf{\Theta}^{(l)})$ where $\mathbf{X}^{(0)} \in \mathbb{R}^{n\times d}$ is the input feature maps, $\mathbf{\Theta}^{(l)} \in \mathbb{R}^{d\times d}$ are the weight matrices and L is the number of layers so $\mathbf{X}^L$ is the output features. Then*

$$\|\mathbf{X}^{(L)} - \mathbf{X}_p^{(L)}\|_F \leq \sqrt{d}L\|\mathbf{E}\|_2 \prod_{l=1}^{L} \|\mathbf{\Theta}^{(l)}\|_2.$$

Proof. We prove this by induction. The base case $L = 1$ follows immediately from Proposition 4. Consider a more general L. Then

$$\begin{aligned}
\|\mathbf{X}^{(L)} - \mathbf{X}_p^{(L)}\|_F &= \|\sigma(\tilde{\mathbf{\Delta}}\mathbf{X}^{(l-1)}\mathbf{\Theta}^{(l)}) - \sigma(\tilde{\mathbf{\Delta}}_p\mathbf{X}_p^{(l-1)}\mathbf{\Theta}^{(l)})\|_F \\
&\leq \|\tilde{\mathbf{\Delta}}\mathbf{X}^{(l-1)}\mathbf{\Theta}^{(l)} - \tilde{\mathbf{\Delta}}_p\mathbf{X}_p^{(l-1)}\mathbf{\Theta}^{(l)}\|_F \\
&\leq \|\tilde{\mathbf{\Delta}}\mathbf{X}^{(l-1)} - \tilde{\mathbf{\Delta}}_p\mathbf{X}_p^{(l-1)}\|_F\|\mathbf{\Theta}^{(l)}\|_2,
\end{aligned}$$

where the first inequality comes from ReLU having unit Lipschitz constant and the second being an application of Lemma 6. By using triangle inequality we bound the following term

$$\begin{aligned}
\|\tilde{\mathbf{\Delta}}\mathbf{X}^{(l-1)} - \tilde{\mathbf{\Delta}}_p\mathbf{X}_p^{(l-1)}\|_F &= \|\tilde{\mathbf{\Delta}}\mathbf{X}^{(l-1)} - \tilde{\mathbf{\Delta}}_p\mathbf{X}^{(l-1)} + \tilde{\mathbf{\Delta}}_p\mathbf{X}^{(l-1)} - \tilde{\mathbf{\Delta}}_p\mathbf{X}_p^{(l-1)}\|_F \\
&\leq \|\tilde{\mathbf{\Delta}}\mathbf{X}^{(l-1)} - \tilde{\mathbf{\Delta}}_p\mathbf{X}^{(l-1)}\|_F + \|\tilde{\mathbf{\Delta}}_p\mathbf{X}^{(l-1)} - \tilde{\mathbf{\Delta}}_p\mathbf{X}_p^{(l-1)}\|_F \\
&\leq \|\mathbf{E}\|_2\|\mathbf{X}^{(l-1)}\|_F + \|\mathbf{X}^{(l-1)} - \mathbf{X}_p^{(l-1)}\|_F.
\end{aligned}$$

Note that

$$\left\|\mathbf{X}^{(l)}\right\|_F = \left\|\sigma(\tilde{\mathbf{\Delta}}\mathbf{X}^{(l-1)}\mathbf{\Theta}^{(l)})\right\|_F \leq \left\|\mathbf{X}^{(l-1)}\right\|_F\left\|\mathbf{\Theta}^{(l)}\right\|_2,$$

so by recursivity we get that

$$\left\|\mathbf{X}^{(l)}\right\|_F \leq \left\|\mathbf{X}^{(0)}\right\|_F\left\|\mathbf{\Theta}^{(1)}\right\|_2 \dots \left\|\mathbf{\Theta}^{(l)}\right\|_2.$$

Recall that $\left\|\mathbf{X}^{(0)}\right\|_F = \sqrt{d}$. Using this observation and the inductive assumption we

get that

$$\begin{aligned}\left(\|\mathbf{E}\|_2\|\mathbf{X}^{(l-1)}\|_F + \|\mathbf{X}^{(l-1)} - \mathbf{X}_p^{(l-1)}\|_F\right)\|\mathbf{\Theta}^{(l)}\|_2 &\leq \sqrt{d}\|\mathbf{E}\|_2\prod_{l=1}^{L}\|\mathbf{\Theta}^{(l)}\|_2 \\ &+ \sqrt{d}(L-1)\|\mathbf{E}\|_2\prod_{l=1}^{L}\|\mathbf{\Theta}^{(l)}\|_2 \\ &= \sqrt{d}L\|\mathbf{E}\|_2\prod_{l=1}^{L}\|\mathbf{\Theta}^{(l)}\|_2. \qquad \square\end{aligned}$$

We note an element of similarity between this bound and the bound introduced in [128] for convolutional neural networks. We finish this section by combining Proposition 5 with the results from Sec. 5.5 to give the following.

Corollary 2. *Consider the GCN outputs $\mathbf{X}^{(L)}$ and $\mathbf{X}_p^{(L)}$ for a graph $\mathcal{G}$ and a perturbed graph $\mathcal{G}_p$ where the perturbed graph is a result of double edge rewiring. Let $\tilde{\mathbf{\Delta}}$ and $\tilde{\mathbf{\Delta}}_p$ be the corresponding normalised augmented adjacency matrices. Define d_u, δ_u and R_u as in Sec. 5.5, then the following holds*

$$\|\mathbf{X}^{(L)} - \mathbf{X}_p^{(L)}\|_F \leq \sqrt{d}L\prod_{l=1}^{L}\|\mathbf{\Theta}^{(l)}\|_2 \max_{u\in\mathcal{V}}\frac{2R_u}{\sqrt{(d_u+1)(\delta_u+1)}}.$$

A similar result holds for SGCN by combining Proposition 4 with the results from Sec. 5.5. We suspect these bounds will likely be loose in practice; nevertheless, they provide conceptional insights into the factors that may be related to the robustness of the GCN and SGCN models. In particular, we can reason when this bound will be small as in the final paragraph of Sec. 5.5, which relates interpretable structural perturbation to the robustness of these models.

5.7 Discussion

In this work, we bound the change in output of graph neural networks. This bound is reminiscent of early bounds introduced for convolutional neural networks [128]. We consider a specific form of topological perturbation, i.e., edge rewiring, to extend the bound to consist of terms which have a structural interpretation. We then demonstrate a practical application of this bound by applying it to the SGCN and GCN models.

Up until now, our focus has been on understanding stability through the use of output bounds and experiments inspired by these bounds. In the following chapter, we segue to another form of studying robustness, through the design and study adversarial attack. In Chapter 4 we saw how adversarial attacks can be used as a

tool for finding "worst-case" perturbations. We will also use our proposed adversarial attack strategy to study patterns of perturbation, similar also to those studied in Chapter 4.

Chapter 6

Adversarial Attacks on Graph Classification via Bayesian Optimisation

In the previous chapter, we proposed bounds on the embeddings generated by graph neural networks. These embeddings can be used for graph classification, whereby the network maps individual graphs to a label. Graph neural networks have been shown to be vulnerable to adversarial attacks. While the majority of the literature focuses on such vulnerability in node-level classification tasks, little effort has been dedicated to analysing adversarial attacks on graph-level classification, an important problem with numerous real-life applications such as biochemistry and social network analysis. The few existing methods often require unrealistic setups, such as access to internal information of the victim models, or an impractically-large number of queries. We present a novel Bayesian optimisation-based attack method for graph classification models. Our method is *black-box*, *query-efficient* and *parsimonious* with respect to the perturbation applied. We empirically validate the effectiveness and flexibility of the proposed method on a wide range of graph classification tasks involving varying graph properties, constraints and modes of attack. Finally, we analyse common interpretable patterns behind the adversarial samples produced, similar in spirit to the analysis presented in Section 4.6.4, which may shed further light on the adversarial robustness of graph classification models. An open-source implementation is available at `https://github.com/xingchenwan/grabnel`.

6.1 Introduction

Graphs are a general-purpose data structure consisting of entities represented by nodes and edges which encode pairwise relationships. Graph-based machine learning

models has been widely used in a variety of important applications such as semi-supervised learning, link prediction, community detection and graph classification [21, 162, 62]. Despite the growing interest in graph-based machine learning, it has been shown that, like many other machine learning models, graph-based models are vulnerable to adversarial attacks [126, 74]. If we want to deploy such models in environments where the risk and costs associated with a model failure are high e.g. in social networks, it would be crucial to understand and assess the model stability and vulnerability by simulating adversarial attacks.

Adversarial attacks on graphs can be aimed at different learning tasks. This paper focuses on graph-level classification, where given an input graph (potentially with node and edge attributes), we wish to learn a function that predicts a property of interest related to the graph. Graph classification is an important task with many real-life applications, especially in bioinformatics and chemistry [95]. For example, the task may be to accurately classify if a molecule, modelled as a graph whereby nodes represent atoms and edges model bonds, inhibits HIV replication or not. Although there are a few attempts on performing adversarial attacks on graph classification [37, 91], they all operate under unrealistic assumptions such as the need to query the target model a large number of times or access a portion of the test set to train the attacking agent. To address these limitations, we formulate the adversarial attack on graph classification as a black-box optimisation problem and solve it with Bayesian optimisation (BO), a query-efficient state-of-the-art zeroth-order black-box optimiser. Unlike existing work, our method is query-efficient, parsimonious in perturbations and does not require policy training on a separate labelled dataset to effectively attack a new sample. Another benefit of our method is that it can be easily adapted to perform various modes of attacks such as deleting or rewiring edges and node injection. Furthermore, we investigate the topological properties of the successful adversarial examples found by our method and offer valuable insights on the connection between the graph topology change and the model robustness.

The main contributions of our paper are as follows. First, we introduce a novel black-box attack for graph classification, GRABNEL[1], which is both query efficient and parsimonious. We believe this is the first work on using BO for adversarial attacks on graph data. Second, we analyse the generated adversarial examples to link the vulnerability of graph-based machine learning models to the topological properties of the perturbed graph, an important step towards interpretable adversarial examples that has been overlooked by the majority of the literature. Finally, we evaluate our method on a range of real-world datasets and scenarios including detecting the spread of fake news on Twitter, which to the best of our knowledge is the first analysis of this kind in the literature.

[1]Stands for *Graph Adversarial attack via BayesiaN Efficient Loss-minimisation.*

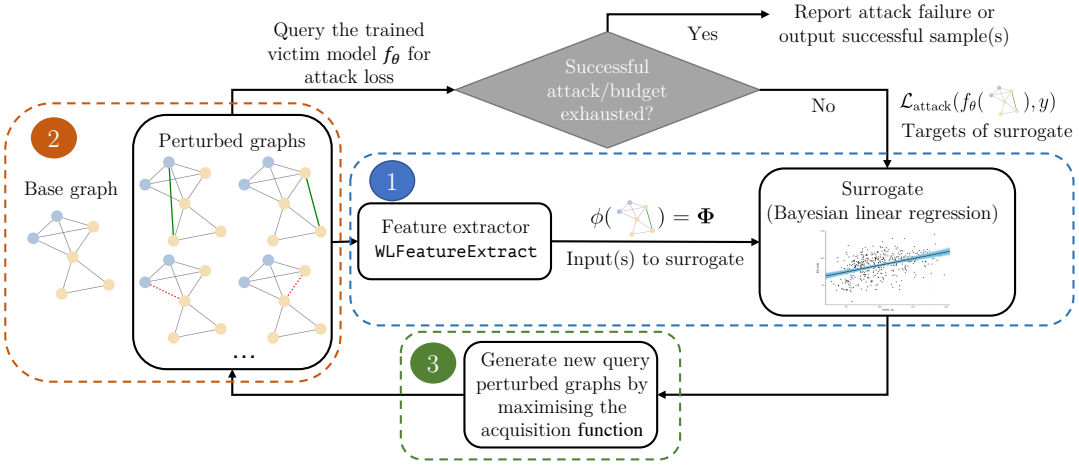


Figure 6.1: The overall pipeline of GRABNEL. The key components are explained in different paragraphs of Sec 6.2: *Surrogate model* describes the construction of the BO surrogate and the feature extractor (Block 1), *Sequential perturbation selection* describes how base graphs and perturbed graphs as candidates of adversarial attack are selected (Block 2), and *Optimisation of acquisition function* describes how new query points are generated by BO via optimising acquisition (Block 3). A detailed algorithmic description for GRABNEL is also available in App. C.1.

6.2 Proposed Method: GRABNEL

Problem Setup A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a set of nodes $\mathcal{V} = \{v_i\}_{i=1}^n$ and edges $\mathcal{E} = \{\mathbf{e}_i\}_{i=1}^m$ where each edge $\mathbf{e}_k = \{v_i, v_j\}$ connects between nodes v_i and v_j. The overall topology can be represented by the adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$ where $\mathbf{A}_{ij} = 1$ if the edge $\{v_i, v_j\}$ is present[2]. The attack objective in our case is to degrade the predictive performance of the trained victim graph classifier f_θ by finding a graph $\mathcal{G}'$ perturbed from the original test graph $\mathcal{G}$ (ideally with the minimum amount of perturbation) such that f_θ produces an incorrect class label for $\mathcal{G}$. In this paper, we consider the *black-box evasion* attack setting, where the adversary agent cannot access/modify the the victim model f_θ (i.e. network architecture, weights θ or gradients) or its training data $\{(\mathcal{G}_i, y_i)\}_{i=1}^L$; the adversary can only interact with f_θ by querying it with an input graph $\mathcal{G}'$ and observe the model output $f_\theta(\mathcal{G}')$ as pseudo-probabilities over all classes in a C-dimensional standard simplex.

Additionally, we assume that *sample efficiency* is highly valued: we aim to find adversarial examples with the minimum number of queries to the victim model. We believe that this is a practical and difficult setup that accounts for the prohibitive monetary, logistic and/or opportunity costs of repeatedly querying a (possibly huge and complicated) real-life victim model. With a high query count, the attacker may also run a higher risk of getting detected. Formally, the objective function of our

[2]We discuss the unweighted graphs for simplicity; our method may also handle other graph types.

BO attack agent can be formulated as a black-box maximisation problem:

$$\max_{\mathcal{G}' \in \Psi(\mathcal{G})} \mathcal{L}_{\text{attack}}\big(f_\theta(\mathcal{G}'), y\big) \quad \text{s.t.} \quad y = \arg\max f_\theta(\mathcal{G}) \tag{6.1}$$

where f_θ is the pretrained victim model that remains fixed in the evasion attack setup and y is the correct label of the original input $\mathcal{G}$. Denote the output logit for the class y as $f_\theta(\mathcal{G})_y$, the *attack* loss $\mathcal{L}_{\text{attack}}$ can be defined as:

$$\mathcal{L}_{\text{attack}}\Big(f_\theta(\mathcal{G}'), y\Big) = \begin{cases} \max_{t \in \mathcal{Y}, t \neq y} \log f_\theta(\mathcal{G}')_t - \log f_\theta(\mathcal{G}')_y & \text{(untargeted attack)} \\ \log f_\theta(\mathcal{G}')_t - \log f_\theta(\mathcal{G}')_y & \text{(targeted attack on class } t\text{)}, \end{cases} \tag{6.2}$$

where $f_\theta(\cdot)_t$ denotes the logit output for class t. Such an attack loss definition is commonly used both in the traditional image attack and the graph attack literature [23, 164] although our method is compatible with any choice of loss function. Furthermore, $\Psi(\mathcal{G})$ refers to the set of possible $\mathcal{G}'$ generated from perturbing $\mathcal{G}$. In this work, we experiment with a diverse modes of attacks to show that our attack method can be generalised to different set-ups:

- creating/removing an edge: we create perturbed graphs by flipping the connection of a small set of node pairs $\delta\mathbf{A} = \{\{u_i, v_i\}\}_{i=1}^{\Delta}$ of $\mathcal{G}$ following previous works [164, 37];
- rewiring or swapping edges: similar to [91], we select a triplet (u, v, s) where we either rewire the edge $(u \to v)$ to $(u \to s)$ (rewire), or exchange the edge weights $w(u, v)$ and $w(u, s)$ (swap);
- node injection: we create new nodes together with their attributes and connections in the graph.

The overall routine of our proposed GRABNEL is presented in Fig 6.1 (and in pseudo-code form in App C.1), and we now elaborate each of its key components.

Surrogate model The success of BO hinges upon the surrogate model choice. Specifically, such a surrogate model needs to 1) be flexible and expressive enough to locally learn the latent mapping from a perturbed graph $\mathcal{G}'$ to its attack loss $\mathcal{L}_{\text{attack}}(f_\theta(\mathcal{G}'), y)$ (note that this is different and generally easier than learning $\mathcal{G}' \to y$, which is the goal of the classifier f_θ), 2) admit a probabilistic interpretation of uncertainty – this is key for the exploration-exploitation trade-off in BO, yet also 3) be simple enough such that the said mapping can be learned with a small number of queries to f_θ to preserve sample efficiency. Furthermore, given the combinatorial nature of the graph search space, it also needs to 4) be capable of scaling to large

graphs (e.g. in the order of 10^3 nodes or more) typical of common graph classification tasks with reasonable run-time efficiency. Additionally, given the fact that BO has been predominantly studied in the continuous domain which is significantly different from the present setup, the design of a appropriate surrogate is highly non-trivial. To handle this set of conflicting desiderata, we propose to first use a *Weisfeiler-Lehman (*WL*) feature extractor* to extracts a vector space representation of $\mathcal{G}$, followed by a *sparse Bayesian linear regression* which balances performance with efficiency and gives an probabilistic output.

With reference to Fig. 6.1, given a perturbation graph $\mathcal{G}'$ as a proposed adversarial sample, the WL feature extractor first extracts a vector representation $\boldsymbol{\phi}(\mathcal{G}')$ in line with the WL subtree kernel procedure (but without the final kernel computation) [114]. For the case where the node features are discrete, let $x^0(v)$ be the initial node feature of node $v \in \mathcal{V}$ (note that the node features can be either scalars or vectors), we iteratively aggregate and hash the features of v with its neighbours, $\{u_i\}_{i=1}^{\deg(v)}$, using the original WL procedure at all nodes to transform them into discrete labels:

$$x^{h+1}(v) = \text{hash}\Big(x^h(v), x^h(u_1), ..., x^h(u_{\deg(v)})\Big), \ \forall h \in \{0, 1, \dots, H-1\}, \tag{6.3}$$

where H is the total number of WL iterations, a hyperparameter of the procedure. At each level h, we compute the feature vector $\boldsymbol{\phi}_h(\mathcal{G}') = [c(\mathcal{G}', \mathcal{X}_{h1}), ..., c(\mathcal{G}', \mathcal{X}_{h|\mathcal{X}_h|})]^\top$, where $\mathcal{X}_h$ is the set of distinct node features x^h that occur in all input graphs at the current level and $c(G', x^h)$ is the counting function that counts the number of times a particular node feature x^h appears in G'. For the case with continuous node features and/or weighted edges, we instead use the modified WL procedure proposed in [131]:

$$x^{h+1}(v) = \frac{1}{2}\Big(x^h(v) + \frac{1}{\deg(v)} \sum_{i=1}^{\deg(v)} w(v, u_i) x^h(u_i)\Big), \ \forall h \in \{0, 1, \dots, H-1\}, \tag{6.4}$$

where $w(v, u_i)$ denotes the (non-negative) weight of edge $e_{\{v,u_i\}}$ (1 if the graph is unweighted) and we simply have feature at level h $\boldsymbol{\phi}_h(\mathcal{G}') = \text{vec}(\mathbf{X}_h)$, where $\mathbf{X}_h$ is the feature matrix of graph $\mathcal{G}'$ at level h by collecting the features at each node $\mathbf{X}_h = \Big[x^h(1), ...x^h(v)\Big]$ and $\text{vec}(\cdot)$ denotes the vectorisation operator. In both cases, at the end of H WL iterations we obtain the final feature vector $\boldsymbol{\phi}(\mathcal{G}') = \text{concat}\big(\boldsymbol{\phi}_1(\mathcal{G}'), ..., \boldsymbol{\phi}_H(\mathcal{G}')\big)$ for each training graph in $[1, n_{\mathcal{G}'}]$ to form the feature matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathcal{G}'_1), ...\boldsymbol{\phi}(\mathcal{G}'_{|n_{\mathcal{G}'}|})]^\top \in \mathbb{R}^{|n_{\mathcal{G}'}| \times D}$ to be passed to the Bayesian regressor – it is particularly worth noting that the training graphs here denote inputs to train the surrogate model of the attack agent and are typically perturbed versions of a *test* graph $\mathcal{G}$ of the victim model; they are *not* the graphs that are used to train the victim model itself: in an evasion attack setup, the model is considered

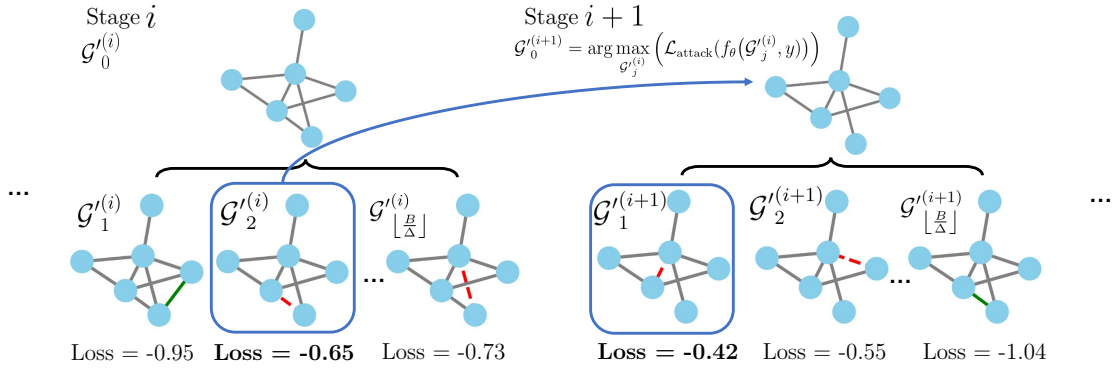


Figure 6.2: Sequential edge selection. At each stage, the BO agent sequentially proposes candidate graphs with edge edit distance of 1 from the base graph $\mathcal{G}'^{(i)}_0$ (which is the original unperturbed graph $\mathcal{G}$ at initialisation, or a perturbed graph that led to the largest increase in loss from the previous stage otherwise) by selecting the graph that maximises the acquisition function value amongst all candidates generated via sampling/genetic algorithm (see details at *Optimisation of acquisition function*). This procedure repeats until either the attack succeeds (i.e. we find a graph $\mathcal{G}'$ with $\mathcal{L}_{\text{attack}}(f_\theta(\mathcal{G}'), y) > 0$) or the maximum number of B queries to f_θ is exhausted.

frozen and the training inputs cannot be accessed by the attack agent any point in the pipeline. The WL iterations capture both information related to individual nodes and topological information (via neighbourhood aggregation), and have been shown to have comparable distinguishing power to some Graph Neural Network (GNN) models [96], and hence the procedure is expressive. Alternative surrogate choices could be, for example, GNNs with the final fully-connected layer replaced by a probabilistic linear regression layer such as the one proposed in [115]. However, in contrast to these, our extraction process $\mathcal{G}' \to \boldsymbol{\phi}(\mathcal{G}')$ requires no learning from data (we only need to learn the Bayesian linear regression weights) and therefore should lead to better sample efficiency. Alternatively, we may also use a Gaussian Process (GP) surrogate, such as the Gaussian Process with Weisfeiler-Lehman Kernel (GPWL) model proposed in [106] that directly uses a GP model together with a WL kernel. Nonetheless, while GPs are theoretically more expressive (although we empirically show in App. C.4.1 that in most of the cases their predictive performances are comparable), they are also much more expensive with a cubic scaling w.r.t the number of training inputs. Furthermore, GPWL is designed specifically for neural architecture search, which features small, directed graphs with discrete node features only; on the other hand, the GRABNEL surrogate covers a much wider scope of applications.

When we select a large H or if there are many training inputs and/or input graph(s) have a large number of nodes/edges, there will likely be many unique WL features and the resulting feature matrix will be very high-dimensional, which would lead to high-variance regression coefficients $\boldsymbol{\alpha}$ being estimated if $n_{\mathcal{G}'}$ (number

of graphs to train the surrogate of the attack agent) is comparatively few. To attain a good predictive performance in such a case, we employ Bayesian regression surrogate with the Automatic Relevance Determination (ARD) prior to learn the mapping $\mathbf{\Phi} \rightarrow \mathcal{L}_{\text{attack}}(f_\theta(\mathcal{G}'), y)$, which regularises weights and encourages sparsity in $\boldsymbol{\alpha}$ [142]:

$$\mathcal{L}_{\text{attack}}|\mathbf{\Phi}, \boldsymbol{\alpha}, \sigma_n^2 \sim \mathcal{N}(\boldsymbol{\alpha}^\top \mathbf{\Phi}, \sigma_n^2 \mathbf{I}), \tag{6.5}$$

$$\boldsymbol{\alpha}|\boldsymbol{\lambda} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Lambda}),\ \text{diag}(\mathbf{\Lambda}) = \boldsymbol{\lambda}^{-1} = \{\lambda_1^{-1}, ..., \lambda_D^{-1}\}, \tag{6.6}$$

$$\lambda_i \sim \text{Gamma}(k, \theta)\ \forall i \in [1, D], \tag{6.7}$$

where $\mathbf{\Lambda}$ is a diagonal covariance matrix. To estimate $\boldsymbol{\alpha}$ and noise variance σ_n^2, we optimise the model marginal log-likelihood. Overall, the WL routines scales as $\mathcal{O}(Hm)$ and Bayesian linear regression has a linear runtime scaling w.r.t. the number of queries; these ensure the surrogate is scalable to both larger graphs and/or a large number of graphs, both of which are commonly encountered in graph classification (See App C.4.6 for a detailed empirical runtime analysis).

Sequential perturbation selection In the default structural perturbation setting, given an attack budget of Δ (i.e. we are allowed to flip up to Δ edges from $\mathcal{G}$), finding exactly the set of perturbations $\delta\mathbf{A}$ that leads to the largest increase in $\mathcal{L}_{\text{attack}}$ entails an combinatorial optimisation over $\binom{n^2}{\Delta}$ candidates. This is a huge search space that is difficult for the surrogate to learn meaningful patterns in a sample-efficient way even for modestly-sized graphs. To tackle this challenge, we adopt the strategy illustrated in Fig. 6.2: given the query budget B (i.e. the total number of times we are allowed to query f_θ for a given $\mathcal{G}$), we assume $B \geq \Delta$ and amortise B into Δ stages and focus on selecting *one* edge perturbation at each stage. While this strategy is greedy in the sense that it always commits the perturbation leading to the largest increase in loss at each stage, it is worth noting that we do *not* treat the previously modified edges differently, and the agent can, and does occasionally as we observe empirically, "correct" previous modifications by flipping edges back: this is possible as the effect of edge selection is permutation invariant. Another benefit of this strategy is that it can potentially make full use of the entire attack budget Δ *while* remaining parsimonious w.r.t. the amount of perturbation introduced, as it only progresses to the next stage and modifies the $\mathcal{G}$ further when it fails to find a successful adversarial example in the current stage.

Optimisation of acquisition function At each BO iteration, acquisition function $\alpha(\cdot)$ is optimised to select the next point(s) to query the victim model f_θ. However, commonly used gradient-based optimisers cannot be used on the discrete

graph search space; a naïve strategy would be to randomly generate many perturbed graphs, evaluate α on all of them, and choose the maximiser(s) to query f_θ next. While potentially effective on modestly-sized $\mathcal{G}$ especially with our sequential selection strategy, this strategy nevertheless discards any known information about the search space.

Inspired by recent advances in BO in non-continuous domains [35, 136], we optimise α via an adapted version of the Genetic algorithm (GA) in [37], which is well-suited for our purpose but is not particularly sample efficient since many evolution cycles could be required for convergence. However, the latter is not a serious issue here as we only use GA for acquisition optimisation where we only query the surrogate instead of the victim model, a subroutine of BO that does not require sample efficiency. We outline its ingredients below:

- *Initialisation:* While GA typically starts with random sampling in the search space to fill the initial population, in our case we are not totally ignorant about the search space as we could have already queried and observed f_θ with a few different perturbed graphs $\mathcal{G}'$. A smoothness assumption on the search space would be that if a $\mathcal{G}'$ with an edge (u, v) flipped from G led to a large $\mathcal{L}_{\text{attack}}$, then another $\mathcal{G}'$ with $(u, s), s \notin \{u, v\}$ flipped is more likely to do so too. To reflect this, we fill the initial population by *mutating* the top-k queried $\mathcal{G}'$s leading to the largest $\mathcal{L}_{\text{attack}}$ seen so far in the current stage, where for $\mathcal{G}'$ with (u, v) flipped from the base graph we 1) randomly choose an end node (u or v) and 2) change that node to another node in the graph except u or v such that the perturbed edges in all children shares one common end node with the parent.

- *Evolution*: After the initial population is built, we follow the standard evolution routine by evaluating the acquisition function value for each member as its *fitness*, selecting the top-k performing members as the breeding population and repeating the mutation procedure in initialisation for a fixed number of rounds. At termination, we simply query f_θ with the graph(s) seen so far (i.e. computing the loss in Fig. 6.2) with the largest acquisition function value(s) seen during GA.

6.3 Related Works

Adversarial attack on graph-based models There has been an increasing attention in the study of adversarial attacks in the context of GNNs [126, 74]. One of the earliest models, Nettack, attacks a Graph Convolution Network (GCN) node classifier by optimising the attack loss of a surrogate model using a greedy algo-

rithm [164]. Using a simple heuristic, DICE attacks node classifiers by adding edges between nodes of different classes and deleting edges connecting nodes of the same class [140]. However, they cannot be straightforwardly transferred graph classification: for Nettack, unlike node classification tasks, we have no access to training input graphs or labels for the victim model during test time to train a similar surrogate in graph classification; for DICE (and also more recent works like [138]), node labels do not exist in graph classification (we only have a single label for the entire graph). We nonetheless acknowledge the other contributions in these works, such as the introduction of constraints to improve imperceptibility, in our experiments in Sec 6.4.

First methods that do extend to graph classification include [37, 91]: [37] propose a number of techniques, including RL-S2V, which uses reinforcement learning to attack both node and graph classifiers in a black-box manner, and the GA-based attack, which we adapt into our BO acquisition optimisation. However, [37] primarily focus on the S2V victim model, do not emphasise on sample efficiency, and to train a policy that attacks in an one-shot manner on the test graphs, RL-S2V has to query repeatedly on a separate validation set. We empirically compare against it in App. C.4.2. Another related work is ReWatt [91], which similarly uses reinforcement learning but through rewiring. Compared to both these methods, GRABNEL does not require an additional validation set and is much more query efficient. Other black-box methods without surrogate models have also been proposed that could be *potentially* be applied to graph classification: [90] exploit common GNN structural bias to attack node features, while [27] relate graph embedding to graph signal processing and construct tailored attack objectives in different GNNs. In comparison to these works that exploit the characteristics of existing architectures to varying degrees, we argue that the optimisation-based method proposed in the our work is more flexible and agnostic to architecture choices, and should be more generalisable to new architectures. Nonetheless, in cases where some architectural information is available, we believe there could be *combinable* benefits: for example, the importance scores proposed in [90] could be used as *sampling weights* as priors to bias GRABNEL towards selecting more vulnerable nodes. We defer detailed investigations of such possibilities to a future work. Finally, there have also been various previous works that focus on a different setup than ours: A white-box optimisation strategy (alternating direction method of multipliers) is proposed in [74]. [159, 149, 14] propose back-door attacks that involve poisoning of the training data before training and/or the test data at inference. [129] attack hierarchical graph pooling networks, but similar to [164] the method requires access to training input/targets. Ultimately, a number of factors, including but not limited to 1) existence/strictness of the query budget, 2) strictness of the perturbation budget, 3) attacker capabilities and 4) sizes

of the graphs, would decide which algorithm/setup is more appropriate and should be adopted in a problem-specific way. Nonetheless, we argue that our setup is both challenging and highly significant as it resembles the capabilities a real-life attacker might have (no access to training data; no access to model parameter/gradients and limited query/perturbation budgets).

Adversarial attacks using bo BO as a means to find adversarial examples in the black-box evasion setting has been successfully proposed for classification models on tabular [127] and image data [105, 161, 116, 97]. However, we address the problem for graph classification models, which work on structurally and topologically fundamentally different inputs. This implies several nontrivial challenges that require our method to go beyond the vanilla usage of BO: for example, the inputs cannot be readily represented as vectors like for tabular or image data and the perturbations that we consider for such inputs are not defined on a continuous, but on a discrete domain.

6.4 Experiments

We validate the performance of the proposed method in a wide range of graph classification tasks with varying graph properties, including but not limited to the typical TU datasets considered in previous works [37, 91]. As a demonstration of the versatility of the proposed method, instead of considering a single mode of attack which is often impossible in real-life, we also select the attack mode specific to each task. All additional details, including the statistics of the datasets used and implementation details of the victim models and attack methods, are presented in App. C.3.

We first conduct experiments on four common TU datasets [95], namely (in ascending order of average graph sizes in the dataset) IMDB-M, PROTEINS, COLLAB and REDDIT-MULTI-5K. In all cases, unless specified otherwise, we define the attack budget Δ in terms of the maximum *structural perturbation ratio* r defined in [30] where $\Delta \leq rn^2$. We similarly link the maximum numbers of queries B allowed for individual graphs to their sizes as $B = 40\Delta$, thereby giving larger graphs and thus potentially more difficult instances higher attack[3] and query budgets similar to the conventional image adversarial attack literature [105]. In this work, unless otherwise specified we set $r = 0.03$ for all experiments, and for comparison we consider a number of baselines, including random search, GA introduced in [37][4].

[3]Due to computational constraints, we cap the maximum number of queries to be 2×10^4 on each graph.

[4]The original implementation of RL-S2V, the primary algorithm in [37], primarily focus on a S2V-based victim model [38]. We compare GRABNEL against it in the same dataset considered in [37] in App. C.4.2.

On some task/victim model combinations, we also consider an additional simple gradient-based method which greedily adds or delete edges based on the magnitude computed input gradient similar to the gradient based method described in [37] (note that this method is *white-box* as access to parameter weights and gradients is required), which is also similar in spirit to methods like Nettack [164]. To verify whether the proposed attack method can be used for a variety of classifier architectures we also consider various victim models: we first use GCN [81] and Graph Isomorphism Network (GIN) [151], which are most commonly used in related works [126]. Considering the strong performance of hierarchical models in graph classification [52, 154], we also conduct some experiments on the Graph-U-Net [52] as a representative of such architectures. We show the classification performance of both victim models before and after attacks using various methods in Table 6.1, and we show the Attack Success Rate (ASR) against the (normalised) number of queries in Fig. 6.3. It is worth noting that in consistency with the image attack literature, we launch and consider attacks on the *graphs that were originally classified correctly*, and statistics, such as the ASR, are also computed on that basis. We report additional statistics, such as the evolution of the attack losses as a function of number of queries of selected individual data points in App. C.4.3.

The results generally show that the attack method is effective against both GCN and GIN models with GRABNEL typically leading to the largest degradation in victim predictions in all tasks, often performing on par or better than *Gradient-based*, a white-box method. It is worth noting that although *Gradient-based* often performs strongly, there is no guarantee that it always does so: first, for general edge flipping problems, *Gradient-based* computes gradients w.r.t. all possible edges (including those that do not currently exist) and an accurate estimation of such high dimensional gradients can be highly difficult. Second, gradients only capture local information and they are not necessarily accurate when used to extrapolate function value beyond that neighbourhood. However, relying on gradients to select edge perturbations constitutes such an extrapolation, as edge addition/deletion is binary and discrete. Lastly, on the tasks with larger graphs (e.g. COLLAB on GCN and GIN), due to the huge search spaces, we find neither random search nor GA could flip predictions effectively except for some "easy" samples already lying close to the decision boundary; GRABNEL nonetheless performs well thanks to the effective constraint of the search space from the sequential selection of edge perturbation, which is typically more significant on the larger graphs.

We report the results on the Graph U-net victim model in Table 6.2: as expected, Graph U-net performs better in terms of clean classification accuracy compared to the GCN and GIN models considered above, and it also seem more robust to all types adversarial attacks on the PROTEINS dataset. Nonetheless, in terms of rel-

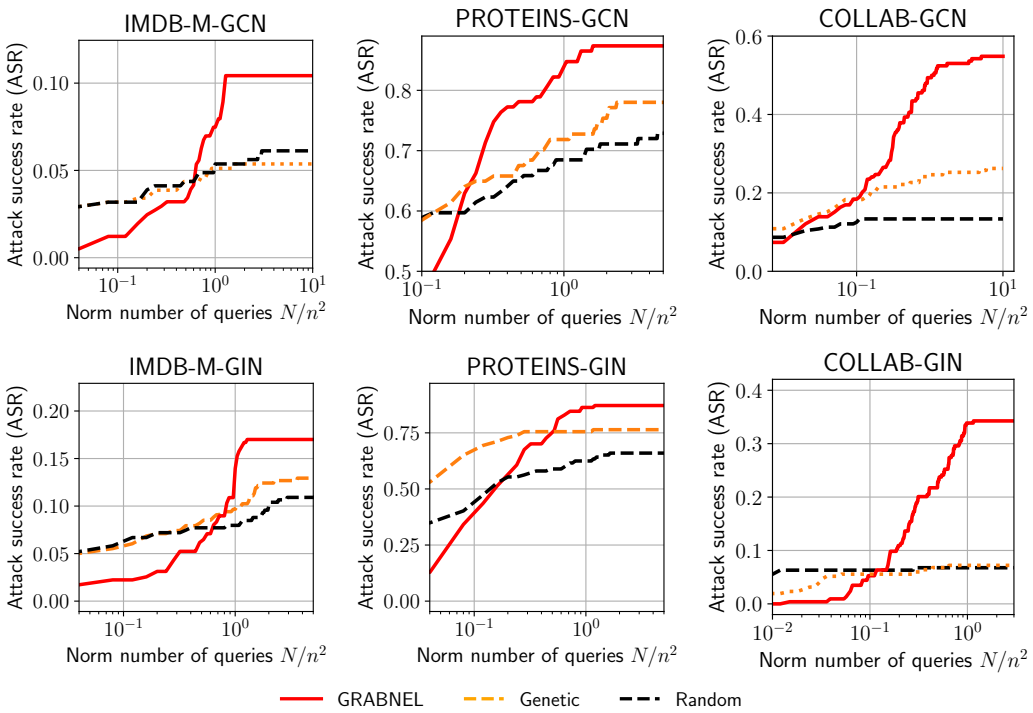


Figure 6.3: ASR against the number of queries to GCN and GIN (normalised by the square of the number of nodes of each graph) of TU datasets. Note the x-axis is on log-scale. Lines and shades denote mean ±1 sd across 3 random initialisations. `GRABNEL` outperforms other attack methods considerably. `Random` and `Genetic` appear to converge faster initially as they always use exploit full perturbation budget allocated, while `GRABNEL` is parsimonious and only attempts a higher perturbation budget when attacks perturbing fewer edges fail. It is also possible to derive a variant of random search with such parsimony, and the readers are referred to the detailed ablation studies in App. C.4.5.

ative performance margin, GRABNEL still outperforms both baselines considerably, demonstrating the flexibility and capability for it to conduct effective attacks even on the more complicated and realistic victim models.

As discussed, in real life, adversarial agents might encounter additional constraints other than the number of queries to the victim model or the amount of perturbation introduced. To demonstrate that our framework can handle such constraints, we further carry out attacks on victim models using identical protocols as above but with a variety of additional constraints considered in several previous works. Specifically, the scenarios considered, in the ascending order of restrictiveness, are:

Table 6.2: Test accuracy of Graph U-net [52] on IMDB-M and PROTEINS before and after attack.

	IMDB-M	PROTEINS
Clean	55.33	79.46
Random	45.33	75.00
Genetic [37]	44.00	75.00
GRABNEL (ours)	**41.33**	**58.80**

- `Base`: The base scenario is identical to the setup in Table 6.1 and Fig 6.3;

Table 6.1: Test accuracy of GCN and GIN victim models on the TU datasets before (*clean*) and after various attack methods. Results shown in mean ± 1 standard deviation across 3 trials. The results for REDDIT-MULTI-5K are shown in App. C.4.4.

	GCN [81]			GIN[151]		
	IMDB-M	PROTEINS	COLLAB	IMDB-M	PROTEINS	COLLAB
Clean	$50.53_{\pm 1.4}$	$71.73_{\pm 2.6}$	$79.73_{\pm 2.1}$	$48.85_{\pm 0.4}$	$70.53_{\pm 2.3}$	$80.80_{\pm 0.9}$
Random	$47.43_{\pm 1.2}$	$19.46_{\pm 1.7}$	$67.00_{\pm 3.7}$	$40.44_{\pm 2.5}$	$23.21_{\pm 14}$	$73.01_{\pm 5.0}$
Genetic [37]	$47.82_{\pm 1.5}$	$14.88_{\pm 1.7}$	$58.61_{\pm 7.9}$	$39.68_{\pm 3.1}$	$15.47_{\pm 10}$	$72.34_{\pm 2.5}$
Gradient-based†	$\mathbf{39.31_{\pm 2.2}}$	$50.60_{\pm 4.5}$	$36.67_{\pm 1.2}$	$\mathbf{37.56_{\pm 2.2}}$	$11.90_{\pm 4.4}$	$\mathbf{54.00_{\pm 2.9}}$
GRABNEL (ours)	$45.23_{\pm 0.2}$	$\mathbf{10.82_{\pm 2.5}}$	$\mathbf{35.38_{\pm 9.3}}$	$38.22_{\pm 3.9}$	$\mathbf{10.72_{\pm 6.3}}$	$57.33_{\pm 4.7}$

†: White-box method

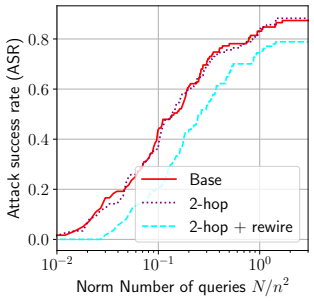


Figure 6.4: ASR vs normalised # queries with constraints on a GCN model trained on PROTEINS.

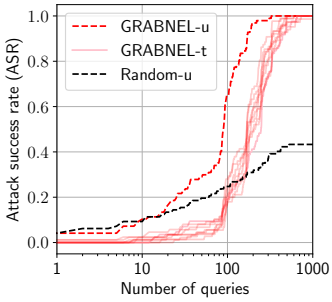


Figure 6.5: ASR vs # queries on a ChebyGIN trained on MNIST-75sp on targeted and untargeted attack setups.

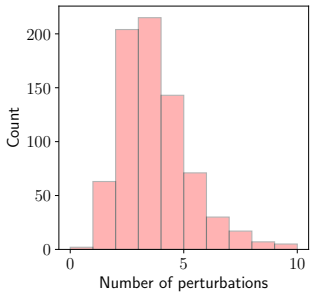


Figure 6.6: Histogram of #edges swapped in successfully attacked MNIST-75sp instances by GRABNEL-t.

- `2-hop`: Edge addition between nodes (u, v) is only permitted if v is within 2-hop distance of u;

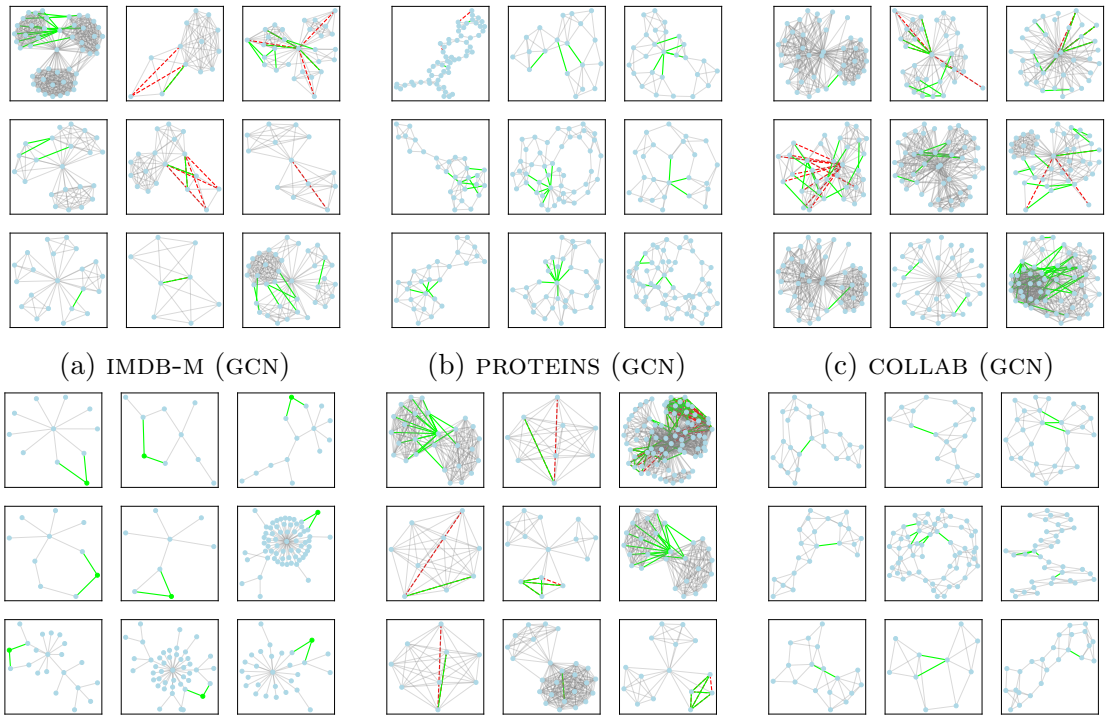
- `2-hop+rewire` [91]: Instead of flipping edges, the adversarial agent is only allowed to rewire from nodes (u, v) (where an edge exists) to (u, w) (where no edge currently exists). Node w must be within 2-hop distance of u;

We test on the PROTEINS dataset, and show the results in Fig. 6.4: interestingly, the imposition of the 2-hop constraint itself leads to no worsening of performance – in fact, as we elaborate in Sec. 6.5, we find the phenomenon of adversarial edges remaining relatively clustered within a relatively small neighbourhood is a general pattern in many tasks. This implies that the 2-hop condition, which constrains the spatial relations of the adversarial edges, might already hold even without explicit specification, thereby explaining the marginal difference between the base and the 2-hop constrained cases in Fig. 6.4. While the additional rewiring constraint leads to (slightly) lower attack success rates, the performance of GRABNEL remains relatively robust in all scenarios considered.

Image Classification Beyond the typical "edge flipping" setup on which existing research has been mainly focused, we now consider a different setup involving attacks on the MNIST-75sp dataset [83, 82] with weighted graphs with continuous attributes – note that . The dataset is generated by first partitioning MNIST image into around 75 superpixels with SLIC [1, 45] as the graph nodes (with average superpixel intensity as node attributes). The pairwise distances between the superpixels form the edge weights. We use the pre-trained ChebyGIN with attention model released by the original authors [82] (with an average validation classification accuracy of around 95%) as the victim model. Given that the edge values are no longer binary, simply flipping the edges (equivalent to setting edge weights to 0 and 1) is no longer appropriate. To generalise the sparse perturbation setup and inspired by edge rewiring studied by previous literature, we instead adopt an attack mode via *swapping edges*: each perturbation can be defined by 3 end nodes (u, v, s) where edge weights $w(u, v)$ is swapped with edge weight $w(u, s)$. We show the results in Fig. 6.5: `GRABNEL-u` and `Random-u` denotes the GRABNEL and random search under the *untargeted* attack, respectively, whereas `GRABNEL-t` denotes GRABNEL under the *targeted attack* with each line denoting 1 of the 9 possible target classes in MNIST. We find that GRABNEL is surprisingly effective in attacking this victim model, almost completely degrading the victim (Fig. 6.5) with very few swapping operations (Fig. 6.6) even in the more challenging targeted setup. This seems to suggest that, at least for the data considered, the victim model is very brittle towards carefully crafted edge swapping, with its predictive power seemingly hinged upon a very small number of key edges. We believe a thorough analysis of this phenomenon is of an independent interest, which we defer to a future work.

Fake news detection As a final experiment, we consider a real-life task of attacking a GCN-based fake news detector trained on a labelled dataset in [135]. Each discussion cascade (i.e. a chain of tweets, replies and retweets) is represented as an undirected graph, where each node represents a Twitter account (with node features being the key properties of the account such as age and number of followers/followees; see App. C.3 for details) and each edge represents a reply/retweet. As a reflection of what a real-life adversary may and may not do, we note that modifying the connections or properties of the existing nodes, which correspond to modifying existing accounts and tweets, is considered impractical and prohibited.

Instead, we consider a *node injection* attack mode (i.e. creating new malicious nodes and connect them to existing ones): injecting nodes is equivalent to creating new Twitter accounts and connecting them to the rest of the graph is equivalent to retweeting/replying existing accounts. We limit the maximum number of injected nodes to be $0.05N$ and the maximum number of new edges that may be created per



(a) IMDB-M (GCN) (b) PROTEINS (GCN) (c) COLLAB (GCN)

(d) Twitter fake news (e) IMDB-M (GIN) (f) PROTEINS (GIN)

Figure 6.7: Adversarial examples found by GRABNEL. Red edges denote deleted edges and green edges indicate added ones. In Twitter fake news detection task, green nodes/edges denote the injected nodes and their connections to the existing graphs. Refer to App C.4.3 for more examples.

each new node is set to the average number of edges an existing node has – in this context, this limits the number of re-tweets and replies the new accounts may have to avoid easy detection. For the injected node, we initialise its node features in a way that reflects the characteristics of a new Twitter user (we outline the detailed way to do so in App. C.3). We show the result in Fig. 6.8, where GRABNEL is capable of reducing the effectiveness of a GCN-based fake news classifier by a third. In this case *Random* also performs reasonably well, as the discussion cascade is typically small, allowing any adversarial examples to be exhaustively found eventually.

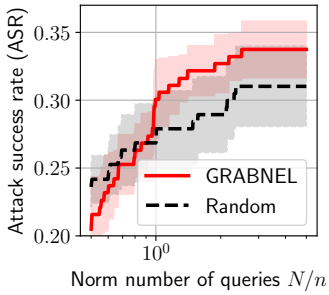


Figure 6.8: ASR vs. #queries (normalised by the number of nodes, since the attack involves node injection) on the Twitter dataset.

Ablation Studies GRABNEL benefits from a number of design choices and it is important to understand the relative contribution of each to the performance. We find that in *some* tasks GRABNEL without surrogate (i.e. random search with sequential perturbation selection. We term this variant *SequentialRandom*) is a very strong baseline in terms of final ASR, although the full GRABNEL is much better in terms of overall performance, sample efficiency and the ability to produce successful examples with few perturbations. The readers are referred to our ablation studies in App. C.4.5.

Runtime Analysis Given the setup we consider (sample- efficient black-box attack with minimal amount of perturbation), the cost of the algorithm should not only be considered from the viewpoint of the computational runtime of the attack algorithm itself alone, and this is a primary reason why we use the (normalised) number of queries as the main cost criterion. Nonetheless, a runtime analysis is still informative which we provide in App. C.4.6. We find that GRABNEL maintains a reasonable overhead even on, e.g., graphs with $\sim 10^3$ nodes/edges that are larger than most graphs in typical graph classification tasks.

6.5 Attack Analysis

Having established the effectiveness of our method, in this section we provide a qualitative analysis on the common interpretable patterns behind the adversarial samples found, which provides further insights into the robustness of graph classification models against structural attacks. We believe such analysis is especially valuable, as it may facilitate the development of even more effective attack methods, and may provide insights that could be useful for identification of real-life vulnerabilities for more effective defence. We show examples of the adversarial samples in Fig. 6.7 (and Fig. C.5 in App. C.4.3). We summarise some key findings below.

- *Adversarial edges tend to cluster closely together*: We find the distribution of the adversarial edges (either removal or addition) in a graph to be highly uneven, with many adversarial edges often sharing common end-nodes or having small spatial distance to each other. This is empirically consistent with recent theoretical findings on the stability of spectral graph filters in [77]. From an attacker point of view, this may provide a "prior" on the attack to constrain the search space, as the regions around existing perturbations should be exploited more; we leave a practical investigation of the possibility of leveraging this to enhance attack performance to a future work.

- *Adversarial edges often attempt to destroy or modify community structures*: for

example, the original graphs in the IMDB-M dataset can be seen to have community structure, a graph-level topological property that is distinct from the existing works analysing attack patterns on node-level tasks [145, 167]. When the GCN model is attacked, the attack tends to flip the edges *between* the communities, and thereby destroying the structure by either merging communities or deleting edges within a cluster. On the other hand, the GIN examples tend to strengthen the community structures by adding edges within clusters and deleting edges between them. With similar observations also present in, for example, PROTEINS dataset, this may suggest that the models may be fragile to modification of the community structure.

- *Beware the low-degree nodes!* While low-degree nodes are important in terms of degree centrality, we find some victim models are vulnerable to manipulations on such nodes. Most prominently, in the Twitter fake news example, the malicious nodes almost never connect directly to the central node (original tweet) but instead to a peripheral node. This finding corroborates the theoretical argument in [77] which shows that spectral graph filters are more robust towards edge flipping involving *high-degree* nodes than otherwise, and is also consistent with observations on node-level tasks [167] with the explanation being lower-degree nodes having larger influence in the neighbourhood aggregation in GCN. Nonetheless, we note that changes in a higher-degree node are likely to cascade to more nodes in the graph than low degree nodes, and since graph classifiers aggregate across all nodes in the readout layer the indirect change of node representations also matter. Therefore, we argue that this phenomenon in graph classification is still non-trivial.

6.6 Conclusion

Summary This work proposes a novel and flexible black-box method to attack graph classifiers using Bayesian optimisation. We demonstrate the effectiveness and query efficiency of the method empirically. Unlike many existing works, we qualitatively analyse the adversarial examples generated. We believe such analysis is important to the understanding of adversarial robustness of graph-based learning models. Finally, we would like to point out that a potential negative social impact of our work is that bad actors might use our method to attack real-world systems such as a fake new detection system on social media platforms. Nevertheless, we believe that the experiment in our paper only serves as a proof-of-concept and the benefit of raising awareness of vulnerabilities of graph classification systems largely outweighs the risk.

Limitations and Future Work Firstly, the current work only considers topological attack, although the surrogate used is also compatible with attack on node/edge features or hybrid attacks. Secondly, while we have evaluated several mainstream victim models, it would also be interesting to explore defences against adversarial attacks and to test GRABNEL in robust GNN setups such as those with advanced graph augmentations [156], randomised smoothing [159, 53] and adversarial detection [28]. Lastly, the current work is specific to graph classification; we believe it is possible to adapt it to attack other graph tasks by suitably modifying the loss function. We leave these for future works.

The vulnerability of graph neural networks to adversarial attack limits their potential for real-world applications, especially in an environment that contains bad actors such as in social networks. Although techniques exists that increase the robustness of graph neural networks, there is no guarantee techniques will stand up to new and improved forms of adversarial attack. In the following chapter we consider the framework of certified robustness via randomised smoothing in the context of graph classifiers. In particular, we show how one can build a classifier which given some test point, one can prove with high probability that it is some distance to the decision boundary and is therefore not vulnerable to adversarial attack (irrespective of the form of the attack).

Chapter 7

Structure-aware robustness certificates for graph classification

In this chapter we consider a third form of robustness. We consider how to generate certificates around an input using the randomised smoothing framework. In this framework the change in output is bounded, however, there are three important distinctions. The first is that the change is with respect to classification labels rather than embeddings. This is usually of more interest in practice. The second is that the bound depends on the input graph. Intuitively this allows one to give much tighter bounds, as a global bound may be loose due to a small number of pathological examples. Local Lipschitz bounds could be used to generate certificates which are in the spirit of both works. Finally, certificates generated using the randomised smoothing framework are probabilistic. This is a sacrifice we make for a framework that applies to any graph classifier model, regardless of the architecture used.

Certifying the robustness of a graph-based machine learning model poses a critical challenge for safety. Current robustness certificates for graph classifiers guarantee output invariance with respect to the total number of node pair flips (edge addition or edge deletion), which amounts to an l_0 ball centred on the adjacency matrix. Although theoretically attractive, this type of isotropic structural noise can be too restrictive in practical scenarios where some node pairs are more critical than others in determining the classifier's output. The certificate, in this case, gives a pessimistic depiction of the robustness of the graph model. To tackle this issue, we develop a randomised smoothing method based on adding an anisotropic noise distribution to the input graph structure. We show that our process generates structural-aware certificates for our classifiers, whereby the magnitude of robustness certificates can vary across different pre-defined structures of the graph. We demonstrate the benefits of these certificates in both synthetic and real-world experiments.

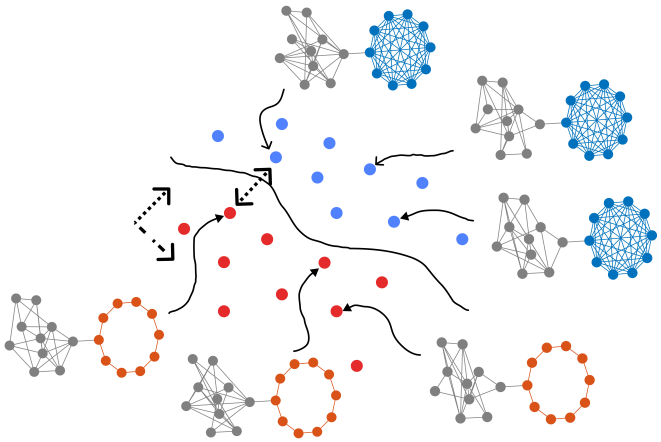


Figure 7.1: An example of graph classification where the class label is determined by the blue or red part of the input graph, but not the grey part. In this case, perturbations to the former will more likely to affect classification outcome than the latter. In other words, the classification decision boundary is sensitive to some perturbations to the input graph structure (which move the point towards the boundary), but robust to others (which keep the point away from the boundary).

7.1 Introduction

Graph-based machine learning models have made considerable strides in the last couple of years, with applications ranging from NLP [146], combinatorial optimization [44] and protein function prediction [58]. As these tools become more common, studying their vulnerability to potential adversarial examples turns paramount for safety purposes.

Robustness certification is an active field of research whose goal is to develop certificates guaranteeing invariance of the model prediction with respect to some input perturbations. Diverse methods have been used to achieve this goal, from interval bound propagation [60], convex relaxation [102], Lipschitz bounds computation [67] or randomised smoothing [137]. Given a data point $\mathbf{x}$ and a set of perturbed inputs $\mathcal{B}(\mathbf{x})$, a robustness certificate verifies that a model's prediction $f(\mathbf{x})$ remains unchanged for all other inputs in the perturbation set. That is, for all $\mathbf{x}' \in \mathcal{B}(\mathbf{x})$ it holds that $f(\mathbf{x}) = f(\mathbf{x}')$. Often the set of perturbed inputs $\mathcal{B}(\mathbf{x})$ is parameterised, for example by a closed-ball $\mathcal{B}_r(\mathbf{x}) = \{\mathbf{x}' : d(\mathbf{x}, \mathbf{x}') \leq r\}$ under some distance function d and radius r. In this case we are interested in knowing the largest r that we can certify for, where r is called the certified radius.

In the context of robustness certification of graph classifiers against structural perturbation, a common choice of perturbation set is the set of all graphs reachable from an input graph $\mathbf{x}$ by up to r node pair flips (edge additions and deletions)[1].

[1]We use the terminology of node pair flip instead of edge flip to emphasise that we are considering the addition of edges that do not exist in the original graph as well as the deletion of existing edges.

This corresponds to a closed ball on the upper triangle entries of the adjacency matrix where the distance is induced by the ℓ_1 norm and the bottom triangle entries are determined by the constraint that the adjacency matrix is symmetric (assuming for simplicity the graph is unweighted and undirected). In some cases, however, different node pairs of the graph can be more predictive of the ground truth label than others. A real-world example is classification of protein structures, where the edges that constitute key substructure (e.g., a ring) are more critical in determining the class label than the rest. A synthetic example is further presented in Fig. 7.1. In such situations, certifying according to a total number of edge additions or deletions might gives a pessimistic certified radius, because the set of perturbed inputs may include perturbations which consist of flipping many critical node pairs (in terms of determining the graph label).

In this work we address this problem by defining disjoint regions of node pairs and proposing robustness certificates that verify that the prediction of the classifier will not change for a potentially different number of node pair flips for each region. Such disjoint regions may be either obtained via domain knowledge (as in the protein example mentioned above) or assigned based on the level of importance of node pairs to the classification task. Our approach then relies on randomised smoothing, which is a powerful framework to produce robustness certificates which hold with high probability. Given some noise distribution over the input, randomised smoothing transforms a base model f into a smoothed model g for which we can provide probabilistic robustness guarantees.

Existing randomised smoothing approaches for certifying graph classification mostly consider an isotropic noise distribution that flips each node pair with a fixed probability [70, 53, 137]. The certificate in this case corresponds to the total number of node pair flips. Instead, we propose using an anisotropic noise distribution based on the predefined regions whereby the probability of flipping a node pair depends on to which region (if any) the node pair belongs. We show that smoothed classifiers constructed using this anisotropic noise distribution naturally lead to structure-aware robustness certificates whereby different number of node flips are certified for each of the regions. We demonstrate the benefits of our approach on both synthetic and real-world experiments. To the best of our knowledge, our method is one of the first of its kind that allows for flexible and structure-aware graph certification in the input graph domain.

7.2 Preliminaries

Let $\mathcal{X}$ be the data space and $f : \mathcal{X} \to \mathcal{Y}$ be a classifier which maps each point $\mathbf{x} \in \mathcal{X}$ to a label $y \in \mathcal{Y} = \{0, ..., C-1\}$. Let $\phi : \mathcal{X} \to \mathcal{P}(\mathcal{X})$ be a noise distribution

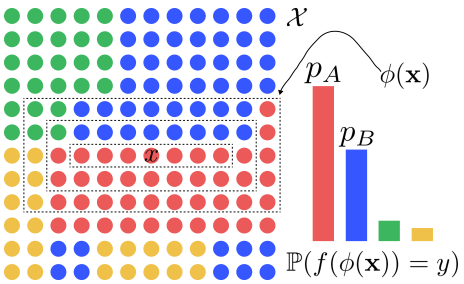


Figure 7.2: Illustration of a smoothed classifier: at every point $\mathbf{x}$ a neighbourhood vote is performed according to a distribution $\phi(\mathbf{x})$ centred on $\mathbf{x}$.

over our data, that is, $\phi(\mathbf{x})$ returns a distribution over $\mathcal{X}$ for every point $\mathbf{x} \in \mathcal{X}$. We write $f(\phi(\mathbf{x}))$ to denote the random variable $\mathbb{P}_{\mathbf{z}\sim\phi(\mathbf{x})}(f(\mathbf{z}))$. This represents the distribution of outputs of the base classifier given the randomisation scheme applied to the input. We define g to be the smoothed classifier of our base classifier f as

$$g(\mathbf{x}) = \arg\max_{y\in\mathcal{Y}} \mathbb{P}(f(\phi(\mathbf{x})) = y). \tag{7.1}$$

The smoothed classifier can be interpreted as a neighbourhood vote, where the output is the mode of the output of the classifier when inputs are sampled from the distribution $\phi(\mathbf{x})$. An illustration of a smoothed classifier is given in Figure 7.2.

7.2.1 Certifying a smoothed classifier

We can construct a lower and upper bound $\underline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p, y)$ and $\overline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p, y)$ on $\mathbb{P}(f(\phi(\tilde{\mathbf{x}})) = y)$, which is the probability of class y under our classifier f smoothed by $\phi(\mathbf{x})$ and evaluated on an arbitrary point $\tilde{\mathbf{x}} \in \mathcal{X}$:

$$\underline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p, y) = \min_{\substack{h\in\mathcal{H}: \\ \mathbb{P}(h(\phi(\mathbf{x}))=y)=p}} \mathbb{P}(h(\phi(\tilde{\mathbf{x}})) = y) \tag{7.2}$$

$$\overline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p, y) = \max_{\substack{h\in\mathcal{H}: \\ \mathbb{P}(h(\phi(\mathbf{x}))=y)=p}} \mathbb{P}(h(\phi(\tilde{\mathbf{x}})) = y) \tag{7.3}$$

In this definition, $\mathcal{H}$ is the class the set of measurable classifiers with respect to ϕ. Because the optimisation constraints include the base classifier $f \in \mathcal{H}$ it follows that

$$\underline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p, y) \leq \mathbb{P}(f(\phi(\tilde{\mathbf{x}})) = y) \leq \overline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p, y). \tag{7.4}$$

We define a perturbation set $\mathcal{B}_r(\mathbf{x})$ as a family of sets $\mathcal{B}_r(\mathbf{x}) \subseteq \mathcal{X}$ parameterised by some $r \geq 0$ such that $\mathcal{B}_r(\mathbf{x}) \subseteq \mathcal{B}_{r'}(\mathbf{x})$ if and only if $r \leq r'$. This includes open or closed balls with respect to a metric over $\mathcal{X}$. We say that the smoothed classifier g is certified at x in some perturbation set $\mathcal{B}_r(\mathbf{x})$ if the output of g is the same for all neighbouring points $\tilde{\mathbf{x}} \in \mathcal{B}_r(\mathbf{x})$ in the perturbation set.

Following [34], we will write c_A to be the output class of $g(\mathbf{x})$ which is returned with probability p_A, and c_B to be the "runner-up" class, i.e., the class distinct from c_A with the next highest probability p_B. From the framework of [85], we define the notion of point-wise certificate around a point $\mathbf{x}$ by verifying if the following holds.

$$\min_{\tilde{\mathbf{x}} \in \mathcal{B}_r(\mathbf{x})} \Phi_{\mathbf{x},\tilde{\mathbf{x}}}(p_A, c_A) > 0, \tag{7.5}$$

where

$$\Phi_{\mathbf{x},\tilde{\mathbf{x}}}(p_A, c_A) \triangleq \underline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p_A, c_A) - \overline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p_B, c_B). \tag{7.6}$$

Equation 7.6 can be thought of as a margin, which gives the difference between a lower bound on p_A and an upper bound on p_B for some point $\tilde{\mathbf{x}}$ in the perturbation set. Equation 7.5 then indicates if this property holds for all $\tilde{\mathbf{x}}$ in the perturbation set.

7.2.2 Certified radius

We can define the certified radius to be the largest value of r so that we can certify with respect to the predefined perturbation set $\mathcal{B}_r(\mathbf{x})$. Formally this can be defined as:

$$R(\mathbf{x}) = \sup r, \text{ s.t. } \min_{\tilde{\mathbf{x}} \in \mathcal{B}_r(\mathbf{x})} \Phi_{\mathbf{x},\tilde{\mathbf{x}}}(p_A, c_A) > 0. \tag{7.7}$$

If the perturbation set is an open or closed ball, $R(\mathbf{x})$ is the radius of the largest ball we can certify over.

7.2.3 Computing the certificate

Computing a point-wise certificate

For a fixed $\mathbf{x}$ and neighbouring point $\tilde{\mathbf{x}}$, we can compute Eq. 7 following the method described by [85]. First, we partition the space $\mathcal{X} = \bigcup_i \mathcal{R}_i$ into disjoint regions of equal likelihood ratios $\mathcal{R}_k = \{\mathbf{z} \in \mathcal{X} : \frac{\mathbb{P}(\phi(\tilde{\mathbf{x}})=\mathbf{z})}{\mathbb{P}(\phi(\mathbf{x})=\mathbf{z})} = \eta_k\}$ where without loss of generality we can assume $\eta_k \in \mathbb{R}$ are in an ascending order. The quantity $\Phi_{\mathbf{x},\tilde{\mathbf{x}}}(p_A, c_A)$ can then be computed by solving the following linear programming (LP) problems [85, Lemma 2]:

$$\underline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p_A, c_A) = \min_{\mathbf{h}} \mathbf{h}^T \tilde{\mathbf{r}} \quad \text{s.t.} \quad \mathbf{h}^T \mathbf{r} = p_A, \quad 0 \leq \mathbf{h} \leq 1 \tag{7.8}$$

and similarly,

$$\overline{\rho_{\mathbf{x},\tilde{\mathbf{x}}}}(p_B, c_B) = \min_{\mathbf{h}} \mathbf{h}^T \tilde{\mathbf{r}} \quad \text{s.t.} \quad \mathbf{h}^T \mathbf{r} = p_B, \quad 0 \leq \mathbf{h} \leq 1. \tag{7.9}$$

The variables $\mathbf{h}$ correspond to optimising over the classifiers that are optimised over in Eq. 7.2 and Eq. 7.3. The vectors $\mathbf{r}$ and $\tilde{\mathbf{r}}$ are $\mathbf{r}_i = \mathbb{P}(\phi(\mathbf{x}) \in \mathcal{R}_i)$ and $\tilde{\mathbf{r}}_i = \mathbb{P}(\phi(\tilde{\mathbf{x}}) \in \mathcal{R}_i)$ respectively. This LP problem can be solved via a greedy approach [85]. Given the ratios η_k are ordered in an ascending manner, starting with $\mathbf{h} = \mathbf{0}$ we can assign $\mathbf{h}_i = 1$ for the indices $i = 1, \ldots, k-1$ such that we choose the largest k with $\mathbf{h}^T\mathbf{r} \leq p_A$, and set the value of $\mathbf{h}_k$ such that $\mathbf{h}^T\mathbf{r} = p_A$. We can solve Equation 7.9 in a similar way.

Given this efficient way to compute a certificate, there remain some quantities that must be calculated. For our certificate, we introduce these methods in Section 7.3. The first is partitioning the space $\mathcal{X}$ into disjoint unions of equal likelihood ratios. We provide a method to do this in Proposition 7.3. Next, the values η_k must be computed, or at least given in closed form, so the regions can be sorted in ascending order. Proposition 7.3 gives an analytic closed-form. Finally, a closed form for $\mathbb{P}(\phi(\mathbf{x}) = \mathbf{z})$ allows us to compute $\mathbf{r}$ and we can compute $\tilde{\mathbf{r}}$, required to solve the linear programs, by noticing that $\mathbb{P}(\phi(\tilde{\mathbf{x}}) = \mathbf{z}) = \eta_k \mathbb{P}(\phi(\mathbf{x}) = \mathbf{z})$. We compute $\mathbb{P}(\phi(\mathbf{x}) = \mathbf{z})$ in Proposition 7.3.

Computing a regional certificate

To certify over some allowed set $\mathcal{B}_r(\mathbf{x})$ we need to find the worst case $\tilde{\mathbf{x}} \in \mathcal{B}_r(\mathbf{x})$ to solve Eq. 7.5. Through particular choices of parameterising $\tilde{\mathbf{x}}$, one can find equal likelihood regions that give rise to equal likelihood ratios values that are independent of the exact value of $\tilde{\mathbf{x}}$, turning the point-wise certificate into a regional certificate, where the region is given by specific parameterisation. We give such an example in Proposition 7.3.

7.2.4 Randomised smoothing for graph classification

Although our method is applicable for any setting with discrete data, we are interested in the robustness of graph classification models to structural perturbation of undirected, unweighted graphs[2]. In this situation the input domain is the set of finite graphs $\mathcal{X} = \cup_{i=1}^{\infty} \mathcal{X}_n$ where $\mathcal{X}_n$ is the space of graphs with n nodes. We can represent a graph with n nodes as a binary vector of size $\binom{n}{2}$ where each entry indicates the presence or absence of an edge. Without loss of generality we will treat graphs as binary vectors $\mathbf{x}$ from here on in[3].

We are interested in robustness with respect to node pair flips which can represent an edge addition (change a zero to a one in $\mathbf{x}$) or edge deletion (change a one to a

[2]This work can be extended to the setting of directed graphs as well as the task of node classification.

[3]Note that due to isomorphism multiple different binary vectors can represent the same graph.

zero). This may be interpreted as adding "structural noise" to the input graph. Two candidates for the distribution of such noise have been proposed in the literature. The first one applies an independent Bernoulli distribution to every node pair. That is, for all $\phi(\mathbf{x})_i = \mathbf{x}_i \oplus \epsilon_i$ with $\epsilon_i \sim \text{Bern}(p)$ for a fixed p, where $\oplus$ is the bitwise XOR operator. We refer to this noise distribution as isotropic, as it flips each node pair with equal probability. This approach is used by [70, 137, 53]. The second approach, a sparsity-aware noise distribution [16], gives a different probability of edge flipping depending on whether the edge is present in the initial graph. If an edge exists between a node pair then it is flipped with probability p_-, whereas if a node pair does not exist between two nodes it is added with probability p_+. This distribution can be written as $P(\phi(\mathbf{x})_i \neq \mathbf{x}_i) = p_-^{\mathbf{x}_i} p_+^{1-\mathbf{x}_i}$. This latter case can be conceptually interpreted as a specific instance of the proposed framework: we consider generic partition of node pairs into one of many node pair sets and these pairs are perturbed according to their membership of the sets. This leads to the derivation of robustness certificates that are aware of the structural characteristics of the input graph with respect to the classification labels.

7.3 Randomised smoothing with anisotropic noise

Given $\mathbf{x} \in \mathcal{X}_n$, suppose we divide our input space of node pairs up into disjoint regions $\bigsqcup_{i \in I} \mathcal{C}_i$ such that each node pair belongs to exactly one region and there are a total of C regions. We define a noise distribution where independent Bernoulli distributions are applied to every node pair, and where the parameter of the Bernoulli distribution is shared within every set $\mathcal{C}_i$:

$$\phi(\mathbf{x})_k = \mathbf{x}_k \oplus \epsilon_k, \text{ where } \epsilon_k \sim \text{Bern}(p_i) \text{ and } k \in \mathcal{C}_i, \tag{7.10}$$

Let $\mathbf{R} \in \mathbb{Z}^C$ be a tuple of integers such that $0 \leq \mathbf{R}_i \leq |\mathcal{C}_i|$ and let $\mathcal{B}_{\mathbf{R}}(x) = \{\mathbf{z} \in \mathcal{X}_n : \|\mathbf{z}_{\mathcal{C}_i} - \mathbf{x}_{\mathcal{C}_i}\|_0 \leq \mathbf{R}_i\}$ be a perturbation set. Let $\tilde{\mathbf{x}} \in \mathcal{B}_{\mathbf{R}}(x)$ and $\mathcal{J} = \{i : \mathbf{x}_i \neq \tilde{\mathbf{x}}_i\}$ be indices of $\mathbf{x}$ which are perturbed to give $\tilde{\mathbf{x}}$. Furthermore, let $\mathcal{J}_i = \mathcal{J} \cap \mathcal{C}_i$ be indices where $\mathbf{x}$ is perturbed in collection $\mathcal{C}_i$. In our case a maximum radius means maximizing the radius on every regions $\mathcal{C}_i$.

First, we develop a set decomposition on which likelihood ratios can be computed. We define regions $\mathcal{R}_{\mathbf{Q}} = \{\mathbf{z} \in \mathcal{X}_n : \|\mathbf{z}_{\mathcal{J}_i} - \mathbf{x}_{\mathcal{J}_i}\|_0 = Q_i : i \in I\}$ that represent points $\mathbf{z}$ which agree with $\mathbf{x}$ by exactly Q_i bits in sub-regions $\mathcal{J}_i$. Then $\mathcal{X}_n$ can be represented by the following disjoint union

$$\bigcup_{\mathbf{0} \leq \mathbf{Q} \leq \mathbf{R}} \mathcal{R}_{\mathbf{Q}}, \tag{7.11}$$

where vector inequalities are element-wise.

Next, the likelihood ratio is fixed for elements $\mathbf{z}$ in any one region. This likelihood ratio has the following closed form. Consider a region $\mathcal{R}_{\mathbf{Q}} = \{\mathbf{z} \in \mathcal{X}_n : \|\mathbf{z}_{\mathcal{J}_i} - \mathbf{x}_{\mathcal{J}_i}\|_0 = Q_i\}$ then for all $\mathbf{z} \in \mathcal{R}_{\mathbf{Q}}$ the following holds

$$\eta^{\mathcal{R}}_{\mathbf{Q}} = \frac{\mathbb{P}(\phi(\tilde{\mathbf{x}}) = \mathbf{z})}{\mathbb{P}(\phi(\mathbf{x}) = \mathbf{z})} = \prod_{i=1}^{C} \left(\frac{1-p_i}{p_i}\right)^{R_i - 2Q_i}. \tag{7.12}$$

Finally, we can compute the likelihood of a smoothed input belonging to these regions: The probability of the output of a smoothed input $\phi(\mathbf{x})$ belonging to a region $\mathcal{R}_{\mathbf{Q}}$ is given by

$$\mathbb{P}(\phi(\mathbf{x}) \in \mathcal{R}_{\mathbf{Q}}) = \prod_{i=1}^{C} \mathrm{Bin}(R_i - Q_i | R_i, p_i), \tag{7.13}$$

where $\mathrm{Bin}(R_i - Q_i | R_i, p_i)$ is the probability mass function of the binomial distribution giving probability of $R_i - Q_i$ successes from R_i trials each with success probability p_i. Using these results we can provide robustness certificates of the smoothed classifier. Given $x \in \mathcal{X}$ and our noise distribution, we can compute the values p_A. In practice, these quantities are not available in closed form and are estimated via sampling, as in [16]. A more detailed description is given in Appendix **2**, which gives probabilistic certificates according to some confidence intervals. Without loss of generality we order the regions as $\mathcal{R}_1, \ldots \mathcal{R}_T$ (where $T = \prod_i (R_i + 1)$). The corresponding ratios $\eta^{\mathcal{R}}_{\mathbf{Q}}$ as given by Proposition 2 are ordered as $\eta_1 \leq \ldots \leq \eta_T$. From these elements, $\Phi_{\mathbf{x},\tilde{\mathbf{x}}}(p_A, c_A)$ can be computed through eq. 7.6 and the optimisation problem eq. 7.7 can be solved. In practice, the optimisation of eq. 7.7 can be solved efficiently by leveraging some symmetries displayed by $\Phi_{\mathbf{x},\tilde{\mathbf{x}}}(p_A, c_A)$ when $\tilde{\mathbf{x}}$ varies. This property is more thoroughly described in Appendix **2**.

7.4 Related work

Robustness certificate. The earliest work proposing robustness certificate for graph-based models is Zügner and Günnemann [165], where the authors consider semi-supervised node classification over graphs with binary attributes using graph neural networks. The admissible perturbations are those bounded under the ℓ_0 seminorm. Following the approach of Wong and Kolter [143], a convex relaxation is considered and a dual problem is solved via linear programming. Other early works that consider certificates under topological change include Bojchevski and Günnemann [15] who propose a certificate for a class of linear graph neural networks based on PageRank diffusion. Jin et al. [73] propose the first certificate for graph classifiers which consist of a single graph convolutional layer followed by pooling

layer and a final linear layer. A convex relaxation of the adversarial polytope based on Lagrange Duality is considered. Although the computed certificates are exact, the framework is specific to this simple graph classifier model and is expensive to compute. Recently, Jin, Yu, and Zhang [71] consider certifying graph classifiers under a different admissible perturbation measure, i.e., the Orthogonal Gromov-Wasserstein discrepency.

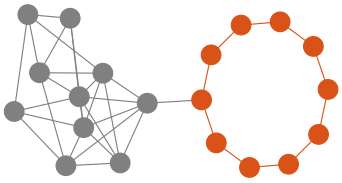
Randomised smoothing based approaches. Randomised smoothing was originally proposed by Lecuyer et al. [84] and Singla and Feizi [119] in the context of image classification. Cohen, Rosenfeld, and Kolter [34] study this approach further, giving a tight certificate for perturbations bounded by an ℓ_2 norm and improving scalability of the approach. They also outline some disadvantages of randomised smoothing, such as the need for high levels of noise or substantial number of samples to certify to a large certified radius. Another consideration of randomised smoothing is their accuracy-robustness trade off, controlled by the level of noise.

Existing randomised smoothing approaches for certifying graph based classifications models consider certificates for the number of edge flips. This is achieved by flipping an edge between each node pair independently with the same probability p. Jia et al. [70] apply this to community detection algorithms, whereas Gao, Hu, and Gong [53] and Wang et al. [137] apply the same principles to graph classification. The closest method from our work is the sparsity-aware certificate proposed in [16]. The authors note that due to the sparse nature of many real-world graphs, adding and deleting edges between node pairs with the same probability leads to many more edges being added than deleted. To account for this, the authors propose using different probabilities for adding and deleting edges. This method is different from ours as the Bernoulli probabilities change with the input graph whereas ours depend on predefined sets of node pairs. These sets in our case account for varying levels of importance of the node pairs in predicting a graph label.

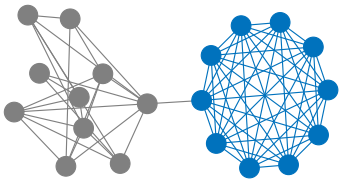
7.5 Experiments

7.5.1 Synthetic experiment

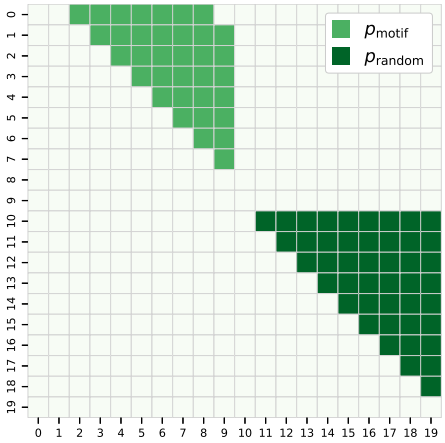
We motivate the use of anisotropic noise by considering inputs $\mathbf{x}$ that are an element of some space $\mathcal{X} = \mathcal{X}_1 \oplus \mathcal{X}_2$ where $\oplus$ is the direct sum. Consider a point that is close to the decision boundary in the $\mathcal{X}_1$ subspace but far from the decision boundary in the $\mathcal{X}_2$ subspace as illustrated in Figure 7.1. An isotropic certificate cannot certify beyond the small distance to the decision boundary in $\mathcal{X}_1$. However, by design our certificate can certify the distances of $\mathcal{X}_1$ and $\mathcal{X}_2$ jointly allowing us to certify a small distance in the $\mathcal{X}_1$ subspace but a large distance in the $\mathcal{X}_2$ subspace.



(a) Graph with negative label.



(b) Graph with positive label.



(c) Noise matrix.

Figure 7.3: Example graphs with a positive and negative label. Blue nodes and edges denote a motif part of a positive label and red nodes and edges denote a motif part of a negative label. The noise matrix show how edges are perturbed with p_{motif} being the noise parameter for node pairs in the motif part and p_{random} being the noise applied to node pairs in the random part. Notice that only the internal edges of the motif are perturbed, and the bridge edge is not perturbed. The upper triangle of the noise is shown only, in practice this is sampled and applied to the upper and lower triangle of the adjacency so the graph remains undirected.

We design a synthetic graph classification data set whereby the graphs are constructed using a motif which determines the class label (corresponding to the important subspace $\mathcal{X}_1$) connected to a randomly generated graph (corresponding to the unimportant subspace $\mathcal{X}_2$) which is independent of the class label. We can consider edges in the motif part to be in one node pair set $\mathcal{C}_{\text{motif}}$ and edges from the random part in $\mathcal{C}_{\text{random}}$. Given a model that solves this task, we would expect changes in the motif to move the input closer or further away from a decision boundary but changes in the random part of the graph to move the point parallel to the direction of the decision boundary. We should be able to certify a large number of node pairs in $\mathcal{C}_{\text{random}}$ by applying a large value of noise p_{random} without hurting the accuracy of the smoothed classifier. We cannot certify a large number of node pairs in $\mathcal{C}_{\text{motif}}$ without drop in accuracy, so we choose a small value of noise p_{motif} to retain high accuracy. The graphs we generate and the noise we use to perturb the graphs are shown in Fig. 7.3.

We generate balanced train, validation and test sets of size 1000, 1000, and 100 respectively. The test set is smaller as the randomised smoothing procedure is computationally expensive. This is because to estimate p_A a large number of random inputs need to be generated and classified using the model. We generate a binary classification problem where each graph has a motif part of $n_{\text{motif}} = 10$ nodes where a cycle determines a negative label and a complete graph determines

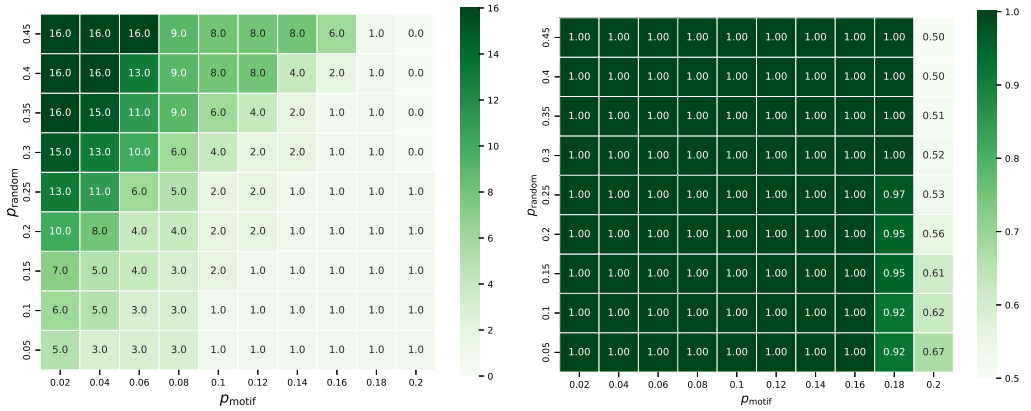
a positive label. We then generate a random part using a connected Erdős-Rényi graph [55] with $n_{\text{random}} = 10$ nodes and parameter $p = 0.5$. We join these graphs using a single edge. See Fig. 7.3a for an example of the negative class and Fig. 7.3b for an example of the positive class.

We train a SVM classifier using a node label histogram kernel [123] where the node label corresponds to the node degree. Let $c(\mathcal{G}, d)$ be a function that counts the number of nodes in a graph $\mathcal{G}$ with degree d. Then the kernel applied to graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ can be written as $\kappa(\mathcal{G}_1, \mathcal{G}_2) = \sum_{d=0}^{\infty} c(\mathcal{G}_1, d) \cdot c(\mathcal{G}_2, d)$ which is well defined for finite graphs. We use this model as we expect it to be sensitive to the motif structure that determines the label; the negative label gives a large value in the $c(\cdot, 2)$ dimension whereas the positive label gives a large value in the in the $c(\cdot, n_{\text{motif}} - 1)$ dimension. Indeed, the base classifier gets 100% accuracy on the train, validation and test data sets.

For the certification procedure, we apply noise separately for node pairs in the motif part and node pairs in the random part. We apply noise to internal edges of the motif part only (i.e. not part of the outer cycle, see Fig. 7.3). We do not apply noise to node pairs where one node lies in the motif and one does not. We also do not perturb the edge that joins the motif part and the random part. The noise matrix is shown in Fig. 7.3c. We generate $100,000$ perturbations per test-sample and use these to estimate the output of the smoothed classifier and generate a certificate. We use a confidence level of $\alpha = 0.99$ to estimate p_A. We also compute certificates using isotropic noise for comparison.

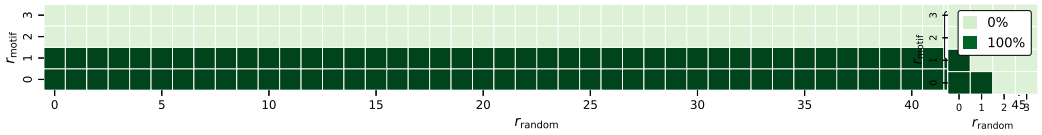
For our certificate with anisotropic noise we consider $\mathbf{p} = (p_{\text{motif}}, p_{\text{random}}) \in \mathbf{P}$ where $\mathbf{P} = \{0.02, 0.04, \ldots, 0.2\} \times \{0.05, 0.1, \ldots, 0.45\}$. Recall that p_{motif} is the noise parameter for the motif part and p_{random} is the noise parameter for the random part. For the isotropic certificate we consider $p \in \{0.02, 0.04, \ldots, 0.2\}$. For the anisotropic certificate we certify over perturbation pairs $\mathbf{r} = (r_{\text{motif}}, r_{\text{random}})$ which means that with high probability r_{motif} edge flips in the motif part and r_{random} edge flips in the random part will not change the label of the smoothed classifier. This is different from the isotropic certificate which guarantees the label does not change for r edge flips anywhere in the graph.

We first analyze how the noise vector $\mathbf{p}$ influences the the certificates. We introduce a score to evaluate the noise parameters. For each $\mathbf{r}$ if we can certify strictly more than half of the test samples in this perturbation space then we add 1 to the score. To motivate the utility of this score consider a smoothed model with large values of noise. In the limit, a perfectly smooth classifier will be constant everywhere. In other words, it will predict the same label for all inputs and give a certified accuracy of 50% for a balanced classification task. This classifier could be certified for arbitrary numbers of edge flips. For this reason, metrics such as total certified



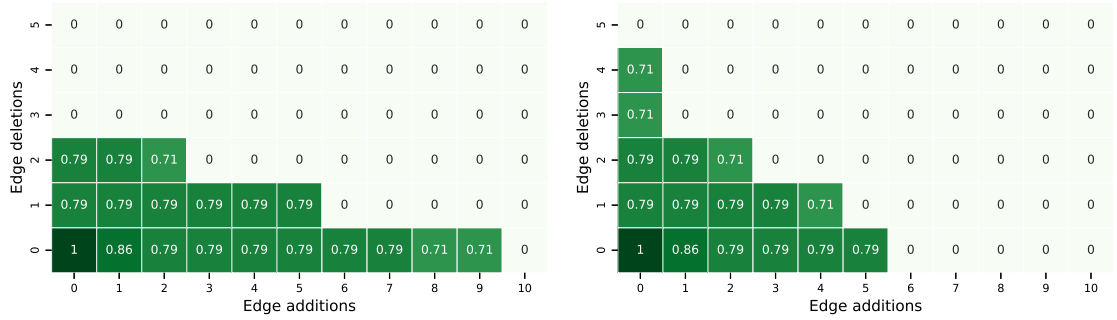
(a) Score. (b) Test-set accuracy.

Figure 7.4: The test-set accuracy of the smoothed classifier and the certified volume for various values of $\mathbf{p} = (p_{\text{motif}}, p_{\text{random}})$.



(a) Anisotropic. (b) Isotropic.

Figure 7.5: Comparison of an anisotropic certificates with $\mathbf{p} = (0.02, 0.45)$ and isotropic certificates for various levels of p. We omit some values of p for the isotropic certificate for readability.



(a) Anisotropic.

(b) Sparsity-aware.

Figure 7.6: A comparison between the anisotropic certificate and the sparsity-aware certificate. Each entry represents the ratio of correctly classified test-set samples that could be certified at a specified number of edge deletions and additions.

area averaged over samples do not necessarily tell us if a noise parameter is useful.

Fig. 7.4a shows our smoothed classifier has the highest score when p_{motif} is small and p_{random} is large. Fig. 7.4b shows large values of p_{random} does not effect the accuracy of the smoothed classifier, but if p_{motif} becomes too large the accuracy begins to drop. These results are expected: p_{motif} cannot be too large as the motif part is more important to determining the label. This motivates to fix p_{motif} to be small and increase p_{random} , retaining high test accuracy whilst increasing the number of edge flips we can certify in $\mathcal{C}_2$.

We take a closer look at the smoothed classifier given by $\mathbf{p} = (0.02, 0.45)$, one of the smoothed classifiers with the highest observed score. We are interested in a smooth model with high test set accuracy that can certify for many values of $\mathbf{r}$. Our model has 100% certified accuracy. The proportion of the test set that can be certified for varying values of $\mathbf{r}$ is shown in Fig. 7.5a. As the figure demonstrates we can certify 100% of the test-set samples to 0 or 1 edge flips in the motif part of the graph and up to 45 edge flips in the random part of the graph. This is the maximum number of possible node pairs in the random part, so we can certify any perturbation in this part of the graph. The smoothed classifier using isotropic noise can also achieve 100% test set accuracy for all values of noise we tested. We show the certification results for when $p = 0.02$, as this is the only value that allows us to certify the entire test-set for one edge flip. We plot the proportion of the test set that can be certified at using this value of isotropic noise in Fig. 7.5b. Using larger values of noise for the isotropic certificate allows for some of the test-set to be certified at a radius of 2, but it can no longer certify the entire test set at radius 1. By using anisotropic noise, and being specific about where edges are being certified, we can certify 46 edge flips with 100% accuracy compared to 1 edge flip with 100% accuracy.

7.5.2 Real-world experiment

We also experiment using the real-world data set MUTAG [41]. In this data set each graph represents a molecule and the goal is to predict the molecules mutagenicity on a specific bacterium, which is encoded into a binary label. Each node has one of 7 discrete node labels corresponding to atomic number which is one-hot encoded. The data set contains a total 188 molecular graphs.

We train a graph neural network which has a single GCN layer [81] with 64 hidden units, followed by a max pooling layer and a linear layer. We use a 80%/10%/10% train/val/test on which to train and optimize the model hyperparameters. We train for a maximum of 500 epochs using the AdamW optimiser [89] with weight decay of 10^{-3}. The initial learning rate is 10^{-3} and it is decayed by 0.5 every 50 epochs.

We compare our certificate to [16], referred here as a sparsity-aware certificate, as this is the only non-isotropic certificate used for graph classification that we are aware of. We consider node pairs where there is an edge in the original graph as $\mathcal{C}_1$ and all other node pairs as $\mathcal{C}_2$. In this scenario, we can certify edge deletions and additions in a comparable way. Following the setup described in [16] we consider $p_1 = 0.4$ which corresponds to the probability of deleting an edge and $p_2 = 0.2$ which corresponds to the probability of adding an edge. We apply noise during training to make the model more robust.

The values computed in Proposition 7.3 differ between the two approaches as $\mathbb{P}(\phi(\tilde{\mathbf{x}}) = \mathbf{z})$ is computed differently. Furthermore, the probability $\phi(\mathbf{x})$ belonging to a region in the anisotropic approach is a product of Binomial distributions (Proposition 7.3) whereas for the sparsity-aware certificate this probability follows a Poison-Binomial distribution. If the assignment of node pairs was dependent on the individual sample, this would generalise our approach further, and would also generalise the sparsity-aware certificate.

Our model has a test-set accuracy of 84%. In Fig. 7.6 we plot the ratio of correctly predicted test points that are certified for varying numbers of edge deletions and additions. We make a few observations from these results. The first is that for values of $\mathbf{r}$ where both methods can certify test points, our method certifies the same quantity of points and in some cases more. The second is that there are two values of $\mathbf{r}$ where the sparsity-aware certificate can certify test samples but the anisotropic certificate cannot. However, there are five values of $\mathbf{r}$ where the anisotropic can certify but the sparsity-aware certificate cannot. Finally, we note that in this experiment, as well as the synthetic experiments, we find our certificates tend to be oblong, i.e. if p_i is larger than we tend to certify for larger values in the r_i direction. This is advantageous in the case where some node pairs are considered more important to the classification label (as demonstrated in the

synthetic experiment).

7.6 Conclusions

We propose in this work, to the best of our knowledge, one of the first methods that introduces structure-aware robustness certificates in the context of classifying undirected, unweighted graphs. To achieve this, we leverage a flexible, anisotropic noise distribution in the framework of randomised smoothing and develope an efficient algorithm to compute certificates. We apply these certificates to a synthetic experiment and demonstrate a clearly improved robustness of graph classifiers that cannot be achieved with an isotropic certificates. We also validate our certificate on real-world experiments and show superior results compared to an existing approach.

Our approach requires defining a priori which edges a user believes to be more or less important to determining the graph label. Such knowledge may come from domain expertise (which we simulate in the synthetic experiment), or we may treat edge deletions and additions differently as in the sparsity-aware approach of Bojchevski, Klicpera, and Günnemann [16]. We may also consider approaches that have been used to identify edges that may be vulnerable to attack. For example, a previous work found edges vulnerable to adversarial attack are those not captured by a low-rank approximation of the adjacency [46]. Another line of work propose that edges where the end-point node features have low Jaccard index are potentially vulnerable [145]. Beyond this, one could learn the importance of the node pairs in a data-driven fashion. One such example is the recent work of Sui et al. [124] where the authors propose to learn the causal relations between node pairs and model outcome. We leave these directions for future work. Finally, even though we have applied our method in the context of graph classification, it can also used for any type of task based on a discrete domain such as binary image classification.

Chapter 8

Conclusion

8.1 Summary

In this thesis, we outline theoretical and empirical results to help understand robustness in the context of graph machine learning. We will briefly summarise our contributions.

Our first theoretical contribution was given in Chapter 3, where we proved that polynomial spectral graph filters are linearly stable. We empirically showed that the term in this bound, which captured the change in graph topology, the error norm, is influenced beyond the number of edge edits of the graph. We did so by showing that the distribution of observed error norm is different when considering different initial graph structures and patterns of perturbation.

We subsequently tightened the bound in Chapter 4 for polynomial spectral graph filters using a more straightforward proof. Furthermore, we proved that a larger class of spectral graph filters have the linear stability property. Furthermore, we expanded on our initial observation that structural properties of the graph can influence stability by providing a bound in terms which involved the degree distribution and how spatially distributed the perturbation is across the graph. Finally, we conducted a thorough empirical study involving perturbing various graphs using different perturbation techniques to validate aspects of the theory.

Following this, in Chapter 5, we proved that two common graph neural network architectures satisfy the linear stability property. We analysed a bound involving double-edge rewiring operations and found a similar interpretation to the more general bound. A limitation of this study is that the results and derivations in this work are specific to the two graph neural network architectures considered.

The stability bounds we provide for graph-based models in Chapters 3, 4 and 5 all consider worst case scenarios, for the input signal, the model parameters and the perturbation. Because of this, we expect the bounds to be very loose, and despite

being of theoretical interest, they are unlikely to be useful in practical applications. This has been noted for other theoretical bounds too [47].

By analysing robustness through a different lens, we proposed an adversarial attack method for graph classifiers in Chapter 6 following a Bayesian optimisation approach. Our approach led to a higher attack success rate and superior sample efficiency than existing methods, which used reinforcement learning. Following the insight that specific patterns can lead to stability, we analysed if patterns emerged in the discovered adversarial examples. From a qualitative perspective, the degree of endpoints nodes of the perturbed edges, changes to the community structure and locality of perturbed edges play a role. Despite our approach being flexible in the settings of attack, a drawback is that the algorithm requires a perturbation to be made via a discrete action space. For example, our approach could not be easily adapted in the weighted graph setting where continuous perturbations to the edge weights are permitted.

Finally, we considered robustness certificates for graph classifiers. Existing certificates provide a lower bound to the decision boundary by considering all directions equally. We can give more refined certificates by considering sub-spaces of the input space. Our certificate allows one to construct a smoothed classifier which gives a deeper understanding of how far points are from the decision boundary in different directions. On the other hand, our certificate is less scalable than isotropic certificates, and the certificates are harder to visualise and analyse as they are inherently high dimensional.

8.2 Potential improvements

We proved stability bounds for the class of polynomial spectral graph filters. Polynomials on a bounded interval are dense in the space of continuous functions on the same bounded interval by the Weierstrass approximation theorem [108, Theorem 7.26]. As all polynomials are linearly stable, this hints that the same may be true of spectral graph filters that use a continuous function applied to the eigenvalues. Related to this, an important future direction is to understand the stability of graph neural networks more generally. By considering graph neural network frameworks such as the graph networks framework [7] or message passing neural networks [57, 19], one may be able to prove a larger class of graph neural networks to be stable.

As noted in the previous section, the stability bounds in the first part of the thesis are too loose to be of practical utility. More refined bounds which consider the input signal, such as a bound based on local Lipschitz continuity [75], will likely lead to more practical bounds. Intuitively, this is because robustness may vary depending on the input. Local Lipschitz bound are closely related to what a regression counterpart

to robustness certificates may look like. If one were to consider what an analogous definition of certified robustness is for a regression problem, a reasonable proposal would be to certify that the output cannot change more than a specified magnitude in a small neighbourhood of the function. A bound on the Lipschitz constant in this neighbourhood could provide such a certificate.

In chapter 6 of the thesis, we considered Bayesian optimisation as an approach to adversarially attacking graph classifiers. In the case where the goal is to attack a simple graph, this problem can be viewed as a search through a large discrete space. One may find optimal adversarial examples via brute force for sufficiently small graphs and perturbation constraints. These may form an interesting benchmark for evaluating adversarial attacks and certified robustness on graph classifiers. For example, a graph on 10 nodes has $15,225$ perturbations of up to 3 edge flips.

Our proposed method for adversarial attack uses a procedure of selecting a single edge to flip at each step to deal with the large size of the search space. However, a more sophisticated search method such as Monto Carlo tree search, which has been shown to be effective in large discrete search spaces such as in the game Go [118], could be incorporated into our framework.

In Chapter 4, we conducted a qualitative analysis of perturbed graphs using a Robust strategy and, separately, an adversarial strategy. In Chapter 6, we evaluated qualitatively the topological properties of the adversarial examples generated. An important future direction for both of these works would be to conduct a statistical analysis to verify that the patterns are significant and quantify the effect's size. Understanding common structural patterns may lead to universal adversarial examples, like those demonstrated to exist for image classifiers [59, 94]. Many graph neural networks are based on fixed (usually low-pass) filtering operations to diffuse information across the graph structure. This common design may be exploitable in an adversarial setting.

In Chapter 7, we propose a certified robustness certificate that jointly verifies different parts of an input graph at different magnitudes. An important future direction would be to combine this with classifiers that can give explanations for the prediction [124] or by using a post hoc subgraph explainer [155]. These methods provide parts of the graph that are important for predicting the label, which gives an automated way of dividing the node pairs into collections required for our proposed certificate. We would expect to be able to certify by a small amount the important node pairs whilst providing much larger certified robustness for the other node pairs.

Throughout this thesis, we have measured the level of perturbation by considering the number of edge flips. As graph-based models are typically functions of the graph topology and the features, future work should also consider the features. A challenge here is that one needs to design a metric that considers both. Because the

domain of graphs and real vectors are inherently quite different, it is not apparent how one may design such a metric. An approach taken in some existing work is to consider binary node features, which are somewhat comparable to binary changes in the adjacency [167]. However, this setting is limited and often not encountered in real-world graph data.

Related to this, we often consider the number of edge changes to the graph as a measure of perturbation. A limitation of this metric is that it assumes a fixed graph ordering which is suitable for tasks such as node classification but not for graph classification due to symmetries that may arise. Furthermore, it is not straightforward to extend to perturbations that change the size of the graph. A challenge here is that for theory involving a graph metric, computing the metric is at least as hard to compute as the isomorphism problem for which no efficient algorithm is known [3], making empirical verification infeasible. However, graph metrics based on optimal transport or graph matching, such as the Gromov Hausdorff metric and the Gromov Wasserstein metric, may provide an appropriate solution as efficient approximations have been recently developed [99, 72].

8.3 Perspectives

We end this thesis with some selected perspectives.

Do we need robustness? In this work and many other works in graph-based machine learning, an explicit assumption is made that robustness is desirable. However, this is not always the case. One can easily imagine synthetic tasks such as predicting if a graph has a property such as bipartiteness, planarity or degree regularity, whereby an ideal classifier would not be robust as single edge flips can change the ground truth label. In real-world graph classification data sets where the graphs represent molecules, the properties of the molecules can radically change by adding or removing a bond. In many cases, the perturbed graph might not even be a valid molecule in the sense that it cannot exist in the physical world. The often overlooked assumption that robustness is desirable may be influenced by machine vision, where machine learning robustness research is most dominant. Here common tasks such as segmentation and classification should always be robust to small l_2 perturbations in the input since it does not change the visible (lower frequency) properties of the image. Researchers should consider if and when robustness is needed and what this might mean for the task.

Robustness beyond inference. Much research has considered robustness at inference time. However, robustness in different parts of the machine-learning pipeline

is important too. For example, in the data collection step, data may be poisoned or a backdoor introduced [14, 149]. We also require models to be robust to how a training distribution is sampled [134] (i.e. reduce variance in the bias-variance tradeoff). The model selection step impacts how flexible a model can be, which has implications for how robust (or not) it can be [78, 128].

Inherent limits on robustness Can a task have inherent robustness limits before even considering the robustness of the model we use, how the model is trained and how the data is sampled? Some works have considered fundamental limits of accuracy and robustness, which can give results inherent to the task and which applies to any model. For example, in [56], the authors generate a simple but high-dimensional task and show an inherent connection between accuracy and the existence of adversarial examples. Going further, Shafahi et al. [113] consider if adversarial examples are inevitable in high dimensional space for arbitrary data distributions. In another line of work, it has been hypothesised that there may be an inherent trade-off between accuracy and robustness [133, 158]. However, it is still an open question how important this trade-off is in practical scenarios [153]. These works consider the geometry of data distributions in a Euclidean metric space. Similar results in graph spaces could elucidate fundamental limits to robustness for learning over graph domains. Recent research into graph neural network expressivity is related to this question, as it restricts how a model can carve up the input space [151].

Appendix A

Appendix for Chapter 3

A.1 Proofs of Lemmas

A.1.1 Proof of Lemma 2

Proof. The function $z^K f(1/z)$ is holomorphic in the disk $|z| \leq 1$ and thus reaches its maximum on the boundary $|z| = 1$ by the maximum modulus principle. For $|z| \leq 1$,

$$|z^K f(1/z)| = |z|^K |f(1/z)| \leq \max_{|z|=1} f(1/z) = \max_{|z|=1} f(z). \tag{A.1}$$

Now, let w be so that $|w| \geq 1$ then $z = w^{-1}$ is inside the unit disk. Using Eq. (A.1) we get

$$|w|^{-K} |f(w)| = |z|^K |f(1/z)| \leq \max_{|z|=1} f(z). \tag{A.2}$$

Multiplying the inequality by $|w|^K$ yields the result. □

A.1.2 Proof of Lemma 3

Proof. Taking the derivative of the expression with respect to ϵ we get

$$\frac{\partial}{\partial \epsilon} \frac{(1+\epsilon)^{K+1}}{\epsilon^2} = \frac{(1+\epsilon)^K ((K-1)\epsilon - 2)}{\epsilon^3}$$

which has a unique zero in the positive reals given by $\epsilon^* = 2/(K-1)$. The second derivative is calculated to be

$$\frac{\partial^2}{\partial \epsilon^2} \frac{(1+\epsilon)^{K+1}}{\epsilon^2} = \frac{(K^2 - 3K + 2)\epsilon^2 - 4(K-2)\epsilon + 6}{\epsilon^4 (\epsilon+1)^{1-K}}.$$

It can be verified that this is strictly positive when evaluated at ϵ^*. By plugging ϵ^* into the original expression we obtain the result. □

Appendix B

Appendix for Chapter 4

B.1 Proofs

Proposition 1. *Polynomial filters $g(\lambda) = \sum_{k=0}^{K} \theta_k \lambda^k$ are linearly stable with respect to any GSO where the spectrum lies in $[-1, 1]$. The stability constant is given by $C = \sum_{k=1}^{K} k|\theta_k|$.*

Proof. The proof technique is the same as in Kenlay, Thanou, and Dong [78]. Note that $\|\mathbf{L}\|_2 \leq 1$ and $\|\mathbf{L}_p\|_2 \leq 1$ so by Levie, Isufi, and Kutyniok [86, Lemma 3] we know that $\|\mathbf{L}^k - \mathbf{L}_p^k\| \leq k\|\mathbf{E}\|_2$. Using this and the the triangle inequality we have:

$$\|g(\mathbf{L}) - g(\mathbf{L}_p)\|_2 = \|\sum_{k=1}^{K} \theta_k(\mathbf{L}^k - \mathbf{L}_p^k)\|_2 \leq \sum_{k=1}^{K} |\theta_k| \|\mathbf{L}^k - \mathbf{L}_p^k\|_2 \leq \sum_{k=1}^{K} k|\theta_k| \|\mathbf{E}\|_2. \quad \square$$

Proposition 2. *The low-pass filter $g(\lambda) = (1 + \alpha\lambda)^{-1}$ is linearly stable with respect to the normalised Laplacian matrix. The constant is given by $C = \alpha$.*

Proof. Let $\mathbf{X} = \mathbf{I}_n + \alpha\mathbf{L}$ and $\mathbf{Y} = \mathbf{I}_n + \alpha\mathbf{L}_p$ where $\mathbf{L}, \mathbf{L}_p$ are normalised Laplacian matrices. Then we have:

$$\|f(\mathbf{L}) - f(\mathbf{L}_p)\|_2 = \|\mathbf{X}^{-1} - \mathbf{Y}^{-1}\|_2 = \|\mathbf{X}^{-1}(\mathbf{Y} - \mathbf{X})\mathbf{Y}^{-1}\|_2 \leq \|\mathbf{X}^{-1}\|_2 \|\mathbf{Y}^{-1}\|_2 \|\mathbf{X} - \mathbf{Y}\|_2. \tag{B.1}$$

Note that $\mathbf{X}$ has eigenvalues in the interval $[1, 1 + 2\alpha]$, so $\mathbf{X}^{-1}$ has eigenvalues in the interval $[(1 + 2\alpha)^{-1}, 1]$. This holds similarly for $\mathbf{Y}$. The matrices $\mathbf{X}^{-1}$ and $\mathbf{Y}^{-1}$ are symmetric and positive definite so their operator norm is the largest eigenvalue $\|\mathbf{X}^{-1}\|_2 = \|\mathbf{Y}^{-1}\|_2 = 1$. Thus,

$$\|f(\mathbf{L}) - f(\mathbf{L}_p)\|_2 \leq \|\mathbf{X}^{-1}\|_2 \|\mathbf{Y}^{-1}\|_2 \|\mathbf{X} - \mathbf{Y}\|_2 \leq \|\mathbf{X} - \mathbf{Y}\|_2 = \alpha\|\mathbf{L} - \mathbf{L}_p\|_2. \quad \square$$

Lemma 1. *Let $\alpha_u \in [0, 1)$. Then the following holds:*

$$\sum_{v \in \mathcal{R}_u} |\frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{d'_u d'_v}}| \leq \sum_{v \in \mathcal{R}_u} \left(\frac{\alpha_u}{1-\alpha_u}\right) \frac{1}{\sqrt{d_u d_v}} \tag{B.2}$$

$$\leq \left(\frac{\alpha_u}{1-\alpha_u}\right) \frac{d_u - \Delta_u^-}{\sqrt{d_u \delta_u}}. \tag{B.3}$$

Proof. The main part of proving this Lemma is proving the following identity:

$$|\frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{d'_u d'_v}}| \leq \left(\frac{\alpha_u}{1-\alpha_u}\right) \frac{1}{\sqrt{d_u d_v}}. \tag{B.4}$$

To prove Eq. (B.4), we will maximise the left hand size bearing in mind that the constraint $\alpha_u \in [0, 1)$ limits the possible values d'_u and d'_v can take. To maximise the left hand size we will consider it as a function of Δ_u and Δ_v and use partial derivatives to reason where the maxima is.

The left hand side of Eq. (B.4) can be written as a function of the change in degree of node u and v

$$f(\Delta_u, \Delta_v) = |\frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{(d_u + \Delta_u)(d_v + \Delta_v)}}| \tag{B.5}$$

on the domain $\Omega = [-\alpha_u d_u, \alpha_u d_u] \times [-\alpha_u d_v, \alpha_u d_v]$. Because the domain is such that $\Omega \subset (0, d_{\max}]^2$, where $d_{\max}$ is the largest degree of all nodes in graph $\mathcal{G}$, this function has no poles. The function f has zeros in its domain and the function is non-differentiable at these points but differentiable away from these points. Since the function is non-negative but not identically zero, the local maximas are strictly positive. Therefore we can consider critical points of the function away from the zeros. We consider the partial derivative of f with respect to Δ_u. The chain rule gives us:

$$\frac{\partial f(\Delta_u, \Delta_v)}{\partial \Delta_u} = \text{sign}(f(\Delta_u, \Delta_v)) \ \frac{\partial}{\partial \Delta_u} \left(\frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{(d_u + \Delta_u)(d_v + \Delta_v)}}\right) \tag{B.6}$$

$$= \text{sign}(f(\Delta_u, \Delta_v)) \ \frac{\partial}{\partial \Delta_u} \left(\frac{-1}{\sqrt{(d_u + \Delta_u)(d_v + \Delta_v)}}\right). \tag{B.7}$$

Because we only consider the function f away from its zeros, $\text{sign}(f(\Delta_u, \Delta_v))$ will be non-zero. Therefore, the partial derivative of f with respect to Δ_u is zero if and only if the partial derivative on the right hand side of Eq. (B.7) is zero. By setting

$z = (d_u + \Delta_u)(d_v + \Delta_v)$ we use the chain rule again to obtain:

$$\frac{\partial}{\partial \Delta_u} \frac{-1}{\sqrt{(d_u + \Delta_u)(d_v + \Delta_v)}} = \frac{\partial}{\partial z} \frac{-1}{\sqrt{z}} \frac{\partial z}{\partial \Delta_u} \tag{B.8}$$

$$= \frac{1}{2z^{3/2}}(d_v + \Delta_v) \tag{B.9}$$

$$= \frac{d_v + \Delta_v}{2\big((d_u + \Delta_u)(d_v + \Delta_v)\big)^{3/2}}. \tag{B.10}$$

The partial derivative for f with respect to Δ_u is zero if and only if the right hand side of Eq. (B.10) is zero. One can see this partial derivative is zero if and only if $d_v + \Delta_v = 0$, but our constraints are such that $d_v + \Delta_v \geq d_v - \alpha_u d_v > 0$. This holds by symmetry for the other partial derivative. Since there are no local maximas in the domain of the function and the function is continuous, the maxima must lie on the boundary of the domain which we write as $\delta\Omega$ [120, Theorem 2.6].

The boundary $\delta\Omega$ describes a closed rectangle. One can parameterise the sides of this rectangle by either fixing $\Delta_u \in \{-\alpha d_u, \alpha d_u\}$ or by fixing $\Delta_v \in \{-\alpha d_v, \alpha d_v\}$. Without loss of generality consider fixing $\Delta_u = -\alpha d_u$ to give a 1D function in Δ_v describing a side of the boundary of this rectangle. The derivative of this 1D function is exactly the partial derivative of f with respect to Δ_v. We proved that both partial derivatives are non-zero in Ω, so in particular the derivative of this 1D function is non-zero and thus the maxima must lie on its boundary [120, Theorem 2.6]. The boundary of a side are the two corner end points in the rectangle adjacent to the side. By considering all four sides we can deduce that the maximum of the function must lie on one or more of the four corners. In the following we will prove that the function has a single maximum and show it takes this value at the corner $(-\alpha_u d_u, -\alpha_u d_v)$.

We first show that the function maps two of the corners to the same value:

$$f(-\alpha_u d_u, \alpha_u d_v) = \left|\frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{(d_u - \alpha_u d_u)(d_v + \alpha_u d_v)}}\right| \tag{B.11}$$

$$= \left|\frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{(1 - \alpha_u)(1 + \alpha_u)}\sqrt{d_u d_v}}\right| \tag{B.12}$$

$$= \left|\frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{(d_u + \alpha_u d_u)(d_v - \alpha_u d_v)}}\right| = f(\alpha_u d_u, -\alpha_u d_v). \tag{B.13}$$

Next, we show that $f(-\alpha_u d_u, -\alpha_u d_v) \geq f(-\alpha_u d_u, \alpha_u d_v)$. We do this by proving that $p_1 = f(-\alpha_u d_u, -\alpha_u d_v) - f(-\alpha_u d_u, \alpha_u d_v)$ is non-negative. Writing this out we have that:

$$p_1 = \left|\frac{1}{\sqrt{d_u d_v}} - \frac{1}{(1 - \alpha_u)\sqrt{d_u d_v}}\right| - \left|\frac{1}{\sqrt{d_u d_v}} - \frac{1}{\sqrt{(1 - \alpha_u^2) d_u d_v}}\right|. \tag{B.14}$$

Since $(1-\alpha_u)$ and $\sqrt{1-\alpha_u^2}$ both lie in the interval $(0,1]$, we know that both the values inside the absolute values are non-positive. By negating the values inside and removing the absolute value we get that this is equal to:

$$p_1 = \left(\frac{1}{(1-\alpha_u)\sqrt{d_u d_v}} - \frac{1}{\sqrt{d_u d_v}}\right) - \left(\frac{1}{\sqrt{(1-\alpha_u^2)d_u d_v}} - \frac{1}{\sqrt{d_u d_v}}\right) \tag{B.15}$$

$$= \frac{1}{(1-\alpha_u)\sqrt{d_u d_v}} - \frac{1}{\sqrt{(1-\alpha_u^2)d_u d_v}}. \tag{B.16}$$

This quantity is non-negative if and only if $(1-\alpha_u) \leq \sqrt{(1-\alpha_u^2)}$. By squaring both sides and rearranging we get that this is true if and only if $\alpha_u \in [0,1]$ which holds true by our assumption.

Finally, we show that $f(-\alpha_u d_u, -\alpha_u d_v) \geq f(\alpha_u d_u, \alpha_u d_v)$. Similar to before we define $p_2 = f(-\alpha_u d_u, -\alpha_u d_v) - f(\alpha_u d_u, \alpha_u d_v)$ is non-negative:

$$p_2 = \left|\frac{1}{\sqrt{d_u d_v}} - \frac{1}{(1-\alpha_u)\sqrt{d_u d_v}}\right| - \left|\frac{1}{\sqrt{d_u d_v}} - \frac{1}{(1+\alpha_u)\sqrt{d_u d_v}}\right| \tag{B.17}$$

$$= \left(\frac{1}{(1-\alpha_u)\sqrt{d_u d_v}} - \frac{1}{\sqrt{d_u d_v}}\right) - \left(\frac{1}{\sqrt{d_u d_v}} - \frac{1}{(1+\alpha_u)\sqrt{d_u d_v}}\right) \tag{B.18}$$

$$= \frac{1}{\sqrt{d_u d_v}}\left(\frac{1}{1-\alpha_u} + \frac{1}{1+\alpha_u} - 2\right) \tag{B.19}$$

$$= \frac{1}{\sqrt{d_u d_v}}\left(\frac{-2\alpha_u^2}{(\alpha_u - 1)(\alpha_u + 1)}\right). \tag{B.20}$$

The above is non-negative for $\alpha_u \in [0,1)$. This proves that $f(-\alpha_u d_u, -\alpha_u d_v)$ is a maxima of the four corners hence a global maxima of the function f. We show that the inequality in Eq. (B.4) holds and is tight by showing equality holds when the left hand side takes its largest value among all possible values of d'_u and d'_v (equivalently Δ_u and Δ_v):

$$\max_{\substack{\Delta_u \in [-\alpha d_u, \alpha d_u] \\ \Delta_v \in [-\alpha d_v, \alpha d_v]}} f(\Delta_u, \Delta_v) = f(-\alpha_u d_u, -\alpha_u d_v) \tag{B.21}$$

$$= \frac{1}{(1-\alpha_u)\sqrt{d_u d_v}} - \frac{1}{\sqrt{d_u d_v}} = \left(\frac{\alpha_u}{1-\alpha_u}\right)\frac{1}{\sqrt{d_u d_v}}. \tag{B.22}$$

The first inequality (Eq. (B.2)) in Lemma 4 follows immediately from this. The second inequality (Eq. (B.3)) comes from noting that $d_v \geq \delta_u \implies 1/\sqrt{d_u d_v} \leq 1/\sqrt{d_u \delta_u}$ and that $|\mathcal{R}_u| = d_u - \Delta_u^-$ giving:

$$\sum_{v \in \mathcal{R}_u}\left(\frac{\alpha_u}{1-\alpha_u}\right)\frac{1}{\sqrt{d_u d_v}} \leq \sum_{v \in \mathcal{R}_u}\left(\frac{\alpha_u}{1-\alpha_u}\right)\frac{1}{\sqrt{d_u \delta_u}} = \left(\frac{\alpha_u}{1-\alpha_u}\right)\frac{d_u - \Delta_u^-}{\sqrt{d_u \delta_u}}. \tag{B.23}$$

□

Theorem 1. *Let $\alpha_u \in [0, 1)$ for all nodes $u \in \mathcal{V}$. Then the following holds:*

$$\|\mathbf{E}\|_2 \leq \max_{u \in \mathcal{V}} \left\{ \frac{\Delta_u^-}{\sqrt{d_u \delta_u}} + \frac{\Delta_u^+}{\sqrt{d'_u \delta'_u}} + \left(\frac{\alpha_u}{1 - \alpha_u} \right) \frac{d_u - \Delta_u^-}{\sqrt{d_u \delta_u}} \right\}.$$

Proof. If $\alpha_u \in [0, 1)$ for all nodes then Eq. (4.9) also holds for all nodes. Substituting Eq. (4.9) into Eq. (4.5) gives the desired result. □

B.2 Random graph models

In this work we consider graphs without isolated nodes for simplicity. Although the bounds do not require graphs to be connected, we always consider the unperturbed graphs to be connected. To achieve this in practice we use rejection sampling, i.e., sampling from the random graph models until we sample a connected graph. We describe the random graph models we use in our experiments in detail below. Where available we use implementations provided by the NetworkX and PyGSP libraries. Summary statistics describing the degree distribution as well as distance between nodes are given in Table B.1.

Graph	Mean degree	Degree standard deviation	Average shortest path length	Diameter	Degree correlation
K-Reg	3.00	0.00	4.83	8.77	N/A
WS	4.00	0.62	5.07	10.58	-0.02
ENZYMES	4.02	1.01	5.35	13.60	0.02
PROTEINS_full	3.89	1.03	6.66	17.37	0.06
SBM	4.22	1.87	3.53	7.43	-0.02
Assortative	4.64	2.02	4.21	10.79	0.80
ER	4.66	2.04	3.15	6.41	-0.02
BA	5.82	4.72	2.59	4.43	-0.16

Table B.1: Summary statistics averaged across the generated graphs. The first two columns give the average and standard deviation of the degree sequence. The average shortest path length and diameter (largest shortest path length) measure node connectivity. The degree correlation measures assortativity.

Erdős-Rényi Erdős-Rényi graphs are constructed by independently including each possible edge between any pair of nodes with probability p [55]. For sufficiently large graphs the graph will be connected with high probability if $p > \ln n/n$ and disconnected if $p < \ln n/n$. We thus chose $p = \ln n/n$. The degree distribution of Erdős-Rényi graphs is approximately a Poisson distribution.

Barabási–Albert Barabási–Albert graphs are randomly generated scale-free (with power-law degree distributions) graphs which are constructed using a preferential

attachment mechanism [2]. The graphs are constructed using parameters n, the number of nodes, and m, the number of edges that are preferentially attached. An initial graph is given by a star graph on $m + 1$ nodes. Then, until the graph has n nodes the following step is repeated. A new node is introduced and connected to m existing nodes with probabilities proportional to their degrees. Barabási–Albert graphs are connected by construction.

Watts-Strogatz Watts-Strogatz graphs with appropriate parameters exhibit small-world properties such as small average path lengths and high clustering [141]. The graph begins as a ring lattice which is obtained by taking a cycle of nodes and connecting each node to its K nearest neighbours. Then, each edge is rewired independently with probability p, which means deleting edge $i \sim j$ and adding edge $i \sim j'$ such that $j' \notin \{i, j\}$ is selected uniformly at random. We connected each node to its $K = 4$ neighbours for the initial configuration and rewire each edge with probability $p = 0.1$.

K-regular A K-regular graph is one such that the degree of every node is K. We use the algorithm described in Steger and Wormald [122] and implemented in NetworkX. For a fixed K, the algorithm samples graphs with n nodes all of degree K almost uniformly at random in the sense that the distribution approaches uniform in the limit $n \to \infty$.

Assortative An assortative graph is one such that there is a high positive correlation between the degree of end points along edges. To generate assortative graphs we use a variant of the Xulvi-Brunet & Sokolov (XBS) Algorithm [152] applied to Erdős-Rényi graphs. The XBS algorithm iteratively rewires edges in the graph to increase assortativity whilst keeping the degree sequence fixed. At each step two edges, corresponding to four nodes, are chosen uniformly at random. With probability p, these edges are rewired in a way that increases the assortativity. Otherwise, the edges are rewired randomly. In our experiments we set this probability to $p = 1$, meaning the assortativity increases in each step. In the original algorithm a graph can become disconnected; we discard an iteration if it disconnects the graph. Instead of running the algorithm for a fixed number of steps we run the algorithm until the degree correlation is at least 0.8. We describe the algorithm in Algorithm 1.

Algorithm 1 XBS Algorithm

Input: An initial graph $\mathcal{G}$, assortative threshold a
repeat
  Initialize $\mathcal{G}_{temp} \leftarrow \mathcal{G}$.
  Sample edges $u \sim v$ and $u' \sim v'$ from $\mathcal{G}_{temp}$ such that u, v, u', v' are unique
  Delete edges $u \sim v$ and $u' \sim v'$ from $\mathcal{G}_{temp}$
  Connect the two nodes in $\mathcal{G}_{temp}$ from $\{u, v, u', v'\}$ with the highest degree
  Connect the two nodes in $\mathcal{G}_{temp}$ from $\{u, v, u', v'\}$ with the lowest degree
  if $\mathcal{G}_{temp}$ is connected **then**
    $\mathcal{G} \leftarrow \mathcal{G}_{temp}$
  end if
until Assortativity of $\mathcal{G}$ more than or equal to a
Output: Perturbed graph $\mathcal{G}$

(a) Before rewiring (b) After rewiring

Figure B.1: In the rewiring operation the red edges are deleted and the blue edges are added. The degree of each node remains the same.

B.3 Perturbation strategies

We make use of four random strategies (Add, Delete, Add/Delete and Rewire), one strategy which gives perturbations according to an optimisation based search strategy (PGD), and a strategy which is based on the theory described in Section 4.5 (Robust). All strategies delete or add a fixed number of edges. The rewiring operation is depicted graphically in Fig. B.2. We describe Robust and PGD in more detail.

Robust The robust strategy works by iteratively building the perturbed graph. Consider two unique nodes u and v, then we say an edge is flipped between them if an edge $u \sim v$ exists and is deleted, or if the edge does not exist and is added. A single iteration consists of finding nodes u and v that have not been flipped in previous iterations such that $\|\mathbf{E}\|_1$ is minimal. The steps of the algorithm are described in Algorithm 2.

PGD Projected gradient descent is a variant of gradient descent where after each gradient update a projection operation is applied. This strategy aims to find adversarial examples and follows closely the strategy described in Xu et al. [150], with three modifications. The first modification is that during the sampling procedure

Algorithm 2 Robust Algorithm

Input: An initial graph $\mathcal{G}$, a budget B
Initialize set of flipped edges $\mathcal{F} \leftarrow \emptyset$.
while $|\mathcal{F}| < B$ **do**
  Initialize candidate edge e
  Initialize candidate edge norm $f \leftarrow \infty$
  for Potential edge $e' \in \{1, \ldots, n\}^2 \setminus \mathcal{F}$ **do**
    Initialize $\mathcal{G}_{temp} \leftarrow \mathcal{G}$
    Flip edge e' in $\mathcal{G}_{temp}$
    Calculate normalised Laplacian matrix $\mathbf{L}_{temp}$ of $\mathcal{G}_{temp}$
    if $\|\mathbf{L}_{temp}\|_1 < f'$ **then**
      $e \leftarrow e'$
      $f \leftarrow \|\mathbf{L}_{temp}\|_1$
    end if
  end for
  Flip edge e in $\mathcal{G}$
  $F \leftarrow F \cup \{e\}$
end while
Output: Perturbed graph $\mathcal{G}$

[**pgd**], we not only discard perturbed graphs that are over budget, but also those containing isolated nodes.. The second modification is that we perform gradient ascent on the relative error between the original signal (before noise is added) and the denoised signal (estimate). This is different from the negative cross-entropy or Carlili-Wagner loss functions used in Xu et al. [150] which are better suited for classification problems. Finally, our filter involves the inverse of a matrix which can become singular making it not possible to propagate gradients. We describe how we calculate the gradient used in **pgd** in Algorithm 3. The variables $\mathbf{A}', \mathbf{s}$ and $\mathbf{S}$ are described further in Xu et al. [150]. We include additional steps 3 and 4 to improve the stability of the filtering operation (step 6). Step 4 projects negative values to 0 and values above 1 to 1. Step 6 makes use of the lower–upper (LU) decomposition for solving linear systems. We use a learning rate of $\eta_t = 200/\sqrt{t}$, and implement $T = 200$ iterations and $K = 250$ random trials.

B.4 Additional results

B.4.1 Experimental setup

We experiment with changing the size of the random graphs, the size of the perturbation and the SNR ratio, whilst keeping other experimental variables fixed. We present the bound given by Theorem 3 for $n = 500$ in Fig. B.2a. As expected, we find our bounds to be looser for large graphs. Furthermore, a larger proportion

Algorithm 3 Numerically stable gradient calculation

1: **Input:** Adjacency matrix $\mathbf{A}$, probability vector $\mathbf{s}$, target signal $\mathbf{y}$, noisy signal $\mathbf{x}$.
2: $\mathbf{A}' = \mathbf{A} + (\bar{\mathbf{A}} - \mathbf{A}) \circ \mathbf{S}$ [**pgd**] where $\mathbf{S}$ is the matrix form of $\mathbf{s}$
3: $\mathbf{A}' = \mathbf{A}' + \mathrm{diag}(\mathcal{N}(\mathbf{0}_n, 10^{-3} \times \mathbf{I}_n))$
4: $\mathbf{A}' = \mathrm{clamp}(\mathbf{A}')$
5: Calculate Laplacian $\mathbf{L}'$ of $\mathbf{A}'$
6: Calculate denoised signal $\hat{\mathbf{y}} = (\mathbf{I}_n + \mathbf{L}')^{-1}\mathbf{x}$
7: Calculate relative error loss $\mathcal{L} = \|\mathbf{y} - \hat{\mathbf{y}}\|/\|\mathbf{y}\|$
8: Calculate gradient $\nabla_s \mathcal{L}$
9: **Output:** $\nabla_s \mathcal{L}$

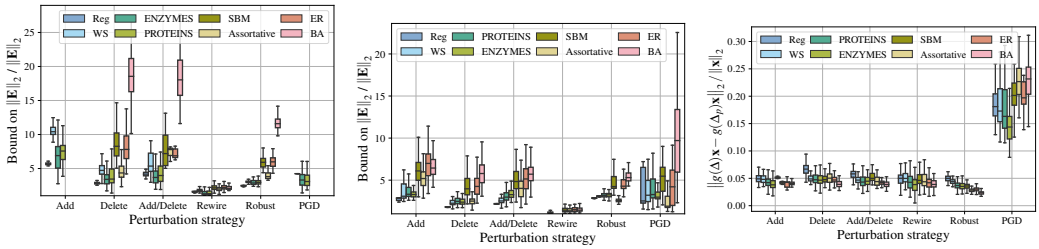
of experiments are invalid more often due a larger number of edge edits and more opportunities for the assumption of Theorem 1 to be violated. Nevertheless, our bounds are still valuable in practice as in the graph classification setting graphs with 100 or less nodes are common. For larger graphs, we approximate the Robust strategy with a sampling scheme as exhaustively searching for an edge at each step became computationally prohibitive. The sampling scheme alters step 6 in Algorithm 2 such that instead of considering every possible edge flip, we randomly sample 50 edges present in the graph, and 50 pairs of nodes which are not present as edges in the graph.

The bound in Theorem 3 tends to be tighter for a smaller perturbation level of $\lfloor 2\% \cdot |\mathcal{E}| \rfloor$ edge edits (Fig.B.2b). [1]. Our approach to generating synthetic signals for the random graphs are entangled with the graph structure. This is reasonable since an implicit assumption behind using graph filters is that the signal and graph are related. Nevertheless, by varying SNR we can control the degree of such entanglement, and we show how the relative output distance changes for a lower SNR of -10dB (Fig. B.2c, patterns are similar for a higher SNR of 10dB). We consider the relative output distance here rather than the bound in Theorem 3. This is because all perturbation strategies apart from PGD are invariant to the level of noise.

B.4.2 How close are the relative output distance to the filter distance?

In Section 4.4 we have established a relationship between the relative output distance in Eq. (4.2) and the filter distance in Eq. (4.3). The looseness of the bound given in Eq. (4.3) in shown in Fig. B.3. As we can see the PGD strategy gives rise to a tighter inequality compares to other strategies.

[1] With the smaller perturbation level, we have that for some graphs in the real-world data sets $\lfloor 2\% \cdot |\mathcal{E}| \rfloor < 4$, meaning a rewire operations cannot take place within the budget of perturbation, hence the absence of the bars in the plot.



(a) Larger graphs ($n = 500$) (b) Smaller perturbation (c) Lower SNR ($-10dB$)

Figure B.2: Further results for varying experimental conditions

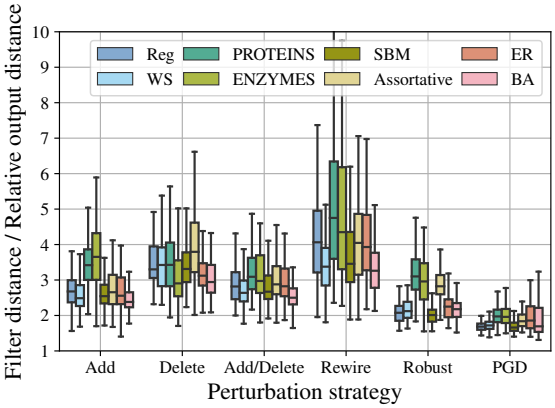


Figure B.3: Looseness of the bound relating the relative output distance to the filter distance.

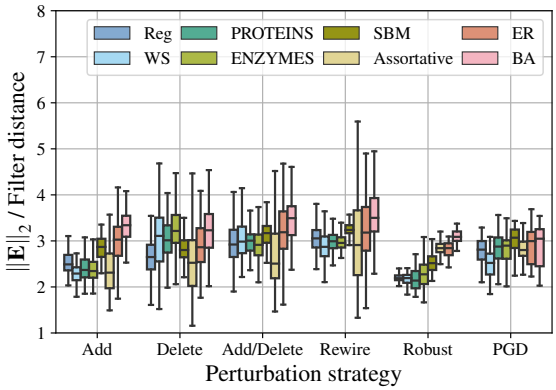


Figure B.4: Looseness of the linear stability bound.

B.4.3 How tight is the linear stability bound?

Linearly stable filters have the property that there exists an upper bound on the filter distance which is linearly proportional to the error norm. In the previous works of Levie, Isufi, and Kutyniok [86] and Kenlay, Thanou, and Dong [78] it has also been shown experimentally that the relationship is linear in practice. The low-pass filter we use in experiments has a stability constant of one meaning $\|f(\mathbf{L}) - f(\mathbf{L}_p)\|_2 \leq \|\mathbf{E}\|_2$. Our experiments (Fig. B.4) show the looseness of this bound to be at least 1.5 in all our experiments.

B.4.4 Validity of experiments

As mentioned in Section 6.4, some experiments give $\alpha_u > 1$ for some node u in the graph meaning that strictly speaking the bound of Theorem 3 cannot be applied. Therefore, we discard the results of these experiments from our experimental analysis for figures relating to the bound. Explicitly we drop invalid experiments for the following figures: Fig. 4.1, Fig. 4.2, the bottom right pane of Fig. 4.3, Fig. B.2b, Fig. B.2c and the right pane of Fig. B.7. Table B.2 shows the proportion out of 100 experiments we run which are valid (in the sense that the assumption on α_u is satisfied) for all combinations of random graph model and perturbation strategy. As noted in Section 4.5, for the assumption on α_u to be violated the degree of at least one node must at least double after perturbation. Since Delete only decreases the degree of nodes, and Rewire preserves the degree of all nodes, experiments with these perturbation strategies are always valid as expected. The graph models with high-variance degree distributions (ER, BA, Assortative) tend to have the assumption violated more often. This is possibly due to a large number of leaf nodes, to which if a single edge is added the assumption on α_u would be violated.

Graph	Add	Delete	Add/Delete	Rewire	Robust	PGD
K-Reg	0.73	1.00	0.98	1.00	1.00	0.65
WS	0.85	1.00	0.95	1.00	1.00	0.69
PROTEINS_full	0.70	1.00	0.90	1.00	1.00	0.68
ENZYMES	0.80	1.00	0.92	1.00	1.00	0.80
SBM	0.03	1.00	0.23	1.00	1.00	0.02
Assortative	0.04	1.00	0.28	1.00	1.00	0.09
ER	0.07	1.00	0.33	1.00	1.00	0.14
BA	0.38	1.00	0.91	1.00	1.00	0.35

Table B.2: Proportion of experiments which are valid ($\alpha_u \in [0, 1)$ for all nodes $u \in \mathcal{V}$) across all graph types and perturbation strategies.

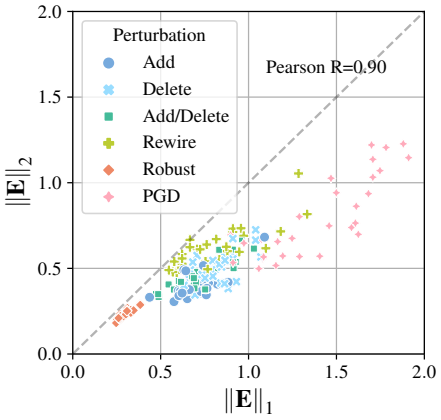


Figure B.5: Comparison of $\|\mathbf{E}\|_1$ and $\|\mathbf{E}\|_2$. The dashed line represents $\|\mathbf{E}\|_2 = \|\mathbf{E}\|_1$.

B.4.5 How tight is the bound $||\mathbf{E}||_2 \leq ||\mathbf{E}||_1$?

In Section 4.6.2 we consider the tightness of the inequality $||\mathbf{E}||_2 \leq ||\mathbf{E}||_1$ for various random graph models and perturbation strategies. For each strategy we plot the absolute value of $||\mathbf{E}||_2$ and $||\mathbf{E}||_1$ for all random graph models in Fig. B.5. We plot just 5 repeats for clarity. In Fig. B.6 we plot the looseness of the bound.

B.4.6 How tight are the bounds on $||\mathbf{E}||_1$ and $||\mathbf{E}||_2$?

In Section 4.6.3 we consider the looseness of Eq. 4.9. We can consider the looseness for each component seperately which is shown in Fig. B.7. It can be seen that the bound on the third term is the loosest.

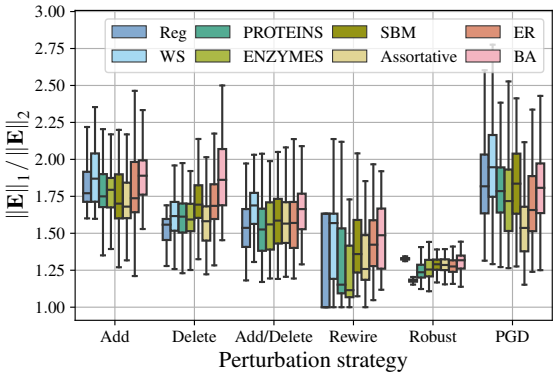


Figure B.6: Looseness of the inequality $\|\mathbf{E}\|_1 \leq \|\mathbf{E}\|_2$ under different perturbation strategies and random graph models.

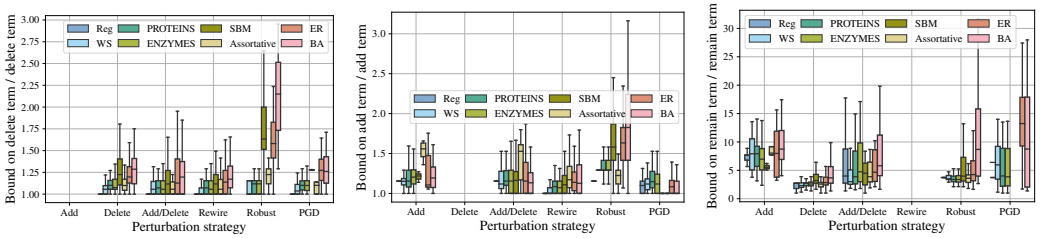


Figure B.7: Looseness of the bound for the three terms in Eq. (5.2). The first term, as well as the bound for the first term, evaluate to zero for the Add strategy and thus looseness is undefined. The same applies to the second term for the Delete strategy, and the third term for the Rewire strategy.

Appendix C

Appendix for Chapter 6

C.1 Algorithms

The overall algorithm of GRABNEL is shown in Algorithm 4.

Algorithm 4 Overall pseudocode of the GRABNEL routine.

Input: Original graph $\mathcal{G}_0$, victim model f_θ, n_{init} (the number of random initialising points), Query budget B, Perturbation budget Δ.
Output: An adversarial graph $\mathcal{G}^*$
Set base graph $\mathcal{G}_{\text{base}} \leftarrow \mathcal{G}_0$; initialise stage count `stage` $\leftarrow 0$.
Randomly sample n_{init} perturbed graphs $\{\mathcal{G}'\}_{i=1}^{n_{\text{init}}}$ that are 1 edit distance different from $\mathcal{G}$ and query each perturbed graph to obtain their attack losses $\mathcal{L}_{\text{attack}}(f_\theta, \mathcal{G}'_i)$ (these random samples are counted towards the query budget of the current stage).
Compute the WL feature encoding for all graphs: $(\Phi(\mathcal{G}'_1), \ldots, \Phi(\mathcal{G}'_{n_{\text{init}}})) =$ `WLFeatureExtract`$(\mathcal{G}_0, (\mathcal{G}'_1, \ldots \mathcal{G}'_{n_{\text{init}}}))$. *// See App. C.2 for details of `WLFeatureExtract`.*
Fit the sparse Bayesian linear regression surrogate with the data $\{\Phi(\mathcal{G}'_i), \mathcal{L}_{\text{attack}}(f_\theta, \mathcal{G}'_i)\}_{i=1}^{n_{\text{init}}}$
Divide total budget of B into Δ stages *// See "Sequential perturbation selection"*
while query budget is not exhausted and attack has not succeeded **do**
    if query budget of the *current stage* is exhausted **then**
        Increment the stage count `stage` $\leftarrow$ `stage` $+ 1$ and update the base graph $\mathcal{G}_{\text{base}}$ with the graph leading to largest increase in attack loss in the previous stage. *// Refer to Fig. 1*
    end if
    Propose graph to be queried next $\mathcal{G}'_{\text{proposal}}$ via acquisition optimisation. *// See "Optimisation of acquisition function"*
    Query f_θ for the graph proposed in the previous step to calculate its attack loss.
    if attack succeeded **then**
        Set $\mathcal{G}^* \leftarrow \mathcal{G}'_{\text{proposal}}$ and **return** it.
    end if
    Augment the observed data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathcal{G}'_{\text{proposal}}, \mathcal{L}_{\text{attack}}(f_\theta, \mathcal{G}'_{\text{proposal}})\}$, update the WL feature encodings of *all* observed graphs $(\Phi(\mathcal{G}'_1), \ldots, \Phi(\mathcal{G}'_{|\mathcal{D}|})) =$ `WLFeatureExtract`$(\mathcal{G}_0, (\mathcal{G}'_1, \ldots \mathcal{G}'_{|\mathcal{D}|}))$ and re-fit the surrogate.
end while
return `None` *// Failed attack within the query budget*

C.2 WL feature extractor

In this section we describe `WLFeatureExtract` in Algorithm 4 in greater detail. The module takes in both the input graph itself and the set of all input graphs (including

itself), as the second argument is to construct a collection of all WL features seen in any of the input graphs and controls the dimensionality of the output feature vector so that the entries in feature vectors of different input graphs represent the same WL feature. For an illustrated example of the procedure, the readers are referred to Fig. C.1.

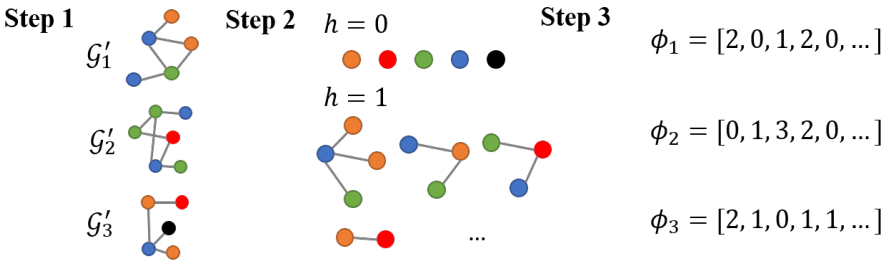


Figure C.1: Illustration of the WL extractor. Consider an example of an input of three graphs to the extractor $\{\mathcal{G}'_1, \mathcal{G}'_2, \mathcal{G}'_3\}$ with colors representing the different (discrete) node labels and we would like to compute $\texttt{WLFeatureExtract}(\{\mathcal{G}'_i, \{\mathcal{G}'_1, \mathcal{G}'_2, \mathcal{G}'_3\}) \; \forall i$ (**Step 1**). The extractor module takes in 2 arguments, as the second argument consisting of the set of all input graphs is used to generate a collection of all possible Weisfeiler-Lehman features seen in *all input graphs* (**Step 2**), up to $H \in \mathbb{N}^+$ where H is the number of WL iterations specified. This step involves computing the Weisfeiler-Lehman embedding on all of the input graphs using the routine introduced in [114]. The extractor finally counts the number of each features present from Step 2 and outputs the feature vector (**Step 3**; only $h = 0$ part of the feature vector is shown in the figure – note that $\mathcal{G}'_1$ has 2 orange nodes, 2 blue nodes and 1 green nodes which yields the corresponding feature vector ϕ_1). Note that if a particular feature present in the entire set of input graphs is not present in a particular graph, the entry is filled with zero. *The graphs here are for illustration only; in our task each input graph is only one edit distance different from the base graph* $\mathcal{G}_0$.

C.3 Implementation Details

Datasets We provide some key descriptive statistics of the TU datasets [95] in Table C.2. All TU datasets may be downloaded at `https://chrsmrrs.github.io/datasets/docs/datasets/`. The MNIST-75sp dataset is generated from scripts available at `https://github.com/bknyaz/graph_attention_pool`, and the ER-graphs dataset used in App. C.4.2 is available at `https://github.com/Hanjun-Dai/graph_adversarial_attack`. The details on the Twitter fake news data are described in the following section.

Twitter dataset We used the Twitter dataset described in [135]. In this dataset, each graph represents a rumour cascade. A cascade is made up of nodes which represents both a tweet and the corresponding user who posted the tweet. An edge

Table C.1: Node features used in the Twitter dataset.

Feature	Datatype	Transform applied	Description
Rumour category	categorical	One hot encoding	Topic of the rumour
Tweet date	float		Date and time the tweet was posted
User account age	float		How old the account is in days
User verified	bool		If the user is verified by Twitter
User followers	int	Log transform	How many followers the user has
User followees	int	Log transform	How many other users the user follows
User engagement	float	Log transform	How active the user has been since joining
Was retweeted	bool		If the tweet was retweeted

exists between nodes u and v if u is a retweet of node v. The graphs are directed but for simplicity we drop the direction of edges. We use node features which are described in table C.1. We apply a log transform to features with a high level of skewness. All data was normalised by subtracting the mean and divided by the standard deviation (estimated using node features from the training set). Further information of these features are give in the supplementary material of [135]. The graph is labelled as true, false or mixed, corresponding to the judged veracity of the rumour, and the learning task is to correctly predict the label of the graph. Many of the graphs in the original dataset are small (and therefore many were topologically similar), we discard samples where the graph has less than 5 nodes. Furthermore, we use downsampling to balance the dataset so each label appears an equal number of times. After all preprocessing steps are complete the dataset is made up of 4746 labelled graphs.

A challenge of node injection is deciding how to choose node features. We reasoned about how to do this for each feature based on the feature semantics. We choose values based on the assumption that the inserted node (tweet) is being inserted from a bot account trying to emulate other users in the cascade. The rumour category is the same for every node in the graph, so inserted nodes used the same as other nodes in the graph. We set the tweet date to the largest of other nodes, to represent an attack in the evasion setting. We set the user account age, followers, followees, retweet status and verified status to the minimum of all other nodes in the graph (using the convention that $\mathit{False} \leq \mathit{True}$). The user engagement was set to the median among other nodes in the graph. In practice, a user may have control over some of these variables such as the number of followees (by following other accounts) or user engagement (by posting tweets).

Computing Environment We conduct all experiments, unless otherwise specified, on a shared server with an Intel Xeon CPU and 256GB of RAM.

Table C.2: Key statistics of the TU datasets used.

Dataset	#graphs	#labels	Avg #nodes	Avg #edges
IMDB-M	1500	3	13.0	65.9
PROTEINS	1113	2	39.1	72.8
COLLAB	5000	3	74.5	2457.8
REDDIT-MULTI-5K	4999	5	508.8	594.9

Victim models We focus our attack on two widely used graph neural networks, namely graph convolutional network (GCN) [81] and graph isomorphism network (GIN) [151]. We also consider an attack on ChebyGIN [82] and Graph U-Net [52]. The graph convolution layers in these models work by aggregating information across the graph edges and then updating combined node features to output new node features. Multiple layers of graph convolution are used. A readout layer transforms the final node embeddings into a fixed-sized graph embedding which can then be fed through a linear layer and a softmax activation function to provide predicted probabilities for each class.

The GCN graph convolutions take the form

$$\mathbf{X}^{(h)} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathcal{X}^{(h-1)}\mathbf{\Theta}^{(h)})$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ is the adjacency matrix with self loops, $\tilde{\mathbf{D}} = \text{diag}(d_1 + 1, d_2 + 1, \dots d_n + 1) = \text{diag}(\mathbf{1}\tilde{\mathbf{A}})$, is a diagonal matrix where d_u is the degree of node u. $\mathbf{\Theta}^{(h)}$ and $\mathcal{X}^{(h)}$ are the weight matrix and node features in layer h, respectively. For the first layer $\mathcal{X}^{(0)}$ is the original node features. We use three GCN convolutions where the dimension of the hidden node representations are 16. A max pooling across feature maps is applied to the final layer to give a fixed length graph representation which is then used as input to a linear layer.

The graph isomorphism architecture (GIN) is provably more expressive (in terms of distinguishing graph topologies) than the GCN architecture [151]. The graph convolution takes the form

$$\mathbf{X}^{(h)} = \text{MLP}^{(h)}\big((1 + \epsilon^{(h)})\mathcal{X}^{(h-1)} + \mathbf{A}\mathcal{X}^{(h-1)}\big),$$

where $^{(h)}$ is a learnable scalar parameter and $\text{MLP}^{(h)}$ is a multilayer perceptron. In our experiments the MLP consists of a single hidden layer of dimension 64 using ReLU activation functions and batch norm applied before applying the activation to the hidden units. We use 5 convolutional layers, applying batchnorm and ReLU activation functions in-between. For the readout function we utilise the representation after each of the GIN convolutions. For each representation, a sum pooling is

applied followed by a linear layer. During training dropout is applied to the output of each of the linear layers with probability $p = 0.5$. The outputs of the linear layer are summed to give a final logit score for each class.

The ChebyGIN architecture is similar to the GIN architecture but aggregates information from nodes across multi-hop neighbourhoods. This is achieved by using higher-order Chebyshev polynomials as the aggregation matrix. The polynomial filter is evaluated using the (shifted) normalised Laplacian matrix $\mathbf{L} = -\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. Chebyshev polynomials can be defined recursively:

$$T_k(\mathbf{L}) = \begin{cases} \mathbf{I}_n, & \text{for } k = 0 \\ \mathbf{L}, & \text{for } k = 1 \\ 2\mathbf{L}T_{k-1}(\mathbf{L}) - T_{k-2}(\mathbf{L}), & \text{for } k > 2 \end{cases}. \tag{C.1}$$

The ChebyGIN convolution is then defined to be

$$\mathbf{X}^{(h)} = \mathrm{MLP}\big((1+\epsilon)\mathcal{X}^{(h-1)} + T_k(\mathbf{L})\mathcal{X}^{(h-1)}\big).$$

We used a pretrained model used in [82] available at `https://github.com/bknyaz/graph_attention_pool`. We use the model with supervised attention, the best-performing pre-trained model available.

Graph U-Nets are an autoencoder like architecture with skip connections [52]. The architecture consists of a differential graph pooling layer *gPool* and a differential graph unpooling layer *gUnpool* which we briefly describe. To do differential graph pooling a $d \times p$ projection matrix $\mathbf{p}$ is used to compute a length n vector $\mathbf{y} = \mathbf{X}\mathbf{p}$. A ranking operation is used to select the indices of the largest k entries $idx = \mathrm{rank}(\mathbf{y}, k)$ which represent nodes to be included in the sub-graph after pooling. Using this we can select the subgraph adjacency $\mathbf{A}^{(l+1)} = \mathbf{A}^{(l)}[idx, idx]$. The corresponding rows of the features matrix are selected $\tilde{\mathbf{X}}^{(l)} = \mathbf{X}^{(l)}[idx, :]$ and then a re-normalisation is applied to give features for the next layer $\mathbf{X}^{(l+1)} = \tilde{\mathbf{X}}^{(l)} \odot (\mathrm{sigmoid}(\mathbf{y})\mathbf{1}_d^T)$. The *gUnpool* operator does a reverse operation by using the indices computed in the pooling layer and using zero vectors for indices that were not selected during pooling. The architecture uses rounds of pooling and unpooling, as well as skip connections between representations of the same size. For a detailed description of the architecture we refer the reader to [52, Section 3]. We used the open source implementation used in [52] and provided by the authors at `https://github.com/HongyangGao/Graph-U-Nets`.

grabnel GRABNEL, which uses the WL feature extractor, involves a number of hyperparameters: the WL procedure is parameterised by a single hyperparameter H, which specifies the number of Weisfiler-Lehman iterations to perform. While

it is possible for H to be selected automatically via, for example, maximising the log-marginal likelihood of the surrogate model (e.g. [106]), in our case we find fixing $H = 1$ to be performing well. For the sparse Bayesian linear regression model used, we always normalise the input data into hypercubes $[0, 1]^d$ and standardise the target by deducting its mean and dividing by standard deviation. We optimise the marginal log-likelihood via a simple gradient optimiser and we set the maximum number of iterations to be 300. As described in Sec. 6.2, we need to specify a Gamma prior over over $\{\lambda_i\}$ and we use shape parameter and inverse scale parameters of 1×10^{-6}. For the acquisition optimisation, we set the maximum number of evaluation of the acquisition function to be 500: we initialise with 50 randomly sampled perturbed graphs, each of which is generated from flipping one pair of randomly selected end nodes from the base graph. To generate the initial population, we fill generate 50 candidates by mutating from the top-3 queried graphs that previously led to the largest attack loss (if we have not yet queried any graphs, we simply sample 100 randomly perturbed graphs). We then evolve the population 10 times, with each evolution cycle involving mutating the current population to generate offspring and popping the oldest members in the population. Finally, we select the top 5 unique candidates seen during the evolution process that have the highest acquisition function value (we use the Expected Improvement (EI) acquisition function) to query the victim model.

C.4 Additional Experiments

C.4.1 Comparison with Alternative Surrogate Models

We compare the surrogate performance between a GP model with RBF kernel and a Bayesian linear regression model (which is equivalent to GP with linear kernel) to justify the usage of the latter in this section. To make a fair comparison, in each of the 3 TU datasets, we randomly select 20 graph samples that the victim model (we use the trained GCN models identical to those used in Section 6.4) originally classify correctly. For each of the sample, we generate $\{25, 50, 100\}$ perturbed samples that are 1 edit distance different from the original graphs and query the victim model to obtain their respective attack losses. We then train the surrogate models with the perturbed graphs and their attack losses as the training inputs and targets, and validate their performance on a further validation set of 200 perturbed graphs with the objective of predicting their attack losses. We use 3 metrics to evaluate the quality of the surrogate model: Root Mean Squared Error (RMSE) between predicted attack losses and the ground-truth atack losses, Spearman correlation between them and the negative log-likelihood on the validation set (which assess the

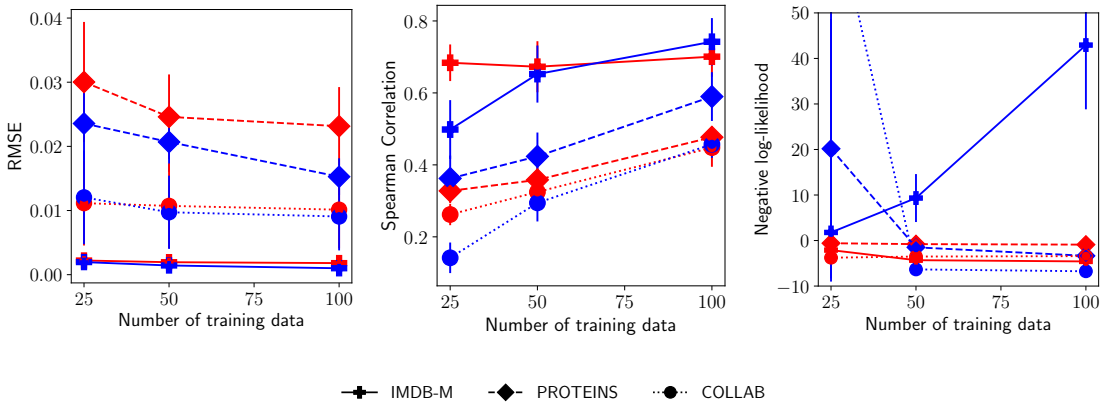


Figure C.2: Comparison of the Bayesian linear regression and GP surrogate models on 3 TU datasets in terms of RMSE (lower is better), Spearman correlation (higher is better) and the negative log-likelihood (lower is better) on validation set. Error bars denote 1 standard error.

quality of the prediction mean as long as its predictive uncertainty, as a principled uncertainty estimation is crucial in Bayesian optimisation). The results are shown in Fig. C.2: it can be seen that the difference between Bayesian linear regression and GP models are insignificant in most cases in terms of RMSE and Spearman correlation (except in the PROTEINS dataset where GP model is arguably better), whereas the uncertainty estimation seems to be more stable throughout for Bayesian linear regression. It therefore justifies our usage of Bayesian linear regression, as its performance is often comparable with GP, but it is much more cheaper in terms of computation, making computations much more tractable especially when the number of queries is modestly large.

C.4.2 Comparison with RL-S2V

We compare GRABNEL with RL-S2V on the graph classification dataset described in [37]. Each input graph is made of 1, 2 or 3 connected components. Each connected component is generated using the Erdős–Rényi random graph model (additional edges are added if the generated graph is disconnected). The label node features are set to a scalar value of 1 and the corresponding graph label is the number of connected components. The authors consider three variants of this dataset using different graph sizes, we consider the variant with the smallest graphs ($15 - 20$ nodes. The victim model, as well as the surrogate model used to compute Q-values in RL-S2V is structure2vec [38]. This embedding has a hyper-parameter determining the depth of the computational graph. We fix both to be the the smallest model considered in [37]. These choices were made to keep the computational budget to a minimum.

To adapt to the settings in [37], we only allow one edge edit (addition/deletion),

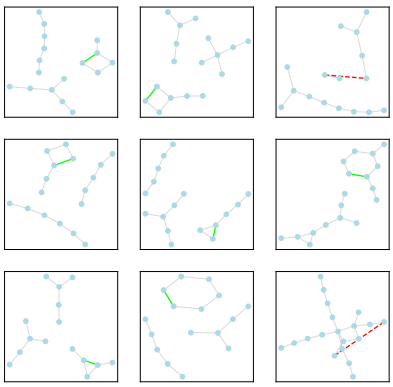


Figure C.3: Adversarial examples found by the proposed method on the ER graphs with S2V being the victim model. Similar to Fig. 6.7, Red edges denote deleted edges from the original samples and green edges indicate those added.

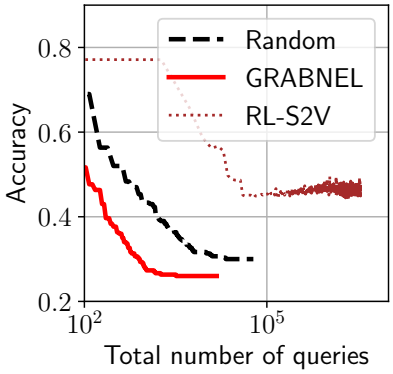


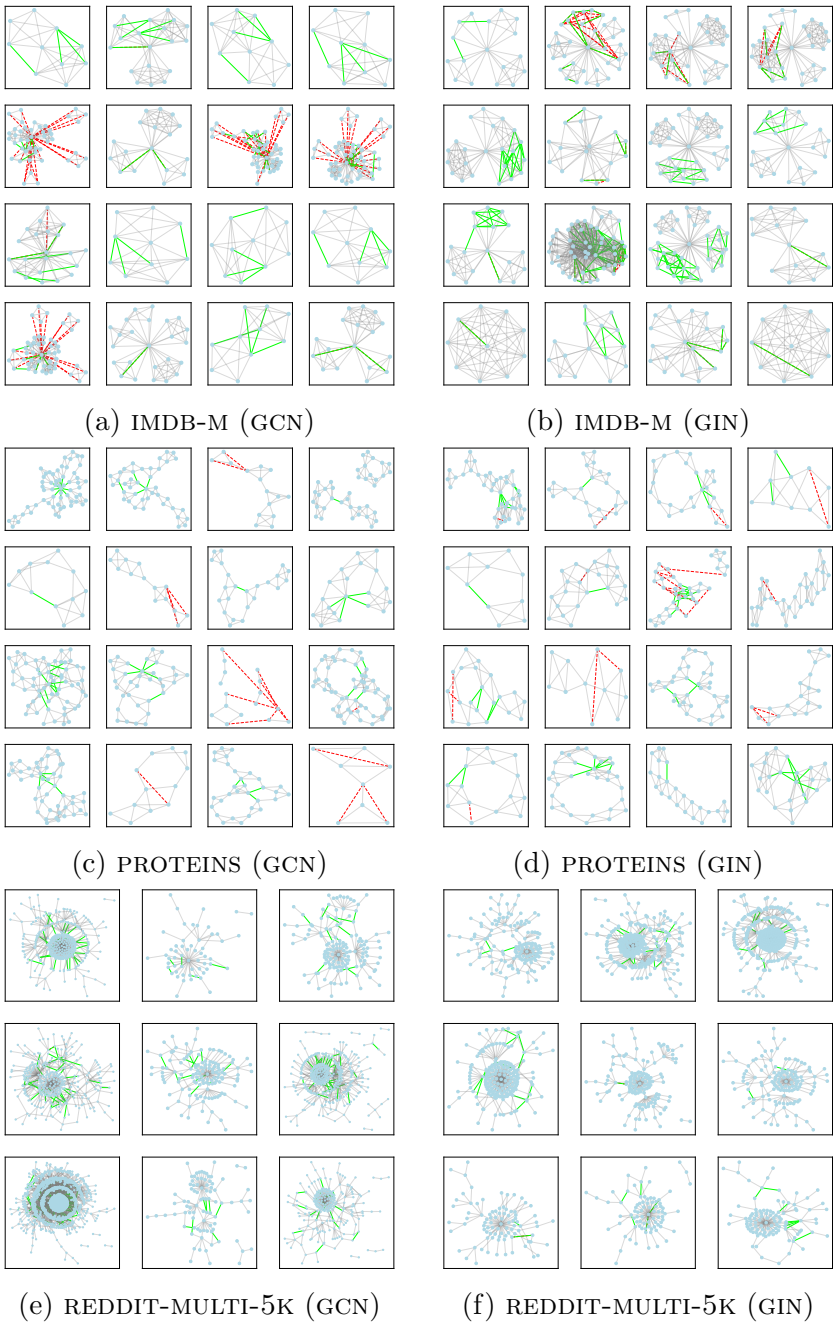
Figure C.4: Validation accuracy vs *total* number of queries to the victim model. RL-S2V requires significantly more victim model queries, as it attempts to learn an attack policy by repeatedly querying a subset of the validation set which is used for policy training.

and for GRABNEL we allow up to 100 queries to the victim model per sample in the validation set. For Random baseline, we instead allow up to 400 queries. Similar to [37], we enforce the constraint such that any edge edit must not result in a change of the number of disconnected components (i.e. the label) and any such edit proposed is rejected before querying the victim model. We show the results in Fig. C.4, and we similarly visualise some of the adversarial samples found by GRABNEL in Fig. C.3. The final performance of RL-S2V is similar to that reported in the [37], whereas we find that random perturbation is actually a very strong baseline if we give it sufficient query budget[1]. Again, we find that GRABNEL outperforms the baselines, offering orders-of-magnitude speedup compared to RL-S2V, with the main reasons being 1) GRABNEL is designed to be sample-efficient, and 2) GRABNEL does not require a separate training set *within* the validation to train a policy like what RL-S2V does. Fig. C.3 shows that the edge addition is more common than deletion in the adversarial examples in this particular case, and often the attack agent forms *ring structures*. Such structures are rather uncommon in the original graphs generated from the Erdos-Renyi generator, and are thus might not be familiar to the classifier during training. This might explain why the victim model seems particularly vulnerable to such attacks.

[1]The random baseline reported in [37] is obtained by only querying victim model with a randomly perturbed graph *once*.

C.4.3 Additional Examples of Adversarial Samples Discovered

We show more examples of the adversarial examples found by GRABNEL on various datasets and victim models in Figs C.5 and C.6.

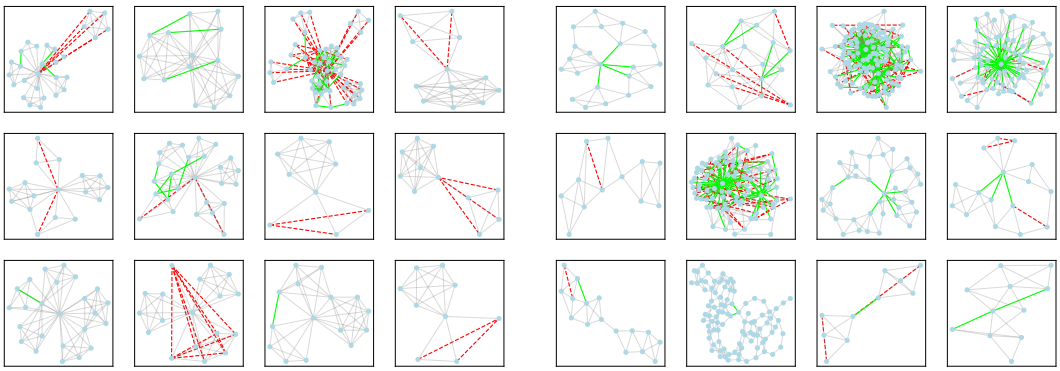


(a) IMDB-M (GCN) (b) IMDB-M (GIN)

(c) PROTEINS (GCN) (d) PROTEINS (GIN)

(e) REDDIT-MULTI-5K (GCN) (f) REDDIT-MULTI-5K (GIN)

Figure C.5: More examples of adversarial examples found by GRABNEL using GCN/GIN victim models. The colors have the same meaning as Fig. 6.7 in the main text.



(a) IMDB-M (b) PROTEINS

Figure C.6: Examples of adversarial examples found by GRABNEL using Graph U-net victim models. The colors have the same meaning as Fig. 6.7 in the main text.

C.4.4 REDDIT-MULTI-5K Results

For the REDDIT-MULTI-5K experiments, the graphs are in general typically much sparser and denoting the perturbation budget Δ with structural perturbation budget in terms of n^2 may be too lenient. Instead, we limit the perturbation budget in terms of the fraction of the *number of edges* of the individual graphs, and we set $\Delta \leq 0.03m$ for all experiments. We report the results in Table C.3 and some examples of the adversarial examples discovered can be found in Fig. C.5.

Table C.3: Test accuracy of GCN and GIN victim models REDDIT-MULTI-5K before (*clean*) and after various attack methods.

	GCN[81]	GIN[151]
Clean	45.20	48.40
Random	32.77	42.77
Genetic [37]	28.25	42.35
GRABNEL (ours)	**23.73**	**29.27**

C.4.5 Ablation Studies

In this section, we conduct ablation studies on GRABNEL on two datasets previously considered in Section 6.4 in the main text, namely the PROTEINS dataset and the MNIST-75sp image classification task. We conduct the following variants of GRABNEL to understand how different components affect the performance:

- `Random` and `GA`: Identical to those in Sec. 6.4.
- `SequentialRandom`: instead of perturbing all edges simultaneously we use the sequential perturbation generation: we divide the total query budget into

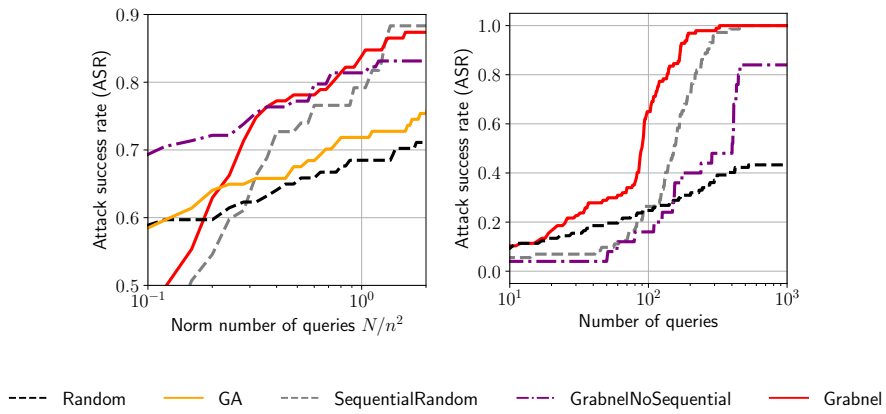


Figure C.7: Ablation studies on PROTEINS and MNIST-75sp datasets.

stages according to the description in Sec. 6.2, and at each stage we only modify one edge from the base graph *via random sampling* and commit to the perturbation that leads to the largest attack loss in the previous stage. This setup is otherwise identical to GRABNEL but the candidates are generated via random sampling instead of surrogate-guided BO.

- `GrabnelNoSequential`: GRABNEL but without the sequential perturbation selection. At each BO iteration, the BO needs to search and propose *all* (instead of one) edges to perturb up to the attack budget

- `Grabnel`: Full GRABNEL with both surrogate-guided BO and sequential perturbation selection.

We show the results in Fig. C.7: it is evident that in both cases the use of surrogates and the use of sequential perturbation selection has led to improvements over baselines. In particular, `SequentialRandom` seems also to be a simple and strong baseline, with their final performance on par with GRABNEL but is less sample efficient. In PROTEINS, GRABNEL converges much faster initially (noting the log scale of the x-axis), while the final performance is comparable between GRABNEL and `SequentialRandom`. In the MNIST-75sp task, GRABNEL is roughly 1-2× faster throughout (it is worth noting that since we explicitly link the number of queries to the amount of perturbation applied, GRABNEL being 1-2× faster also suggests that it requires 1-2× less perturbation compared to `SequantialRandom` to reach the same ASR). The strong performance on PROTEINS of `SequentialRandom` is because 1) the perturbation budget $\Delta \leq 0.03n^2$ we set is relatively lenient: with many possible adversarial examples present in the search space, the importance of the different search algorithms may diminish (as we will show later, when we set a much stricter perturbation budget, the out-performance of GRABNEL is markedly larger. This could also be seen by the margin of out-performance in Fig. C.7 when the number of queries is small), and 2) the dataset mainly features modestly-sized graphs on

which simple a relatively small number random search might already be sufficient in finding vulnerable edges. On the other hand, `GrabnelNoSequential` improves significantly over `Random`, showing the effectiveness of the surrogate in BO and in PROTEINS it starts off with a much higher ASR because it utilises the entire attack budget throughout instead of the approaches using sequential perturbation selection, which is parsimonious with respect the amount of perturbation applied. Nonetheless, in both tasks its final performance is worse than the full GRABNEL, presumably due to the fact that the unconstrained search space, which scales exponentially with the attack budget (i.e. the number of edges to edit), is too large for the surrogate model to explore effectively even for modestly-sized graphs.

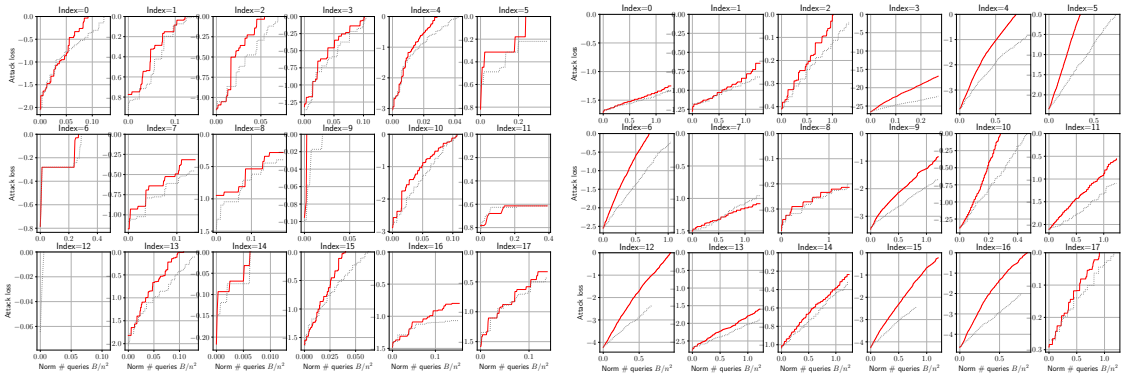
In view of the strong performance of `SequentialRandom`, we conduct more detailed experiments to compare it against the full GRABNEL and we show the results in Table C.4: with exceptions of GCN on PROTEINS and Graph U-net on IMDB-M where the two perform similarly, possibly within margin of error, GRABNEL outperforms in all other cases: it is easy to see that the margin of GRABNEL over SequentialRandom increases significantly as the task difficulty increases. This is because SequentialRandom takes more queries to reach the similar level of ASR of GRABNEL and is less efficient in increasing the attack loss.

Table C.4: Test accuracy of a GCN victim model after attacks by GRABNEL and `SequentialRandom` under perturbation budget $\Delta \leq 0.03n^2$. Mean (and $\pm$ standard deviation, if available) shown.

Victim model	GCN			Graph U-Net	
Dataset	IMDB-M	PROTEINS	COLLAB	IMDB-M	PROTEINS
Clean	$50.53_{\pm 1.4}$	$71.23_{\pm 2.6}$	$79.93_{\pm 2.1}$	55.33	79.46
GRABNEL*	$\mathbf{45.23}_{\pm 0.2}$	$10.82_{\pm 2.5}$	$\mathbf{35.38}_{\pm 9.3}$	**41.33**	**58.80**
SequentialRandom	$46.22_{\pm 0.2}$	$\mathbf{10.12}_{\pm 1.5}$	$49.80_{\pm 6.8}$	42.05	66.62

*: Taken from results in the main text.

However, as discussed, looking at the performance *only when a relatively lenient budget has been exhausted* can be misleading. We thus test the algorithms while allowing fewer edits, hence making the task more difficult. Taking the example on the PROTEINS dataset where `SequentialRandom` performs the strongest, we show a set of results with reduced perturbation budgets in Table C.5 and the margin of GRABNEL over `SequentialRandom` increases significantly as the task difficulty increases. This is because `SequentialRandom` takes more queries to reach the similar level of ASR of GRABNEL and is less efficient in increasing the attack loss: as concrete examples, we show the attack loss trajectory as a function of the number of queries to the victim model for both methods in randomly selected attack samples in Fig



(a) GCN (b) Graph U-Net

Figure C.8: Comparison of the attack loss as a function of the (normalised) number of queries to the GCN/Graph U-net victim models of GRABNEL and SequentialRandom on the PROTEINS dataset.

C.8: in successfully attacked samples (i.e. the attack loss reaches 0), it is clear to see that GRABNEL typically requires fewer queries. Even for the samples that neither managed to successfully attack, GRABNEL pushes the losses closer to 0.

Table C.5: Test acc. of GCN after attacks by GRABNEL and SequentialRandom on the PROTEINS dataset under varying perturbation budgets. Mean (± standard deviation, if available) shown.

Δ	$0.03n^2$	$0.003n^2$	$0.001n^2$
GRABNEL	$10.82_{\pm 2.5}$	**26.43**	**52.59**
SequentialRandom	$\mathbf{10.12}_{\pm 1.5}$	32.09	60.51

Another concrete example would be the attack on the MNIST-75sp task: while both GRABNEL and SequentialRandom converge eventually to 100% ASR, GRABNEL converges 2 times faster: on average, in a successfully attacked sample, GRABNEL rewires 1.84 ± 0.7 edges but SequentialRandom rewires 2.54 ± 1.0 edges (Fig C.9), which suggests that SequentialRandom needs to modify 38% more edges on average to succeed. Imperceptibility in graph attack is usually measured in terms the number of edge edits, and thus while in some cases the end performance of SequentialRandom and GRABNEL can be similar especially when a lenient perturbation budget is given, GRABNEL is both more sample-efficient and produces less perceptible attacks. Therefore, we conclude that while SequentialRandom could perform strongly for easier tasks (e.g. high perturbation budget and smaller graphs) but otherwise GRABNEL is significantly better.

C.4.6 Runtime Analysis

In this work, we particularly emphasise the desideratum of sample efficiency that we aim to find adversarial examples with the minimum number of queries to the victim model. We believe that this is a practical, and difficult, setup that accounts for the prohibitive monetary, logistic and/or opportunity costs of repeatedly querying a (possibly huge and complicated) real-life victim model. With a high query count, the attacker may also run a higher risk of getting detected. Given this objective, the cost of the algorithm should not only be considered from the viewpoint of computational runtime of the attack algorithm itself alone, and this is a primary reason why we use number of queries as the main cost criterion in our paper (this emphasis on the number of queries over runtime as the main cost metric is common in adversarial attacks in other data structures emphasising sample efficiency [105, 68, 160] and other related domains, like hyperparameter optimisation). The common assumption is that the cost of the BO itself is secondary to the cost of querying the objective function (the cost here should not be interpreted as being the computing cost alone, but includes all the potential costs discussed above). Nevertheless, the runtime analysis is still a relevant metric, and we provide a more comprehensive analysis of the algorithm runtime below.

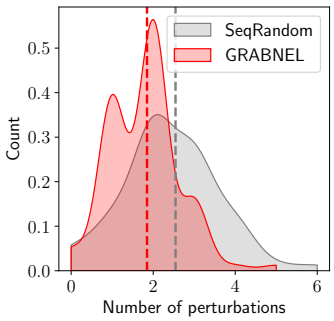


Figure C.9: Distribution of the number of edge rewiring/swapping required to successfully attack MNIST-75sp samples in GRABNEL and `SequentialRandom`. The dotted lines denote the median numbers of edge operations required.

Each iteration in the main loop of our algorithm can be broadly separated into two parts:

1. Initialisation/updates of the surrogates: this step involves the WL feature extraction, and initialisation/update of the Bayesian Linear Regression (BLR) surrogate (the complexity of this step was discussed in Sec 6.2 with the new data. Note that BLR scales much better than GPs used in related works [105].

2. Acquisition function optimisation: this step involves using genetic algorithms to optimise the acquisition function. It is further broken into 2 sub-parts:

   (a) GA steps, which involve the selection of the population and mutation and crossover operations on the parents. For the manipulations here, we do not have to store the full graphs, but we instead only have to maintain a tuple of the edges that are flipped/rewired.

   (b) Conversion of the tuple into full graph objects (we use Deep Graph Library (DGL) [139] for implementation), and call the trained BLR to obtain

predicted mean/variance and compute the acquisition function value.

Table C.6: Runtime analysis of GRABNEL in terms of average second per iteration (standard deviation in brackets; slowest step in bold). H denotes the number of WL iterations performed. Benchmarked on an otherwise idle machine with AMD Ryzen 7 CPU and 32 GB RAM. We used GCN victim model. Results may vary significantly depending on the hardware, system load and the hyperparameters used

# nodes	# edges	H	Step 1	Step 2a	Step 2b
17	106	1	0.022 (0.0027)	0.0344 (0.0002)	**0.482 (0.006)**
72	719	1	0.251 (0.003)	0.0371 (0.0006)	**0.458 (0.009)**
1961	5336	0	**1.76 (0.12)**	0.0555 (0.0007)	1.52 (0.016)

Note that even for a graph with almost $2k$ nodes and $> 5k$ edges (which is larger than most graphs in the TU dataset for graph classification), the runtime is still manageable on a mainstream desktop-grade PC. In fact, the GA itself is efficient, and the much slower step is Step 2b: instead of our algorithm itself being inefficient, this is because of the large overhead in graph representation conversion between the list of changed edges (e.g. $([1, 2], [3, 4])$, which denotes a perturbed graph with edges $e_{(1,2)}$ and $e_{(3,4)}$ flipped from the original graph) and an actual DGL graph object. For better efficiency, it should be possible to reduce the number of such conversions as the perturbed and original graphs differ only at the flipped edges which only make up a very small fraction of all edges. At the very least, since each conversion is independent for other graphs, we can parallelise it for candidates in the population of the genetic algorithm (the current code does this sequentially).

To give a better context, on a shared Intel Xeon Gold server where we conduct the majority of experiments (we unfortunately could not provide very reliable and accurate statistics due to the varying load by other users), each attack on a single graph on the IMDB-M (average $\sim$ 66 edges per graph) dataset usually takes < 1 minute. On COLLAB, which on average is much larger, each attack on average takes $\sim$ 1 hour (Note we set the maximum number of queries to be dependent on the sizes of the graph, so larger graphs are also proportionally allocated a higher query budget and hence each run could be much longer). Furthermore, the sizes of graphs within a dataset can vary a lot, and runtime also depends a lot on the difficulty of attack (if an adversarial perturbation is easily found the run is terminated early). A further comparison with RL-S2V [37] is that to run the same task on ER-graphs attack with 15-20 nodes, our method takes 30 minutes to an hour. RL-S2V, which requires a separate validation set to train policy on, requires approx. 1.5 hours with GPU acceleration and approx. 12h without (we do not currently use any GPU acceleration for our method).

Appendix D

Appendix for Chapter 7

D.1 Proofs of propositions

D.1.1 Proof of Proposition 1

Disjoint Unions. Let $\mathbf{z} \in \mathcal{R}_{\mathbf{Q}}$ and $\tilde{\mathbf{z}} \in \mathcal{R}_{\mathbf{Q}'}$ such that for some $i \in I$ we have $Q_i \neq Q_i'$. If $\mathbf{z} = \tilde{\mathbf{z}}$, it implies that $||\mathbf{z}_{\mathcal{J}_i} - \mathbf{x}_{\mathcal{J}_i}|| = Q_i$ and $||\tilde{z}_{\mathcal{J}_i} - \mathbf{x}_{\mathcal{J}_i}|| = Q_i'$ which is a contradiction.

Partition. $|\mathcal{J}_i| \leq R_i$, and $||\mathbf{z}_{\mathcal{J}_i} - \mathbf{x}_{\mathcal{J}_i}|| \leq Q_i$ hence $\mathcal{X} = \cup_{Q \leq R} \mathcal{R}_Q^R$.

D.1.2 Proof of Proposition 2

As the noise for each entry is independent we can decompose the probabilities as so

$$\frac{P(\phi(\tilde{\mathbf{x}}) = \mathbf{z})}{P(\phi(\mathbf{x}) = \mathbf{z})} = \prod_{k \in [N]} \frac{P(\phi(\tilde{\mathbf{x}})_k = \mathbf{z}_k)}{P(\phi(\mathbf{x})_k = \mathbf{z}_k)}. \tag{D.1}$$

Furthermore, as each components belongs to exactly one edge community.

$$\prod_{k \in [N]} \frac{P(\phi(\tilde{\mathbf{x}})_k = \mathbf{z}_k)}{P(\phi(\mathbf{x})_k = \mathbf{z}_k)} = \prod_{i=1}^{I} \prod_{k \in \mathcal{C}_i} \frac{P(\phi(\tilde{\mathbf{x}})_k = \mathbf{z}_k)}{P(\phi(\mathbf{x})_k = \mathbf{z}_k)}. \tag{D.2}$$

We note that for k where $\tilde{\mathbf{x}}_k = \mathbf{x}_k$ this fraction is one, so we can focus on terms when $\tilde{\mathbf{x}}_k \neq \mathbf{x}_k$. In equations this can be written as

$$\prod_{i=1}^{I} \prod_{k \in \mathcal{C}_i} \frac{P(\phi(\tilde{\mathbf{x}})_k = \mathbf{z}_k)}{P(\phi(\mathbf{x})_k = \mathbf{z}_k)} = \prod_{i=1}^{I} \prod_{k \in \mathcal{J}_i} \frac{P(\phi(\tilde{\mathbf{x}})_k = \mathbf{z}_k)}{P(\phi(\mathbf{x})_k = \mathbf{z}_k)} \tag{D.3}$$

We can consider what the terms are equal to when $\mathbf{x}_k = \mathbf{z}_k$ and when $\mathbf{x}_k \neq \mathbf{z}_k$

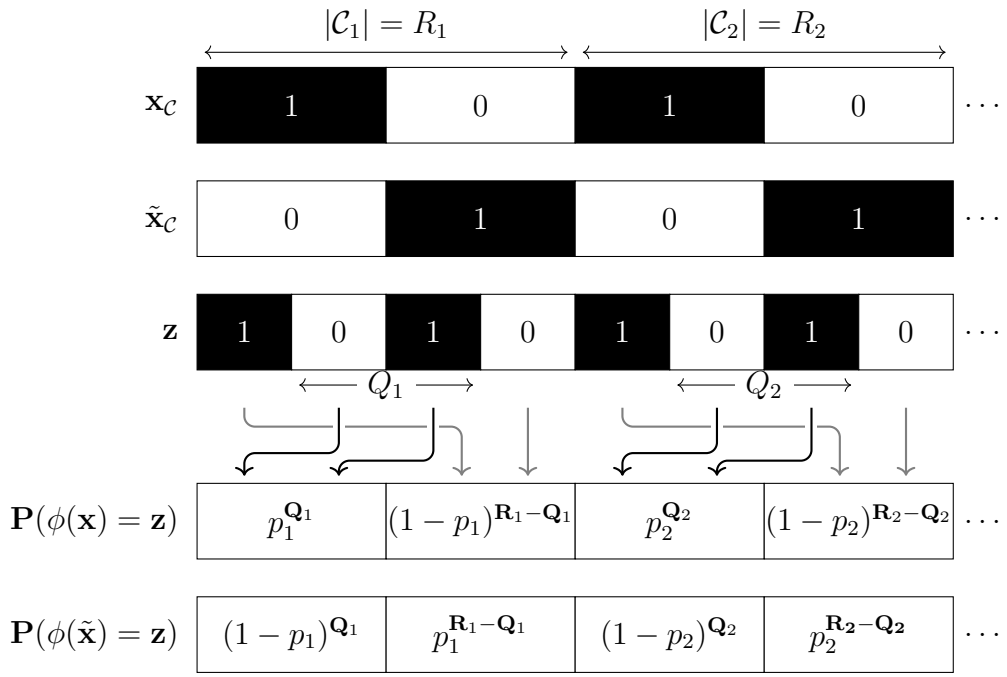


Figure D.1: Pictorial representation of where the terms in Proposition 2 come from.

(assuming that $\mathbf{x}_k \neq \tilde{\mathbf{x}}_k$). We get

$$\frac{P(\phi(\tilde{\mathbf{x}})_k = \mathbf{z}_k)}{P(\phi(\mathbf{x})_k = \mathbf{z}_k)} = \begin{cases} \frac{p_i}{1-p_i} & \text{if } \mathbf{x}_k = \mathbf{z}_k \text{ and } \mathbf{x}_k \neq \tilde{\mathbf{x}}_k \\ \frac{1-p_i}{p_i} & \text{if } \mathbf{x}_k \neq \mathbf{z}_k \text{ and } \mathbf{x}_k \neq \tilde{\mathbf{x}}_k \end{cases}. \tag{D.4}$$

In total there are R_i terms in each product, of which Q_i are the first case and $R_i - Q_i$ are in case two. Thus

$$\prod_{i=1}^{I} \prod_{k \in \mathcal{J}_i} \frac{P(\phi(\tilde{\mathbf{x}})_k = \mathbf{z}_k)}{P(\phi(\mathbf{x})_k = \mathbf{z}_k)} = \prod_{i=1}^{I} \left(\frac{p_i}{1-p_i}\right)^{Q_i} \left(\frac{1-p_i}{p_i}\right)^{R_i - Q_i} \tag{D.5}$$

$$= \prod_{i=1}^{C} \left(\frac{p_i}{1-p_i}\right)^{2Q_i - R_i} \tag{D.6}$$

$$= \prod_{i=1}^{C} \left(\frac{1-p_i}{p_i}\right)^{R_i - 2Q_i} \tag{D.7}$$

as required. We provide Fig. D.1 as a visual aid to the proof.

D.1.3 Proof of Proposition 3

We have $\mathcal{R}_\mathbf{Q} = \{\mathbf{z} \in \mathcal{X} : \|\mathbf{z}_{\mathcal{J}_i} - \mathbf{x}_{\mathcal{J}_i}\|_0 = Q_i\}$. The probability $\mathbb{P}(\phi(\mathbf{x}) \in \mathcal{R}_\mathbf{Q})$ corresponds to each set R_i having Q_i entries not being flipped or equivalently $R_i - Q_i$ entries being flipped. Each node pair is flipped with a probability of p_i. Since all flips

are independent we can express the probability as $\mathbb{P}(\phi(\mathbf{x}) \in \mathcal{R}_{\mathbf{Q}}) = \prod_{i=1}^{C} \text{Bin}(R_i - Q_i | R_i, p_i)$.

D.2 Implementation

D.2.1 Estimations of probabilities

The quantities $p_y(\mathbf{x})$ cannot be computed in closed form for general f. Hence, we resolve to lower bound p^* and upper bound $p_y(\mathbf{x}), y \neq y^*$ via sampling. To achieve this, we use the Clopper-Pearson interval [22].

D.2.2 Symmetries certification

Solving the optimization problem defined in Eq. 7.7 is difficult as certificates have to be computed for every $\tilde{\mathbf{x}}$ in the ball around $\mathbf{x}$: $\mathcal{B}_r(\mathbf{x})$. However, in practice, $\Phi_{\mathbf{x},\tilde{\mathbf{x}}}(p^*, y^*)$ displays some symmetries depending on the noise distribution $\phi(\mathbf{x})$.

In the case of isotropic noise, the regions $\mathcal{H}_k$ and values c_k only depends on $\|\mathbf{x} - \tilde{\mathbf{x}}\|_0$. This implies $\Phi_{\mathbf{x},\tilde{\mathbf{x}}}(p^*, y^*) = \Phi_{\mathbf{x},\tilde{\mathbf{x}}'}(p^*, y^*)$ for all $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \in \mathcal{S}_r(\mathbf{x})$ which reduce the search on every spheres.

In the case of anisotropic noise, the regions $\mathcal{H}_k$ and values c_k only depends on $\|\mathbf{x}_{\mathcal{C}_i} - \tilde{\mathbf{x}}_{\mathcal{C}_i}\|_0$. This implies $\Phi_{\mathbf{x},\tilde{\mathbf{x}}}(p^*, y^*) = \Phi_{\mathbf{x},\tilde{\mathbf{x}}'}(p^*, y^*)$ for all $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \in \mathcal{S}_{\mathbf{R}}(\mathbf{x})$.

Bibliography

[1] Radhakrishna Achanta et al. "SLIC superpixels compared to state-of-the-art superpixel methods". In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2274–2282.

[2] Réka Albert and Albert-László Barabási. "Statistical mechanics of complex networks". In: *Reviews of modern physics* 74.1 (2002), p. 47.

[3] László Babai. "Graph isomorphism in quasipolynomial time". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 2016, pp. 684–697.

[4] Muhammet Balcilar et al. "Bridging the gap between spectral and spatial domains in graph neural networks". In: *arXiv preprint arXiv:2003.11702* (2020).

[5] A. Barabási and R. Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (Oct. 1999), pp. 509–512.

[6] Albert-László Barabási. "Network science". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1987 (2013), p. 20120375.

[7] Peter Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[8] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. "Tikhonov regularization and semi-supervised learning on large graphs". In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3. IEEE. 2004, pp. iii–1000.

[9] Rajendra Bhatia. *Matrix analysis*. Vol. 169. Springer Science & Business Media, 2013.

[10] F. Bianchi et al. "Graph Neural Networks With Convolutional ARMA Filters". In: *IEEE Transactions on Pattern Analysis Machine Intelligence* 44.07 (2022), pp. 3496–3507.

[11] Daniel Bienstock and Oktay Günlük. "A degree sequence problem related to network design". In: *Networks* 24.4 (1994), pp. 195–205.

[12] Battista Biggio, Blaine Nelson, and Pavel Laskov. "Poisoning Attacks against Support Vector Machines". In: ICML'12. Edinburgh, Scotland: Omnipress, 2012, pp. 1467–1474. ISBN: 9781450312851.

[13] Battista Biggio et al. "Evasion attacks against machine learning at test time". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2013, pp. 387–402.

[14] Aleksandar Bojchevski and Stephan Günnemann. "Adversarial attacks on node embeddings via graph poisoning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 695–704.

[15] Aleksandar Bojchevski and Stephan Günnemann. "Certifiable Robustness to Graph Perturbations". In: *Advances in Neural Information Processing Systems*. 2019, pp. 8319–8330.

[16] Aleksandar Bojchevski, Johannes Klicpera, and Stephan Günnemann. "Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more". In: *International Conference on Machine Learning* (2020), pp. 1003–1013.

[17] Béla Bollobás and Oliver Riordan. "The diameter of a scale-free random graph". In: *Combinatorica* 24.1 (2004), pp. 5–34.

[18] M. M Bronstein et al. "Geometric deep learning: Going beyond Euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[19] Michael M Bronstein et al. "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges". In: *arXiv preprint arXiv:2104.13478* (2021).

[20] J. Bruna et al. "Spectral networks and deep locally connected networks on graphs". In: *International Conference on Learning Representations*. 2014.

[21] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 30.9 (2018), pp. 1616–1637.

[22] T Tony Cai. "One-sided confidence intervals in discrete distributions". In: *Journal of Statistical planning and inference* 131.1 (2005), pp. 63–88.

[23] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *2017 ieee symposium on security and privacy (sp)*. IEEE. 2017, pp. 39–57.

[24] E. Ceci and S. Barbarossa. "Robust Graph Signal Processing in the Presence of Uncertainties on Graph Topology". In: *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. June 2018, pp. 1–5.

[25] E. Ceci and S. Barbarossa. "Small Perturbation Analysis of Network Topologies". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Apr. 2018, pp. 4194–4198.

[26] Ines Chami et al. "Machine Learning on Graphs: A Model and Comprehensive Taxonomy". In: *arXiv:2005.03675* (2020).

[27] Heng Chang et al. "A restricted black-box adversarial framework towards attacking graph embedding models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 3389–3396.

[28] Jinyin Chen et al. "Adversarial Detection on Graph Structured Data". In: *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*. 2020, pp. 37–41.

[29] Jinyin Chen et al. "Fast gradient attack on network embedding". In: *arXiv preprint arXiv:1809.02797* (2018).

[30] Jinyin Chen et al. "GraphAttacker: A General Multi-Task GraphAttack Framework". In: *arXiv preprint arXiv:2101.06855* (2021).

[31] Liang Chen et al. "A survey of adversarial learning on graphs". In: *arXiv preprint arXiv:2003.05730* (2020).

[32] Yizheng Chen et al. "Practical attacks against graph-based clustering". In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017, pp. 1125–1142.

[33] F. Chung. *Spectral graph theory*. American Mathematical Society, 1997.

[34] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified adversarial robustness via randomized smoothing". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1310–1320.

[35] Alexander I Cowen-Rivers et al. "HEBO: Heteroscedastic Evolutionary Bayesian Optimisation". In: *arXiv preprint arXiv:2012.03826* (2020).

[36] D. I Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (May 2013), pp. 83–98.

[37] H. Dai et al. "Adversarial Attack on Graph Structured Data". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1115–1124.

[38] Hanjun Dai, Bo Dai, and Le Song. "Discriminative embeddings of latent variable models for structured data". In: *International conference on machine learning*. PMLR. 2016, pp. 2702–2711.

[39] Nilesh Dalvi et al. "Adversarial classification". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 99–108.

[40] E. Deadman and S. D. Relton. "Taylor's theorem for matrix functions with applications to condition number estimation". In: *Linear Algebra and its Applications* 504 (Sept. 2016), pp. 354–371.

[41] Asim Kumar Debnath et al. "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity". In: *Journal of medicinal chemistry* 34.2 (1991), pp. 786–797.

[42] M. Defferrard, X. Bresson, and P. Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29*. 2016, pp. 3844–3852.

[43] X. Dong et al. "Graph Signal Processing for Machine Learning: A Review and New Perspectives". In: *IEEE Signal Processing Magazine* 37.6 (2020), pp. 117–127.

[44] Iddo Drori et al. "Learning to solve combinatorial optimization problems on real-world graphs in linear time". In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2020), pp. 19–24.

[45] Vijay Prakash Dwivedi et al. "Benchmarking graph neural networks". In: *arXiv* (2020). ISSN: 23318422. arXiv: 2003.00982.

[46] Negin Entezari et al. "All you need is low (rank) defending against adversarial attacks on graphs". In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. 2020, pp. 169–177.

[47] F. Gama, J. Bruna, and A. Ribeiro. "Stability Properties of Graph Neural Networks". In: *arXiv:1905.04497* (2019).

[48] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. "Stability properties of graph neural networks". In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 5680–5695.

[49] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. "Stability of graph scattering transforms". In: *Advances in Neural Information Processing Systems* 32 (2019).

[50] Fernando Gama et al. "Convolutional neural network architectures for signals supported on graphs". In: *IEEE Transactions on Signal Processing* 67.4 (2018), pp. 1034–1049.

[51] B. Gao and L. Pavel. "On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning". In: *arXiv:1704.00805* (2017).

[52] Hongyang Gao and Shuiwang Ji. "Graph u-nets". In: *international conference on machine learning*. PMLR. 2019, pp. 2083–2092.

[53] Zhidong Gao, Rui Hu, and Yanmin Gong. "Certified Robustness of Graph Classification against Topology Attack with Randomized Smoothing". In: *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE. 2020, pp. 1–6.

[54] Simon Geisler, Daniel Zügner, and Stephan Günnemann. "Reliable graph neural networks via robust aggregation". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 13272–13284.

[55] Edgar N Gilbert. "Random graphs". In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144.

[56] Justin Gilmer et al. "Adversarial spheres". In: *arXiv preprint arXiv:1801.02774* (2018).

[57] Justin Gilmer et al. "Neural message passing for quantum chemistry". In: *International conference on machine learning*. PMLR. 2017, pp. 1263–1272.

[58] Vladimir Gligorijević et al. "Structure-based protein function prediction using graph convolutional networks". In: *Nature communications* 12.1 (2021), pp. 1–14.

[59] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations*. 2015. URL: http://arxiv.org/abs/1412.6572.

[60] Sven Gowal et al. "Scalable verified training for provably robust image classification". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 4842–4851.

[61] Stephan Günnemann. "Graph Neural Networks: Adversarial Robustness". In: *Graph Neural Networks: Foundations, Frontiers, and Applications*. Ed. by Lingfei Wu et al. Singapore: Springer Singapore, 2022, pp. 149–176.

[62] William L Hamilton. "Graph representation learning". In: *Synthesis Lectures on Artifical Intelligence and Machine Learning* 14.3 (2020), pp. 1–159.

[63] N. J. Higham. *Functions of Matrices: Theory and Computation*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008.

[64] N. J. Higham and S. D. Relton. "Higher Order Fréchet Derivatives of Matrix Functions and the Level-2 Condition Number". In: *SIAM Journal on Matrix Analysis and Applications* 35.3 (July 2014), pp. 1019–1037.

[65] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. Society for Industrial and Applied Mathematics, 2002.

[66] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[67] Yujia Huang et al. "Training certifiably robust neural networks with efficient local lipschitz bounds". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22745–22757.

[68] Zhichao Huang, Yaowei Huang, and Tong Zhang. "CorrAttack: Black-box Adversarial Attack with Structured Search". In: *arXiv preprint arXiv:2010.01250* (2020).

[69] E. Isufi et al. "Filtering Random Graph Processes Over Random Time-Varying Graphs". In: *IEEE Transactions on Signal Processing* 65.16 (Aug. 2017), pp. 4406–4421.

[70] Jinyuan Jia et al. "Certified robustness of community detection against adversarial structural perturbation via randomized smoothing". In: *Proceedings of The Web Conference 2020* (2020), pp. 2718–2724.

[71] Hongwei Jin, Zishun Yu, and Xinhua Zhang. "Certifying Robust Graph Classification under Orthogonal Gromov-Wasserstein Threats". In: *Advances in Neural Information Processing Systems*. 2022.

[72] Hongwei Jin, Zishun Yu, and Xinhua Zhang. "Orthogonal Gromov Wasserstein Distance with Efficient Lower Bound". In: *The 38th Conference on Uncertainty in Artificial Intelligence*. 2022. URL: https://openreview.net/forum?id=rtZlKwLo9x5.

[73] Hongwei Jin et al. "Certified robustness of graph convolution networks for graph classification under topological attacks". In: *Advances in neural information processing systems* 33 (2020), pp. 8463–8474.

[74] Wei Jin et al. "Adversarial Attacks and Defenses on Graphs: A Review, A Tool and Empirical Studies". In: *SIGKDD Explor. Newsl.* 22.2 (2021), pp. 19–34. DOI: 10.1145/3447556.3447566. URL: https://doi.org/10.1145/3447556.3447566.

[75] Matt Jordan and Alexandros G Dimakis. "Exactly computing the local lipschitz constant of relu networks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7344–7353.

[76] Guy Katz et al. "Reluplex: An efficient SMT solver for verifying deep neural networks". In: *International conference on computer aided verification*. Springer. 2017, pp. 97–117.

[77] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. "Interpretable Stability Bounds for Spectral Graph Filters". In: *International conference on machine learning*. 2021.

[78] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. "On the Stability of Graph Convolutional Neural Networks under Edge Rewiring". In: *arXiv preprint arXiv:2010.13747* (2020).

[79] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. "On The Stability of Polynomial Spectral Graph Filters". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2020), pp. 5350–5354.

[80] Nicolas Keriven, Alberto Bietti, and Samuel Vaiter. "Convergence and stability of graph convolutional networks on large random graphs". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 21512–21523.

[81] T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations (ICLR)*. 2017.

[82] Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. "Understanding attention and generalization in graph neural networks". In: *NeurIPS* (2019). ISSN: 23318422. arXiv: 1905.02850.

[83] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[84] Mathias Lecuyer et al. "Certified robustness to adversarial examples with differential privacy". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 656–672.

[85] Guang-He Lee et al. "Tight certificates of adversarial robustness for randomly smoothed classifiers". In: *Advances in Neural Information Processing Systems* 32 (2019).

[86] R. Levie, E. Isufi, and G. Kutyniok. "On the Transferability of Spectral Graph Filters". In: *arXiv:1901.10524* (2019).

[87] R. Levie et al. "CayleyNets: Graph Convolutional Neural Networks With Complex Rational Spectral Filters". In: *IEEE Transactions on Signal Processing* 67.1 (2019), pp. 97–109.

[88] Ron Levie et al. "Transferability of Spectral Graph Convolutional Neural Networks." In: *J. Mach. Learn. Res.* 22 (2021), pp. 272–1.

[89] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[90] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. "Towards more practical adversarial attacks on graph neural networks". In: *Advances in Neural Information Processing Systems 33* (2020).

[91] Yao Ma et al. "Graph Adversarial Attack via Rewiring". In: KDD '21. Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 1161–1169. ISBN: 9781450383325. DOI: 10.1145/3447548.3467416. URL: https://doi.org/10.1145/3447548.3467416.

[92] Aleksander Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rJzIBfZAb.

[93] Sohir Maskey, Ron Levie, and Gitta Kutyniok. "Transferability of graph neural networks: an extended graphon approach". In: *arXiv preprint arXiv:2109.10096* (2021).

[94] Seyed-Mohsen Moosavi-Dezfooli et al. "Universal adversarial perturbations". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1765–1773.

[95] Christopher Morris et al. "TUDataset: A collection of benchmark datasets for learning with graphs". In: *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. 2020. arXiv: 2007.08663. URL: www.graphlearning.io.

[96] Christopher Morris et al. "Weisfeiler and leman go neural: Higher-order graph neural networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 4602–4609.

[97] Luis Munoz-González. "Bayesian Optimization for Black-Box Evasion of Machine Learning Systems". PhD thesis. Imperial College London, 2017.

[98] Hoang Nt and Takanori Maehara. "Revisiting graph neural networks: All we have is low-pass filters". In: *arXiv preprint arXiv:1905.09550* (2019).

[99] Vladyslav Oles, Nathan Lemons, and Alexander Panchenko. "Efficient estimation of a Gromov–Hausdorff distance between unweighted graphs". In: *arXiv preprint arXiv:1909.09772* (2019).

[100] Antonio Ortega. *Introduction to graph signal processing*. Cambridge University Press, 2022.

[101] Antonio Ortega et al. "Graph Signal Processing: Overview, Challenges, and Applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[102] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. "Semidefinite relaxations for certifying robustness to adversarial examples". In: *Advances in Neural Information Processing Systems* 31 (2018).

[103] Raksha Ramakrishna, Hoi To Wai, and Anna Scaglione. "A User Guide to Low-Pass Graph Signal Processing and Its Applications: Tools and Applications". In: *IEEE Signal Processing Magazine* 37.6 (2020), pp. 74–85.

[104] Emanuele Rossi et al. "SIGN: Scalable Inception Graph Neural Networks". In: *arXiv preprint arXiv:2004.11198* (2020).

[105] Binxin Ru et al. "Bayesopt adversarial attack". In: *International Conference on Learning Representations*. 2020.

[106] Binxin Ru et al. "Interpretable Neural Architecture Search via Bayesian Optimisation with Weisfeiler-Lehman Kernels". In: *International Conference on Learning Representations (ICLR)* (2021).

[107] Benjamin IP Rubinstein et al. "Antidote: understanding and defending against poisoning of anomaly detectors". In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. 2009, pp. 1–14.

[108] Walter Rudin et al. *Principles of mathematical analysis*. Vol. 3. McGraw-hill New York, 1976.

[109] Luana Ruiz, Luiz FO Chamon, and Alejandro Ribeiro. "Graphon signal processing". In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 4961–4976.

[110] Luana Ruiz, Luiz FO Chamon, and Alejandro Ribeiro. "The graphon fourier transform". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 5660–5664.

[111] Aliaksei Sandryhaila and José MF Moura. "Discrete signal processing on graphs". In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[112] S. Segarra and A. Ribeiro. "Stability and continuity of centrality measures in weighted graphs". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Apr. 2018, pp. 3387–3391.

[113] Ali Shafahi et al. "Are adversarial examples inevitable?" In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=r1lWUoA9FQ.

[114] Nino Shervashidze et al. "Weisfeiler-lehman graph kernels." In: *Journal of Machine Learning Research* 12.9 (2011).

[115] Han Shi et al. "Bridging the gap between sample-based and one-shot neural architecture search with bonas". In: *Advances in Neural Information Processing Systems 33* (2020).

[116] Satya Narayan Shukla et al. "Black-box adversarial attacks with bayesian optimization". In: *arXiv preprint arXiv:1909.13857* (2019).

[117] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. "Vertex-frequency analysis on graphs". In: *Applied and Computational Harmonic Analysis* 40.2 (2016), pp. 260–291.

[118] David Silver et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.

[119] Sahil Singla and Soheil Feizi. "Second-order provable defenses against adversarial attacks". In: *International conference on machine learning*. PMLR. 2020, pp. 8981–8991.

[120] Michael Spivak. *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*. CRC press, 2018.

[121] Ljubisa Stankovic et al. "Understanding the Basis of Graph Signal Processing via an Intuitive Example-Driven Approach [Lecture Notes]". In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 133–145. DOI: 10.1109/MSP.2019.2929832.

[122] Angelika Steger and Nicholas C Wormald. "Generating random regular graphs quickly". In: *Combinatorics, Probability and Computing* 8.04 (1999), pp. 377–396.

[123] Mahito Sugiyama and Karsten Borgwardt. "Halting in random walk kernels". In: *Advances in neural information processing systems* 28 (2015).

[124] Yongduo Sui et al. "Causal attention for interpretable and generalizable graph classification". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 1696–1705.

[125] Peter Súkenık, Aleksei Kuvshinov, and Stephan Günnemann. "Intriguing Properties of Input-dependent Randomized Smoothing". In: *arXiv preprint arXiv:2110.05365* (2021).

[126] Lichao Sun et al. "Adversarial attack and defense on graph data: A survey". In: *arXiv preprint arXiv:1812.10528* (2018).

[127] Fnu Suya et al. "Query-limited black-box attacks to classifiers". In: *NIPS Workshop* (2017).

[128] Christian Szegedy et al. "Intriguing properties of neural networks". In: *International Conference on Learning Representations*. 2014. URL: http://arxiv.org/abs/1312.6199.

[129] Haoteng Tang et al. "Adversarial attack on hierarchical graph pooling neural networks". In: *arXiv preprint arXiv:2005.11560* (2020).

[130] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. "Evaluating Robustness of Neural Networks with Mixed Integer Programming". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HyGIdiRqtm.

[131] Matteo Togninalli et al. "Wasserstein weisfeiler-lehman graph kernels". In: *Advances in Neural Information Processing Systems* 32 (2019).

[132] L. N. Trefethen and M. Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, 2005.

[133] Dimitris Tsipras et al. "Robustness May Be at Odds with Accuracy". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=SyxAb30cY7.

[134] Saurabh Verma and Zhi-Li Zhang. "Stability and generalization of graph convolutional neural networks". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 1539–1548.

[135] Soroush Vosoughi, Deb Roy, and Sinan Aral. "The spread of true and false news online". In: *Science* 359.6380 (2018), pp. 1146–1151.

[136] Xingchen Wan et al. "Think Global and Act Local: Bayesian Optimisation over High-Dimensional Categorical and Mixed Search Spaces". In: *International Conference on Machine Learning (ICML)* (2021).

[137] Binghui Wang et al. "Certified robustness of graph neural networks against adversarial structural perturbation". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (2021), pp. 1645–1653.

[138] Binghui Wang et al. "Evasion Attacks to Graph Neural Networks via Influence Function". In: *arXiv preprint arXiv:2009.00203* (2020).

[139] Minjie Wang et al. "Deep graph library: A graph-centric, highly-performant package for graph neural networks". In: *arXiv preprint arXiv:1909.01315* (2019).

[140] Marcin Waniek et al. "Hiding individuals and communities in a social network". In: *Nature Human Behaviour* 2.2 (2018), pp. 139–147.

[141] Duncan J Watts and Steven H Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393.6684 (1998), pp. 440–442.

[142] David P Wipf et al. "A new view of automatic relevance determination." In: *NIPS*. 2007, pp. 1625–1632.

[143] Eric Wong and Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5286–5295.

[144] Felix Wu et al. "Simplifying Graph Convolutional Networks". In: *International Conference on Machine Learning*. 2019, pp. 6861–6871.

[145] Huijun Wu et al. "Adversarial examples on graph data: Deep insights into attack and defense". In: *arXiv preprint arXiv:1903.01610* (2019).

[146] Lingfei Wu et al. "Graph neural networks for natural language processing: A survey". In: *arXiv preprint arXiv:2106.06090* (2021).

[147] Z. Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *arXiv:1901.00596* (2019).

[148] Han Xu et al. "Adversarial attacks and defenses in images, graphs and text: A review". In: *International Journal of Automation and Computing* 17.2 (2020), pp. 151–178.

[149] Jing Xu, Stjepan Picek, et al. "Explainability-based Backdoor Attacks Against Graph Neural Networks". In: *arXiv preprint arXiv:2104.03674* (2021).

[150] Kaidi Xu et al. "Topology attack and defense for graph neural networks: An optimization perspective". In: *International Joint Conference on Artificial Intelligence*. 2019, pp. 3961–3967.

[151] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. 2019. URL: `https://openreview.net/forum?id=ryGs6iA5Km`.

[152] Ramon Xulvi-Brunet and Igor M Sokolov. "Reshuffling scale-free networks: From random to assortative". In: *Physical Review E* 70.6 (2004), p. 066102.

[153] Yao-Yuan Yang et al. "A closer look at accuracy vs. robustness". In: *Advances in neural information processing systems* 33 (2020), pp. 8588–8601.

[154] Rex Ying et al. "Hierarchical graph representation learning with differentiable pooling". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2018).

[155] Zhitao Ying et al. "Gnnexplainer: Generating explanations for graph neural networks". In: *Advances in neural information processing systems* 32 (2019).

[156] Yuning You et al. "Graph contrastive learning with augmentations". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 5812–5823.

[157] Xiaoyong Yuan et al. "Adversarial examples: Attacks and defenses for deep learning". In: *IEEE transactions on neural networks and learning systems* 30.9 (2019), pp. 2805–2824.

[158] Hongyang Zhang et al. "Theoretically principled trade-off between robustness and accuracy". In: *International conference on machine learning*. PMLR. 2019, pp. 7472–7482.

[159] Zaixi Zhang et al. "Backdoor attacks to graph neural networks". In: *arXiv preprint arXiv:2006.11165* (2020).

[160] Zhuosheng Zhang and Shucheng Yu. "BO-DBA: Query-Efficient Decision-Based Adversarial Attacks via Bayesian Optimization". In: *arXiv preprint arXiv:2106.02732* (2021).

[161] Pu Zhao et al. "On the Design of Black-box Adversarial Examples by Leveraging Gradient-free Optimization and Operator Splitting Method". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 121–130.

[162] Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81.

[163] Dongmian Zou and Gilad Lerman. "Graph convolutional neural networks via scattering". In: *Applied and Computational Harmonic Analysis* 49.3 (2020), pp. 1046–1074.

[164] D. Zügner, Akbarnejad A, and S. Günnemann. "Adversarial Attacks on Neural Networks for Graph Data". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. July 2018, pp. 2847–2856.

[165] D. Zügner and S. Günnemann. "Certifiable Robustness and Robust Training for Graph Convolutional Networks". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 246–256.

[166] Daniel Zügner and Stephan Günnemann. "Adversarial attacks on graph neural networks via meta learning". In: 2019.

[167] Daniel Zügner et al. "Adversarial attacks on graph neural networks: Perturbations and their patterns". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14.5 (2020), pp. 1–31.